Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

SISTEMA DE PREVENCIÓN DE INTRUSOS PARA ENTORNOS INALÁMBRICOS Y DOMÉSTICOS

Autor: Francisco Javier Santos Rodríguez

Tutor: Ignacio Campos Rivera

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

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

SISTEMA DE PREVENCIÓN DE INTRUSOS PARA ENTORNOS INALÁMBRICOS Y DOMÉSTICOS

Autor:
Francisco Javier Santos Rodríguez

Tutor:
Ignacio Campos Rivera
Profesor asociado

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: SISTEMA DE PREVENCIÓN DE INTRUSOS PARA ENTORNOS INALÁMBRICOS Y DOMÉSTICOS

Autor: Francisco Javier Santos Rodríguez

Tutor: Ignacio Campos Rivera

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

Agradecimientos

En primer lugar, agradecer a mi familia por todo su apoyo y todo lo que han hecho por mí, ya que, sin ellos, sería imposible llegar a este instante.

Agradecer también a mis amigos y compañeros de carrera, que desde el primer año hasta el último han compartido momentos conmigo haciendo que estos cuatro años de carrera sean más fáciles. En especial a aquellos que aun estando estos últimos años en distintas ramas, no hemos perdido el contacto y han estado ahí siempre que lo necesitaba.

Por último, agradecer a todos aquellos profesores que durante esta etapa me han ayudado a formarme intelectualmente.

Francisco Javier Santos Rodríguez

Sevilla, 2017

Resumen

El proyecto consiste en la realización de un *WIPS (Wireless Intrusion Prevention System),* sistema de prevención de intrusos para redes inalámbricas, orientado a redes domésticas, el software está desarrollado en *C* en su mayoría, aunque también se hace uso de *HTML*, *Java* y *Shell-Script* para la interfaz web.

Este *WIPS*, pretende detectar y parar los ataques *DoS*, *MITM* y suplantación de *SSID*, incluyendo también una base de datos con el registro de los clientes habituales. El motor del WIPS es *Snort* pero también integra otras herramientas cómo *Arpwatch*, *Iptables*, la suite *Aircrack-ng*, *PostgreSQL* y *Apache/Tomcat* para la interfaz web.

El objetivo del proyecto es unificar todas las herramientas anteriormente citadas en una única que permita dotar de una mayor seguridad al hogar y mediante su interfaz de gestión, facilitar el uso al usuario.

Abstract

The project is about a *WIPS (Wireless Intrusion Prevention System)* for domestic networks, the software has developed in *C* principally, but it includes *HTML*, *Java* and *Shell-Script* too.

This *WIPS* was designed to detect and stop *DoS*, *MITM* and *fake SSID* attack, it includes a data base for regular clients. The engine of the *WIPS* is *Snort*, but it combines another tools as *Arpwatch*, *Iptables*, the suite *Aircrack-ng*, *PostgreSQL* and *Apache/Tomact* for the web interface.

The main goal is integrated all above tools in a unique tool to increase the security of the home and through the web interface make easier for the user.

Índice

Agradecimientos i

Resumen i

Abstract i

Índice i

Índice de Tablas i

Índice de Figuras i

1 Introducción 1

1.1 Objetivo 1

1.2 Descripción del WIPS 1

1.3 Estado del arte 2

1.4 Alcance 2

1.5 Motivación 2

2 Arquitectura de red 5

2.1 Hardware 5

2.2 Interconexión 6

2.3 Esquema de red 6

3 Arquitectura software 7

3.1 Herramientas 7

3.1.1 Snort 7

3.1.2 Aircarck-ng 9

3.1.3 Arpwatch 10

3.1.4 Iptables 11

3.1.5 Apache/Tomcat 12

3.1.6 PostgreSQL 13

3.2 Funcionamiento del WIPS 13

3.3 Estructura del código 14

3.4 Módulos 16

3.4.1 Integración 16

3.4.2 Módulo 1: *DoS* 21

3.4.3 Módulo 2: MITM 24

3.4.4 Módulo 3: SSID Duplicado 27

3.4.5 Módulo 4: BBDD 31

4 Conclusiones 35

4.1 Beneficios 35

4.2 Líneas futuras 35

5 Anexos 37

5.1 Anexo A. Instalación de herramientas 37

5.1.1 Snort 37

5.1.2 Arpwatch 39

5.1.3 Aircrack-ng 40

5.1.4 PostgreSQL 40
5.1.5 Apache/Tomcat 41
5.2 Anexo B. Configuración de la Raspberry cómo punto de acceso *43*
5.3 Anexo C. Reglas de Snort *45*
5.4 Anexo D. Código *48*

Referencias **139**

ÍNDICE DE TABLAS

Tabla 1 Esquema de ficheros 16

ÍNDICE DE FIGURAS

Figura 1 Raspberry Pi 2 Modelo B 5
Figura 2 Antena WiFi TP-Link 5
Figura 3 Diagrama de interconexión 6
Figura 4 Diagrama de Red 6
Figura 5 Tablas Iptables 11
Figura 6 Diagrama WBS del proyecto 14
Figura 7 Diagrama de paquetes del proyecto 15
Figura 8 Diagrama general de ficheros 15
Figura 9 Diagrama flujo integración 17
Figura 10 Estructura contenido_fichero 17
Figura 11 Estructura estructura_regla 18
Figura 12 Diagrama flujo lee_fichero() 19
Figura 13 Diagrama flujo crea_y_escribe_regla() 20
Figura 14 Diagrama flujo recarga_Snort() 21
Figura 15 Diagrama flujo Módulo1: DoS 22
Figura 16 Diagrama flujo función comprueba_Coincidencia() 23
Figura 17 Diagrama flujo Módulo 2: MITM 24
Figura 18 Diagrama flujo busca_Cambio_En_Mac() 25
Figura 19 Diagrama flujo registra_Regla() 26
Figura 20 Diagrama flujo detecta_Registro_Caducado() 27
Figura 21 Diagrama flujo Módulo3: SSID duplicado 28
Figura 22 Estructura módulo 3 29
Figura 23 Diagrama flujo lee_fichero_csv() 29
Figura 24 Diagrma flujo procesa_fichero_CSV() 30
Figura 25 Diagrama flujo bloquea_SSID() 31
Figura 26 Diagrama flujo Módulo 4: BBDD 32
Figura 27 Diagrama flujo inserta_en_BBDD() 33

1 INTRODUCCIÓN

"El único sistema seguro es aquél que está apagado en el interior de un bloque de hormigón protegido en una habitación sellada rodeada por guardias armados".

- Gene Spafford -

El objetivo de este capítulo es describir el alcance del proyecto, los objetivos y las principales motivaciones que nos han llevado a realizarlo

1.1 Objetivo

El objetivo del proyecto es la realización de un Sistema de Prevención de Intrusos para redes inalámbricas *(WIPS)*, implementado mediante la integración de distintas herramientas de código abierto.

Este dispositivo se implantará en el hogar y lo dotará de mayor seguridad permitiendo:

- Controlar el tráfico de red
- Registrar los clientes habituales
- Detectar los ataques recibidos
- Actuar frente a los ataques

1.2 Descripción del WIPS

Un *WIPS* (*Wireless Intrusion Prevention System*) o Sistema de Prevención de Intrusos para redes inalámbricas, es un dispositivo que se encarga de analizar el espectro para detectar falsos puntos de acceso y denegar el servicio a estos, también, se encarga de detectar posibles ataques cómo *Man In The Middle (MITM)* o Denegación de servicio (*DoS*) entre otros.

El *WIPS* típicamente está formado por tres elementos:

- Sensores: Son los que contienen las antenas para analizar el espectro
- Servidor: Analiza los paquetes capturados por los sensores
- Consola: Interfaz para la gestión del dispositivo

En cuanto a los ataques que puede detectar:

- Los ataques *Man In The Middle (MITM)* [1] son ataques en los que, en una comunicación extremo a extremo, se establece un punto intermedio (atacante) siendo transparente a ambos. Este ataque persigue la recolección del tráfico intercambiado entre extremos. Una forma común de realizar este ataque es envenenando la caché *ARP* (*Address Resolution Protocol*) [2] y hacer creer en los extremos que la dirección de su par es la del atacante.

La caché *ARP*, es una tabla que contienen todos los dispositivos y permite la comunicación, esta, se encarga de almacenar una relación entre la dirección *IP* y la dirección *MAC (Media Access Control)*

- Los ataques de denegación de servicio (*DoS*) [3], son los ataques que tienen como objetivo "*tumbar"* una red o servicio. Para realizar esto, el atacante sobrecarga la red provocando que esta sea inaccesible, una variante de estos ataques son los denominados *DDoS* o ataques de denegación de servicios distribuidos, en los que, bajo el mismo objetivo, la sobrecarga no es realiza por un único punto de conexión, sino que es realiza por varios puntos.

1.3 Estado del arte

Esta tecnología no es novedosa en el ámbito industrial, fabricantes cómo *Cisco* [4], por ejemplo, la comercializan. No obstante, hay pocas implementaciones *"Open Source"*. Tras realizar una búsqueda, sólo se encuentra una, *OpenWIPS-ng* [5], de los creadores de la suite *Aircrack-ng*.

El proyecto *OpenWIPS-ng* comenzó en 2012 con la idea de realizar un servidor que recolectase los datos llegados de distintos sensores y actuara en función a esos datos, el proyecto a día de hoy (04/Abril/2017) está estancado y sólo implementa soporte para un único sensor y dos tipos de ataques; desautenticación y fragmentación.

1.4 Alcance

El radio de actuación del *WIPS* se ajustará a un sensor y un servidor que actuará ante tres tipos distintos de ataques, un registro de clientes habituales y una interfaz web que nos facilitará la gestión.

Los ataques sobre los que actuará serán:

- DoS (*Denial Of Service*)
- MITM (*Man In The Middle*)
- Suplantación SSID

Estos, son tres de los diez ataques más comunes [6] que se realizan a las redes, de ahí la importancia de protegernos ante ellos.

Los ataques de suplantación de SSID tienen por objetivo simular un punto de acceso de confianza para el usuario, de esta forma al conectarse a él, obtendremos las credenciales del verdadero punto de acceso y además tendremos control absoluto sobre el tráfico de los usuarios.

1.5 Motivación

El *Instituto Nacional de Ciberseguridad (INCIBE)* recogió en un artículo [7] algunos de los riesgos que suponía en nuestra vida cotidiana el *IoT*, *Sophos* también informó en otro artículo [8] sobre los consejos de seguridad que debemos tomar cuando instalamos y usamos estos dispositivos.

Algunos de los aspectos más importantes que recogían eran:

- Recolección de datos personales
- Falta de estándares
- Compatibilidad con redes Wifi

- Aperturas de puertas traseras en los router

Por otra parte, la creciente instalación de dispositivos inteligentes [9] cómo "*Cerraduras automáticas*", "*Detector de humos inteligentes*" o "*Iluminación automática*" hacen aún más importante la necesidad de dotar a los hogares de seguridad.

Esto, ligado al aumento de los dispositivos *IoT* [10], motivó la necesidad de realizar un sistema que permitiese añadir una capa de seguridad al hogar.

2 ARQUITECTURA DE RED

"¿Dónde está la tecla 'ANY'?".

- Homer Simpson, frente a un mensaje "press any key" -

En este capítulo se describe el hardware necesario para el funcionamiento del *WIPS*, la interconexión de este y el esquema de red resultante.

2.1 Hardware

Para poder reproducir en cualquier lugar la implementación que se llevará a cabo en este proyecto se hará uso de:

- Router del proveedor de servicio
- Raspberry Pi 2 Modelo B

Figura 1 Raspberry Pi 2 Modelo B

- Cable Ethernet
- Antena WiFi[1]

Figura 2 Antena WiFi TP-Link

El motivo por el que se usan estos dispositivos es su relación rendimiento/precio, ya que, con la *Raspberry*, se puede obtener un gran rendimiento en pequeñas redes, cómo la doméstica, a un coste bajo.

[1] Para este proyecto se ha optado por la antena TP-LINK TL-WN722N

2.2 Interconexión

El *WIPS* se ejecutará en la *Raspberry*, por tanto, la conexión se realizará por cable Ethernet entre el router proveedor de servicio y la *Raspberry*, será necesario desactivar la capacidad de punto de acceso del router (*Ver manual del router*) y activar esta capacidad en la *Raspberry* (*Ver Anexos*).

Gracias a este cambio, la *Raspberry* será nuestro nuevo punto de acceso para el hogar y de esta forma el *WIPS* tiene el control absoluto tanto del tráfico de red cómo de los clientes que hacen uso de esta.

Por último, la antena WiFi, que será la que proporcione la señal a los dispositivos, se conectará a cualquiera de los puertos USB de la *Raspberry*. Esta antena tiene un doble objetivo, proporcionar señal a los dispositivos del hogar y analizar el espectro para detectar anomalías.

En el siguiente diagrama, se puede observar la interconexión explicada.

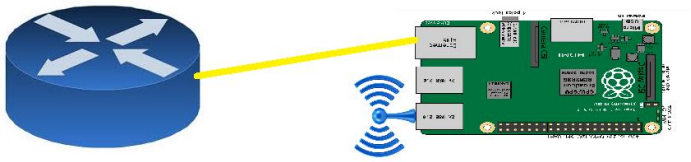


Figura 3 Diagrama de interconexión

2.3 Esquema de red

La interconexión que se realiza entre los dispositivos produce dos subredes privadas, una en la conexión Ethernet entre el router y la *Raspberry* y otra que crea la propia *Raspberry* al actuar como punto de acceso.

La primera subred será la 192.168.1.0/24, ya que, en nuestro caso, es la que asigna el router automáticamente, por otra parte, el rango que asignará la *Raspberry* será 10.10.10.0/24, esto provoca que la *Raspberry* tenga direcciones pertenecientes a distintas subredes en sus interfaces.

Para solucionarlo, la *Raspberry* realizará un *NAT (Network Address Translation)*. Con esta técnica lo que conseguimos es que la *Raspberry* cuando necesite encaminar paquetes desde la red WiFi al exterior, sustituya las direcciones de origen de los paquetes, 10.10.10.X, por la perteneciente a la primera subred 192.168.1.0/24, gracias a esto conseguimos aislar la red interna y así proporcionar también un mayor grado de seguridad, ya que el router solo conocerá de la *Raspberry* la dirección perteneciente a la subred 192.168.1.0/24.

En el siguiente diagrama se puede observar el esquema de red resultante.

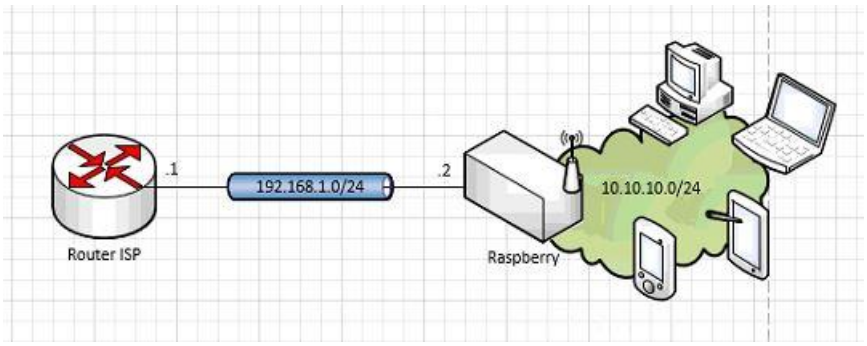


Figura 4 Diagrama de Red

3 ARQUITECTURA SOFTWARE

"La programación en bajo nivel es buena para el alma del programador".

-John Carmack-

En este capítulo profundizaremos en las herramientas usadas para el *WIPS*, la estructura del código implementado, y un análisis en profundidad de este.

3.1 Herramientas

En esta subsección trataremos las distintas herramientas que hemos utilizado para implementar el *WIPS*, haciendo especial hincapié en la integración de estas en el software creado.

3.1.1 Snort

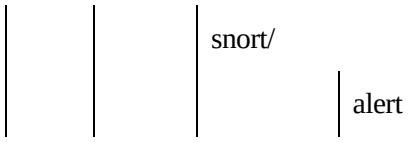
Es el *"alma"* del *WIPS, Snort* se define en su página oficial cómo "*Un sistema de prevención de intrusos de código abierto capaz de analizar el tráfico y registrar paquetes en tiempo real*" [11].

Esta herramienta, como bien indica su definición es un *IPS (Intrusion Prevention System)*, será el encargado de monitorizar todo el tráfico y en función de las reglas de actuación que definamos, dejará pasar, cortará o notificará de aquel tráfico que coincida con las reglas. Aprovechando esto, *Snort* nos será útil para detectar los ataques *DoS* y *DDoS* ya que, gracias a su potente comunidad, existen un conjunto de reglas predefinidas para detectar posibles amenazas de *DoS* y *DDoS.* Además, nos servirá para bloquear todo el tráfico a nivel tres[2] que sea necesario, para esto, será necesario que *Snort* coopere con *Iptables,* herramienta que veremos un poco más adelante.

Una vez que hemos instalado *Snort* (*Ver Anexos*), la estructura de directorios resultante será:

```
/etc/
    snort/
        snort.conf
        rules/
            local.rules
/var/
    log/
```

[2] Respecto al modelo OSI [31]



Bajo esta distribución de directorios nos encontramos el fichero de configuración de *Snort* (*snort.conf*) y una carpeta que, por defecto, contiene el fichero *local.rules*, este fichero contendrá las reglas que creará el *WIPS*, pero también, bajo esa carpeta, y configurándolo adecuadamente *(Ver Anexos)* podremos introducir el fichero con las reglas comunitarias para los ataques *DoS y DDoS*.

Las reglas deben seguir una estructura fija definida por *Snort*

---

acción protocolo ip_origen puerto_origen <sentido> ip_destino puerto_destino (msg:" Mensaje de información extra"; sid:sid_id; rev: rev_id)

- Acción: Acción que se tomará cuándo coincida una regla. Existen varias acciones, aunque sólo nos interesan las siguiente:
  - Alert: Notifica que el tráfico coincide con alguna de las reglas
  - Drop: Rechaza el tráfico si coincide con alguna de las reglas

- Protocolo: Protocolo ante el que actuará la regla.
  - IP, TCP, UDP e ICMP

- IP origen/destino: Define las direcciones *IP* origen y destino que deben tener los paquetes.
  - Subred
  - Dirección
  - "any"
  - $HOME_NET

- Puerto origen/destino: Define los valores de los puertos que deben tener los paquetes
  - Puerto
  - "any"

- Sentido: Define el sentido de la comunicación necesario para activar la regla
  - " -> ": Origen – destino
  - " <- ": Destino – origen
  - " <> ": Ambos sentidos de la comunicación

- Sid: Número único que identifica a cada regla

- Rev: Número de revisión de la regla por si se ha modificado en alguna ocasión

Otro fichero de especial interés para nuestro software será */var/log/snort/alert*, en este fichero, *Snort* irá registrando todo el tráfico que coincida con alguna de las reglas definidas. La estructura que siguen se puede observar en el siguiente ejemplo.

[**] [1:1000001:1] ICMP Testing Rule [**]

[Priority: 0]

01/01-00:00:00.000000 10.10.10.3 -> 216.58.214.174

ICMP TTL:64 TOS:0x0 ID:22388 IpLen:20 DgmLen:84 DF

Type:8 Code:0 ID:13616 Seq:1 ECHO

En primer lugar, aparece el mensaje de información que definimos para esa regla, tras esto, la prioridad que, en nuestro caso, todas las reglas tendrán la misma. Seguidamente, las direcciones *IP* y puerto, si existiese, junto con el sentido de la comunicación. Por último, aparece el protocolo y los datos relativos a este.

Aprovechando esto, y las reglas comunitarias, podremos detectar los ataques *DoS* y *DDoS*, y bloquear al atacante, completando así una de las funcionalidades del *WIPS*.

Por tanto, esta herramienta, nos será de especial interés en nuestro proyecto para detectar ataques *DoS*, *DDoS* y actuar ante ellos bloqueando al atacante, también no servirá para actuar siempre que sea necesario en el tráfico a nivel tres.

3.1.2 Aircarck-ng

Aircrack-ng [12] es una *suite* de herramientas para evaluar la seguridad de las redes WiFi, en nuestra implementación, la usaremos para analizar el espectro y detectar posibles suplantaciones de *SSID*.

De entre todas sus herramientas, haremos uso de dos de ellas.

- ***Airmon-ng***
- ***Airodump-ng***

Airmon-ng [13] se usa para habilitar o deshabilitar el modo *monitor* en la interfaz WiFi.

Habilitar el modo *monitor* en una interfaz es similar a habilitar el modo *promiscuo*, es decir, típicamente, una interfaz de red cuando recibe un paquete mira su dirección *IP* destino, si no coincide con la dirección *IP* de la interfaz, rechaza ese paquete. Si habilitamos el modo *monitor* o *promiscuo*, la interfaz no rechazará ningún paquete y escuchará todo lo que esté a su alcance.

Airodump-ng [14], se encarga de capturar los paquetes de nivel dos del protocolo 802.11. Una vez que ejecutemos esta herramienta con el comando adecuado (*Ver Anexos*), se creará un fichero de extensión *CSV* que sigue la estructura del siguiente ejemplo.

BSSID, First time seen, Last time seen, channel, Speed, Privacy, Cipher, Authentication, Power, # beacons, # IV , LAN IP, ID-length, ESSID, Key

00:1C:10:26:22:41, 2007-10-07 12:48:58, 2007-10-07 12:49:44, 6, 48, WEP , WEP, , 171, 301, 0, 0. 0. 0. 0, 5, zwang,

00:1A:70:51:B5:71, 2007-10-07 12:48:58, 2007-10-07 12:49:44, 6, 48, WEP , WEP, , 175, 257, 1, 0. 0. 0. 0, 9, brucey123,

00:09:5B:7C:AA:CA, 2007-10-07 12:48:58, 2007-10-07 12:49:44, 11, 54, OPN , , , 189, 212, 0, 0. 0. 0. 0, 7, NETGEAR,

Station MAC, First time seen, Last time seen, Power, # packets, BSSID, Probed ESSIDs

00:1B:77:7F:67:94, 2007-10-07 12:49:43, 2007-10-07 12:49:43, 178, 3, (not associated) ,

Podemos observar como en la primera línea nos aparecen los datos de los puntos de acceso que mostraremos en cada columna, y tras esto, todos los puntos de acceso activos. En el segundo conjunto de datos, nos aparecen todos los clientes detectados y a qué punto de acceso están conectados.

El *WIPS* leerá este fichero cuando la captura termine, y centrándose en la sección de los puntos de acceso, analizará el fichero en busca de dos situaciones:

- ***BSSID* duplicado**
- ***ESSID* duplicado**

El *BSSID (Basic Service Set Identifier)* es la dirección *MAC* del punto de acceso, mientras que el *ESSID (Extended Service Set Identifier)* es el nombre de la red que crea el punto de acceso.

Para el caso de *ESSID* duplicado, obtendrá el *BSSID* correspondiente y lo bloqueará mediante *Iptables*. En cambio, si encontramos un *BSSID* duplicado, el *WIPS*, notificará al usuario ya que no puede actuar.

El motivo de la no actuación radica en que al estar analizando una comunicación en capa dos, la única información que tenemos del punto de acceso es su dirección *MAC*, y si bloqueamos esta, nos quedaremos sin comunicación con el punto de acceso original. Por tanto, la única acción posible es notificar.

Con estas herramientas, queda cubierta otra de las necesidades del *WIPS* cómo la suplantación de *SSID*.

3.1.3 Arpwatch

Aprwatch [15] es una herramienta que se encarga de monitorizar el tráfico *ARP* de la red en busca de actividad sospechosa.

Una vez instalado y configurado correctamente (*Ver Anexos*), la herramienta registrará en el fichero */var/log/arpwatch.log* toda la actividad que detecte en la red. A continuación, se muestra un ejemplo de ese registro.

Feb 21 10:31:13 raspberryFran arpwatch: changed ethernet address 10.10.10.3 50:5c:1a:2d:27:9a (00:0c:12:9a:cd:00) eth0

Lo interesante de este registro es que nos indica qué ha sucedido en la red, la dirección *IP* que ha provocado ese registro, la dirección *MAC* y la interfaz en la que se ha producido.

La actividad que *Arpwatch* [16] puede detectar en la red es la siguiente:

- ***new station***: Se ha registrado un nuevo equipo en la red
- ***changed ethernet address***: Un equipo ya registrado ha cambiado su dirección *IP*
- ***flip flop***: Un equipo que ya había cambiado su dirección *IP* ha vuelto a cambiarla a la más antigua.

A partir del fichero de *logs*, dónde registramos estas actividades, el *WIPS* será capaz de detectar un posible ataque *MITM* y bloquear al atacante a través de *Snort*.

Gracias a otra característica de *Arpwatch* como es la de almacenar todos los clientes que se han asociado a la red, podremos almacenar estos en una base de datos. Estos ficheros se encuentran bajo el directorio */var/lib/arpwatch/* y contiene tantos ficheros, cuya extensión es *dat*, cómo interfaz esté monitorizando.

Un ejemplo del contenido:

50:5c:1a:2d:27:9	*10.10.10.1*	*1488909824*	*eth0*
00:0c:12:9a:cd:00	*10.10.10.2*	*1488909824*	*eth0*

El fichero contiene los datos divididos por columnas en el siguiente orden; *MAC*, dirección *IP*, instante de tiempo del registro e interfaz detectada, aunque para nuestra base de datos, sólo nos centraremos en las tres primeras columnas.

Por tanto, con esta herramienta, completamos dos funcionalidades más del *WIPS*, cómo ataques *MITM* y registro de clientes habituales.

3.1.4 Iptables

Iptables pertenece al proyecto *Netfilter* [17] y es una herramienta, integrada por defecto en el *Kernel* de *Linux*, utilizada para el filtrado de paquetes a través de reglas.

El instante de actuación de la regla, depende de la tabla y la cadena en la que la introduzcamos, existen tres tablas [18], aunque las que nos interesan son:

- ***filter**:* Encargada del filtrado de paquetes, cuyas cadenas son:
  - ***INPUT***: Cadena de los paquetes destinados al sistema
  - ***OUTPUT***: Cadena de los paquetes que atraviesan el sistema
  - ***FORWARD***: Cadena de los paquetes que van a ser reenviados
- ***nat**:* Responsable de las reglas de reescritura de direcciones o puertos, con las cadenas:
  - ***PREROUTING***: Cadena para los paquetes que no se han encaminado
  - ***POSTROUTING***: Cadena para los paquetes sobre los que se ha tomado la decisión de encaminamiento
  - ***OUTPUT***: Cadena para realizar un *DNAT (Destination Network Address Translation)* en los paquetes generados localmente
  - ***INPUT***: Cadena para hacer un *DNAT* en los paquetes destinados al sistema

En la siguiente imagen extraída de *Wikipedia* [19], se muestran todas las tablas, cadenas y dónde actúan cada una de ellas

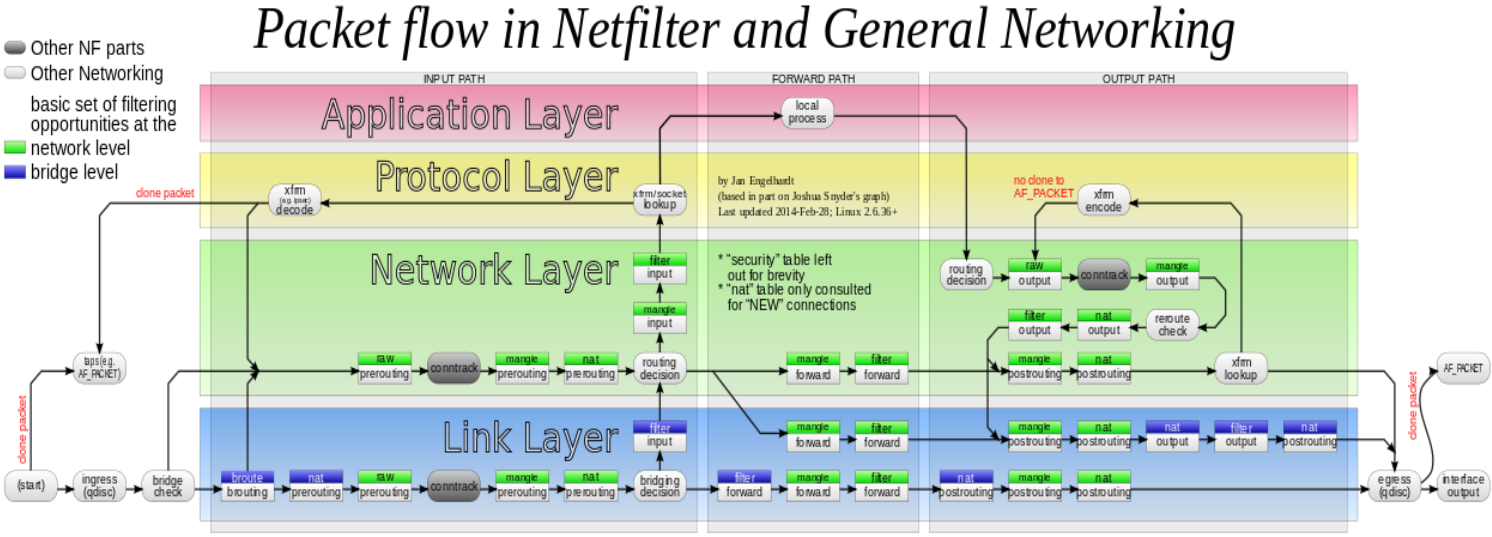


Figura 5 Tablas Iptables

El *WIPS*, hará uso de esta herramienta en tres ocasiones:

- ***NAT***
- **Integración con *Snort***
- **Suplantación de *SSID***

Para el caso del *NAT*, lo que se realizará es un *SNAT (Source Network Address Translation)*, es decir, cambiamos la dirección origen de los paquetes, eso lo haremos en la tabla NAT y en la cadena *POSTROUTING*. A continuación, se muestra el comando necesario para realizar esta acción.

```
iptables -t nat -A POSTROUTING -s (IP_addr_masq) -j SNAT --to (IP_addr_gw)
```

La integración con *Snort* es esencial ya que permite bloquear el tráfico cuándo sea necesario, sin esto, lo que tendríamos es un *IDS (Intrusion Detection System)*. Para convertirlo en *IPS*, además de la correcta configuración de *Snort*, tendremos que crear la siguiente regla en *Iptables*.

```
iptables -t nat -A PREROUTING -j NFQUEUE --queue-num 0 --queue-bypass
```

Con esta regla, conseguimos que mediante *NFQUEUE* cooperen ambos. La opción *queue-bypass* indica que cuando *Snort* esté desactivado esta regla no actúe.

Por último, para actuar ante la suplantación de *SSID*, el *WIPS*, creará una regla en la tabla *filter* para cortar el tráfico dirigido al *BSSID* correspondiente.

3.1.5 Apache/Tomcat

Apache [20] es un servidor web de código abierto, es el más usado [21] y permite ejecutar una gran cantidad de lenguajes. *Tomcat* [22] es un servidor web con soporte a aplicaciones *Java*, algo que *Apache* no soporta, de ahí la necesidad de integrar ambos.

Con la integración, *Apache* recibirá todas las peticiones por parte de los clientes y reenviará a *Tomcat* todas las peticiones que contengan código *Java*, es decir, cuya extensión sea *JSP*, de esta forma podemos crear interfaces web con código estático y dinámico.

En el *WIPS* implantaremos este tipo de interfaces, ya que, a través de la interfaz web podremos ver y eliminar las reglas que hayamos definido en */etc/snort/local.rules*, visualizar las reglas creadas en *Iptables*, pertenecientes a la tabla *filter*, y realizar consultas a la base de datos de clientes habituales entre otras cosas.

Para el código estático, utilizaremos *HTML5*, mientras que para el dinámico utilizaremos dos lenguajes distintos, *C* para el *CGI (Common Gateway Interface)*, de esta forma reutilizamos gran parte del código ya implementado en el *WIPS*, y *Java* para la parte de *JSP (Java Server Pages)*, así podemos realizar consultas a una base de datos o mostrar el contenido de algunos ficheros.

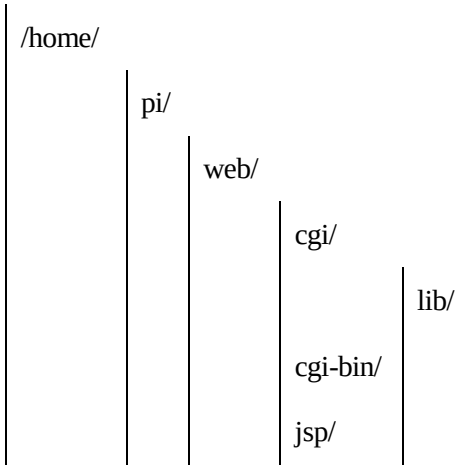
El *CGI* [23] es la parte del servidor que permite comunicarse con otros programas que se ejecutan en el servidor, de esta forma se puede llamar a un programa con los datos específicos del usuario para que este procese los datos, y devolver la respuesta de vuelta al navegador del usuario.

Por su parte, *JSP* [24], es una tecnología que permite incorporar contenido dinámico basándose en código *Java*. Apoyándonos en *JSP* y *Tomcat*, podemos integrar también las consultas a bases de datos, esto lo haremos mediante *JDBC (Java Database Connectivity)*.

JDBC [25] *es una API (Application Programming Interface)* de *Java* que permite ejecutar sentencias *SQL*

(Structure Query Language), y consta de un conjunto de clases e interfaces escritas en *Java*.

Una vez instalado y configurado correctamente (*Ver Anexos*), la estructura de directorios resultantes es la siguiente:



La interfaz web del *WIPS*, se colocará bajo esta estructura de directorios, estando todo el código perteneciente a la parte dinámica en el directorio *cgi-bin/* para el escrito en *C* y *jsp/* para el código *Java*, y la parte estática bajo el directorio *cgi/*.

Por tanto, con estos servidores web tendremos completa la interfaz, y la herramienta de gestión del *WIPS*.

3.1.6 PostgreSQL

PostgreSQL [26] es un sistema gestor de bases de datos objeto-relacional de código abierto que permite almacenar tablas, y devolver los resultados de las consultas realizadas por los usuarios.

Una vez instalada, configurada y con las tablas creadas (*Ver Anexos*), el *WIPS*, hará uso de la base de datos para almacenar los clientes que *Arpwatch* detectó en el enlace.

Los clientes se almacenarán en una tabla creada previamente. Esta tabla, contiene tres columnas:

- Dirección *MAC* (***dir_mac***): Será del tipo *macaddr* y no podrá ser un valor nulo
- Dirección *IP* (***dir_ip***): No podrá ser nulo y será del tipo *inet*
- Fecha registro (***fecha***): Del tipo "*timestamp without time zone*" y no podrá ser nulo.

Para la tercera columna, la fecha, será necesario realizar una conversión antes de introducirla en la base de datos, ya que, *Arpwatch* almacena el instante de tiempo como una marca de tiempo *Unix*, luego tenemos que convertirlo en una fecha legible.

La clave primaria de esta tabla, es decir, los identificadores únicos de cada tupla (fila), será la combinación de la dirección *MAC* y la fecha de esta forma nos garantizamos que estamos introduciendo siempre un registro nuevo.

Con esto, ya tenemos las herramientas necesarias para implementar todas las funcionalidades que definimos en el alcance del proyecto para el *WIPS*.

3.2 Funcionamiento del *WIPS*

Para la realización del proyecto se ha optado por dividir el alcance en pequeñas partes, en concreto, una por

funcionalidad, quedando así el proyecto dividido en cuatro módulos y una interfaz web de gestión.

En el siguiente diagrama *WBS* (*Work Breakdown Structure*) [27] se muestra la subdivisión del proyecto.

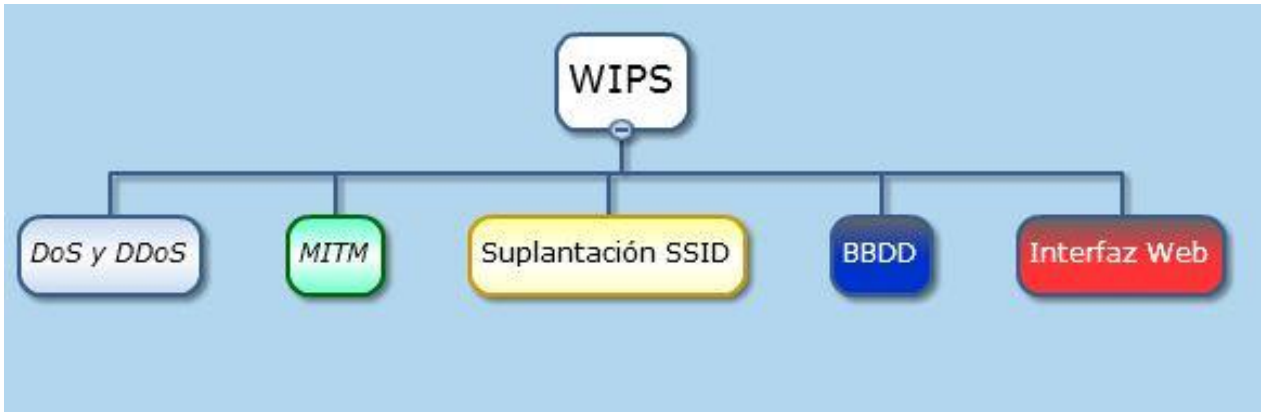


Figura 6 Diagrama WBS del proyecto

El primer módulo, encargado de los ataques *DoS* y *DDoS*, utiliza las herramientas explicadas anteriormente para detectar las alertas generadas ante posibles ataques y bloquear la dirección *IP* del atacante para detener el ataque.

Tras este, el módulo encargado de los ataques *MITM*, detecta las posibles amenazas en la red local y bloquea al atacante a nivel tres, una característica de este módulo es que el bloqueo es temporal, esto se debe a que la dirección *IP* bloqueada es local y si el bloqueo es permanente, se podría llegar a una situación de inoperatividad en la red. Para evitar esto, se bloquea la dirección *IP* 60 minutos, un tiempo adecuado para detener el ataque y no dejar inoperativa la red.

El tercero detecta la suplantación de puntos de acceso mediante el análisis del espectro, cuando detecta un punto de acceso con el mismo *ESSID*, obtiene su dirección *MAC* y bloquea el tráfico hacia ese punto de acceso. Si lo que detecta es un *BSSID* duplicado, informa mediante la interfaz web de esto, por los motivos que se explicaron en el punto anterior.

El último módulo de funcionalidades es el de la base de datos, este, vuelca el contenido de todos los ficheros de los clientes registrados en la red local y los registra en la base de datos, la cual es accesible a través de la interfaz web.

Todos estos módulos se integrarán median hilos, de esta forma, cada módulo se ejecuta de manera independiente e ininterrumpidamente.

Por último, la interfaz de gestión, que integrando varias tecnologías permite listar y eliminar las reglas creadas en *Snort*, mostrar las reglas de *Iptables*, realizar consultas a la base de datos de los clientes, ver los registros activos de los ataques *MITM* y ver los *logs* que ha generado el propio *WIPS*.

3.3 Estructura del código

En esta sección nos centraremos en la estructura del código que implementa el *WIPS*.

Para la implementación del *WIPS*, cómo hemos comentado anteriormente, hemos dividido el proyecto en varios módulos, en concreto, cuatro módulos y una interfaz web para la gestión. En el siguiente diagrama, observamos esa división.

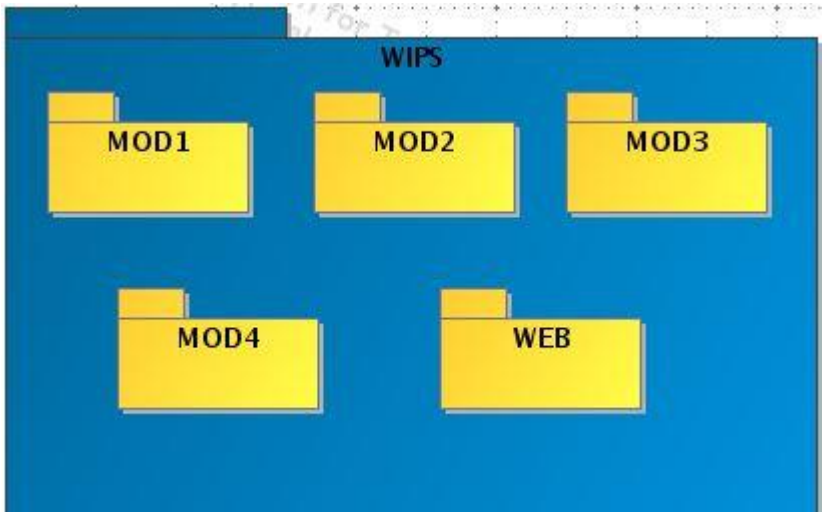


Figura 7 Diagrama de paquetes del proyecto

El diagrama, es una abstracción a alto nivel de la estructura del proyecto, si profundizamos en cada módulo, nos encontraremos con la siguiente estructura general.

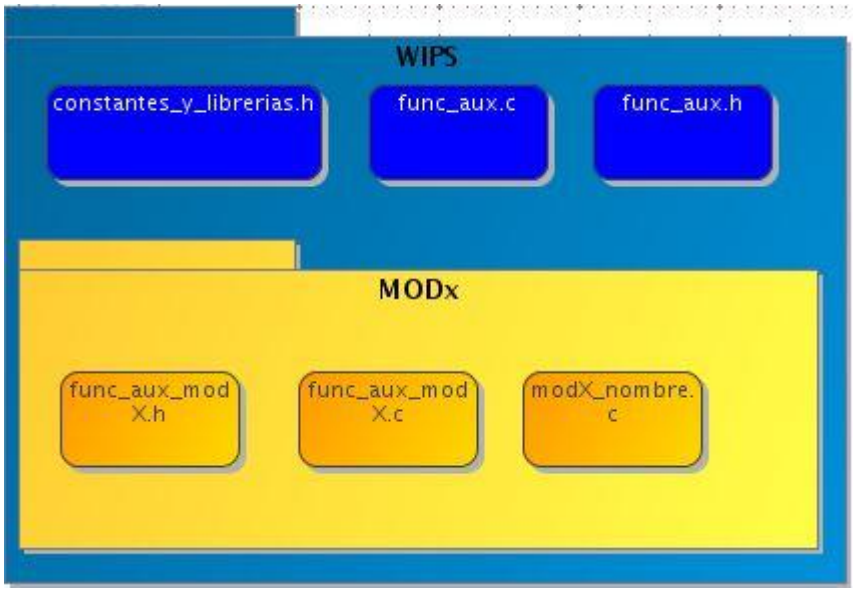


Figura 8 Diagrama general de ficheros

En el diagrama anterior podemos observar que todos los módulos contendrán tres ficheros comunes y tres específicos de ese módulo.

De los ficheros comunes, *constantes_y_librerias.h* contendrá todas las constantes usadas en las diferentes funciones, además contiene las librerías comunes a todos los módulos, por otro lado, *func_aux.h* almacena las definiciones de las funciones implementadas en *func_aux.c* que son las funciones comunes a todos los módulos y las que se usarán para realizar la integración con la interfaz web.

En cuanto a los ficheros comunes a los módulos, nos encontramos que *modX_nombre.c* será el fichero principal de ese módulo y contendrá una función llamada *modX* que será la función a la que se llamará desde el fichero principal del *WIPS* (*main.c*) para que se ejecute cada módulo como un proceso independiente.

Los ficheros restantes, *func_aux_modX.h* y *finc_aux_modX.c*, contienen las definiciones e implementaciones, respectivamente, de las funciones específicas a ese módulo, que no serán usadas en ningún otro.

En general, la distribución de los ficheros se esquematiza en la siguiente tabla.

Ficheros Comunes			
	constantes_y_librerias.h	func_aux.h	func_aux.c
Ficheros Específicos			
	func_aux_mod1.h	func_aux_mod1.c	mod1_DoS.c
	func_aux_mod2.h	func_aux_mod2.c	mod2_MITM.c
	func_aux_mod3.h	func_aux_mod3.c	mod3_SSID.c
	func_aux_mod4.h	func_aux_mod4.c	mod4_database.c

Tabla 1 Esquema de ficheros

Por último, la interfaz web, no sigue una estructura específica. De esta forma, tendremos definida toda la estructura del código, en la siguiente sección, profundizaremos sobre el código que contiene cada fichero.

3.4 Módulos

En esta subsección, profundizaremos en los distintos módulos que componen el *WIPS*, analizando el código por el que está compuesto cada uno de ellos.

3.4.1 Integración

Todos los módulos descritos a continuación, se unifican en un fichero principal.

Figura 9 Diagrama flujo integración

Como se observa en el diagrama de flujo anterior, en primer lugar, se crean los semáforos que se usarán para evitar que dos módulos accedan a la vez a un fichero ya sea en modo lectura o escritura, de esta forma garantizamos la exclusividad de cambios en los ficheros.

Tras esto, se crean los hilos, se creará un hilo por módulo y de forma independiente, de esta forma, los cuatro módulos estarán ejecutándose en paralelo.

Por último, queda en un bucle para permitir que los procesos queden activos y el *WIPS* permanezca en funcionamiento.

Este módulo, contiene las estructuras y funciones auxiliares a todos los módulos en su propia librería. Las estructuras que se definen son las siguientes;

```
//Estructura para almacenar el contenido leído de un fichero
typedef struct CONTENIDO_FICHERO{

    //Variable auxiliar para almacenar cuantas palabras tiene cada frase
    int palabras_por_frase[NUM_FRASES];

    //Variable para contabilizar el número de frases del fichero
    int num_frases_fichero;

    //Array para almacenar el contenido del documento
        //NUM_FRASES líneas y NUM_PALABRAS_POR_FRASE palabras por líneas
    char *contenido_leido_del_fichero[NUM_FRASES][NUM_PALABRAS_POR_FRASE];

} contenido_fichero;
```

Figura 10 Estructura contenido_fichero

Es la estructura que se devuelve cuando se lee un fichero, en ella, tenemos un contador para las líneas existentes en el fichero, un array para almacenar cuántas palabras hay en cada línea o frase y por último un array doble que almacena cada palabra de cada frase, de esta forma, cuando se lee un fichero, todo el contenido queda almacenado en esta estructura y en concreto en el array doble.

La siguiente, es la estructura utilizada para almacenar los datos de las reglas de *Snort*.

```
//Estructura para almacenar la información relativa a una regla
typedef struct ESTRUCTURA_REGLA {

    //Array para almacenar la dirección IP
    char *dir_IP[NUM_REGLAS];
    /*-------------------------------------
        Bandera para conocer si se la
        dirección IP lleva asociada un puerto
        o no
            --true: La dir_IP lleva un puerto
            -- false: No lo lleva
    -------------------------------------*/
    bool dir_Con_Puerto[NUM_REGLAS];
    //Array para almacenar el número de puerto
    char *puerto[NUM_REGLAS];
    //Array de string para almacenar el protocolo
    char *protocolo[NUM_REGLAS];
    /*-------------------------------------
        Bandera para conocer si la dirección
        IP se encontraba en el campo de origen
        o en el de destino
            -- true: se encontraba en origen
            -- false: se encontraba en destino
    -------------------------------------*/
    bool dir_en_origen[NUM_REGLAS];
    //Array para almacenar la acción que queremos ejecutar
    char *accion[NUM_REGLAS];
    //Variable para almacenar el número total de coincidencias
    int numero_lineas;

}estructura_regla;
```

Figura 11 Estructura estructura_regla

Esta estructura, está compuesta por un campo para almacenar la dirección IP de la regla, una bandera para indicar si esa dirección IP lleva asociada un puerto, el campo del puerto, un array para almacenar el protocolo de la regla, una bandera para indicar si la posición de la dirección IP y el puerto era en origen o en destino, esto es útil a la hora de crear la regla, para conocer en qué posición debemos colocar esa información y por último un array para almacenar la acción de la regla y un contador de las reglas existentes en la estructura.

A continuación, se definen las principales funciones comunes que se han creado para todos los módulos.

La primera, lee_fichero(), función que recibe cómo parámetro la ruta de un fichero, lo lee y como resultado, devuelve una estructura con el contenido del fichero.

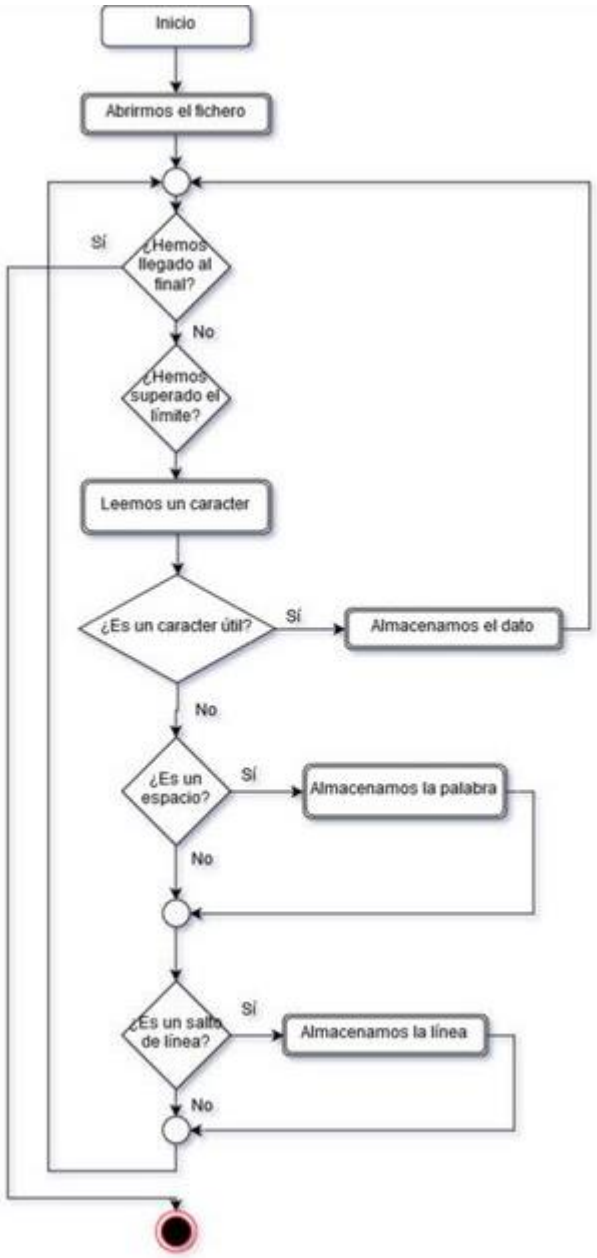


Figura 12 Diagrama flujo lee_fichero()

Esta función, tras abrir el fichero que se desea leer, lo recorre caracter a caracter, si el caracter es útil, es decir, no es un espacio, una coma, un punto o un salto de línea, lo almacenamos en una variable auxiliar, ya que se está leyendo una palabra. Si se lee un espacio, una coma o un punto, almacenamos el valor de la variable auxiliar en el campo correspondiente a la estructura, ya que se habrá finalizado la lectura de una palabra.

En caso que se lea un salto de línea, se almacena el valor de la variable dentro de la estructura ya que se habrá finalizado la lectura de una palabra, pero, además, se actualizan los contadores de la estructura, ya que se ha finalizado la lectura de una línea y, por tanto, a continuación, le sigue una nueva que habrá que almacenar en una posición distinta en la estructura.

La siguiente función es crea_y_escribe_regla(), la cuál recibe cómo parámetro el fichero dónde se desean escribir las reglas a crear, la estructura con la información de las reglas y una cadena de texto por si se quiere añadir información extra, cómo un comentario o algo similar.

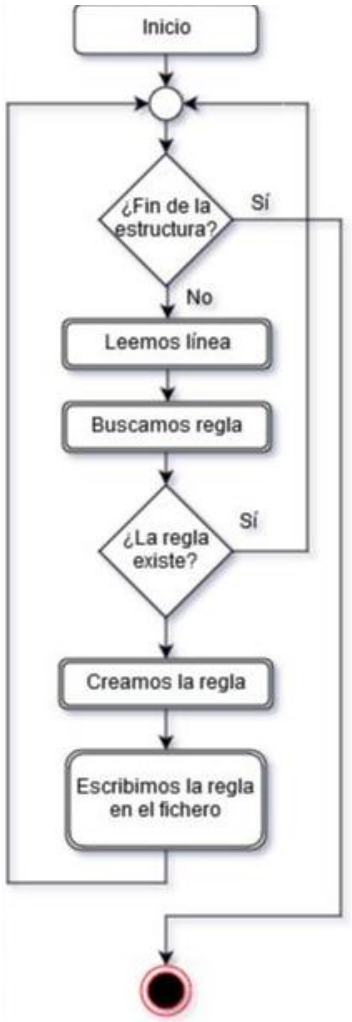


Figura 13 Diagrama flujo crea_y_escribe_regla()

Comienza recorriendo la estructura de reglas, en primer lugar, comprobamos que la regla que queremos crear en el fichero de reglas de *Snort (/etc/snort/rules/local.rules)*, no existía, ya que, en caso de existir, la regla no se creará para evitar así tener reglas duplicadas. Tras esta comprobación, si la regla que se desea introducir es nueva, se forma a través de los datos almacenados en la estructura y se escribe en el fichero de reglas. Esto, se realizará mientras queden en la estructura reglas por crear.

recarga_Snort(), es una función que no recibe ningún parámetro y se encarga de recargar el fichero de configuración de *Snort*.

Figura 14 Diagrama flujo recarga_Snort()

Esta función, almacena el PID del proceso en ese instante de *Snort*, una vez que obtiene el identificador, reinicia el proceso a través del comando "*sudo kill -SIGHUP PID*", comando extraído de la documentación de *Snort* por el cuál *Snort* vuelve a cargar el fichero de configuración y por tanto el fichero de reglas.

3.4.2 Módulo 1: *DoS*

El primer módulo, es el encargado de los ataques *DoS*, en la siguiente imagen, podemos observar mediante un diagrama de flujo el funcionamiento de este.

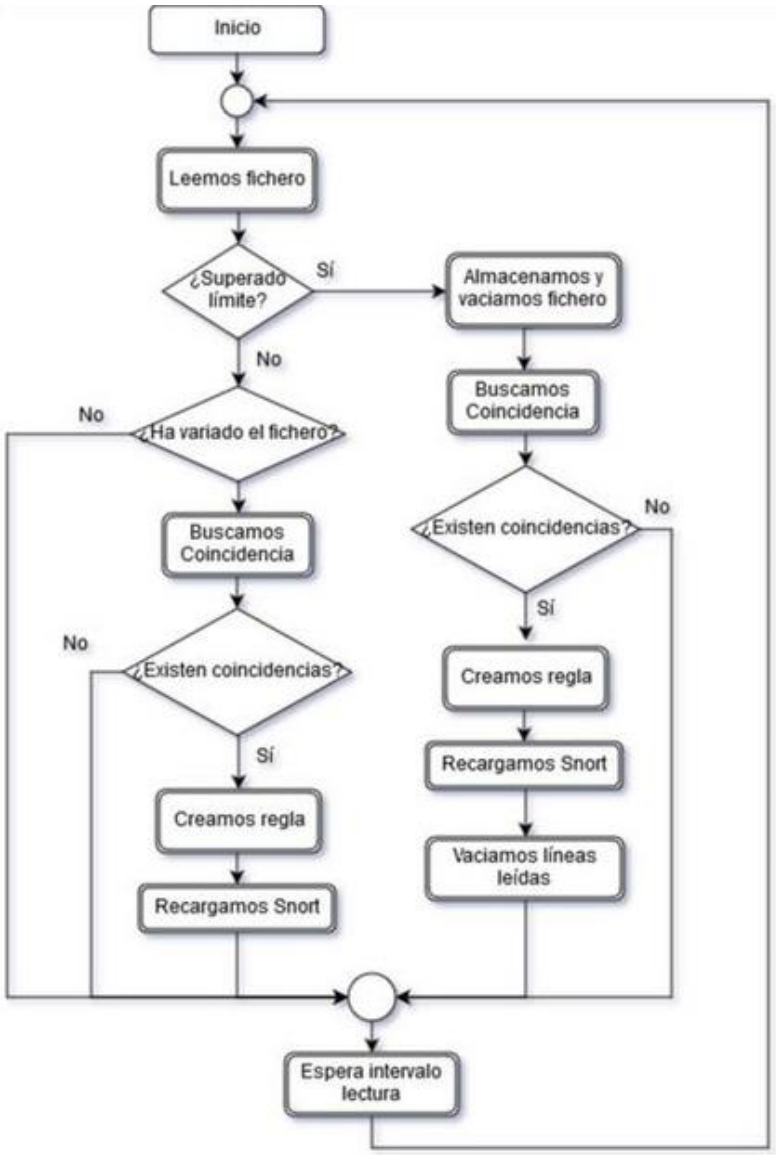


Figura 15 Diagrama flujo Módulo1: DoS

Como se observa en el diagrama superior, mediante un bucle, vamos leyendo el fichero donde registramos las alertas que ha generado *Snort* (*/var/log/snort/alert*), tras leer línea por línea este fichero, comprobamos en primer lugar que no se ha superado el tamaño máximo de líneas que admite la estructura donde almacenamos los datos. Para esta condición, se pueden dar dos casuísticas, que la estructura supere o no el límite.

En el primer caso, lo que se realizará será lo siguiente; Almacenar el fichero que está superando el límite, vaciar el fichero de alertas (*/var/log/snort/alert*), buscar si el fichero antiguo contenía alguna alerta nueva, de esta forma, nos aseguramos que se detectan todas las alertas y en el caso que se detecte alguna alerta, se crean las reglas necesarias, se recarga *Snort* para que las reglas se apliquen y se reinician las variables ya que el fichero vuelve a estar vacío.

Si el fichero no ha superado aún el límite, realizamos las mismas acciones que en el caso anterior, se buscan las alertas nuevas, se crean las reglas para esas alertas y se recarga *Snort* para que estas se apliquen.

Tras esto, antes de comenzar una nueva iteración del bucle, esperamos un intervalo de lectura general definido en la librería principal.

Centrándonos en la función que se encarga de detectar si existen nuevas alertas.

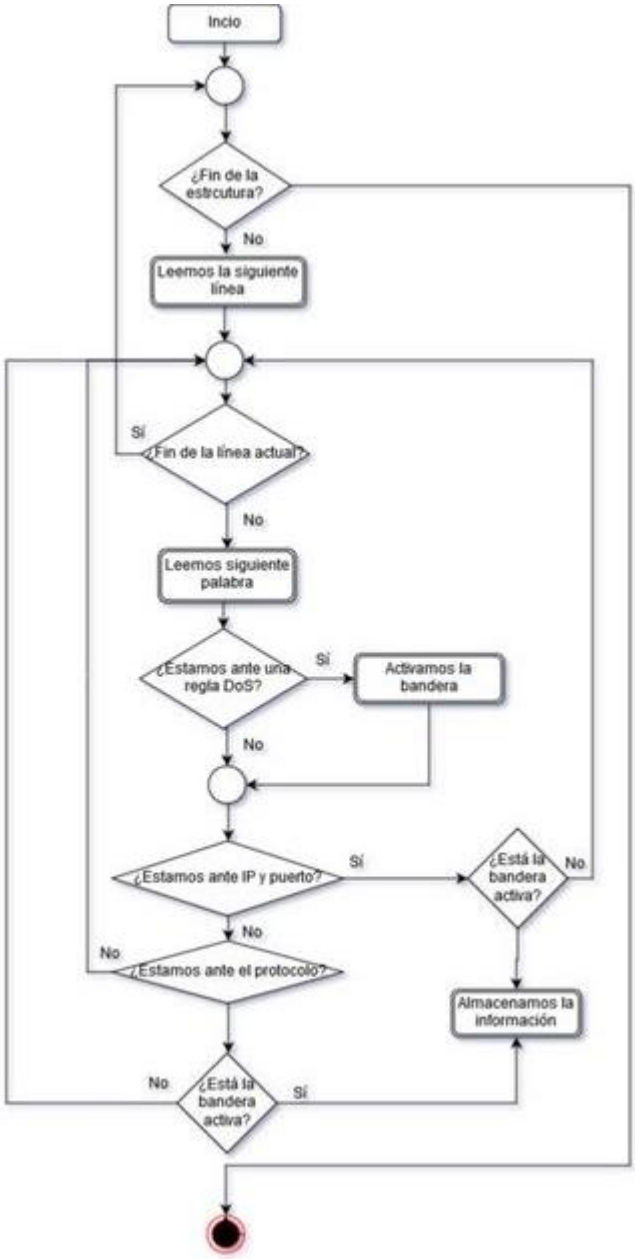


Figura 16 Diagrama flujo función comprueba_Coincidencia()

Es una función que recibe cómo parámetro una estructura, una palabra clave a buscar, en este caso "*DoS*", y la posición de comienzo en la estructura.

Mientras no se haya llegado al final de la estructura, la recorremos en busca de la palabra clave, en el momento que esta se detecte, activamos una bandera y pasamos a buscar la posición relativa a la dirección IP, puerto y protocolo, una vez que estamos en dicha posición, almacenamos los datos en una nueva estructura que será la devuelta para crear las reglas en *Snort.*

El tercer parámetro es necesario para agilizar la búsqueda, ya que la comienza dónde termino la última, de esta forma evitamos recorrer la estructura de forma completa innecesariamente.

3.4.3 Módulo 2: MITM

El segundo módulo, encargado de los ataques *MITM*, se comporta de la siguiente forma.

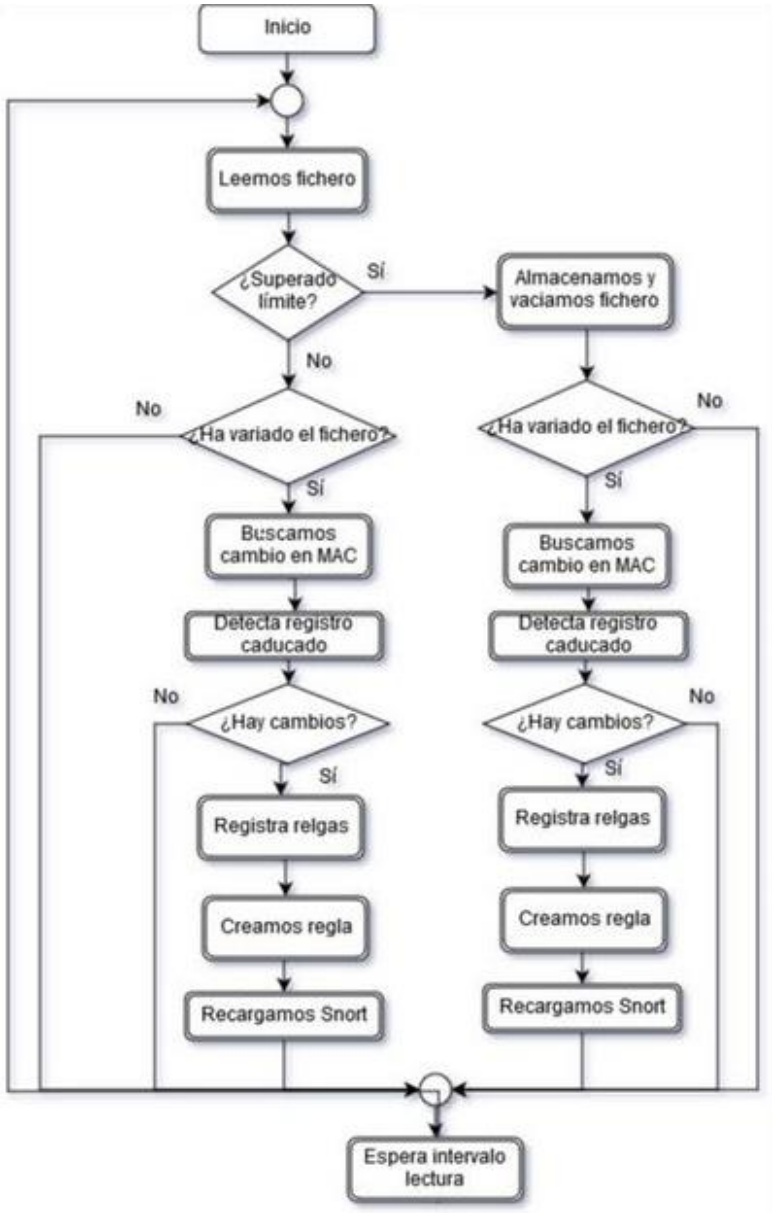


Figura 17 Diagrama flujo Módulo 2: MITM

Al igual que en el primer módulo, mediante un bucle leemos el fichero donde *Arpwatch* registra las alertas producidas (*/var/lib/arpwatch/*.dat*) y la primera comprobación que se realiza, es si dicho fichero ha superado el límite de la estructura tomando las mismas acciones en caso positivo.

Una vez realizada esta comprobación, comprobamos si algún registro ha caducado y buscamos en el fichero si se ha producido algún cambio sospechoso en la dirección *MAC*.

Si se ha producido algún cambio, creamos la regla en *Snort* para bloquear la dirección *IP* sobre la que se ha producido el cambio de dirección *MAC* ya que podríamos estar ante un ataque *MITM* y recargamos *Snort* para que los cambios se apliquen. A la vez que se crea la regla, esta se registra en */tmp/WIPS/registro_reglas_MITM* añadiendo la hora a la que se creó.

En relación a la comprobación de registros caducados citado anteriormente, dado que estas reglas son temporales, es necesario comprobar si ha pasado el intervalo de tiempo definido. De ahí la necesidad de registrar el instante en el que se crea una regla. Tras esto, el último paso antes de la nueva iteración del bucle es la espera del intervalo de lectura.

Entre las funciones auxiliares que utiliza este módulo destacan.

buscaCambioEnMAC(), función encargada de procesar el fichero generado por *Arpwatch* y detectar si existen posibles ataques.

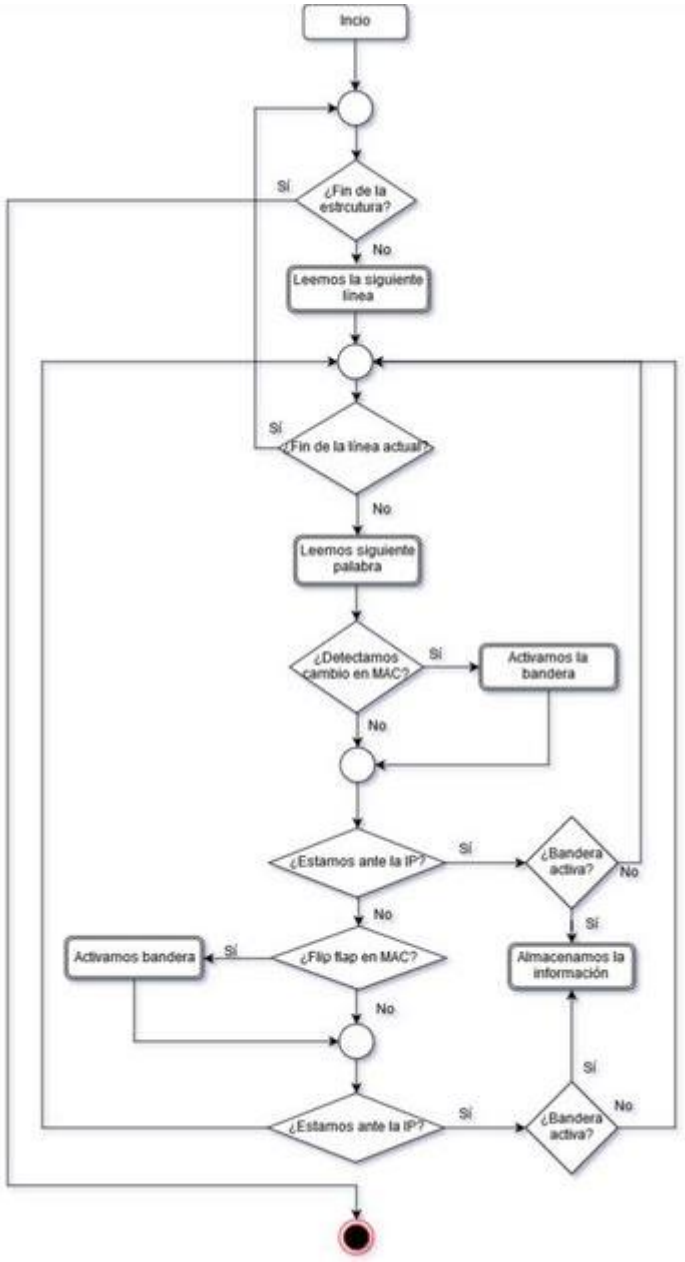


Figura 18 Diagrama flujo busca_Cambio_En_Mac()

Esta función, recibe cómo parámetros la estructura resultante de la lectura del fichero y la posición de inicio en dicha estructura.

Comienza recorriendo la estructura en busca de un mensaje de alerta ya sea "*changed ethernet address*" cómo "*flip flop*", si localiza alguno de estos mensajes, activa la bandera correspondiente y pasa a buscar la dirección

IP que provocó dicha alerta para almacenarla en la estructura que será devuelta para crear la regla en *Snort*.

La función registra_Regla(), recibe una estructura con la información de la regla a registrar en el fichero temporal citado anteriormente en este punto.

Figura 19 Diagrama flujo registra_Regla()

Se encarga de recorrer la estructura que recibe cómo parámetro y para cada, obtiene el instante en el que se va a registrar y extrae la información de la estructura para escribirla en el fichero. Antes de escribir, comprueba que para la regla que se va a crear no existía un registro similar previamente.

Es necesario almacenar el instante en el que se registran las reglas ya que estas estarán activas durante un intervalo definido en la librería principal.

Para detectar el instante en el que estas reglas han sobrepasado su intervalo, está la función detecta_Registro_Caducado().

Figura 20 Diagrama flujo detecta_Registro_Caducado()

Esta función, la cual recibe cómo parámetro la ruta donde está dicho fichero temporal, lo lee y almacena su contenido en una estructura. Tras esto, comienza a recorrer la estructura.

Para cada línea del fichero, obtiene la marca de tiempo actual y la compara con la que está registrada en el fichero, si la diferencia es mayor al intervalo de tiempo definido, se procede a eliminar dicho registro y su regla correspondiente en *Snort*, será necesario recargar *Snort* para que esa regla no actúe más.

3.4.4 Módulo 3: SSID Duplicado

Este módulo se encarga de detectar SSIDs duplicados, en la siguiente imagen, se puede observar el funcionamiento de este módulo.

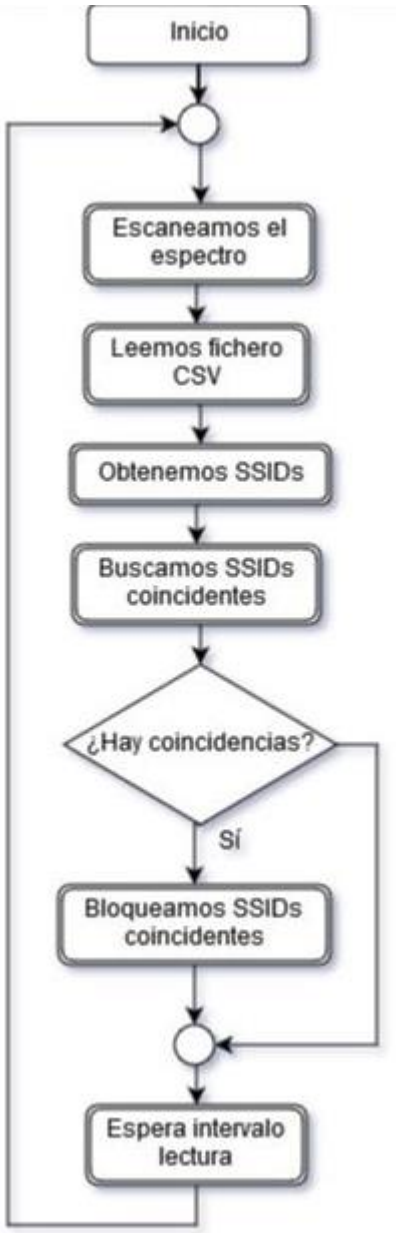


Figura 21 Diagrama flujo Módulo3: SSID duplicado

Observando el diagrama de flujo superior, el primer paso que se realiza es analizar el espectro mediante la herramienta *airodump-ng*, la salida de este, es un fichero *CSV* que se lee cuando se termina el análisis. Una vez leído, se extraen los SSIDs detectados y se busca en esos SSIDs si alguno coincide con el que está propagando nuestro dispositivo.

Si hemos detectado algún SSID similar, procedemos a bloquearlo mediante *Iptables*. Una vez hecho esto, antes de iniciar una nueva iteración en el bucle, se esperar el intervalo de lectura definido en la librería principal.

Este módulo, además de las funciones auxiliares, en su librería incluye una estructura propia para almacenar la información relativa al *SSID*, esta estructura es la siguiente.

```
//Estructura para almacenar los SSID leidos
typedef struct INFO_SSID {

    //Array para almacenar los SSID que se lean
    char *essid[NUM_SSID];
    //Array para almacenar los BSSID que se lean
    char *bssid[NUM_SSID];
    /*-------------------------------------------
        Bandera para conocer si la coincidencia
        fue ESSID o BSSID
            -- true; La coincidencia es en BSSID
            -- false; La coincidencia es en ESSID
    -------------------------------------------*/
    bool coincide_BSSID[NUM_SSID];
    //Variable para almacenar la cantidad de SSID leidos
    int num_ssid;

}info_ssid;
```

Figura 22 Estructura módulo 3

La estructura está compuesta de cuatro campos; una bandera para indicar si la coincidencia se da en *BSSID* o *ESSID*, un entero para indicar el número de *SSID* leídos y dos arrays para almacenar el *ESSID* y *BSSID* correspondiente a cada *SSID*.

Centrándonos en las funciones auxiliares, lee_fichero_csv(), es una función que recibe la ruta dónde se ha almacenado el fichero resultante del escaneo del espectro.

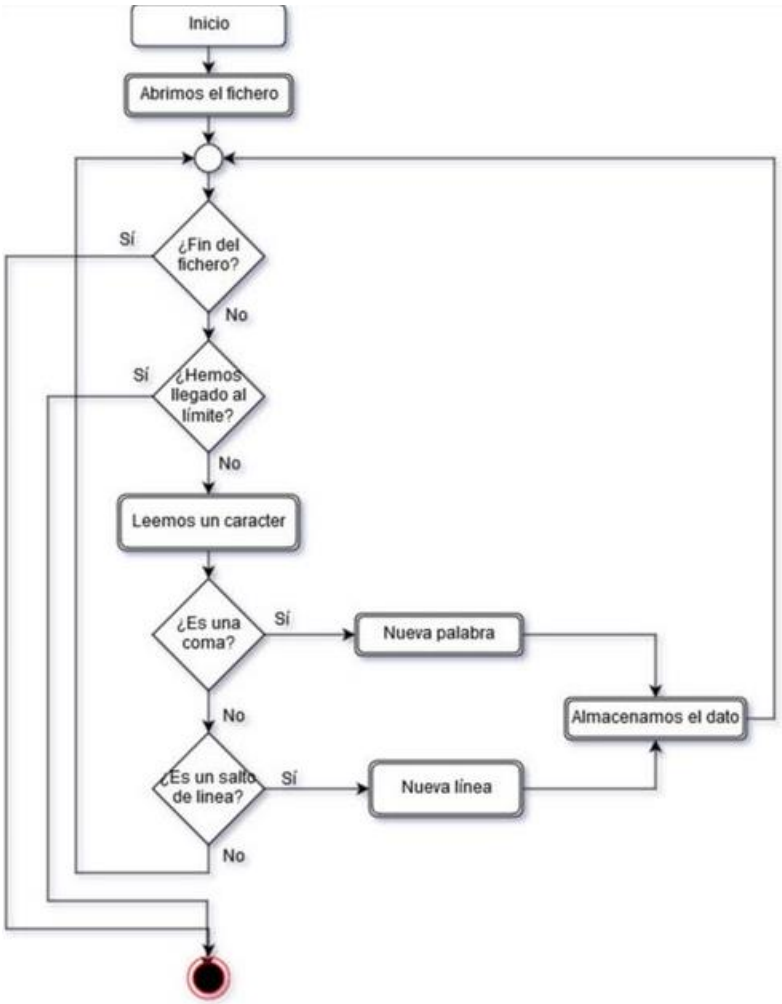


Figura 23 Diagrama flujo lee_fichero_csv()

Una vez abierto el fichero, mientras no lleguemos al final, lo leemos caracter a caracter, mientras no leamos una coma o un salto de línea, lo iremos almacenando en una variable auxiliar, una vez que leamos uno de estos caracteres, tomaremos distintas acciones.

Si es una coma, lo anterior era una única palabra, por tanto, la almacenamos en la estructura. Si es un salto de línea, realizamos lo mismo que en el caso anterior pero adicionalmente, reiniciaremos las variables y se añade en la estructura una nueva línea.

Tras esto, la función procesa_fichero_CSV(), recibe cómo parámetro la estructura creada por la función anterior y devuelve la estructura propia del módulo descrita anteriormente.

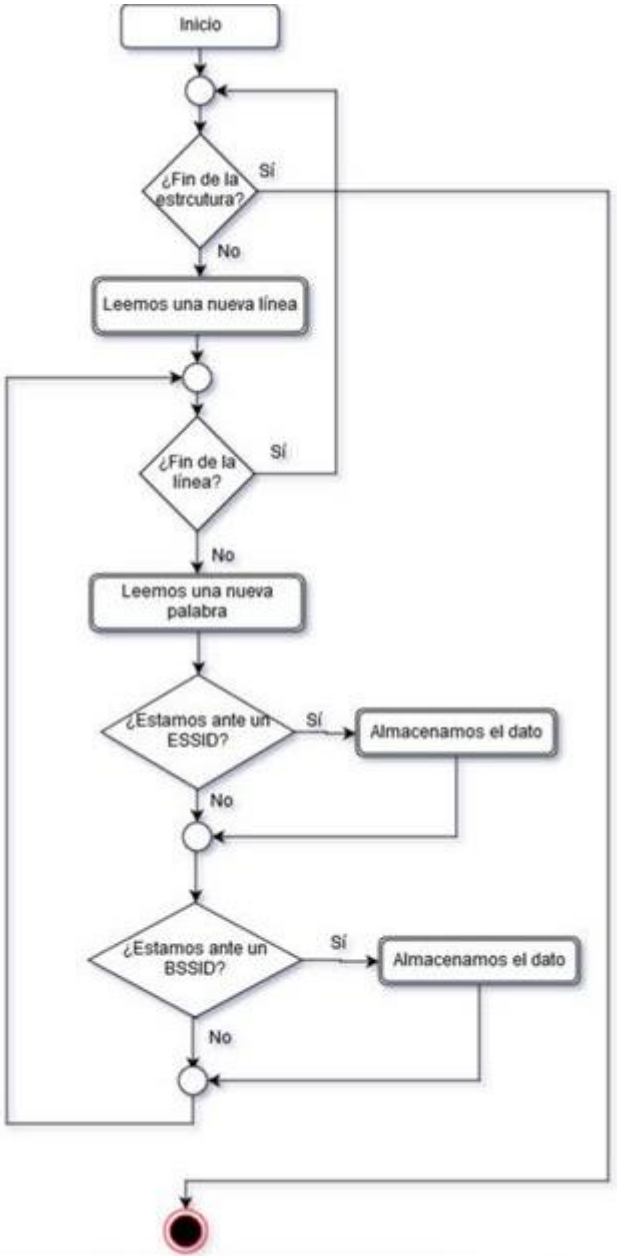


Figura 24 Digrama flujo procesa_fichero_CSV()

Esta función, recorre la estructura en busca de los *ESSIDs* y *BSSIDs* existentes en el espectro para almacenarlos junto con el número de *SSIDs* leídos en la estructura que será devuelta.

Por último, la función bloquea_SSID(), recibe cómo parámetro la estructura devuelta por la función anterior y tras comprobar si los *SSIDs* analizados coinciden con el que está propagando nuestro dispositivo;

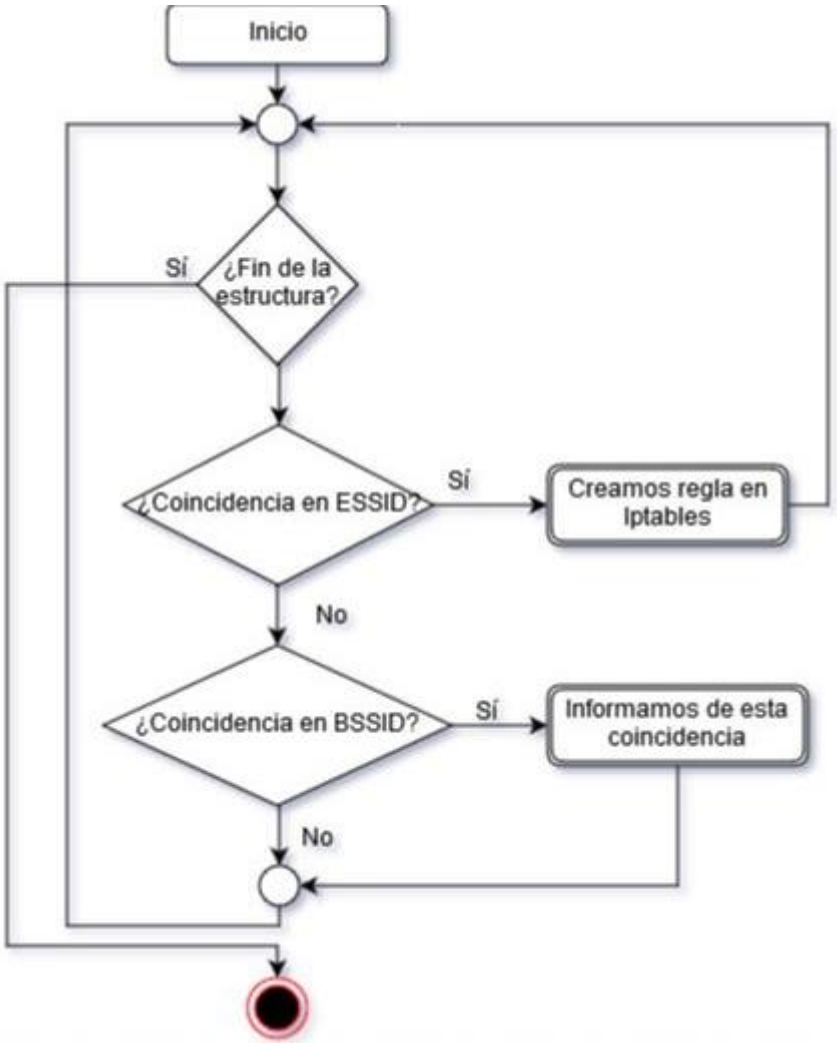


Figura 25 Diagrama flujo bloquea_SSID()

Recorre la estructura y comprueba si la coincidencia se dio en el *ESSID* o en *BSSID* mirando el valor de la bandera de la estructura. Si la coincidencia fue en *ESSID*, crea la regla correspondiente en *Iptables* para bloquear la dirección *MAC* del *SSID*. Si la coincidencia se da en el *BSSID*, la única acción a tomar será la de notificar al usuario, ya que no se podrá tomar ninguna acción correctiva porque lo que se está analizando son comunicaciones a nivel dos.

3.4.5 Módulo 4: BBDD

El cuarto y último módulo, se encarga de registrar en una base de datos los clientes de nuestra red.

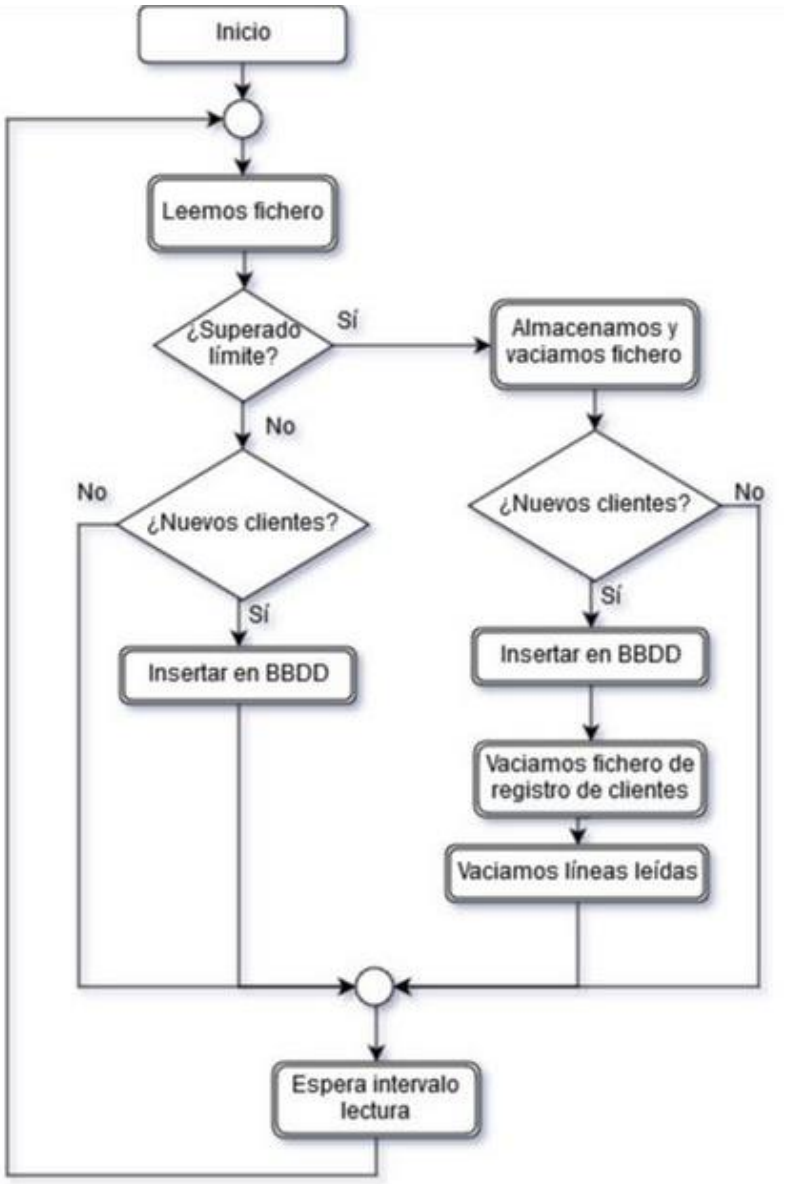


Figura 26 Diagrama flujo Módulo 4: BBDD

Este módulo, mediante un bucle, lee los ficheros de Arpwatch (*/var/lib/arpwatch/*.dat*) y tras comprobar si el fichero ha superado el límite, busca nuevos clientes, en caso que existan, los registra en la base de datos y tras esperar el tiempo de lectura definido, comienza una nueva iteración del bucle.

Este módulo, sólo contiene una función auxiliar, inserta_en_BBDD() recibe cómo parámetros una estructura y la posición de comienzo.

Figura 27 Diagrama flujo inserta_en_BBDD()

Recorre la estructura en busca de nuevos clientes, si existen, extrae los datos de la estructura y crea el comando necesario para insertar al cliente en el base de datos.

4 CONCLUSIONES

En este capítulo trataremos los beneficios que provoca la instalación del *WIPS* y plantearemos los posibles módulos que en un futuro podemos añadir al *WIPS* para dotarlo de mayor funcionalidad.

4.1 Beneficios

Como se comentó en capítulos anteriores, con el *WIPS* conseguimos añadir una capa de seguridad a los hogares que incorporan dispositivos *IoT*.

En las pruebas realizadas al *WIPS*, se ha conseguido parar los ataques *DoS* y *DDoS*, provocados desde el interior y el exterior de la red.

También se ha conseguido detectar y cortar los ataques *MITM* producidos en nuestra red local y la detección de la suplantación de *SSID*.

Por lo que, con este dispositivo, conseguimos proteger a nuestro hogar de los ataques más comunes a redes locales y así evitar que se perturbe el estado de los dispositivos interconectados "*tapando*" las principales brechas de seguridad implícitas en los hogares con dispositivos *IoT*.

La interfaz web nos permitirá visualizar los clientes que se han conectado a nuestra red, para así detectar posibles anomalías, y gestionar todas las acciones que realiza el *WIPS* para alterarlo en cualquier momento, si fuese necesario.

Por tanto, hemos conseguido alcanzar los objetivos propuestos en el alcance del proyecto y así aportar una mayor seguridad a redes locales.

4.2 Líneas futuras

Para este proyecto, se han pensado varias funcionalidades adicionales que en futuro podrán dotar al *WIPS* de una mayor capacidad para detectar posibles amenazas en el ámbito doméstico. Estas funcionalidades son:

- Aumentar el número de sensores y centralizar los datos en un punto.
- Localizar los dispositivos *IoT* estáticos.
- Aumentar los tipos de ataques que puede detectar.

Analizando estas funcionalidades en profundidad, en la primera lo que buscamos es la escalabilidad del *WIPS*, actualmente, nuestro dispositivo sólo posee un sensor (antena WiFi) y la *Raspberry* es la que procesa todos los datos que recoge este, aumentando el número de sensores, podremos cubrir un área más amplia y analizar todo el espectro con una precisión mayor, estos sensores, enviarían todos los datos que recogen a la *Raspberry* que actuaría como servidor central y procesaría todos los datos para tomar una decisión a partir de ellos.

De la funcionalidad anterior, sacamos indirectamente la segunda, ya que, gracias al aumento de sensores, podemos realizar técnicas de localización en interiores cómo la triangulación.

Dado que la mayoría de los sensores *IoT* que posee un hogar son estáticos, podemos triangular la posición de ellos y monitorizarlos constantemente, si se detecta que el dispositivo ha sufrido un cambio de posición, podemos notificar al usuario inmediatamente y bloquear la dirección *IP* o *MAC* del dispositivo hasta que el usuario informe de que el cambio se realizó intencionadamente.

Por último, la tercera funcionalidad que se ha pensado es la del aumento de los tipos de ataques a detectar.

El *WIPS* está limitado a los tipos de ataques definidos en el alcance del proyecto, de esta formar teníamos acotado el rango de actuación de este y podíamos probarlo todo correctamente, en futuro, interesaría incluir la detección de más ataques, esto se podría hacer gracias a la comunidad de *Snort* y el conjunto de reglas que estos crean, de esta forma, podríamos añadir reglas para alertar o detener cualquier tipo de ataque y así dotar de mayor seguridad aún al hogar.

Estas son todas las funcionalidades extras que se han pensado para que en un futuro se pueda ampliar la capacidad del *WIPS*.

5 ANEXOS

En este capítulo profundizaremos en la instalación de las herramientas, la configuración de la *Raspberry* y el código utilizado.

El contenido de este capítulo está pensado para su reproducción en *Raspbian* y puede variar si se intentan reproducir en otro sistema operativo.

5.1 Anexo A. Instalación de herramientas

En esta sección trataremos la instalación de las diferentes herramientas, así como la configuración necesaria para el correcto funcionamiento.

5.1.1 Snort

Para la instalación de *Snort*, en primer lugar, debemos instalar las dependencias externas:

```
sudo apt-get install bison flex zlib1g-dev
sudo apt-get install libpcre3 libpcre3-dev
sudo apt-get install libpcap0.8 libpcap0.8-dev
sudo apt-get install libnfnetlink-dev
sudo apt-get install libnetfilter-queue-dev
```

Una vez hecho esto, pasamos a instalar otra dependencia externa a través de su código fuente y no mediante repositorios cómo se realizó anteriormente.

```
cd /usr/src
sudo wget
http://downloads.sourceforge.net/project/libdnet/libdnet/libdnet-
1.11/libdnet-1.11.tar.gz
sudo tar xvfz libdnet-1.11.tar.gz
cd libdnet-*
sudo ./configure --prefix=/usr
sudo make
sudo make install
```

Tras esto, instalamos el módulo *DAQ (Data Acquisition)* [28], el cual nos permitirá que *Snort* actúe cómo *IPS*.

```
cd /usr/src
sudo wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
sudo tar xvfz daq-2.0.6.tar.gz
cd daq-*
sudo ./configure
sudo make
sudo make install
```

Por último, instalamos *Snort*.

```
cd /usr/src
```

```
sudo wget https://www.snort.org/downloads/snort/snort-2.9.9.0.tar.gz
sudo tar xvfz snort-2.9.9.0.tar.gz
cd snort-*
sudo ./configure --enable-sourcefire --enable-active-response --
enable-reload
sudo make
sudo make install
sudo ldconfig
```

Los parámetros que se le pasan a "*configure"* se encargan de:

- --enable-active-response: Soporta la interrupción de una sesión cuándo una regla coincide.
- --enable-reload: Permite recargar los ficheros de configuración de *Snort* sin pararlo, de esta forma, podemos ir actualizándolo cada vez que se cree una regla nueva.

Tras esto, ya tendríamos instalado *Snort* en nuestro dispositivo, pero será necesario configurarlo para que funcione correctamente y actúe cómo *IPS.*

En primer lugar, creamos la estructura de directorios.

```
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /etc/snort/so_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /var/log/snort
```

Tras esto, copiamos los ficheros de configuración de *Snort* en estos directorios

```
cd /usr/src/snort-*
sudo cp etc/* /etc/snort/
cd /usr/src/snort-*/etc
sudo cp attribute_table.dtd /etc/snort/
cd /usr/src/snort-*
sudo cp preproc_rules/* /etc/snort/preproc_rules/
```

Creamos los ficheros de reglas

```
cd /etc/snort/rules/
sudo touch local.rules
sudo touch /etc/snort/rules/white_list.rules
sudo touch /etc/snort/rules/black_list.rules
```

Modificamos el fichero de configuración de *Snort (/etc/snort/snort.conf)* para que este opere cómo un *IPS*. Para ello, en este fichero deberemos realizar los siguientes cambios:

1. Comentar a partir de la línea 575 (*Step 7*), las inclusiones de los ficheros de reglas, de esta forma garantizamos el control absoluto de las reglas que están cargadas y evitamos errores.
   La única línea que no se comenta es "*$RULE_PATH/local.rules*" y se añade "*$RULE_PATH/reglas_DoS.rules*" al final de este paso para así tener cargadas las reglas ante ataques *DoS* y DDoS creadas por la comunidad

2. Cambiamos las rutas de las reglas a:

- var RULE_PATH /etc/snort/rules
- var SO_RULE_PATH /etc/snort
- var PREPROC_RULE_PATH /etc/snort/preproc_rules
- var WHITE_LIST_PATH /etc/snort/rules
- var BLACK_LIST_PATH /etc/snort/rules

3. Nos aseguramos que en la línea 279 (*Step 5*), no está comentado:
   - prepocessor normalize_ip4
   - prepocessor normalize_tcp:ips ecn stream
   - prepocessor normalize_icmp4
   - prepocessor normalize_ip6
   - prepocessor normalize_icmp6

4. En la línea 202 (*Step 2*), añadimos lo siguiente:
   - config policy_mode:inline

   De esta forma indicamos que *Snort* se pueda ejecutar cómo IPS

5. A partir de la línea169 (*Step 2*), configuramos el módulo *DAQ* para que *Snort* actúe cómo *IPS*
   - config daq:nfq
   - config daq_dir:/usr/local/lib/daq
   - config daq_mode:inline
   - config daq_var:queue=0

Una vez hecho esto, guardamos el fichero de configuración de *Snort* para almacenar los cambios realizados y ya podemos ejecutarlo mediante el siguiente comando:

```
sudo snort -Q -c /etc/snort/snort.conf -A full
```

Aunque no funcionará cómo *IPS* ya que será necesario crear una regla en *Iptables*.

```
iptables -t nat -A PREROUTING -j NFQUEUE --queue-num 0 --queue-bypass
```

Esta regla permite la integración de ambos y que se pueda bloquear el tráfico, por tanto, ya tendríamos funcionando nuestro *IPS*.

En el caso que queramos recargar los ficheros de configuración, será necesario ejecutar el siguiente comando:

```
kill -SIGHUP pid_snort
```

5.1.2 Arpwatch

Arpwatch no precisa dependencias externas, por lo que podemos instalarla directamente mediante repositorios:

```
sudo apt-get install arpwatch
```

Una vez instalado, debemos redirigir los *logs* que genera *Arpwatch* por defecto a un fichero específico. Para ello, debemos modificar el fichero */etc/rsyslog.conf* añadiendo al final:

```
if $programname contains 'arpwatch' then /var/log/arpwatch.log
```

Reiniciamos el servicio para que este cambio cobre efecto

```
service rsyslog restart
```

A partir de este momento, todas las notificaciones de *Arpwatch* se registrarán en el fichero */var/log/arpwatch.log*

Por último, pasamos a modificar el fichero de configuración (*/etc/arpwatch.conf*), añadiendo al final:

```
eth0 -a -n 10.10.10.0/24
wlan0 -a -n 192.168.1.0/24
```

Con estas líneas indicamos a la herramienta las subredes a las que pertenecen las interfaces. Esta herramienta, no requiere ningún comando para iniciarla ya que se ejecuta automáticamente.

5.1.3 Aircrack-ng

Aircrack-ng no requiere dependencias externas, por tanto, podemos instalarla directamente:

```
apt-get install aircrack-ng
```

Con el comando anterior, ya tenemos instalada la suite en nuestro dispositivo, por lo que podremos ejecutar las herramientas deseadas ya que tampoco necesita ninguna configuración previa.

Si queremos realizar un análisis del espectro será necesario en primer lugar establecer la interfaz por la que se realizará en modo monitor.

```
airmon-ng start <interfaz>
```

Con este comando, crearemos una interfaz llamada mon0 y podremos ejecutar la segunda herramienta, encargada de realizar el análisis. Los resultados se mostrarán por pantalla, cambiando periódicamente de canal, si deseamos parar la captura bastará con pulsar *Ctrl+C*

```
airodump-ng mon0
```

5.1.4 PostgreSQL

Para instalar *PostgreSQL* ejecutamos el siguiente comando:

```
apt-get install postgresql
```

Al instalarlo, se crea un usuario por defecto con las siguientes credenciales

- Usuario: postgres
- Contraseña: pgsqlroot

Para cambiar esta contraseña, debemos ejecutar:

```
sudo passwd postgres
```

Tras esto, vamos a crear un usuario propio para el *WIPS*, para lo que ejecutamos:

```
su postgres
psql
CREATE USER clientes WITH ENCRYPTED PASSWORD 'clientes' CREATEDB;
```

```
\q
exit
```

El usuario creado se llama "clientes" con contraseña igual al nombre de usuario.

Cambiamos la configuración para así permitir la autenticación por contraseña del nuevo usuario, para ello añadimos en el fichero de configuración de *PostgreSQL*, antes de la línea que contiene "l*ocal all all peer*" lo siguiente:

```
local all clientes md5
```

Reiniciamos el servicio para que los cambios tengan efecto

```
/etc/init.d/postgresql restart
```

Por último, creamos la base de datos y la tabla que contendrá los datos de los clientes.

Para crear la base de datos:

```
createdb -U clientes clientes_asociados
```

Con este comando, hemos creado en el usuario "*clientes"* una base de datos llamada *"clientes_asociados"*. Creamos la tabla dentro de esa base de datos.

```
psql -U clientes -d clientes_asociados -f crea_tabla_bbdd.sql
```

Siendo *crea_tabla_bbdd.sql* un fichero que contiene:

```
CREATE TABLE datos_clientes
(
dir_mac macaddr NOT NULL,
dir_ip inet NOT NULL,
fecha timestamp without time zone NOT NULL,
CONSTRAINT primary_key PRIMARY KEY (dir_mac, fecha)
);
```

Ya tenemos la base de datos y la tabla creada, sólo falta rellenar la tabla con los datos de los clientes mediante la siguiente sentencia:

```
export PGPASSWORD='clientes'; psql -U clientes -d clientes_asociados
-c "INSERT INTO datos_clientes VALUES ('direccionMAC', 'direccionIP',
to_timestamp(valortimestamp)); "
```

En primer lugar, indicamos la contraseña de la base de datos, tras esto, accedemos a ella y, por último, insertamos los datos deseados.

Para la inserción, será necesario realizar una conversión en la tercera columna, ya que el valor que se le pasa es una marca de tiempo *Unix* y lo que queremos registrar es una fecha legible.

5.1.5 Apache/Tomcat

Comenzamos instalando tanto *Apache* cómo *Tomcat*

```
apt-get install apache2
apt-get install tomcat7
```

En el fichero de configuración de *Apache (/etc/apache2/apache2.conf)* comentamos todas las líneas que estén incluidas entre las etiquetas "<*Directory*>" y "<*/Directory*>" estas incluidas.

En el directorio "*/etc/apache2/conf-enabled/"* creamos el fichero "*config_wips.conf*" con el siguiente contenido:

```
ScriptAlias /wips/cgi-bin/ /home/pi/web/cgi-bin/

<Directory "/home/pi/web/cgi-bin/">
AllowOverride All
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
Order allow,deny
Allow from all
</Directory>

Alias /wips "/home/pi/web/"

<Directory "/home/pi/web/">
AllowOverride All
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch +Includes
DirectoryIndex wips.html
Order allow,deny
Allow from all
</Directory>
```

En el fichero "*/etc/apache2/sites-enabled/000-default.conf*" comentamos la línea que comienza por "*DocumentRoot*" y añadimos:

```
DocumentRoot /home/pi/web/cgi/wips.html
```

Copiamos el módulo que habilita el uso de *CGI*.

```
cp /etc/apache2/mods-available/cgi.load /etc/apache2/mods-enabled/cgi.load
```

Reiniciamos el servicio para que se efectúen los cambios

```
service apache2 restart
```

Tras esto, creamos los directorios donde almacenaremos el código de la interfaz web

```
mkdir /home/pi/web
mkdir /home/pi/web/cgi
mkdir /home/pi/web/cgi-bin
mkdir /home/pi/web/jsp
mkdir /home/pi/web/WEB-INF
mkdir /home/pi/web/WEB-INF/classes
mkdir /home/pi/web/WEB-INF/lib
mkdir /home/pi/web/WEB-INF/src
```

Ya tenemos configurado *Apache* para que procese las páginas estáticas y las que hacen uso de *CGI*, ahora pasamos a configurar *Tomcat* para que coopere con *Apache*.

En primer lugar, debemos instalar el módulo que permite la integración de ambos servicios.

```
apt-get install libapache2-mod-jk
```

Añadimos al final del fichero "*/etc/apache2/sites-enabled/000-default.conf*" lo siguiente:

```
JkMount         /*.jsp              ajp13_worker
```

En el fichero "*/etc/tomcat7/server.xml*" descomentamos la línea

```
<Connector port="8009" protocol="AJP/1.3" />
```

Y creamos el fichero "*/etc/tomcat7/Catalina/localhost/wips.html*" con el siguiente contenido:

```
<Context      path="/wips"      docBase="/home/pi/web"      debug="0"
reloadable="true" />
```

Reiniciamos los servicios para que tengan efecto los cambios

```
service tomcat7 restart
service apache2 restart
```

Tras esto, la interfaz web funcionará correctamente, procesando las páginas *Java Tomcat* y el resto *Apache*.

5.2 Anexo B. Configuración de la *Raspberry* cómo punto de acceso

En esta sección trataremos los pasos necesarios para configurar la *Raspberry* cómo punto de acceso.

Comenzamos instalando los programas que nos harán falta, para ello, ejecutamos el siguiente comando:

```
sudo apt-get install hostapd isc-dhcp-server
```

Una vez que hayamos instalado los programas, los configuraremos correctamente.

En primer lugar, configuramos el servidor *DHCP* (*Dynamic Host Configuration Protocol*) [29], el cual proporcionará las direcciones *IP* a los dispositivos que se conecten al punto de acceso.

Para configurarlo, accedemos al fichero *dhcp*.conf situado en */etc/dhcp/*.

```
sudo nano /etc/dhcp/dhcp.conf
```

Buscamos las siguientes líneas y las comentamos

```
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;
```

En este mismo fichero, debemos descomentar la línea en la que aparece lo siguiente:

```
authoritative;
```

Para terminar con la configuración del servidor *DHCP*, debemos añadir al final del fichero lo siguiente:

```
subnet 10.10.10.0 netmask 255.255.255.0 {
range 10.10.10.2 10.10.10.254;
option broadcast-address 10.10.10.255;
option routers 10.10.10.1;
default-lease-time 600;
max-lease-time 7200;
```

```
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

De esta forma, estamos indicamos que la subred a la que proporcionamos direcciones es la *10.10.10.0*, dentro de esta, sólo asignaremos desde la *10.10.10.2* a la *10.10.10.254*, la dirección de *broadcast* (difusión) es la *10.10.10.255* y la del router la *10.10.10.1*, en este caso entendemos la del router cómo la de la interfaz de la *Raspberry* que se propagará la señal *WiFi*.

Tras esto, modificamos el fichero *isc-dhcp-server* que se encuentra bajo el directorio */etc/default/*.

```
sudo nano /etc/default/isc-dhcp-server
```

Añadimos *wlan0* dentro de *INTERFACES=""*, para indicar que el servidor *DHCP* actúe en la interfaz *wlan0*, deberá quedar de la siguiente forma:

```
INTERFACES=”wlan0”
```

El siguiente paso, será el de asignar una dirección *IP* estática a la interfaz *wlan0*. En primer lugar, desactivamos la interfaz y accedemos al fichero de configuración:

```
sudo ifdown wlan0
sudo nano /etc/network/interfaces
```

En este fichero, tras “*allow-hotplug wlan0*” debemos añadir:

```
iface wlan0 inet static
address 10.10.10.1
netmask 255.255.255.0
```

Y comentar lo siguiente:

```
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Una vez que hemos modificado el fichero, podemos arrancar la interfaz y comprobar que ha tomado la dirección *IP* deseada.

```
sudo ifup wlan0
ifconfig wlan0
```

El último paso, es la configuración del punto de acceso, para ello, creamos el fichero *hostapd.conf* situado bajo el directorio */etc/hostapd/*.

```
sudo touch -f /etc/hostapd/hostapd.conf
```

Tras esto, escribimos en él lo siguiente.

```
interface=wlan0
ssid=Red_RasPi
hw_mode=g
channel=6
macaddr_acl=0
```

```
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=contrasena_red
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Es importante que el fichero no contenga ninguna línea extra ni espacios entre líneas.

Finalizamos modificando el fichero */etc/default/hostapd*, en él debemos buscar la línea *DAEMON_CONF* y cambiarla por la siguiente:

```
DAEMON_CONF=”/etc/hostapd/hostapd.conf”
```

Con esto, ya tendríamos configurada la *Raspberry* cómo punto de acceso. Para que comience a propagar el *SSID* y así poder conectarnos a ella, debemos ejecutar lo siguiente:

```
sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

De esta forma, cualquier dispositivo puede conectarse a la *Raspberry*, aunque no tendría acceso a internet, ya que quedan algunos aspectos de la configuración de la Raspberry que se verán en el siguiente punto.

5.3 Anexo C. Reglas de Snort

En capítulos anteriores, hemos hablado sobre el funcionamiento de *Snort* y la necesidad de tener varios ficheros de reglas para poder actuar ante el tráfico de la red, en concreto, se habló de dos ficheros de reglas, uno vacío al iniciar por primera vez el *WIPS* y otro de la comunidad que contenía las reglas necesarias para detectar los ataques *DoS* y *DDoS*. Este último en concreto, se descargó directamente de la página oficial de *Snort* [30] y se modificó para generar un mensaje de información acorde con el software desarrollado para el *WIPS*.

Se comentó también en la sección anterior la ubicación y nombre de estos ficheros, a continuación, se detalla el contenido del fichero “*reglas_DoS*”

```
######### DoS #########

######### IP #########
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Jolt attack"; dsize:408;
fragbits:M; reference:cve,1999-0345; classtype:attempted-dos; sid:268; rev:4;)

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS IGMP dos attack"; fragbits:M+;
ip_proto:2; reference:bugtraq,514; reference:cve,1999-0918; classtype:attempted-dos;
sid:272; rev:9;)
##################

######### UDP #########
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Teardrop attack"; fragbits:M;
id:242; reference:bugtraq,124; reference:cve,1999-0015; reference:nessus,10279;
reference:url,www.cert.org/advisories/CA-1997-28.html; classtype:attempted-dos;
sid:270; rev:6;)

alert udp $EXTERNAL_NET any -> $HOME_NET 9 (msg:"DoS Ascend Route";
content:"NAMENAME"; depth:50; offset:25; reference:arachnids,262;
reference:bugtraq,714; reference:cve,1999-0060; classtype:attempted-dos; sid:281;
rev:5;)
```

```
  alert udp $EXTERNAL_NET any -> $HOME_NET 500 (msg:"DoS ISAKMP invalid identification
payload attempt"; content:"|05|"; depth:1; offset:16; byte_test:2,>,4,30;
byte_test:2,<,8,30; reference:bugtraq,10004; reference:cve,2004-0184;
classtype:attempted-dos; sid:2486; rev:5;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 2048 (msg:"DoS squid WCCP I_SEE_YOU message
overflow attempt"; content:"|00 00 00 08|"; depth:4; byte_test:4,>,32,16;
reference:cve,CAN-2005-0095; reference:bugtraq,12275; classtype:attempted-user;
sid:3089; rev:1;)
  ###################


  ######### TCP #########
  alert tcp $EXTERNAL_NET any -> $HOME_NET 617 (msg:"DoS arkiea backup";
flow:to_server,established; dsize:>1445; reference:arachnids,261;
reference:bugtraq,662; reference:cve,1999-0788; classtype:attempted-dos; sid:282;
rev:8;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 3372 (msg:"DoS MSDTC attempt";
flow:to_server,established; dsize:>1023; reference:bugtraq,4006; reference:cve,2002-
0224; reference:nessus,10939; classtype:attempted-dos; sid:1408; rev:10;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 6004 (msg:"DoS iParty attempt";
flow:to_server,established; content:"|FF FF FF FF FF FF|"; offset:0;
reference:bugtraq,6844; reference:cve,1999-1566; classtype:misc-attack; sid:1605;
rev:6;)
  ###################


  ######### ICMP #########
  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS ath"; itype:8;
content:"+++ath"; nocase; reference:arachnids,264; reference:cve,1999-1228;
classtype:attempted-dos; sid:274; rev:5;)
  ###################


  ######### DDoS #########


  ######### UDP #########
  alert udp $EXTERNAL_NET any -> $HOME_NET 18753 (msg:"DoS shaft handler to agent";
content:"alive tijgu"; reference:arachnids,255; classtype:attempted-dos; sid:239;
rev:2;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 20433 (msg:"DoS shaft agent to handler";
content:"alive"; reference:arachnids,256; classtype:attempted-dos; sid:240; rev:2;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DoS Trin00 Daemon to Master
message detected"; content:"144"; reference:arachnids,186; classtype:attempted-dos;
sid:231; rev:3;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DoS Trin00 Daemon to Master
*HELLO* message detected"; content:"*HELLO*"; reference:arachnids,185;
reference:url,www.sans.org/newlook/resources/IDFAQ/trinoo.htm; classtype:attempted-
dos; sid:232; rev:5;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 27444 (msg:"DoS Trin00 Master to Daemon
default password attempt"; content:"144adsl"; reference:arachnids,197;
classtype:attempted-dos; sid:237; rev:2;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 6838 (msg:"DoS mstream agent to handler";
content:"newserver"; classtype:attempted-dos; sid:243; rev:2;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 10498 (msg:"DoS mstream handler to agent";
content:"stream/"; reference:cve,2000-0138; classtype:attempted-dos; sid:244; rev:3;)
```

```
  alert udp $EXTERNAL_NET any -> $HOME_NET 10498 (msg:"DoS mstream handler ping to
agent"; content:"ping"; reference:cve,2000-0138; classtype:attempted-dos; sid:245;
rev:3;)

  alert udp $EXTERNAL_NET any -> $HOME_NET 10498 (msg:"DoS mstream agent pong to
handler"; content:"pong"; classtype:attempted-dos; sid:246; rev:2;)
  ##################


  ######### TCP #########
  alert tcp $HOME_NET 20432 -> $EXTERNAL_NET any (msg:"DoS shaft client login to
handler"; flow:from_server,established; content:"login|3A|"; reference:arachnids,254;
reference:url,security.royans.net/info/posts/bugtraq_ddos3.shtml; classtype:attempted-
dos; sid:230; rev:5;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"DoS Trin00 Attacker to Master
default startup password"; flow:established,to_server; content:"betaalmostdone";
reference:arachnids,197; classtype:attempted-dos; sid:233; rev:3;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"DoS Trin00 Attacker to Master
default password"; flow:established,to_server; content:"gOrave"; classtype:attempted-
dos; sid:234; rev:2;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 27665 (msg:"DoS Trin00 Attacker to Master
default mdie password"; flow:established,to_server; content:"killme"; classtype:bad-
unknown; sid:235; rev:2;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 12754 (msg:"DoS mstream client to handler";
flow:to_server,established; content:">"; reference:cve,2000-0138; classtype:attempted-
dos; sid:247; rev:4;)

  alert tcp $HOME_NET 12754 -> $EXTERNAL_NET any (msg:"DoS mstream handler to client";
flow:to_client,established; content:">"; reference:cve,2000-0138; classtype:attempted-
dos; sid:248; rev:4;)

  alert tcp $EXTERNAL_NET any -> $HOME_NET 15104 (msg:"DoS mstream client to handler";
flow:stateless;    flags:S,12;    reference:arachnids,111;    reference:cve,2000-0138;
classtype:attempted-dos; sid:249; rev:8;)

  alert tcp $HOME_NET 15104 -> $EXTERNAL_NET any (msg:"DoS mstream handler to client";
flow:from_server,established;          content:">";          reference:cve,2000-0138;
classtype:attempted-dos; sid:250; rev:4;)
  ##################


  ######### ICMP #########
  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS TFN Probe"; icmp_id:678;
itype:8; content:"1234"; reference:arachnids,443; classtype:attempted-recon; sid:221;
rev:4;)

  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS tfn2k icmp possible
communication"; icmp_id:0; itype:0; content:"AAAAAAAAAA"; reference:arachnids,425;
classtype:attempted-dos; sid:222; rev:2;)

  alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"DoS TFN server response";
icmp_id:123;    icmp_seq:0;    itype:0;    content:"shell    bound    to    port";
reference:arachnids,182; classtype:attempted-dos; sid:238; rev:6;)

  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS - TFN client command LE";
icmp_id:51201; icmp_seq:0; itype:0; reference:arachnids,183; classtype:attempted-dos;
sid:251; rev:3;)

  alert icmp 3.3.3.3/32 any -> $EXTERNAL_NET any (msg:"DoS Stacheldraht server spoof";
icmp_id:666;  itype:0;  reference:arachnids,193;  classtype:attempted-dos;  sid:224;
rev:3;)

  alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"DoS Stacheldraht gag server
response";   icmp_id:669;   itype:0;   content:"sicken";   reference:arachnids,195;
classtype:attempted-dos; sid:225; rev:6;)
```

```
  alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"DoS Stacheldraht server
response";  icmp_id:667;  itype:0;  content:"ficken";  reference:arachnids,191;
classtype:attempted-dos; sid:226; rev:6;)

  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Stacheldraht client
spoofworks"; icmp_id:1000; itype:0; content:"spoofworks"; reference:arachnids,192;
classtype:attempted-dos; sid:227; rev:6;)

  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Stacheldraht client check
gag";  icmp_id:668;  itype:0;  content:"gesundheit!";  reference:arachnids,194;
classtype:attempted-dos; sid:236; rev:6;)

  alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Stacheldraht client check
skillz";  icmp_id:666;  itype:0;  content:"skillz";  reference:arachnids,190;
classtype:attempted-dos; sid:229; rev:5;)
  ###################
```

5.4 Anexo D. Código

En este apartado, se recoge el código desarrollado en el proyecto.

En primer lugar, el fichero *Makefile* que permite la compilación de todo el código fuente.

```
#Makefile WIPS
WIPS:  main.o  mod1_DoS.o  func_aux_mod1.o  mod2_MITM.o  func_aux_mod2.o  mod3_SSID.o
mod3_SSID.o func_aux_mod3.o mod4_database.o func_aux_mod4.o func_aux.o
	gcc  -o  WIPS  main.o  mod1_DoS.o  func_aux_mod1.o  mod2_MITM.o  func_aux_mod2.o
mod3_SSID.o func_aux_mod3.o mod4_database.o func_aux_mod4.o func_aux.o -pthread

main.o: main.c constantes_y_librerias.h
	gcc -W -Wall -c main.c

mod4_database.o: mod4_database.c func_aux_mod4.h
	gcc -W -Wall -c mod4_database.c

func_aux_mod4.o: func_aux_mod4.c func_aux_mod4.h
	gcc -W -Wall -c func_aux_mod4.c

mod3_SSID.o: mod3_SSID.c func_aux_mod3.h
	gcc -W -Wall -c mod3_SSID.c

func_aux_mod3.o: func_aux_mod3.c func_aux_mod3.h
	gcc -W -Wall -c func_aux_mod3.c

mod2_MITM.o: mod2_MITM.c func_aux_mod2.h
	gcc -W -Wall -c mod2_MITM.c

func_aux_mod2.o: func_aux_mod2.c func_aux_mod2.h
	gcc -W -Wall -c func_aux_mod2.c

mod1_DoS.o: mod1_DoS.c func_aux_mod1.h
	gcc -W -Wall -c mod1_DoS.c

func_aux_mod1.o: func_aux_mod1.c func_aux_mod1.h
	gcc -W -Wall -c func_aux_mod1.c

func_aux.o: func_aux.c func_aux.h
	gcc -W -Wall -c func_aux.c

eliminaSem: elmina_semaforos.c constantes_y_librerias.h func_aux.c func_aux.h
	gcc -W -Wall -o eliminaSem elmina_semaforos.c func_aux.c -pthread

clean:
```

```
	rm *.o
```

El fichero principal *main.c*;

```
/*----------------------------------------
Autor: Francisco Javier Santos
Fecha: 10-03-2017
Descripción: Fichero principal que integra
todos los módulos
----------------------------------------*/

#include "constantes_y_librerias.h"
#include "func_aux.h"

//Constante para definir el número de módulos creados
#define NUM_MODULOS 4

int main () {

	//Variables para la creación de hilos
	pthread_attr_t attr;
	pthread_t thid[NUM_MODULOS];

	//Creamos los semáforos
	crea_semaforo(SEM_LECTURA);
	crea_semaforo(SEM_SNORT);
	crea_semaforo(SEM_SID);
	crea_semaforo(SEM_REGLAS_SNORT);

	//Iniciamos los atributos de los hilos
	pthread_attr_init (&attr);
	pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);

	/*-------------------------------------
		Creamos los hilos independientes
		de esta forma cada función es un
		proceso independiente
	-------------------------------------*/
	registra_log("Iniciando WIPS\n");
	pthread_create(&thid[0], &attr, mod1, NULL);
	pthread_create(&thid[1], &attr, mod2, NULL);
	pthread_create(&thid[2], &attr, mod3, NULL);
	pthread_create(&thid[3], &attr, mod4, NULL);

	pthread_attr_destroy (&attr);

	//Dejamos en una espera continua al proceso principal
	while(true)
		sleep(INTERVALO_LECTURA);

	//Eliminamos los semáforos
	/*----------------------------------------
	Se ha creado el fichero eliminaSem.c
	para eliminar los semáforos cuando
	se detiene el proceso principal y
	los semáforos siguen abiertos
	----------------------------------------*/
	elimina_semaforo(SEM_LECTURA);
	elimina_semaforo(SEM_SNORT);
	elimina_semaforo(SEM_SID);
	elimina_semaforo(SEM_REGLAS_SNORT);

	return 0;
}
```

El fichero *constantes_y_librerias.h* el cuál es la librería principal.

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 13-03-2017
Descripción: Fichero que contiene todas las
constantes y librerias necesarias para
la ejecución del programa
-----------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>
#include <semaphore.h>
#include <pthread.h>

#ifndef constantes_y_librerias
#define constantes_y_librerias


/*-------------------------------------------------------------------------
       Constantes para definir las rutas y nombres de los ficheros que se
       va a usar en las funciones
-------------------------------------------------------------------------*/
#define FICHERO_ALERT_SNORT "/var/log/snort/alert"
#define FICHERO_REGLAS_SNORT "/etc/snort/rules/local.rules"
#define FICHERO_SID "/home/pi/TFG/sid"
#define FICHERO_PID_SNORT "/tmp/WIPS/pid.txt"
#define FICHERO_ARPWATCH "/var/log/arpwatch.log"
#define FICHERO_REGISTRO_REGLAS_MITM "/tmp/WIPS/registro_reglas_MITM"
#define FICHERO_CAPTURA_WIFI "/tmp/WIPS/captura-01.csv"
#define FICHERO_CAPTURA_WIFI_SIN_EXTENSION "/tmp/WIPS/captura"
#define EXTENSION_FICHERO_CAPTURA_WIFI "-01.csv"
#define FICHERO_PID_AIRODUMP "/tmp/WIPS/pid_airodump.txt"
#define FICHERO_REGISTRO_CLIENTES "/tmp/WIPS/registro_clientes.txt"
#define FICHERO_LOG "/var/log/wips.log"

//Constante para definir la contraseña de la base de datos
#define CONTRASENIA_BBDD "clientes"
//Constante para definir el usuario con el que nos queremos registrar
#define USUARIO_BBDD  "clientes"
//Constante para definir la bbdd a la que deseamos acceder
#define NOMBRE_BBDD "clientes_asociados"
//Constante para definir la tabla a la que vamos a acceder
#define NOMBRE_TABLA_BBDD "datos_clientes"

/*-------------------------------------------------------
       Constante para definir los errores que pueden
       existir
-------------------------------------------------------*/
#define ERROR_NUM_LINEAS "El fichero %s supera las %d líneas útiles máximas\n"
#define ERROR_APERTURA_FICHERO "No se puede abrir el fichero %s \n"
#define ERROR_APERTURA_SEMAFORO "Error al abrir el semáforo %s\n"
#define ERROR_APERTURA_VARIOS_SEMAFOROS "Error al abrir los semáforos %s y %s\n"
#define ERROR_CREACION_SEMAFORO "Error al crear el semáforo %s\n"
#define ERROR_CERRAR_SEMAFORO "Error al cerrar el semáforo %s\n"
#define BSSID_DUPLICADO "Encontrado BSSID duplicado %s %s\n"

#define INICIO 0
#define IGUAL 0
//Constante que devolverá la funcion busca_Regla cuando no encuentre una igual
#define NO_COINCIDE -1
#define NO_EXISTE -1


/*-------------------------------------------------------
       Constante para definir el número máximo de línea que
```

```
       contiene el fichero, sin contar con las líneas
       en blanco
-------------------------------------------------------*/
#define NUM_FRASES 500
//Constante para definir el límite a partir del cuál crearemos un nuevo fichero
#define LIMITE_LINEAS_LEIDAS NUM_FRASES-30
//Constante para definir el número máximo de palabras en una frase
#define NUM_PALABRAS_POR_FRASE 50
//Constante para definir el número máximo de caracteres para una palabra
#define NUM_CARACTERES_PALABRA 100
//Número máximo de reglas que almacenarán las tablas
#define NUM_REGLAS 100
//Constante para definir el número máximo de caracteres para una regla
#define NUM_CARACTERES_REGLAS 500
//Constante para definir el número máximo de digitos que puede tener el SID
#define NUM_DIG_SID 10
//Constante para definir el número máximo de SSID que vamos a almacenar en la estructura
#define NUM_SSID 50


#define TABULADOR '\t'
#define SALTO_DE_LINEA '\n'
#define ESPACIO ' '
#define PUNTO '.'
#define TERMINADOR '\0'
//Constante para definir el caracter :
#define DOS_PUNTOS ':'
//Constante para definir el caracter ','
#define COMA ','


//Constates para definir la posición dentro de una tabla de cada fragmento
#define FRAG_1 0
#define FRAG_2 1
#define FRAG_3 2
//Constantes para definir el valor de cada fragmento de IP de la red local
#define VALOR_FRAG_1 "10"
#define VALOR_FRAG_2 "10"
#define VALOR_FRAG_3 "10"
//El número máximo de digitos que puede tener un fragmento de IP
#define MAX_DIG_FRAGMENTO_IP 3


//Constante para definir los protocolos
#define UDP "UDP"
#define TCP "TCP"
#define ICMP "ICMP"
#define IP "IP"


//Constante para definir la línea de fin de una coincidencia
#define FIN_REGLA_ALERTA 3
#define SIGUIENTE_LINEA 1
//Constante para definir la posición relativa del protocolo
#define LINEA_PROTOCOLO 2
//Constante donde definimos la posición de la hora en el fichero de registro
#define HORA 0
//Constante para indicar la posición de la fecha en la bbdd
#define FECHA 2
//Constante para definir la posición de la IP origen dentro de una línea
#define POSICION_IP_ORIGEN 3
//Igual que el anterior pero relativa a la IP destino
#define POSICION_IP_DESTINO 5
//Constantes para definir la posición que ocupan los campos relevantes en una regla
#define POS_ACCION 0
#define POS_PROTOCOLO 1
#define POS_DIR_IP_ORIG 2
#define POS_PUERTO_ORIG 3
#define POS_DIR_IP_DEST 5
```

```
#define POS_PUERTO_DEST 6
//Posición que ocupa la dirección IP
#define POS_DIR_IP 3
#define POS_DIR_IP_INM POS_DIR_IP-1
//Posición que ocupa la nueva dirección MAC
#define POS_DIR_MAC_NUEVA 4
#define POS_DIR_MAC_NUEVA_INM POS_DIR_MAC_NUEVA-1
//Posición que ocupa la vieja dirección MAC
#define POS_DIR_MAC_VIEJA 5
#define POS_DIR_MAC_VIEJA_INM POS_DIR_MAC_VIEJA-1
//Constante para definir la posición de la bandera que indica si la IP es origen o
destino
#define POS_BANDERA_IP 4
//Constante para definir la posición de la bandera que indica si la IP lleva puerto
asociado
#define POS_BANDERA_PUERTO 5


//Constante para definir el número máximo de digitos de un puerto
#define TAM_PUERTO 6
//Constante para definir el número máximo de digitos que puede tener la fecha
#define TAM_FECHA 100
//Tamaño de la dirección MAC
#define TAM_MAC 17
//Constante para definir el número máximo de octetos en una dirección IP
#define TAM_IP 4


//Constante para definir el valor inicial del semáforo
#define INICIO_SEM 1
//Constante para almacenar el nombre del semáforo para los ficheros de lecturas
#define SEM_LECTURA "semaforo_lectura"
//Constante para definir el nombre del semáforo que se usará para la recarga de Snort
#define SEM_SNORT "semaforo_snort"
//Constante para definir el nombre del semáforo que usará el fichero donde se almacena el
sid
#define SEM_SID "semaforo_sid"
//Constante para definir el nombre del semáforo que usará el fichero de reglas de snort
#define SEM_REGLAS_SNORT "semaforo_reglas_snort"


//Constante de tiempo para definir los segundos que pasaran entre actividad
#define INTERVALO_LECTURA 3
//Constante paara definir el paso de una hora
#define PASA_UNA_HORA 100
//Constante para definir el tiempo en el que estaremos capturando tráfico de la red
#define TIEMPO_CAPTURA 30


//Indica que la dirección MAC ha cambiado para una misma IP
#define CAMBIO_EN_MAC "changed"
//Indica que la MAC ha cambiado en un intervalo muy corto de tiempo
#define CAMBIO_MAC_INMEDIATO "flip"


//Constante para definir la palabra ESSID
#define ESSID "ESSID"
//Connstante para definir la palabra BSSID
#define BSSID "BSSID"
//Constante para definir la palabra que indica que comienza la info de los clientes
#define INFO_CLIENTES "Station MAC"


/*----------------------------------------
       Función que implementa el primer
       módulo

              Recibe: Nada
              Devuelve: Nada
```

```
-----------------------------------------*/

void *mod1();

/*-----------------------------------------
       Función que se encarga de implementar
       las funcionalidades del módulo 2

       Recibe: Nada
       Devuelve: Nada
-----------------------------------------*/
void *mod2();

/*-----------------------------------------
       Función que se encarga de implementar
       las funcionalidades del módulo 3

       Recibe: Nada
       Devuelve: Nada
-----------------------------------------*/
void *mod3 ();

/*-----------------------------------------
       Función que se encarga de implementar
       las funcionalidades del módulo 4

       Recibe: Nada
       Devuelve: Nada
-----------------------------------------*/
void *mod4 ();
#endif
```

Func_aux.h es la librería propia del fichero *func_aux.c* donde se recogen las funciones comúnes a todos los módulos.

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 12-02-2017
Descripción: Fichero que contiene las declaraciones
de las funciones auxiliares y constantes
necesarias para que estas funcionen
-----------------------------------------*/
#include "constantes_y_librerias.h"

#ifndef func_aux
#define func_aux

//Estructura para almacenar el contenido leído de un fichero
typedef struct CONTENIDO_FICHERO{

       //Variable auxiliar para almacenar cuantas palabras tiene cada frase
       int palabras_por_frase[NUM_FRASES];

       //Variable para contabilizar el número de frases del fichero
       int num_frases_fichero;

       //Array para almacenar el contenido del documento
              //NUM_FRASES líneas y NUM_PALABRAS_POR_FRASE palabras por líneas
       char *contenido_leido_del_fichero[NUM_FRASES][NUM_PALABRAS_POR_FRASE];

} contenido_fichero;


//Estructura para almacenar la información relativa a una regla
typedef struct ESTRUCTURA_REGLA {

       //Array para almacenar la dirección IP
       char *dir_IP[NUM_REGLAS];
```

```
	/*-------------------------------------
		Bandera para conocer si se la
		dirección IP lleva asociada un puerto
		o no
			--true: La dir_IP lleva un puerto
			-- false: No lo lleva
	-------------------------------------*/
	bool dir_Con_Puerto[NUM_REGLAS];
	//Array para almacenar el número de puerto
	char *puerto[NUM_REGLAS];
	//Array de string para almacenar el protocolo
	char *protocolo[NUM_REGLAS];
	/*-------------------------------------
		Bandera para conocer si la dirección
		IP se encontraba en el campo de origen
		o en el de destino
			-- true: se encontraba en origen
			-- false: se encontraba en destino
	-------------------------------------*/
	bool dir_en_origen[NUM_REGLAS];
	//Array para almacenar la acción que queremos ejecutar
	char *accion[NUM_REGLAS];
	//Variable para almacenar el número total de coincidencias
	int numero_lineas;

}estructura_regla;



/* ----------------------------------------------------------
	Función que se encarga de la lectura de los ficheros

	Recibe: El nombre del fichero que queremos leer
	Devuelve: Una estructura del tipo CONTENIDO_FICHERO
-----------------------------------------------------------*/
struct CONTENIDO_FICHERO lee_fichero (char *nombre_fichero);




/* ----------------------------------------------------------
	Función que se encarga de crear las reglas a partir de la
	estructura y la acción a tomar,

	Recibe: El nombre del fichero donde estan  almacenadas las
		reglas , una estructura del tipo ESTRUCTURA_REGLA
		y un string para añadir información

	Devuelve: Un booleano indicando si se ha escrito la
	regla o no
-----------------------------------------------------------*/
bool   crea_y_escribe_regla(char   *nombre_fichero_escritura,   struct   ESTRUCTURA_REGLA
contenido_fichero_alerta,char *info_extra);




/* ----------------------------------------------------------
	Función que se encarga de buscar en el fichero
	de reglas donde se quiera insertar si existe
	una igual para evitar introducir reglas duplicadas

	Recibe: Estructura del tipo CONTENIDO_FICHERO,
	estructura del tipo ESTRUCTURA_REGLA, y la posición
	dentro de la estructura ESTRUCTURA_REGLA donde se
	encuentra la info relativa a la nueva regla

	Devuelve: Un entero
		-- >0: Ya hay una regla igual. Es la línea que
```

```
              ocupa en el fichero donde están las reglas
              -- -1: No la hay

-------------------------------------------------------------*/
int    busca_Regla(struct    CONTENIDO_FICHERO    contenido_del_fichero_reglas,    struct
ESTRUCTURA_REGLA contenido_fichero_alerta,
       int pos_dentro_estruct_regla);


/* -----------------------------------------------------
       Función para que Snort vuelva a leer las reglas
       nuevas que se han introducido

       Recibe: Nada
       Devuelve: Nada
-------------------------------------------------------------*/
void recarga_Snort ();


/*-----------------------------------------------------
       Función que se encarga de vaciar un fichero
       que está en el límite de líneas que puede
       leer la función que se encarga de ello

       Devuelve: string con el nombre del fichero creado

       Recibe: El nombre del fichero que se desea
              vaciar
-----------------------------------------------------*/
char *vacia_fichero(char *nombre_fichero);


/*-----------------------------------------------------------
       Función que se encarga de eliminar una regla
       del fichero de reglas de Snort

       Recibe: El fichero del que se quiere eliminar
       las reglas, la estructura del tipo ESTRUCTURA_REGLA
       que contiene la información de la regla que se
       desea borrar y la posición dentro de la
       estructura

       Devuelve: Nada
-----------------------------------------------------*/
void      elimina_Regla(char      *nombre_fichero_borrar,      struct      ESTRUCTURA_REGLA
informacion_regla, int pos_dentro_estruct_regla);



/*-----------------------------------------------------
       Función que se encarga de crear un semáforo

       Recibe: Nombre del semaforo a crear
       Devuelve: Nada
-----------------------------------------------------*/
void crea_semaforo(char *nombre_semaforo);



/*-------------------------------------------
       Función que se encarga de eliminar un
       semáforo que fue creado previamente

       Recibe: El nombre del semáforo a eliminar
       Devuelve: Nada
-------------------------------------------*/
void elimina_semaforo(char *nombre_semaforo);

/*-------------------------------------------
```

```
	Función que se encarga de registrar en
	el fichero de logs del WIPS las incidencias
	que aparecen

	Recibe: Un string indicando la causa del log
	Devuelve: Nada
-----------------------------------------------*/
void registra_log(char *log);
#endif
```

Y *func_aux.c*

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 12-02-2017
Descripción: Fichero que contiene las
implementaciones de las funciones auxiliares
-----------------------------------------*/

#include "func_aux.h"
//Libería para poder usar la función "umask()"
#include <sys/stat.h>

/* ----------------------------------------------------------
	Función que se encarga de la lectura de los ficheros

	Recibe: El nombre del fichero a leer

	Devuelve: Una estructura con
		-- Número de frases leidas
		-- Número de palabras en cada frase
		-- Array doble que almacena cada palabra de cada frase

	Características:
		El fichero no debe contener más de NUM_FRASES
		formada por no más de NUM_PALABRAS_POR_FRASE
		cada palabra de las frase no debe tener más de
		NUM_CARACTERES_PALABRA, estos parámetros están definidos
		en la estructura
------------------------------------------------------------*/

struct CONTENIDO_FICHERO lee_fichero (char *nombre_fichero){

	//Variable donde almacenaremos el fichero que queremos leer
	FILE *fichero_lectura;
	//Variable para leer el fichero caracter a carater
	signed char c;
	//Variable para almacenar el valor del caracter anterior
	signed char c_ant;

	//Variable para contabilizar el número de frases del fichero
	int frase;
	//Variable para contabilizar en que palabra estamos
	int palabra;
	//Variable para contabilizar el número de letras
	int letra;
	//Bandera para bloquear el aumento de frases si estamos leyendo un salto de parrafo
	bool flag;
	//Bandera para evitar que se lean más líneas que el tamaño de la tabla
	bool sobrepasa_tamanio;
	//Variable para almacenar la dirección del semáforo al abrirlo
	sem_t *semaforo_lectura;

	//Array auxiliar para almacenar cada palabra de la frase
		//NUM_CARACTERES_PALABRA caracteres por palabras
	char *palabras_frase = (char *)malloc (sizeof(char)*NUM_CARACTERES_PALABRA);
	//Cadena para almacenar el informe de log
```

```
	char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

	//Estructura que será devuelta
	struct CONTENIDO_FICHERO contenido_del_fichero;

	/*-----------------------------------
		Abrimos el semáforo pasandole
		los siguientes argumentos a la
		función:
			-- SEM_LECTURA: Nombre del
				semáforo
			-- INICIO: Bandera para
			indicar que el semáforo ya
			existe(valor=0)
	-----------------------------------*/
	semaforo_lectura = sem_open(SEM_LECTURA, INICIO);

	//Si lo hemos abierto correctamente
	if (semaforo_lectura != NULL){

		//Bajamos el semáforo
		sem_wait(semaforo_lectura);

		//Inicialización de variables
		fichero_lectura = fopen(nombre_fichero, "r");
		frase = INICIO;
		c_ant = ESPACIO;
		palabra = INICIO;
		letra = INICIO;
		flag = false;
		sobrepasa_tamanio = false;

		//Si hemos abierto el fichero correctamente
		if (fichero_lectura != NULL){


			//Comenzamos a leer el fichero
			while ((c=fgetc(fichero_lectura))!=EOF && sobrepasa_tamanio==false){

				//Comprobamos que no sobrepasamos el número máximo de líneas
que se pueden leer
				if (frase < NUM_FRASES){

					//Comprobamos que el caracter que estamos leyendo es
útil
					if (c!= ESPACIO && c!=SALTO_DE_LINEA && c!= TABULADOR){

						//Almacenamos el caracter leído
						palabras_frase[letra] = c;
						//Aumentamos el número de letras leídas
						letra++;
					}

					//Comprobamos si el caracter leído anteriormente y el
actual son un salto de linea
					else if (c_ant == SALTO_DE_LINEA && c ==
SALTO_DE_LINEA){

						//Si lo es, estamos en un nuevo parrafo
						flag=true;
						palabra=INICIO;

					}

					//Comprobamos si se ha leído alguna letra,si esta es un
salto de línea y no estamos en el salto de parrafo
					else if (c == SALTO_DE_LINEA && letra>0 && flag!=
true){
```

```
                                        //Añadimos un terminador
                                        palabras_frase[letra++] = TERMINADOR;
                                        //Almacenamos la palabra de la frase deseada en
nuestra tabla

       contenido_del_fichero.contenido_leido_del_fichero[frase][palabra]                =
strdup(palabras_frase);

                                        //Volvemos a iniciar el número de letras ya que
estamos en una nueva frase y palabra
                                        letra = INICIO;

                                        //Almacenamos el número de palabras que tenía la
frase que hemos terminado de leer
                                        contenido_del_fichero.palabras_por_frase[frase]
= palabra;

                                        //Reiniciamos el contador de palabras para la
próxima iteración
                                        palabra=INICIO;

                                        //En ese caso aumentamos el número de frases
leidas
                                        frase++;

                                }

                                //Comprobamos si el caracter es un espacioo tabulador y
si se ha leído alguno
                                else if ((c == ESPACIO || c == TABULADOR) && letra >0
){

                                        //Añadimos el terminador
                                        palabras_frase[letra++] = TERMINADOR;
                                        //Almacenamos la palabra de la frase deseada en
nuestra estructura

       contenido_del_fichero.contenido_leido_del_fichero[frase][palabra]                =
strdup(palabras_frase);

                                        //Aumentamos el número de palabras leídas
                                        palabra++;
                                        //Volvemos a iniciar el número de letras leídas
ya que estamos en una nueva palabra
                                        letra=INICIO;
                                }

                                //Almacenamos el caracter que acabamos de leer para la
próxima iteración
                                c_ant=c;;
                                flag=false;
                        }

                        //En caso contrario
                        else {

                                //Cambiamos el valor de la bandera para abandonar el
bucle
                                sobrepasa_tamanio = true;

                                //Creamos la cadena para indicar la incidencia
                                sprintf(log,      ERROR_NUM_LINEAS,      nombre_fichero,
NUM_FRASES);

                                //Registramos la incidencia en el fichero
                                registra_log(log);
                        }
                }
```

```
                        //Cerramos el fichero
                        fclose(fichero_lectura);
                }

                //Si no se ha podido abrir el fichero correctamente
                else {

                        //Creamos una cadena para notificar dicha incidencia
                        sprintf(log, ERROR_APERTURA_FICHERO, nombre_fichero);

                        //Registramos la incidencia en el fichero de logs
                        registra_log(log);
                }


                //Subimos el semáforo
                sem_post(semaforo_lectura);

                //Cerramos el semáforo
                sem_close(semaforo_lectura);

                //Rellenamos la estructura para poder usarla en otro punto del codigo
                contenido_del_fichero.num_frases_fichero = frase;
        }

        //Si hemos obtenido un error al abrir el semáforo
        else {

                //Creamos la cadena para notificar la incidencia
                sprintf(log, ERROR_APERTURA_SEMAFORO, SEM_LECTURA);

                //Registramos dicha incidencia en el fichero de logs
                registra_log(log);
        }

        //Liberamos la memoria
        free(palabras_frase);
        free(log);

        //Devolvemos la estructura
        return contenido_del_fichero;
}



/*-------------------------------------------------------
        Función que se encarga de buscar en el fichero
        de reglas donde se quiera insertar si existe
        una igual para evitar introducir reglas duplicadas

        Recibe: Estructura del tipo CONTENIDO_FICHERO,
        estructura del tipo ESTRUCTURA_REGLA,
        y la posición dentro de la estructura ESTRUCTURA_REGLA
        donde se encuentra leyendo la info relativa a la nueva
        regla

        Devuelve: Un entero
                -->0: Ya hay una regla igual. Es la línea que
                ocupa en el fichero donde están las reglas
                -- -1: No la hay

        Caracteristicas:
                Mediante un bucle, recorre el fichero de las
                reglas y va comprobando una a una si la
                info que tiene almacenada para crear la nueva
                regla coincide con alguna ya creada, para ello,
                va avanzando campo a campo de cada regla hasta
                llegar al puerto destino, la info opcional
```

```
            no se evalua ya que esta puede ser muy
            cambiante
-----------------------------------------------------*/

int    busca_Regla(struct    CONTENIDO_FICHERO    contenido_del_fichero_reglas,    struct
ESTRUCTURA_REGLA contenido_fichero_alerta,
       int pos_dentro_estruct_regla) {


       //Bandera para detectar la coincidencia en una regla
       int coincide;
       //Variable auxiliar para recorrer la tabla devuelta al leer el fichero
       int cont_aux_linea;

       //Inicializamos las variables
       cont_aux_linea = INICIO;
       coincide = NO_COINCIDE;


       //Recorremos la estructura leida del fichero de reglas para buscar coincidencias
con la que vamos a introducir
       //Estaremos aquí mientras que no se haya encontrado ninguna coincidencia
       for (cont_aux_linea=0;
                     cont_aux_linea<contenido_del_fichero_reglas.num_frases_fichero     &&
coincide == NO_COINCIDE;
                     cont_aux_linea++){

              //Comprobamos si los valores a introducir coinciden con los que ya están
              /*-----------------------------------------------------------------
                     Aprovechamos la estructura de las reglas que es la siguiente:
                     accion protocolo dir_IP_orig puerto_orig ->
                            dir_IP_dest puerto_dest

                     Estos valores serán los que comprobaremos
              -----------------------------------------------------------------*/

              //Comprobamos conjuntamente si la pareja acción protocolo son iguales
              if       ((strcmp(contenido_fichero_alerta.accion[pos_dentro_estruct_regla],
contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_ACCION])   ==
IGUAL) &&
                     (strcmp(contenido_fichero_alerta.protocolo[pos_dentro_estruct_regla],

       contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_PROTOC
OLO]) == IGUAL )){

                            //Comprobamos si la dirección IP estaba en el campo origen
                            if
(contenido_fichero_alerta.dir_en_origen[pos_dentro_estruct_regla] == true){

                                   //En ese caso, comprobamos que coincida la dir_IP con
la origen de la regla y la destino con "$HOME_NET"
                                   if                                          (strcmp
(contenido_fichero_alerta.dir_IP[pos_dentro_estruct_regla],

       contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_DIR_IP
_ORIG]) == IGUAL &&
                                          strcmp("$HOME_NET",
contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_DIR_IP_DEST]
)== IGUAL){

                                          //Comprobamos  si  la  dir_IP  tenía  asociada  un
puerto
                                          if
(contenido_fichero_alerta.dir_Con_Puerto[pos_dentro_estruct_regla] == true){

                                                 //Si  es  así  comprobamos  si  el  puerto
coincide
                                                 if                                (strcmp
(contenido_fichero_alerta.puerto[pos_dentro_estruct_regla],
```

```
	contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_PUERTO
_ORIG]) == IGUAL ){

                                                        //Si todo lo anterior coincide, la
regla ya existía
                                                        coincide = cont_aux_linea;
                                                }

                                                else {
                                                        //En caso contrario, estamos ante
una nueva regla
                                                        coincide = NO_COINCIDE;
                                                }
                                        }

                                        //Si no lleva asociado un puerto
                                        else {

                                                //Si  no  tenía  asociada  ningún  puerto,
comprobamos que es "any"
                                                if (strcmp("any",

	contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_PUERTO
_ORIG]) == IGUAL) {

                                                        //Si  coincide,  estamos  ante  una
regla ya existente
                                                        coincide = cont_aux_linea;

                                                }

                                                else {

                                                        //En caso de que no coincida, es
una nueva regla
                                                        coincide = NO_COINCIDE;

                                                }
                                        }
                                }

                                else {

                                        //En  caso  contrario,  estamos  ante  una  nueva
regla
                                        coincide = NO_COINCIDE;

                                }

                        }

                        //Si la dirección IP no estaba en el campo origen
                        else {

                                //Comprobamos que el campo origen sea "$HOME_NET" y el
destno la dir_IP de la regla que queremos crear
                                if                                      (strcmp("$HOME_NET",
contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_DIR_IP_ORIG]
) == IGUAL &&
                                        strcmp
(contenido_fichero_alerta.dir_IP[pos_dentro_estruct_regla],

	contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_DIR_IP
_DEST]) == IGUAL ){

                                        //Comprobamos si esa dirección llevaba asociada
un puerto
```

```
                                        if
(contenido_fichero_alerta.dir_Con_Puerto[pos_dentro_estruct_regla] == true){

                                                //Si lo llevaba asociado, comprobamos si
el puerto coincide
                                                if                                      (strcmp
(contenido_fichero_alerta.puerto[pos_dentro_estruct_regla],

      contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_PUERTO
_DEST]) == IGUAL ){

                                                        //En ese caso estamos ante una
regla creada anteriormente
                                                        coincide = cont_aux_linea;

                                                }

                                                //En otro caso, estamos ante una nueva
regla
                                                else {

                                                        coincide = NO_COINCIDE;

                                                }
                                        }

                                        //Si   no   llevaba   puerto   asociado   seguimos
comprobando
                                        else {

                                                //Comprobamos que el puerto destino es
"any"
                                                if (strcmp("any",

      contenido_del_fichero_reglas.contenido_leido_del_fichero[cont_aux_linea][POS_PUERTO
_DEST]) == IGUAL){

                                                        //Estamos ante una regla ya creada
si todo lo anterior es igual
                                                        coincide = cont_aux_linea;

                                                }

                                                else {

                                                        //Estamos ante una nueva regla
                                                        coincide = NO_COINCIDE;

                                                }

                                        }

                                }

                                else {
                                        //En otro caso, es una nueva regla
                                        coincide = NO_COINCIDE;

                                }
                        }

                }

                else {

                        //En caso contrario estamos ante una nueva regla
                        coincide = NO_COINCIDE;

                }
```

```
	}

	//Devolvemos el valor
	return coincide;
}


/*-------------------------------------------------------------
	Función que se encarga de crear las reglas a partir de la
	estructura y la acción a tomar.

	Recibe: El nombre del fichero donde estan  almacenadas las
		reglas , una estructura del tipo ESTRUCTURA_REGLA
		y un string para añadir informacion a la regla

	Devuelve: Booleano indicando si se ha escrito la regla

	Caracteristicas:
		Lo primero que realiza la función es comprobar que
		la regla que queremos crear no está ya introducida
		en el fichero de esta forma evitamos tener reglas
		duplicadas. Tras esto, la crea y la escribe en el
		fichero. Para crear las reglas es necesario un
		identificador único (sid) que para guardar siempre
		una referencia, este se almacena en otro fichero
		distinto
-------------------------------------------------------------*/

bool	crea_y_escribe_regla(char	*nombre_fichero_escritura,	struct	ESTRUCTURA_REGLA
contenido_fichero_alerta, char *info_extra){

	//Variable a devolver que indica si se ha creado la regla
	bool regla_creada;
	//Variable para el fichero donde almacenaremos el sid
	FILE *fichero_sid;
	//Variable auxiliar para la lectura del fichero donde almacenaremos el sid
	signed char c;
	//Array donde almacenaremos el número de sid leido
	char sid[NUM_DIG_SID];
	//Variable auxiliar para indicar la posición de la tabla donde nos encontramos
	int pos_sid;

	//Variable	para	almacenar	el	fichero	de	las	reglas,	de	donde	leeremos	y
escribiremos
	FILE *fichero_escritura;
	//Estructura que almacenará los datos relativos al fichero de las reglas
	struct CONTENIDO_FICHERO contenido_del_fichero_reglas;
	//Cadena para almacenar el informe de log
	char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

	//String auxiliar para ir formando la regla
	char *reglas_aux= (char *) malloc (sizeof (char)*NUM_CARACTERES_REGLAS);
	//Variable auxiliar para recorre el bucle
	int cont_aux_regla;
	//Bandera para detectar la coincidencia en una regla
	int coincide;

	//Variable para almacenar la dirección del semáforo al abrirlo
	sem_t *semaforo_reglas_snort;
	sem_t *semaforo_sid;


	//Abrimos los semáforos
	semaforo_reglas_snort = sem_open(SEM_REGLAS_SNORT, INICIO);
	semaforo_sid = sem_open(SEM_SID, INICIO);

	//Inicializamos
	cont_aux_regla = INICIO;
```

```
	coincide = INICIO;
	regla_creada = false;

	//Comprobamos que se han abierto correctamente
	if (semaforo_reglas_snort != NULL && semaforo_sid != NULL) {

		//Recorremos la estructura de coincidencias que le pasasmos como parámetro
		for (cont_aux_regla=0;
cont_aux_regla<contenido_fichero_alerta.numero_lineas; cont_aux_regla++) {

			//Leemos el fichero de reglas ya creadas y almacenamos la info en la
estructura
			contenido_del_fichero_reglas = lee_fichero(nombre_fichero_escritura);

			//Comprobamos si la regla que queremos crear está ya en el fichero o
no
			coincide = busca_Regla(contenido_del_fichero_reglas,
contenido_fichero_alerta, cont_aux_regla);


			//En el caso en el que no encontramos ninguna coincidencia en el
fichero
			if (coincide == NO_COINCIDE){

				//Comenzamos añadiendo la acción que queremos ejecutar en esa
regla

	strcpy(reglas_aux,contenido_fichero_alerta.accion[cont_aux_regla]);
				strcat(reglas_aux, " ");
				//Añadimos el protocolo
				strcat(reglas_aux,
contenido_fichero_alerta.protocolo[cont_aux_regla]);
				strcat(reglas_aux, " ");

				//Comprobamos si cuando detectamos la alerta la dirección
estaba en origen o en destino
				if (contenido_fichero_alerta.dir_en_origen[cont_aux_regla] ==
true) {

					//En ese caso añadimos la dirección al campo origen
					strcat(reglas_aux,
contenido_fichero_alerta.dir_IP[cont_aux_regla]);
					strcat(reglas_aux, " ");

					//Comprobamos si dicha dirección IP llevaba asociada un
puerto
					if
(contenido_fichero_alerta.dir_Con_Puerto[cont_aux_regla] == true){

						//Si lo llevaba, almacenamos el puerto en la
regla
						strcat(reglas_aux,
contenido_fichero_alerta.puerto[cont_aux_regla]);

					}

					else {

						//En caso contrario, añadimos el valor "any"
						strcat(reglas_aux, "any");
					}
				}

				//Si la dir_IP no estaba en el campo origen añadimos los
valores por defecto
				else{

					strcat(reglas_aux, "$HOME_NET");
					strcat(reglas_aux, " ");
```

```
                                strcat(reglas_aux, "any");

                        }

                        //Añadimos el sentido de la comunicación
                        strcat(reglas_aux, " ");
                        strcat(reglas_aux, "->");
                        strcat(reglas_aux, " ");

                        //Realizamos la misma comprobación que antes
                        if  (contenido_fichero_alerta.dir_en_origen[cont_aux_regla]  ==
false) {

                                strcat(reglas_aux,
contenido_fichero_alerta.dir_IP[cont_aux_regla]);
                                strcat(reglas_aux, " ");

                                if
(contenido_fichero_alerta.dir_Con_Puerto[cont_aux_regla] == true){

                                        strcat(reglas_aux,
contenido_fichero_alerta.puerto[cont_aux_regla]);
                                }
                                else {

                                        strcat(reglas_aux, "any");
                                }
                        }

                        else{

                                strcat(reglas_aux, "$HOME_NET");
                                strcat(reglas_aux, " ");
                                strcat(reglas_aux, "any");
                        }

                        strcat(reglas_aux, " ");

                        //Añadimos los campos opcionales como mensaje de información
                        /*---------------------------------------------
                                El mensaje de info tendrá la estructura de:
                                        info_extra + accion + protocolo
                        ---------------------------------------------*/
                        strcat(reglas_aux, "(msg:\"");
                        strcat(reglas_aux, info_extra);
                        strcat(reglas_aux, ": ");

      strcat(reglas_aux,contenido_fichero_alerta.accion[cont_aux_regla]);
                        strcat(reglas_aux, " ");
                        strcat(reglas_aux,
contenido_fichero_alerta.protocolo[cont_aux_regla]);
                        strcat(reglas_aux, "\"; sid:");

                        //Abrimos el fichero donde almacenamos el sid
                        /*---------------------------------------------
                                El sid lo debemos almacenar en un fichero
                                ya que debe ser único dentro de local.rules
                                de esta forma siempre tenemos la referencia
                                del último que hemos usado para la última
                                regla introducida
                        ---------------------------------------------*/
                        //Bajamos el semáforo
                        sem_wait(semaforo_sid);

                        //Leemos el fichero
                        fichero_sid = fopen(FICHERO_SID, "r");

                        //Comprobamos que hemos abierto el fichero correctamente
                        if (fichero_sid != NULL){
```

```
                        //Reiniciamos la posición para recorrer el string
                        pos_sid = INICIO;

                        //Leemos el fichero
                        while ((c=fgetc(fichero_sid))!=EOF){

                                //Comprobamos que no estamos leyendo un '\n'
                                if (c != SALTO_DE_LINEA){

                                        //Almacenamos el número
                                        sid[pos_sid] = c;
                                        pos_sid++;
                                }
                        }

                        //Aumentamos en uno el número que estaba almacenado en
el fichero
                        sprintf(sid, "%d", atoi(sid)+1);

                        //Cerramos el fichero
                        fclose(fichero_sid);

                }

                //En caso de error
                else{

                        //Creamos la cadena para notificar de la incidencia
                        sprintf(log, ERROR_APERTURA_FICHERO, FICHERO_SID);

                        //Registramos la incidencia en el fichero de logs
                        registra_log(log);
                }


                //Lo volvemos a crear y almacenamos el nuevo número
                fichero_sid = fopen(FICHERO_SID, "w");

                //Comprobamos que hemos abierto correctamente el fichero
                if (fichero_sid != NULL){
                        fputs(sid, fichero_sid);
                        fclose(fichero_sid);
                }

                //En caso de error
                else{

                        //Creamos la cadena para notificar de la incidencia
                        sprintf(log, ERROR_APERTURA_FICHERO, FICHERO_SID);

                        //Registramos la incidencia en el fichero de logs
                        registra_log(log);
                }

                //Subimos el semáforo
                sem_post(semaforo_sid);


                //Almacenamos este sid en la regla que estamos creando
                strcat(reglas_aux, sid);

                //Añadimos el número de revisión, en nuestro caso será siempre
1
                strcat(reglas_aux, "; rev:1;)");
                strcat(reglas_aux, "\n");

                //Bajamos el semáforo
                sem_wait(semaforo_reglas_snort);
```

```
                        //Abrimos el fichero de las reglas y nos situamos al final de
este
                        fichero_escritura = fopen(nombre_fichero_escritura, "a+");

                        if (fichero_escritura != NULL){

                               //Escribimos la regla que acabamos de crear
                               fputc('\n', fichero_escritura);
                               fputs(reglas_aux, fichero_escritura);

                               //Cerramos el fichero
                               fclose(fichero_escritura);

                               //Indicamos que hemos escrito la regla correctamente
                               regla_creada = true;
                        }

                        //En caso de error
                        else{

                               //Creamos la cadena para notificar de la incidencia
                               sprintf(log,                    ERROR_APERTURA_FICHERO,
nombre_fichero_escritura);

                               //Registramos la incidencia en el fichero de logs
                               registra_log(log);
                        }

                        //Subimos el semáforo
                        sem_post(semaforo_reglas_snort);


                }
        }

        //Cerramos el semáforo
        sem_close(semaforo_reglas_snort);
        //Cerramos el semáforo
        sem_close(semaforo_sid);

    }

    //En caso contrario
    else {


        //Comprobamos cuál es el que no se ha abierto correctamente
        if (semaforo_reglas_snort != NULL) {

               //Cerramos el que se abrió correctamente
               sem_close(semaforo_reglas_snort);

               //Informamos mediante un log del que no
               sprintf(log, ERROR_APERTURA_SEMAFORO, SEM_SID);
               registra_log(log);

        }

        else if (semaforo_sid != NULL) {

               sem_close(semaforo_sid);

               sprintf(log, ERROR_APERTURA_SEMAFORO, SEM_REGLAS_SNORT);
               registra_log(log);
        }

        //En el caso en el que ninguno se haya abierto correctamente
        else {
```

```
                    //Notificamos que no se pudo abrir ninguno
                    sprintf(log,        ERROR_APERTURA_VARIOS_SEMAFOROS,        SEM_SID,
SEM_REGLAS_SNORT);
                    registra_log(log);
             }
      }

      //Liberamos memoria
      free(reglas_aux);
      free(log);

//Devolvemos la bandera
return regla_creada;
}



/*-----------------------------------------
      Función para que Snort vuelva a leer
      las reglas nuevas que se han introducido

      Recibe: Nada
      Devuelve: Nada

      Caracteristicas:
            Obtiene el PID del proceso que almacena
            en un fichero, posteriormente ese
            fichero se lee y se ejecuta la orden
            de recarga para ese PID
-----------------------------------------*/
void recarga_Snort () {

      //Variable auxiliar para almacenar el PID
      char *pid = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
      //Variable auxiliar para almacenar el nombre del fichero
      char *nombre_fichero_pid = (char * )malloc(sizeof(char *)*NUM_CARACTERES_PALABRA);
      //Variable auxiliar para almacenar el comando que vamos a ejecutar
      char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
      //Cadena para almacenar el informe de log
      char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

      //Estructura que almacenará los datos relativos al fichero
      struct CONTENIDO_FICHERO contenido_del_fichero;
      //Variable auxiliar para saber en la línea en la que nos encontramos
      int cont_aux_linea;
      //Variable auxiliar para saber en que palabra nos encontramos
      int cont_aux_palara;
      //Variable para almacenar la dirección del semáforo al abrirlo
      sem_t *semaforo_snort;

      /*-------------------------------------
            Abrimos el semáforo pasandole
            los siguientes argumentos a la
            función:
                  -- SEM_SNORT: Nombre del
                        semáforo
                  -- INICIO: Bandera para
                  indicar que el semáforo ya
                  existe(valor=0)
      -------------------------------------*/
      semaforo_snort = sem_open(SEM_SNORT, INICIO);

      //Comprobamos que hemos abierto correctamente el semáforo
      if (semaforo_snort != NULL) {

            //Bajamos el semáforo
            sem_wait(semaforo_snort);
```

```
		//Inicializamos las variables
		strcpy(nombre_fichero_pid, FICHERO_PID_SNORT);
		cont_aux_linea = INICIO;
		cont_aux_palara = INICIO;

		//Creamos el comando para obtener el PID y redireccionarlo al fichero
		strcpy(comando, "sudo pidof snort > ");
		strcat(comando, nombre_fichero_pid);
		//Ejecutamos dicho comando
		system(comando);

		//Leemos el fichero
		contenido_del_fichero = lee_fichero(nombre_fichero_pid);

		//Recorremos el array devuelto para obtener el PID
		for (cont_aux_linea=0;
				cont_aux_linea < contenido_del_fichero.num_frases_fichero;
				cont_aux_linea++){

			for (cont_aux_palara=0;

	cont_aux_palara<=contenido_del_fichero.palabras_por_frase[cont_aux_linea];
				cont_aux_palara++) {

				//Almacenamos el PID en nuestra variable específica
				strcpy(pid,

	contenido_del_fichero.contenido_leido_del_fichero[cont_aux_linea][cont_aux_palara])
;

			}
		}

		//Creamos el comando para recargar Snort
		strcpy(comando, "sudo kill -SIGHUP ");
		strcat(comando, pid);
		//Ejecutamos dicho comando
		system(comando);

		//Subimos el semáforo
		sem_post(semaforo_snort);

		//Cerramos el semáforo
		sem_close(semaforo_snort);

	}
	//En caso de que uno de los semáforo no se abra correctamente
	else {

		//Creamos la cadena para notificar de la incidencia
		sprintf(log, ERROR_APERTURA_SEMAFORO, SEM_SNORT);

		//Registramos la incidencia en el fichero de logs
		registra_log(log);
	}

	//Liberamos la memoria
	free(nombre_fichero_pid);
	free(comando);
	free(pid);
	free(log);
}



/*-------------------------------------------------
	Función que se encarga de vaciar un fichero
	que está en el límite de líneas que puede
```

```
	leer la función que se encarga de ello

	Devuelve: String con el nombre del fichero creado

	Recibe: El nombre del fichero

	Caracteristicas:
		Para vaciar el fichero, crea uno nuevo con
		el contenido que se desea eliminar
		añadiendo al final del nombre la fecha en
		la que se ha realizado para tenerlo siempre
		almacenado
-------------------------------------------------*/

char *vacia_fichero(char *nombre_fichero){

	//Variable auxiliar para almacenar el instante en el que se cambia el fichero
	time_t tiempo_cambio_fichero = time(NULL);
	//Array para almacenar la fecha
	char t_cambio_fichero[TAM_FECHA];
	//Estructura para almacenar la fecha y hora en el momento deseado
	struct tm *tm;
	//Variable para almacenar el comando que vamos a ejecutar
	char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);




	//Obtenemos el instante en el que lo hemos detectado
	tm = localtime(&tiempo_cambio_fichero);

	/*-------------------------------------------------
		La estructura con la que se almacena el nuevo
		fichero es la siguiente:
			nombre_fichero-dia-mes-año_hora:min:seg
	-------------------------------------------------*/
	strftime(t_cambio_fichero, sizeof(t_cambio_fichero), "-%d-%m-%Y_%H:%M:%S", tm);

	//Almacecenamos el fichero actual en una copia para no perderlo
	strcpy(comando, "cp -f ");
	strcat(comando, nombre_fichero);
	strcat(comando, " ");

	//La copia estará formada por el nombre y la fecha anteriormente obtenida
	strcat(comando, nombre_fichero);
	strcat(comando, t_cambio_fichero);
	//Ejecutamos el comando
	system(comando);

	//Eliminamos el fichero que está apunto de sobrecargarse
	strcpy(comando, "rm -f ");
	strcat(comando, nombre_fichero);
	//Ejecutamos el comando
	system(comando);

	//Creamos uno nuevo vacío
	strcpy(comando, "touch -f ");
	strcat(comando, nombre_fichero);
	//Ejecutamos el comnando
	system(comando);

	strcpy(comando, nombre_fichero);
	strcat(comando, t_cambio_fichero);

	//Devolvemos el comando
	return comando;

	//Liberamos la memoria
	free(comando);
```

```
}


/*-------------------------------------------------------
	Función que se encarga de eliminar una regla
	del fichero de reglas de Snort

	Recibe: El fichero del que se quiere eliminar
	las reglas, la estructura del tipo ESTRUCTURA_REGLA
	que contiene la información de la regla que se
	desea borrar y la posición dentro de la
	estructura

	Devuelve: Nada

	Caracteristicas:
		En primer lugar lee el fichero del que se
		quieren borrar sus líneas para tener almacenada
		una copia, tras esto se busca la línea donde
		está la regla que queremos elminiar y tras esto
		abre el fichero vacio y lo reescribe con los
		datos que almacenó al principio saltandose la
		línea que deseamos eliminar

		Tendremos la opción de pasar directamente la
		línea que queremos eliminar, para ello,
		pasaremos la estructura vacía y en el tercer
		parámetro la línea que queremos eliminar
-------------------------------------------------------*/

void elimina_Regla(char *nombre_fichero_borrar,struct ESTRUCTURA_REGLA informacion_regla,
	int pos_dentro_estruct_regla){

	//Variable para almacenar el fichero de donde vamos a borrar
	FILE *fichero_borrar;
	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;
	//Variables para recorrer la tabla con el contenido del fichero
	int cont_aux_frases_fichero;
	int cont_aux_palabras_fichero;
	//Variable auxiliar para almacenar la regla que vamos a sobreescribir
	char *regla_aux = (char *)malloc(sizeof(char) *NUM_CARACTERES_REGLAS);
	//Cadena para almacenar el informe de log
	char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Variable para almacenar la línea en la que hemos encontrado la regla
	int linea_regla;
	//Variable para almacenar la dirección del semáforo al abrirlo
	sem_t *semaforo_reglas_snort;

	//Inicializamos las variables
	cont_aux_frases_fichero = INICIO;
	cont_aux_palabras_fichero = INICIO;
	linea_regla = INICIO;

	//Abrimos el semáforo
	semaforo_reglas_snort = sem_open(SEM_REGLAS_SNORT, INICIO);

	//Comprobamos que se ha abierto correctamente
	if (semaforo_reglas_snort != NULL) {

		//Bajamos el semáforo
		sem_wait(semaforo_reglas_snort);

		//Leemos el fichero del que queremos borrar la regla
		contenido_del_fichero = lee_fichero(nombre_fichero_borrar);

		//Comprobamos que la estructura no está vacía
		if (informacion_regla.numero_lineas > INICIO) {
```

```
                    //En ese caso, buscamo dicha regla en el fichero
                    linea_regla  =  busca_Regla(contenido_del_fichero,  informacion_regla,
pos_dentro_estruct_regla);

             }

             /*----------------------------------------
                    En caso de estar vacía, el tercer
                    parámetro nos indica la línea que
                    deseamos eliminar
             ----------------------------------------*/
             else {

                    linea_regla = pos_dentro_estruct_regla;
             }

             //Si hemos encontrado la regla
             if (linea_regla >= INICIO){

                    //Abrimos el fichero vacio para volver a escribirlo
                    fichero_borrar = fopen(nombre_fichero_borrar, "w");

                    //Comprobamos que hemos abierto correctamente el fichero
                    if (fichero_borrar != NULL) {

                           //Recorremos  el  array  que  rellenamos  cuando  lo  leimos  por
primera vez
                           for (cont_aux_frases_fichero=0;
                           cont_aux_frases_fichero                                        <
contenido_del_fichero.num_frases_fichero;
                           cont_aux_frases_fichero++){

                                  //Comprobamos que la línea que vamos a escribir no es
la que queremos eliminar
                                  if (cont_aux_frases_fichero != linea_regla) {

                                         for(cont_aux_palabras_fichero =0;
                                                cont_aux_palabras_fichero                <=
contenido_del_fichero.palabras_por_frase[cont_aux_frases_fichero];
                                                cont_aux_palabras_fichero++) {

                                                //Si estamos ante la primera palabra de la
frase
                                                if (cont_aux_palabras_fichero == 0){

                                                       //Copiamos  para  iniciar  una  nueva
regla
                                                       strcpy(regla_aux,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][cont_aux_palab
ras_fichero]);

                                                }
                                                //Si no encontramos ante otra palabra
                                                else {

                                                       //Concatenamos  las  palabras  para
seguir creando la regla
                                                       strcat(regla_aux, " ");
                                                       strcat(regla_aux,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][cont_aux_palab
ras_fichero]);

                                                }

                                         }

                                  //Añadimos un salto de línea
                                  strcat(regla_aux, "\n");
                                  //Escribimos la regla en el fichero
```

```
                                fputs(regla_aux, fichero_borrar);

                                }
                        }

                        //Cerramos el fichero
                        fclose(fichero_borrar);
                }

                //En caso de error
                        else{

                                //Creamos la cadena para notificar de la incidencia
                                sprintf(log,                    ERROR_APERTURA_FICHERO,
nombre_fichero_borrar);

                                //Registramos la incidencia en el fichero de logs
                                registra_log(log);
                        }

        }

        //Subimos el semáforo
        sem_post(semaforo_reglas_snort);

        //Cerramos el semáforo
        sem_close(semaforo_reglas_snort);
    }

    //En caso de error al abrir el semáforo
    else {

        //Creamos la cadena para indicar de la incidencia
        sprintf(log, ERROR_APERTURA_SEMAFORO, SEM_REGLAS_SNORT);

        //Registramos la incidencia en el fichero de logs
        registra_log(log);
    }

    //Liberamos memoria
    free(regla_aux);
    free(log);
}


/*-----------------------------------------------------
    Función que se encarga de crear un semáforo

    Recibe: Nombre del semaforo a crear

    Devuelve: Nada
-----------------------------------------------------*/

void crea_semaforo(char *nombre_semaforo){

    //Variable auxiliar para almacenar el valor inicial del semáforo
    int valor_sem;
    //Cadena para almacenar el informe de log
    char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
    //Variable que almacenará la dirección del semáforo una vez que se cree
    sem_t *semaforo;

    //Inicializamos
    valor_sem = INICIO_SEM;

    /*-----------------------------------------------
        Creamos el semáforo pasando los siguientes
        parámetros:
            -- nombre_semaforo: Nombre que tendrá
```

```
                        el semáforo a crear

                --O_CREAT: Indicamos que lo queremos crear

                --0666: Permisos que tendrá el semáforo

                --valor_sem: Valor inicial del semáforo
        ------------------------------------------------*/
        /*------------------------------------------------
                Cambiamos los valores por defecto de umask
                a "000" de esta forma, a los semáforos
                se le asignarán excatamente los valores
                que se le pasan en la función (0666) en este
                caso
        ------------------------------------------------*/
        umask(000);
        semaforo = sem_open(nombre_semaforo, O_CREAT, 0666, valor_sem);

        /*------------------------------------------------
                Tras crear los semáforos, volvemos a
                asignar a umask los valores por defectos
                para las próxima ocasión en las que se
                cree cualquier fichero este tenga los valores
                por defecto
        ------------------------------------------------*/
        umask(022);

        //Si hemos obtenido un error al crear el semáforo
        if (semaforo == NULL) {

                //Creamos una cadena para almacenar la incidencia
                sprintf(log, ERROR_CREACION_SEMAFORO, nombre_semaforo);

                //Almacenamos la incidencia en el fichero de logs
                registra_log(log);

        }

        //Liberamos memoria
        free(log);
}


/*------------------------------------------------
        Función que se encarga de eliminar un
        semáforo que fue creado previamente

        Recibe: El nombre del semáforo a eliminar

        Devuelve: Nada
------------------------------------------------*/
void elimina_semaforo(char *nombre_semaforo) {

        //Variable auxiliar para almacenar el valor devuelto a la hora de eliminar el
semáforo
        int valor_devuelto;
        //Cadena para almacenar el informe de log
        char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

        //Inicializamos
        valor_devuelto = INICIO;

        //Eliminamos el semáforo
        valor_devuelto = sem_unlink(nombre_semaforo);

        //Si no se eliminó correctamente
        if (valor_devuelto != INICIO) {

                //Notificamos la incidencia
```

```
                sprintf(log, ERROR_CERRAR_SEMAFORO, nombre_semaforo);
                registra_log(log);
        }

        //Liberamos memoria
        free(log);
}


/*-----------------------------------------
        Función que se encarga de registrar en
        el fichero de logs del WIPS las incidencias
        que aparecen

        Recibe: La causa del log

        Devuelve: Nada
-----------------------------------------*/
void registra_log(char *log) {

        //Variable para almacenar el fichero de donde vamos a almacenar los logs
        FILE *fichero_logs;

        //Variable auxiliar para almacenar el instante en el que se recibe el log
        time_t instante_log = time(NULL);
        //Array para almacenar la fecha
        char instante_registro_log[TAM_FECHA];
        //Estructura para almacenar la fecha y hora en el momento deseado
        struct tm *tm;


        //String para almacenar el texto que se va a registrar
        char *registro = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

        //Inicializamos
        fichero_logs = fopen(FICHERO_LOG, "a+");
        //Obtenemos el instante en el que lo hemos detectado
        tm = localtime(&instante_log);

        //Comprobamos que hemos abierto correctamente el fichero
        if (fichero_logs != NULL){

                /*-------------------------------------------------
                        La estructura con la que se registra el log
                        es la siguiente
                                dia/mes/año-hora:min:seg causa_del_registro
                -------------------------------------------------*/
                strftime(instante_registro_log,   sizeof(instante_registro_log),   "%d/%m/%Y-
%H:%M:%S", tm);

                //Formamos el texto que vamos a escribir
                strcpy(registro, instante_registro_log);
                strcat(registro, " ");
                strcat(registro, log);

                //Escribimos el texto en el fichero
                fputs(registro, fichero_logs);

                //Cerramos el fichero
                fclose(fichero_logs);
        }

        //Liberamos memoria
        free(registro);
}
```

A continuación, se detalla el código perteneciente al primer módulo.

Mod1_DoS.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 10-02-2017
Descripción: Fichero que contiene la
implementación del primer módulo centrado
en los ataques DoS
-----------------------------------------*/

#include "func_aux_mod1.h"

/*---------------------------------------
       Función que implementa el primer
       módulo

              Recibe: Nada

              Devuelve: Nada
---------------------------------------*/

void *mod1 (){


       //Estructura que almacenará los datos relativos al fichero
       struct CONTENIDO_FICHERO contenido_del_fichero;
       //Estructura que almacenará los datos relativos a las coincidencias
       struct ESTRUCTURA_REGLA contenido_fichero_alerta;
       //Variable donde almacenamos el fichero que queremos leer
       char *nombre_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
       //Variable auxiliar para almacenar el nombre del fichero que se crea
       char *nuevo_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
       //Variable auxiliar para conocer el número de líneas que leimos anteriormente
       int num_lineas_anterior;
       //Variable para conocer si hemos creado alguna regla
       bool regla_creada;


       //Inicializamos las variables
       strcpy(nombre_fichero, FICHERO_ALERT_SNORT);
       num_lineas_anterior = INICIO;
       regla_creada=false;


       while(true){

              //Leemos el fichero
              contenido_del_fichero = lee_fichero(nombre_fichero);


              //Comprobamos si el número de líneas leídas está cerca del máximo
(NUM_FRASES)
              if ( contenido_del_fichero.num_frases_fichero < LIMITE_LINEAS_LEIDAS){

                     //Comprobamos que hemos leído una línea nueva
                     if (contenido_del_fichero.num_frases_fichero > num_lineas_anterior){

                            //Buscamos alguna alerta de DoS
                            contenido_fichero_alerta                                    =
comprueba_Coincidencia_Fichero_Leido(contenido_del_fichero, "DoS", num_lineas_anterior);

                            //Si hemos detectado alguna alerta nueva
                            if (contenido_fichero_alerta.numero_lineas > INICIO){

                                   //Creamos las reglas con las coincidencias que hayamos
encontrado
                                   regla_creada                                         =
crea_y_escribe_regla(FICHERO_REGLAS_SNORT, contenido_fichero_alerta, "DOS");
```

```
                                    //Comprobamos si hemos escrito alguna regla
                                    if (regla_creada==true){

                                          //Cambiamos el valor de la bandera
                                          regla_creada = false;

                                          //Recargamos  Snort  para  hacer  efectivo  los
cambios
                                          recarga_Snort();
                                    }
                              }
                        }

                  }

                  //En el caso en el que estemos cerca del límite
                  else {

                        //Llamamos a la función deseada para que nos vacie el fichero
                        nuevo_fichero = vacia_fichero(nombre_fichero);
                        /*-------------------------------------------------------
                              Para asegurarnos que no perdemos información
                              en ningún momento, volvemos a leer el fichero
                              al que le hemos cambiado el nombre siempre y
                              cuando no supere el número máximo (NUM_FRASES)

                              Esta comprobación la hacemos por si el fichero
                              se encontraba en el rango entre LIMITE y NUM_FRASES
                              y hay información que se pueda perder

                        -------------------------------------------------------*/
                        contenido_del_fichero = lee_fichero(nuevo_fichero);



                        //Comprobamos que ese fichero no ha superado el límite
                        if (contenido_del_fichero.num_frases_fichero < NUM_FRASES){

                              //Buscamos alguna alerta de DoS
                              contenido_fichero_alerta                                    =
comprueba_Coincidencia_Fichero_Leido(contenido_del_fichero, "DoS", num_lineas_anterior);

                              //Si hemos detectano alguna alerta nueva
                              if (contenido_fichero_alerta.numero_lineas > INICIO){

                                    //Creamos las reglas con las que hayamos encontrado
                                    regla_creada                                          =
crea_y_escribe_regla(FICHERO_REGLAS_SNORT, contenido_fichero_alerta, "DOS");

                                    //Comprobamos si hemos creado la regla
                                    if (regla_creada == true){

                                          //Cambiamos el valor de la bandera
                                          regla_creada = false;

                                          //Recargamos Snort
                                          recarga_Snort();
                                    }
                              }

                        }

                        //Inicializamos el número de líneas leídas para la próxima iteración
                        contenido_del_fichero.num_frases_fichero = INICIO;
                  }

                  //Almacenamos el número de líneas leídas anteriormente
                  num_lineas_anterior = contenido_del_fichero.num_frases_fichero;
```

```
		//Esperamos un tiempo para volver a leer el fichero
		sleep(INTERVALO_LECTURA);
	}

	//Liberamos la memoria
	free(nombre_fichero);
	free(nuevo_fichero);

	pthread_exit(NULL);
}
```

Func_aux_mod1.h

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 13-03-2017
Descripción: Fichero que contiene las declaraciones
de las funciones auxiliares del primer módulo
-----------------------------------------*/
#include "constantes_y_librerias.h"
#include "func_aux.h"


#ifndef func_aux_mod1
#define func_aux_mod1

/* ------------------------------------------------------
	Función que para un string dado comprueba si dicho
	string está en el contenido leido de un fichero

	Recibe: Una estructura del tipo CONTENIDO_FICHERO
	el nombre que vamos a buscar y la línea por la
	que debe empezar a buscar

	Devuelve: Estructura del tipo ESTRUCTURA_REGLA
------------------------------------------------------------*/
struct	ESTRUCTURA_REGLA	comprueba_Coincidencia_Fichero_Leido(struct	CONTENIDO_FICHERO
contenido_del_fichero,
	char * nombre_Coincidencia, int linea_inicio);

/* ------------------------------------------------------
	Función que se encarga de determinar si una IP pertenece
	a una red local o no

	Recibe: Un string que contiene la dirección IP
	Devuelve: Un booleano indicando si pertenece a la red local
------------------------------------------------------------*/
bool comprueba_IP(char *direccion_IP);




#endif
```

Func_aux_mod1.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 13-03-2017
Descripción: Fichero que contiene las declaraciones
de las funciones auxiliares del primer módulo
-----------------------------------------*/
#include "constantes_y_librerias.h"
#include "func_aux.h"
```

```
#ifndef func_aux_mod1
#define func_aux_mod1

/* ----------------------------------------------------
       Función que para un string dado comprueba si dicho
       string está en el contenido leido de un fichero

       Recibe: Una estructura del tipo CONTENIDO_FICHERO
       el nombre que vamos a buscar y la línea por la
       que debe empezar a buscar

       Devuelve: Estructura del tipo ESTRUCTURA_REGLA
------------------------------------------------------------*/
struct   ESTRUCTURA_REGLA   comprueba_Coincidencia_Fichero_Leido(struct   CONTENIDO_FICHERO
contenido_del_fichero,
       char * nombre_Coincidencia, int linea_inicio);

/* ----------------------------------------------------
       Función que se encarga de determinar si una IP pertenece
       a una red local o no

       Recibe: Un string que contiene la dirección IP
       Devuelve: Un booleano indicando si pertenece a la red local
------------------------------------------------------------*/
bool comprueba_IP(char *direccion_IP);



#endif
```

El código correspondiente al segundo módulo es el siguiente;

Mod2_MITM.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 20-02-2017
Descripción: Fichero que contiene las
implementaciones del segundo módulo centrado
en los ataques MITM
-----------------------------------------*/
#include "func_aux_mod2.h"

/*-----------------------------------------
       Función que se encarga de implementar
       las funcionalidades del módulo 2

       Recibe: Nada

       Devuelve: Nada
-----------------------------------------*/

void *mod2 () {

       //Estructura que almacenará los datos relativos al fichero
       struct CONTENIDO_FICHERO contenido_del_fichero;
       //Estructura que almacenará los datos relativos a las reglas que queremos crear
       struct ESTRUCTURA_REGLA informacion_regla;
       //Variable donde almacenamos el fichero que queremos leer
       char *nombre_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
       //Variable auxiliar para conocer el número de línea que leímos anteriormente
       int num_lineas_anterior;
       //Variable auxiliar para almacenar el nombre del fichero que se crea
       char *nuevo_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
       //Bandera para indicar si se han creado reglas
       bool crear_regla;

       //Inicalizamos las variables
       strcpy(nombre_fichero, FICHERO_ARPWATCH);
```

```
	num_lineas_anterior = INICIO;
	crear_regla = false;

	while(true){

		//Leemos el fichero
		contenido_del_fichero = lee_fichero(nombre_fichero);

		//Miramos a ver si para algún registro ha pasado el tiempo
		/*-------------------------------------------------
			Esto se hace ya que para los ataques MITM, las
			direcciones que vamos a bloquear son IPs
			pertenecientes a la red interna, luego,este
			bloqueo sólo será efectivo durante un periodo
			"largo" de tiempo para impedir el ataque y
			permitir que esa dirección esté disponible
			en un futuro
		-------------------------------------------------*/
		detecta_Registro_caducado(FICHERO_REGISTRO_REGLAS_MITM);

		//Comprobamos si el número de líneas leídas está cerca del máximo
(NUM_FRASES)
		if ( contenido_del_fichero.num_frases_fichero < LIMITE_LINEAS_LEIDAS){

			//Comprobamos si hemos leído alguna línea nueva
			if (contenido_del_fichero.num_frases_fichero > num_lineas_anterior){

				//Buscamos si hemos detectado un cambio en la MAC
				informacion_regla=busca_CAMBIO_EN_MAC(contenido_del_fichero,
num_lineas_anterior);

				//Registramos la regla que vamos a crear
				registra_Regla(informacion_regla);

				//Comprobamos si tenemos que crear alguna regla
				if (informacion_regla.numero_lineas > INICIO){

					//Creamos la regla
					crear_regla                                                    =
crea_y_escribe_regla(FICHERO_REGLAS_SNORT, informacion_regla, "MITM");


					//Comprobamos si la hemos creado correctamente
					if (crear_regla == true) {

						//Cambiamos el valor de la bandera para que solo
reiniciemos cuando se crea una regla nueva
						crear_regla = false;

						//Reiniciamos Snort
						recarga_Snort();
					}
				}
			}
		}

		//En el caso en el que estemos cerca del límite
		else {

			//Llamamos a la función deseada para vaicar el fichero
			nuevo_fichero = vacia_fichero(nombre_fichero);

			/*-------------------------------------------------------
				Para asegurarnos que no perdemos información
				en ningún momento, volvemos a leer el fichero
				al que le hemos cambiado el nombre siempre y
				cuando no supere el número máximo (NUM_FRASES)

				Esta comprobación la hacemos por si el fichero
```

```
                        se encontraba en el rango entre LIMITE y NUM_FRASES
                        y hay información que se puede perder
                  ----------------------------------------------------------*/
                  //Leemos el fichero
                  contenido_del_fichero = lee_fichero(nuevo_fichero);

                  //Comprobamos que ese fichero no ha superado el límite
                  if (contenido_del_fichero.num_frases_fichero < NUM_FRASES){

                        //Buscamos si se ha producido algún cambio en las direcciones
MAC
                        informacion_regla = busca_CAMBIO_EN_MAC(contenido_del_fichero,
num_lineas_anterior);

                        //Comprobamos si hemos detectado la necesidad de crear alguna
regla
                        if (informacion_regla.numero_lineas > INICIO){

                              //Registramos la regla que vamos a crear
                              registra_Regla(informacion_regla);

                              //Creamos la regla
                              crear_regla                                          =
crea_y_escribe_regla(FICHERO_REGLAS_SNORT, informacion_regla, "MITM");

                              //Comprobamos que la hemos creado correctamente
                              if (crear_regla == true){

                                    //Cambiamos el valor de la bandera
                                    crear_regla = false;

                                    //Recargamos Snort
                                    recarga_Snort();
                              }
                        }

                  }

                  //Reiniciamos el número de líneas para la próxima iteración
                  contenido_del_fichero.num_frases_fichero = INICIO;
            }

            //Almacenamos el número de líneas leídas
            num_lineas_anterior = contenido_del_fichero.num_frases_fichero;

            //Esperamos un intervalo de tiempo para la próxima lectura
            sleep(INTERVALO_LECTURA);
      }

      //Liberamos memoria
      free(nombre_fichero);
      free(nuevo_fichero);

      pthread_exit(NULL);

}
```

Func_aux_mod2.h

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 21-02-2017
Descripción: Fichero que contiene las
declaraciones y constantes de las
funciones auxiliares que se usarán en el
segundo módulo
-----------------------------------------*/
#include "constantes_y_librerias.h"
```

```
#include "func_aux.h"

#ifndef func_aux_mod2
#define func_aux_mod2

/*-------------------------------------------------------------
       Función que se encarga de leer un fichero y buscar
       la palabra deseada para almacenar información

       Recibe: Una estructura del tipo CONTENIDO_FICHERO y
       la línea en la que debe empezar a buscar

       Devuelve: Estructura del tipo ESTRUCTURA_REGLA
-------------------------------------------------------------*/
struct        ESTRUCTURA_REGLA        busca_CAMBIO_EN_MAC(struct        CONTENIDO_FICHERO
contenido_del_fichero, int linea_inicio);


/*-------------------------------------------------------------
       Función que se encarga de registrar las reglas ante los
       ataques MITM que se puedan producir para posteriormente
       poder borrarlas ya que estas serán temporales

       Recibe: Estructura del tipo ESTRUCTURA_REGLA con la info
       de las relglas que queremos crear

       Devuelve: Nada
-------------------------------------------------------------*/
void registra_Regla(struct ESTRUCTURA_REGLA informacion_regla);


/*-------------------------------------------------------------
       Función que se encarga de detectar cuando una regla
       sobrepasa un tiempo dado, en este caso una hora

       Recibe: El nombre del fichero

       Devuelve: Nada
-------------------------------------------------------------*/
void detecta_Registro_caducado(char *nombre_fichero);


/*-------------------------------------------------------
       Función que se encarga de eliminar una línea del
       fichero de registro de reglas

       Recibe: El nombre del fichero de registro y la
       línea que se desea eliminar

       Devuelve: Nada
-------------------------------------------------------*/
void elimina_Registro(char *nombre_fichero, int numero_linea);



/*-----------------------------------------------
       Función que se encarga de localizar en el
       fichero de reglas creadas para los ataques
       MITM si existen coincidencias ante una regla
       en concreto

       Recibe: El nombre del fichero donde están
       almacenados los registros, una estructura
       del tipo ESTRUCTURA_REGLA y la posición
       dentro de esta estructura de la regla
       que se quiere buscar

       Devuelve: Un entero indicando la línea en
       la que encontró la coincidencia o "-1" si
```

```
	no encontrnó ninguna
-----------------------------------------------*/
int busca_Registro(char * fichero_registro, struct ESTRUCTURA_REGLA informacion_regla,
int pos_estructura);
#endif
```

Func_aux_mod2.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 21-02-2017
Descripción: Fichero que contiene las
implementaciones de las funciones auxiliares
que se usarán en el segundo módulo
-----------------------------------------*/

#include "func_aux_mod2.h"

/*-------------------------------------------------------------
	Función que se encarga de leer un fichero y buscar
	la palabra deseada para almacenar información

	Recibe: Una estructura del tipo CONTENIDO_FICHERO
	y la línea en la que debe empezar a buscar

	Devuelve: Una estructura del tipo ESTRUCTURA_REGLA

	Caracteristicas:
		Función pensada para leer el fichero de la
		herramienta arpwatch para buscar en él posibles
		conflictos de direcciones cuando se está realizando
		un ataque MITM
-------------------------------------------------------------*/

struct ESTRUCTURA_REGLA busca_CAMBIO_EN_MAC(struct CONTENIDO_FICHERO
contenido_del_fichero, int linea_inicio) {

	//Estructura que se va a devolver para almacenar los datos de las reglas que
queremos crear
	struct ESTRUCTURA_REGLA informacion_regla;
	//Variable auxiliar para recorrer las frases leidas
	int cont_aux_frases;
	//Variable auxiliar para recorrer las palabras de cada frase
	int cont_aux_palabras;
	//Variable para almacenar la linea en la que hemos detectado la coincidencia
	int linea_coincidencia;
	//Variable para almacenar el número dentro del array en el que detectamos la
coincidencia
	int palabra_coincidencia;
	//Contador auxiliar para almacenar el número de reglas de las que hemos almacenado
información
	int num_reglas;
	//Bandera para indicar si hemos detectado una coincidencia de cambio
	bool flag_cambio_MAC;
	//Bandera para indicar si hemos detectado una coincidencia de cambio inmediato
	bool flag_cambio_MAC_Inmediato;


	//Inicializamos
	cont_aux_frases = INICIO;
	cont_aux_palabras = INICIO;
	linea_coincidencia = INICIO;
	palabra_coincidencia= INICIO;
	num_reglas=INICIO;
	flag_cambio_MAC = false;
	flag_cambio_MAC_Inmediato = false;

	//Bucle para recorrer las frases que hemos leido del fichero
```

```
	for (cont_aux_frases=linea_inicio;
		cont_aux_frases < contenido_del_fichero.num_frases_fichero;
		cont_aux_frases++){

		/*-----------------------------------------------------
			En el fichero arpwatch.log sólo vamos a encontrar
			información de la herramienta arpwatch y cada
			notificación se hará en una única línea
			de ahí que cada línea sea una notificación distinta
		-----------------------------------------------------*/
		//Comprobamos si hemos encontrado la palabra que indica el cambio de MAC y
estamos en la siguiente línea
		if	(cont_aux_frases	==	linea_coincidencia+SIGUIENTE_LINEA	&&
(flag_cambio_MAC ==true || flag_cambio_MAC_Inmediato == true)){
			//Cambiamos el valor de la bandera para señalizar el instante en el
que encontramos otra
			flag_cambio_MAC = false;
			flag_cambio_MAC_Inmediato = false;

		}

		//Recorremos las paralbras que hemos leido dentro de cada frase
		for(cont_aux_palabras=0;
			cont_aux_palabras
<=
contenido_del_fichero.palabras_por_frase[cont_aux_frases];
			cont_aux_palabras++) {


			//Comprobamos si la palabra actual es la que estamos buscando y hemos
detectado un cambio de MAC para una misma dir_IP
			if	(flag_cambio_MAC	==	false	&&
strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabr
as],
				CAMBIO_EN_MAC) == IGUAL){

				//Almacenamos	la	línea	en	la	que	hemos	encontrado	la
coincidencia
				linea_coincidencia = cont_aux_frases;
				//Almacenamos	la	palabra	en	la	que	hemos	encontrado	la
coincidencia
				palabra_coincidencia = cont_aux_palabras;
				//Cambiamos	el	valor	de	la	bandera	para	indicar	que	hemos
encontrado una coincidencia
				flag_cambio_MAC = true;
			}

			/*-----------------------------------------------------------------
				Detectamos si ARPWATCH nos ha notificado de un cambio
				en la dirección MAC para una IP que tenía almacenada
				anteriormente
			-----------------------------------------------------------------*/
			//Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección IP
			if	(	flag_cambio_MAC	==	true	&&	cont_aux_palabras	==
palabra_coincidencia + POS_DIR_IP){

				//Almacenamos la dirección IP
				informacion_regla.dir_IP[num_reglas] =

	strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_
palabras]);

				//Indicamos	que	la	dirección	IP	no	lleva	asociado	ningún
puerto
				informacion_regla.dir_Con_Puerto[num_reglas] = false;

				//Indicamos que queremos que la dirección IP vaya en destino
				informacion_regla.dir_en_origen[num_reglas] = false;
```

```
                        //Indicamos que el protocolo sea IP para que cubra a todos
                        informacion_regla.protocolo[num_reglas] = strdup(IP);
                        /*-----------------------------------------------
                               En este caso la acción que vamos a tomar
                               es la de "alert" ya que esto no conlleva
                               gran peligro, pero puede ser útil saberlo
                        -----------------------------------------------*/
                        //Almacenamos la acción que queremos realizar
                        informacion_regla.accion[num_reglas] = strdup("alert");
                        //Aumentamos el número de reglas
                        num_reglas++;

                }
/*-----------------------------------------
       Si en un futuro queremos almacenar
       la dirección MAC e IP de cada cambio
       bastará con descomentar este apartado
-----------------------------------------*/

/*
                //Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección MAC nueva
                else if (flag_cambio_MAC == true && cont_aux_palabras ==
palabra_coincidencia + POS_DIR_MAC_NUEVA){
                        printf("     Dirección MAC nueva: %s\n",
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras]);
                }
                //Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección MAC antigua
                else if (flag_cambio_MAC == true && cont_aux_palabras ==
palabra_coincidencia + POS_DIR_MAC_VIEJA){

                        //Eliminamos los paréntesis con los que ARPWATCH nos notifica
en los logs sobre la antigua MAC
                        int i=0;

                        for (i=1; i <
(int)strlen(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_p
alabras])-1;i++){

                                mac_antigua[i-1] =
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras][i];

                        }
                        mac_antigua[i-1] = TERMINADOR;
                        printf("     Dirección MAC vieja: %s\n", mac_antigua);
                }

*/
                /*-----------------------------------------------------------------
                        Detectamos si ARPWATCH nos ha notificado de un cambio
                        en la dirección MAC en un intervalo corto de tiempo
                        para una misma dirección IP
                -----------------------------------------------------------------*/
                //Comprobamos si hemos detectado algún cambio inmediato de MAC
                if (flag_cambio_MAC_Inmediato == false &&
strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabr
as],
                        CAMBIO_MAC_INMEDIATO ) == IGUAL){

                        //Almacenamos la línea en la que hemos encontrado la
coincidencia
                        linea_coincidencia = cont_aux_frases;
                        //Almacenamos la palabra en la que hemos encontrado la
coincidencia
                        palabra_coincidencia = cont_aux_palabras;
                        //Cambiamos el valor de la bandera para indicar que hemos
encontrado una coincidencia
                        flag_cambio_MAC_Inmediato = true;
```

```
                    }

                    //Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección IP
                    if  (  flag_cambio_MAC_Inmediato  ==  true  &&  cont_aux_palabras  ==
palabra_coincidencia + POS_DIR_IP_INM){

                            //Almacenamos la dirección IP
                            informacion_regla.dir_IP[num_reglas] =

      strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_
palabras]);

                            //Indicamos  que  la  dirección  IP  no  lleva  asociado  ningún
puerto
                            informacion_regla.dir_Con_Puerto[num_reglas] = false;

                            //Indicamos que queremos que la dirección IP vaya en destino
                            informacion_regla.dir_en_origen[num_reglas] = false;

                            //Indicamos que el protocolo sea IP para que cubra a todos
                            informacion_regla.protocolo[num_reglas] = strdup(IP);
                            /*-------------------------------------------------
                                  Para este caso la acción que vamos a tomar es
                                  la de "drop" ya que esto puede ser un ataque
                                  MITM y nos interesa cortarlo
                            -------------------------------------------------*/
                            //Almacenamos la acción que queremos realizar
                            informacion_regla.accion[num_reglas] = strdup("drop");
                            //Aumentamos el número de reglas
                            num_reglas++;

                    }

/*-----------------------------------------
      Si en un futuro queremos almacenar
      la dirección MAC bastará con
      descomentar este apartado
-----------------------------------------*/

/*
                    //Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección MAC nueva
                    else  if  (flag_cambio_MAC_Inmediato  ==  true  &&  cont_aux_palabras  ==
palabra_coincidencia + POS_DIR_MAC_NUEVA_INM){
                            printf("     Dirección        MAC        nueva:        %s\n",
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras]);
                    }
                    //Comprobamos si hemos detectado una coincidencia y la palabra en la
que nos encontramos es la relativa a la dirección MAC antigua
                    else  if  (flag_cambio_MAC_Inmediato  ==  true  &&  cont_aux_palabras  ==
palabra_coincidencia + POS_DIR_MAC_VIEJA_INM){

                            //Eliminamos los paréntesis con los que ARPWATCH nos notifica
en los logs sobre la antigua MAC
                            int i=0;

                            for                 (i=1;                 i                 <
(int)strlen(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_p
alabras])-1;i++){

                                  mac_antigua[i-1]                                      =
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras][i];

                            }
                            mac_antigua[i-1] = TERMINADOR;
                            printf("     Dirección MAC vieja: %s\n", mac_antigua);
                    }
```

```
*/

		}
	}

	//Almacenamos el número de reglas que deseamos crear
	informacion_regla.numero_lineas = num_reglas;

	//Devolvemos la estructura
	return informacion_regla;
}



/*-----------------------------------------------------------------
	Función que se encarga de registrar las reglas ante los
	ataques MITM que se puedan producir para posteriormente
	poder borrarlas ya que estas serán temporales

	Recibe: Estructura del tipo ESTRUCTURA_REGLA con la info
	de las relglas que queremos crear

	Devuelve: Nada

	Caracteristicas:
		Esta función en primer lugar, comprueba si la info que
		tenemos de la nueva regla que vamos a registrar es la
		relativa a  un registro ya existente, si esto ocurre no
		registraremos nada
-----------------------------------------------------------------*/
void registra_Regla(struct ESTRUCTURA_REGLA informacion_regla){

	//Fichero para almacenar la regla y la hora de creación
	FILE *fichero_registro;
	//String auxiliar para almacenar la información que vamos a escribir
	char *info_reglas = (char *)malloc(sizeof(char)*NUM_CARACTERES_REGLAS);
	//Variable auxiliar para almacenar el instante en el que se cambia el fichero
	time_t tiempo_creacion_regla = time(NULL);
	//Array para almacenar la fecha
	char t_creacion_regla[TAM_FECHA];
	//Variable auxiliar para recorrer la estructura
	int cont_aux_reglas;
	//Variable que indica si se ha detectado un registro similar
	int detecta_registro;
	//Estructura para almacenar hora en el momento deseado
	struct tm *tm;

	//Inicializamos las variables
	cont_aux_reglas = INICIO;
	detecta_registro = NO_COINCIDE;


	//Recorremos la estructura donde tenemos la info de las reglas
	for (cont_aux_reglas=0;
		cont_aux_reglas< informacion_regla.numero_lineas;
		cont_aux_reglas++){

		//Buscamos si ya había un registro similar
		detecta_registro                                                    =
busca_Registro(FICHERO_REGISTRO_REGLAS_MITM,informacion_regla, cont_aux_reglas);

		//Si la regla no estaba registrada anteriormente
		if (detecta_registro == NO_COINCIDE){

			//Obtenemos el instante en el que lo hemos detectado
			tm = localtime(&tiempo_creacion_regla);

			//Nos interesa obtener la hora y los minutos de creación de la regla
```

```
                        strftime(t_creacion_regla, sizeof(t_creacion_regla), "%H%M", tm);
                        strcpy(info_reglas, t_creacion_regla);
                        strcat(info_reglas, " ");

                        //Añadimos la información de la reglas

                        //Acción a tomar
                        strcat(info_reglas, informacion_regla.accion[cont_aux_reglas]);
                        strcat(info_reglas, " ");


                        //Protocolo
                        strcat(info_reglas, informacion_regla.protocolo[cont_aux_reglas]);
                        strcat(info_reglas, " ");

                        //IP
                        strcat(info_reglas, informacion_regla.dir_IP[cont_aux_reglas]);
                        strcat(info_reglas, " ");

                        //Posición de la dirección IP
                        if (informacion_regla.dir_en_origen[cont_aux_reglas] == true){
                                strcat(info_reglas, "true");
                        }
                        else {
                                strcat(info_reglas, "false");
                        }
                        strcat(info_reglas, " ");



                        /*-----------------------------------------------
                                Sabemos de antemano que los ataques MITM no
                                llevan ningún puerto asociado
                        -----------------------------------------------*/
                        //Asociamos la bandera al puerto
                        strcat(info_reglas, "false");
                        strcat(info_reglas, "\n");

                        //Abrimos el fichero para escribir el registro creado
                        fichero_registro = fopen(FICHERO_REGISTRO_REGLAS_MITM, "a+");


                        if (fichero_registro != NULL){
                                //Escribimos en el fichero
                                fputs(info_reglas, fichero_registro);

                                //Cerramos el fichero
                                fclose(fichero_registro);

                        }
                }
        }




        //Liberamos memoria
        free(info_reglas);
}


/*-----------------------------------------------------------
        Función que se encarga de detectar cuando una regla
        sobrepasa un tiempo dado, en este caso una hora

        Recibe: El nombre del fichero

        Devuelve: Nada
```

```
	Características:
		Lo primero que realiza es la lectura del fichero
		tras esto, toma la hora actual y comienza a
		recorrer el array del fichero que hemos leido
		para comparar los tiempos en los que se realizó
		la creación de la regla y la hora actual
		Si este tiempo es mayor a una hora, procedemos
		a obtener la información de ese registro y
		lo eliminamos, por último eliminamos la regla
		que generó el registro

En este caso, queremos eliminar las reglas  que se
realizan en este módulo ya que bloquean el tráfico
para direcciones de la red local y así solo prohibimos
esa dirección durante un intervalo de tiempo, no para
siempre
-------------------------------------------------------*/


void detecta_Registro_caducado(char *nombre_fichero){

	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;
	//Estructura para almacenar la información de la regla
	struct ESTRUCTURA_REGLA informacion_regla;
	//Variable para recorrer la estructura
	int cont_aux_frases_fichero;
	//Variable auxiliar para almacenar el instante de tiempo actual
	time_t tiempo_actual = time(NULL);
	//Array para almacenar la fecha
	char t_actual[TAM_FECHA];
	//Estructura para almacenar hora en el momento deseado
	struct tm *tm;


	//Inicializamos
	cont_aux_frases_fichero = INICIO;

	//Leemos el fichero
	contenido_del_fichero = lee_fichero(nombre_fichero);

	//Obtenemos el instante en el que lo hemos detectado
	tm = localtime(&tiempo_actual);

	//Nos interesa obtener la hora y los minutos para realizar la comparación
	strftime(t_actual, sizeof(t_actual), "%H%M", tm);

	//Recorremos la estructura
	for (cont_aux_frases_fichero=0;
		cont_aux_frases_fichero < contenido_del_fichero.num_frases_fichero;
		cont_aux_frases_fichero++){

		//Comprobamos si ha pasado una hora desde la hora de creación de la regla
		if                              ((atoi(t_actual)                                -
atoi(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][HORA])  >=
PASA_UNA_HORA) ||
		    (atoi(t_actual)                                                             -
atoi(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][HORA])   <
INICIO)) {

		    //Rellenamos la estructura con la información que hemos recogido del
fichero donde registramos las reglas
		    informacion_regla.accion[INICIO]                                            =
strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][POS_ACC
ION+1]);
		    informacion_regla.protocolo[INICIO]                                         =
strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][POS_PRO
TOCOLO+1]);
```

```
                    informacion_regla.dir_IP[INICIO]                                    =
strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][POS_DIR
_IP_ORIG+1]);

                    if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][POS_BA
NDERA_IP],"true")==IGUAL) {
                           informacion_regla.dir_en_origen[INICIO] = true;
                    }
                    else {
                           informacion_regla.dir_en_origen[INICIO] = false;
                    }

                    if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][POS_BA
NDERA_PUERTO],"true")==IGUAL){
                           informacion_regla.dir_Con_Puerto[INICIO] = true;
                    }
                    else {
                           informacion_regla.dir_Con_Puerto[INICIO] = false;
                    }
                    //Eliminamos el registro
                    elimina_Registro(nombre_fichero, cont_aux_frases_fichero);
                    //Eliminamos la regla que creó ese registro
                    elimina_Regla(FICHERO_REGLAS_SNORT,informacion_regla, INICIO);
                    //Recargamos la configuración de SNORT para que elimine la regla
                    recarga_Snort();
             }

      }

}


/*-------------------------------------------------------
      Función que se encarga de eliminar una línea del
      fichero de registro de reglas

      Recibe: El nombre del fichero de registro y la
      línea que se desea eliminar

      Devuelve: Nada

      Características:
            Lee el fichero y lo almacena en un array,
            posteriormente recorre ese array saltandose
            la línea que se desea elminar y escribe cada
            línea
-------------------------------------------------------*/

void elimina_Registro(char *nombre_fichero, int numero_linea){

      //Estructura que almacenará los datos relativos al fichero
      struct CONTENIDO_FICHERO contenido_del_fichero;
      //Fichero donde escribiremos todos los registros menos el que se desea borrar
      FILE *fichero_borrar;
      //Variable auxiliar para almacenar el registro que vamos a sobreescribir
      char *registro_aux = (char *)malloc(sizeof(char) *NUM_CARACTERES_REGLAS);
      //Variables para recorrer la tabla con el contenido del fichero
      int cont_aux_frases_fichero;
      int cont_aux_palabras_fichero;

      //Inicializamos las variables
      cont_aux_frases_fichero = INICIO;
      cont_aux_palabras_fichero = INICIO;


      //Leemos el contenido del fichero
      contenido_del_fichero = lee_fichero(nombre_fichero);
```

```
	//Abrimos el fichero vacío
	fichero_borrar = fopen(nombre_fichero, "w");

	//Recorremos el array que rellenamos cuando lo leimos por primera vez
	for (cont_aux_frases_fichero=0;
		cont_aux_frases_fichero < contenido_del_fichero.num_frases_fichero;
		cont_aux_frases_fichero++){

		//Comprobamos que la línea que vamos a escribir no es la que queremos
eliminar
		if (cont_aux_frases_fichero != numero_linea) {

			//Recorremos las palabras de esa frase
			for(cont_aux_palabras_fichero =0;
				cont_aux_palabras_fichero                                    <=
contenido_del_fichero.palabras_por_frase[cont_aux_frases_fichero];
				cont_aux_palabras_fichero++) {

				//Si es la pirmera palabra
				if (cont_aux_palabras_fichero == INICIO){

					//Copiamos esa palabra en nuestro string auxiliar
					strcpy(registro_aux,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][cont_aux_palab
ras_fichero]);

				}

				//Si es una palabra distinta a la primera
				else {
					//Concatenamos el contenido al string auxiliar
					strcat(registro_aux, " ");
					strcat(registro_aux,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases_fichero][cont_aux_palab
ras_fichero]);

				}
			}

			//Añadimos un salto de línea
			strcat(registro_aux, "\n");
			//Escribimos el contenido en el fichero
			fputs(registro_aux, fichero_borrar);
		}
	}

	//Cerramos el fichero
	fclose(fichero_borrar);

	//Liberamos memoria
	free(registro_aux);
}


/*-------------------------------------------------
	Función que se encarga de localizar en el
	fichero de reglas creadas para los ataques
	MITM si existen coincidencias ante una regla
	en concreto

	Recibe: El nombre del fichero donde están
	almacenados los registros, una estructura
	del tipo ESTRUCTURA_REGLA y la posición
	dentro de esta estructura de la regla
	que se quiere buscar

	Devuelve: Un entero indicando la línea en
	la que encontró la coincidencia o "-1" si
```

```
	no encontrnó ninguna
-------------------------------------------------*/
int busca_Registro(char * fichero_registro, struct ESTRUCTURA_REGLA informacion_regla,
int pos_estructura) {

	//Variable para indicar la línea en la que se ha encontrado la coincidencia entre
registros
	int linea_coincidencia;
	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;

	//Variables auxiliares para recorrer la estructura
	int cont_aux_lineas;


	//Inicializamos
	linea_coincidencia = NO_COINCIDE;

	//Leemos el fichero del registro de reglas
	contenido_del_fichero = lee_fichero(fichero_registro);

	//Recorremos la estructura línea a línea
	for (cont_aux_lineas=0;
			cont_aux_lineas<	contenido_del_fichero.num_frases_fichero	&&
linea_coincidencia == NO_COINCIDE;
			cont_aux_lineas++) {

		//Comprobamos si existe coincidencia

		//Comenzamos comprobando si hay coincidencia en el conjunto acción/protocolo
		if (strcmp(informacion_regla.accion[pos_estructura],
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][POS_ACCION+1])	==
IGUAL
			&&
strcmp(informacion_regla.protocolo[pos_estructura],contenido_del_fichero.contenido_leido_
del_fichero[cont_aux_lineas][POS_PROTOCOLO+1])==IGUAL) {

			//Comprobamos si coinciden las direcciones IP
			if (strcmp(informacion_regla.dir_IP[pos_estructura],

	contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][POS_DIR_IP_ORIG+
1]) == IGUAL) {

				//En caso de coincidir, comprobamos que estaban ambas en
origen o destino
				if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][POS_BANDERA_IP
],"true") == IGUAL) {

					//Comprobamos que el valor de la regla es el mismo
					if (informacion_regla.dir_en_origen[pos_estructura] ==
true) {

						//En ese caso, estamos ante la misma regla
						linea_coincidencia = cont_aux_lineas;
					}

					else {
						//En caso contrario, es una regla nueva
						linea_coincidencia = NO_COINCIDE;
					}
				}

				//En caso de que la dir_IP no estuviese en el origen
				else {

					//Si coincide, estamos ante la misma regla
					if (informacion_regla.dir_en_origen[pos_estructura] ==
false) {
```

```
                                        linea_coincidencia = cont_aux_lineas;
                                }

                                else {

                                        linea_coincidencia = NO_COINCIDE;
                                }
                        }
                }

                //En caso de no coincidir
                else {
                        linea_coincidencia = NO_COINCIDE;
                }
        }

        else {

                linea_coincidencia = NO_COINCIDE;
        }

    }

    return linea_coincidencia;
}
```

Respecto al tercer módulo;

Mod3_SSID.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 25-02-2017
Descripción: Fichero que contiene las
implementaciones del tercer módulo encargado
de detectar la suplantación de SSID
-----------------------------------------*/
#include "func_aux_mod3.h"

//Constante para definir el ESSID y BSSID propio para buscar las coincidencias
#define ESSID_propio "M"
#define BSSID_propio "D"

/*-----------------------------------------
        Función que se encarga de implementar
        las funcionalidades del módulo 3

        Recibe: Nada

        Devuelve: Nada
-----------------------------------------*/
void *mod3 () {

        //Estructura que almacenará el contenido del fichero
        struct CONTENIDO_FICHERO contenido_del_fichero;
        //Estructura que almacenará los SSID leidos
        struct INFO_SSID info_ssid;
        //Estructura que almacenará los SSID coincidentes
        struct INFO_SSID ssid_coincidentes;
        //Variable para almacenar el nombre del fichero
        char *nombre_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
        //String para almacenar los comandos auxiliares
        char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

        //Inicializamos
        strcpy(nombre_fichero, FICHERO_CAPTURA_WIFI);
```

```
	while(true){

		//Escanemaos la red WiFi
		escanea_WiFi();

		//Leemos el fichero
		contenido_del_fichero = lee_fichero_csv(nombre_fichero);

		//Obtenemos los SSID leídos
		info_ssid = procesa_fichero_CSV(contenido_del_fichero);

		//Buscamos posibles SSID duplicados
		ssid_coincidentes = busca_SSID(info_ssid, ESSID_propio, BSSID_propio);

		//Creamos las reglas para esos SSIDs que hemos detectado
		bloquea_SSID(ssid_coincidentes);

		//Esperamos un intervalo de tiempo para el próximo escanéo
		sleep(INTERVALO_LECTURA);
	}

	//Liberamos memoria
	free(nombre_fichero);
	free(comando);

	pthread_exit(NULL);
}
```

Func_aux_mod3.h

```
/*-------------------------------------------
Autor: Francisco Javier Santos
Fecha: 27-02-2017
Descripción: Fichero que contiene las
declaraciones y constantes de las
funciones auxiliares que se usarán en el
tercer módulo
-------------------------------------------*/
#include "constantes_y_librerias.h"
#include "func_aux.h"

#ifndef func_aux_mod3
#define func_aux_mod3s


//Estructura para almacenar los SSID leidos
typedef struct INFO_SSID {

	//Array para almacenar los SSID que se lean
	char *essid[NUM_SSID];
	//Array para almacenar los BSSID que se lean
	char *bssid[NUM_SSID];
	/*-----------------------------------------
		Bandera para conocer si la coincidencia
		fue ESSID o BSSID
			-- true; La coincidencia es en BSSID
			-- false; La coincidencia es en ESSID
	-----------------------------------------*/
	bool coincide_BSSID[NUM_SSID];
	//Variable para almacenar la cantidad de SSID leidos
	int num_ssid;

}info_ssid;

/*-------------------------------------------------------------
	Función que se encarga de leer un fichero de tipo CSV
```

```
       Recibe: El nombre del fichero a leer

       Devuelve: Una estructura del tipo CONTENIDO_FICHERO
                           con los datos obtenidos del fichero
-------------------------------------------------------------*/
struct CONTENIDO_FICHERO lee_fichero_csv(char *fichero);


/*-----------------------------------------------------
       Función que se encarga de extraer del fichero leido
       la información de los SSID

       Devuelve: Estructura del tipo INFO_SSID
       Recibe: Estructura del tipo CONTENIDO_FICHERO
-----------------------------------------------------*/
struct INFO_SSID procesa_fichero_CSV(struct CONTENIDO_FICHERO contenido_del_fichero);


/*-------------------------------------------------------------------
       Función que se encarga de detectar la posibilidad de SSID
       duplicado.

       Devuelve: La estructura del tipo INFO_SSID
       Recibe: Estructura del tipo INFO_SSID, el ssid y la MAC que
       queremos comprobar
-------------------------------------------------------------------*/
struct INFO_SSID busca_SSID(struct INFO_SSID info_ssid, char *ssid, char *mac_ssid);


/*-----------------------------------------------------------
       Función que se encarga de crear una regla en IPTABLES
       si encuentra un ESSID duplicado y notifica si encuentra
       un BSSID duplicado

       Devuelve: Nada
       Recibe: Una estructura del tipo INFO_SSID
-----------------------------------------------------------*/
void bloquea_SSID (struct INFO_SSID ssid_coincidentes);


/*-----------------------------------------------------
       Función que se encarga de capturar los SSID
       existentes en la red WiFi

       Recibe: Nada
       Devuelve: Nada
-----------------------------------------------------*/
void escanea_WiFi();


#endif
```

Func_aux_mod3.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 27-02-2017
Descripción: Fichero que contiene las
implementaciones de las funciones auxiliares
que se usarán en el tercer módulo
-----------------------------------------*/
#include "func_aux_mod3.h"


/*-------------------------------------------------------------
       Función que se encarga de leer un fichero de tipo CSV

       Recibe: El nombre del fichero a leer
```

```
	Devuelve: Una estructura del tipo CONTENIDO_FICHERO
			con los datos obtenidos del fichero

	Características:
		Los fichero CSV diferencian sus valores por el caracter
		',' por lo que esto nos marcará el fin de cada palabra
-------------------------------------------------------------*/
struct CONTENIDO_FICHERO lee_fichero_csv(char *fichero){

	//Estructura que será devuelta
	struct CONTENIDO_FICHERO contenido_del_fichero;
	//String para almacenar la palabra antes de introducirla en la estructura
	char *palabra_aux = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Cadena para almacenar el informe de log
	char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Variable para contabilizar el número de palabras
	int palabras;
	//Variables para contabilizar el número de frases
	int frases;
	//Variable auxiliar para recorrer el string
	int letra;
	//Variable para el fichero que vamos a leer
	FILE *fichero_lectura;
	//Variable para recorrer el fichero caracter a caracter
	signed char c;
	//Variable para almacenar el caracter que se ha leido anteriormente
	signed char c_ant;


	//Abrimos el fichero
	fichero_lectura = fopen(fichero, "r");

	//Inicializamos
	c_ant = ESPACIO;
	palabras = INICIO;
	frases = INICIO;
	letra = INICIO;

	//Si hemos abierto el fichero correctamente
	if (fichero_lectura != NULL){

		//Comenzamos a leer el fichero
		while ((c=fgetc(fichero_lectura))!=EOF){

			//Comprobamos que no superamos el límite
			if (frases < NUM_FRASES){

				// Si estamos ante una línea en blanco
				if ( (c == SALTO_DE_LINEA && c_ant == SALTO_DE_LINEA)){

					//Reiniciamos el contador de palabras y letras
					palabras = INICIO;
					letra= INICIO;
				}

				//Si no estamos ante una línea en blanco
				else  {

					//Comprobamos que no estamos ante un doble espacio en
blanco
					if ( !(c  == ESPACIO && c_ant == ESPACIO) ){

						//Nos aseguramos que no es el caracter ',' que
se usa para separar palabras
						if (c != COMA){

							//Comprobamos si estamos ante un caracter
que no sea '\n' ni ' '
```

```
                                        if (c != SALTO_DE_LINEA && c != ESPACIO ){

                                                //Almacenamos ese caracter
                                                palabra_aux[letra] = c;
                                                //Aumentamos el contador para
almacenar el próximo caracter en esa posicion
                                                letra++;

                                        }

                                        //Si estamos ante un salto de línea
                                        else if (c == SALTO_DE_LINEA ) {

                                                //Añadimos el terminador
                                                palabra_aux[letra++] = TERMINADOR;

                                                //Almacenamos la palabra en la
estructura

        contenido_del_fichero.contenido_leido_del_fichero[frases][palabras]             =
strdup(palabra_aux);
                                                //Almacenamos el número de palabras
que tiene esa frase

        contenido_del_fichero.palabras_por_frase[frases] = palabras;

                                                //Aumentamos el número de frases
                                                frases++;

                                                //Reiniciamos los contadores
                                                palabras = INICIO;
                                                letra = INICIO;

                                        }

                                        //Si estamos ante un espcacio
                                        else if (c == ESPACIO && c_ant != COMA){

                                                //Almacenamos el espacio
                                                palabra_aux[letra] = ESPACIO;
                                                //Aumentamos el contador
                                                letra++;

                                        }
                                }

                                //Si estamos leyendo el caracter ',' estamos
ante una nueva palabra
                                else {

                                        //Añadimos el terminador
                                        palabra_aux[letra++] = TERMINADOR;

                                        //Almacenamos la palabra en la estructura

        contenido_del_fichero.contenido_leido_del_fichero[frases][palabras]             =
strdup(palabra_aux);

                                        //Aumentamos el número de palabras leiddas
                                        palabras++;
                                        //Reiniciamos el contador
                                        letra = INICIO;
                                }
                        }
                }

                //Guardamos el caracter que acabamos de leer para la próxima
iteración
                c_ant = c;
```

```
                    }

                    //En caso contrario indicamos que hemos sobrepasado el número de
lí-neas
                    else {

                           //Creamos la cadena que nos informará de la incidencia
producida
                           sprintf(log, ERROR_NUM_LINEAS, fichero, NUM_FRASES);

                           //Registramos dicha incidencia en el fichero de logs
                           registra_log(log);
                    }
             }

             //Almacenamos el número total de líneas leídas
             contenido_del_fichero.num_frases_fichero = frases;



        //Cerramos el fichero
        fclose(fichero_lectura);

      }

      else {

             //Creamos la cadena que nos informará de la incidencia producida
             sprintf(log, ERROR_APERTURA_FICHERO, fichero);

             //Registramos dicha incidencia en el fichero de logs
             registra_log(log);

      }


      //Liberamos la memoria
      free(palabra_aux);
      free(log);

      //Devolvemos la estructura
      return contenido_del_fichero;
}



/*-------------------------------------------------------
      Función que se encarga de extraer del fichero leído
      la información de los SSID

      Devuelve: Estructura del tipo INFO_SSID formada por:
             -- essid[NUM_SSID]: Array que almacena los
                                 ESSID leídos
             -- bssid[NUM_SSID]: Array que almacena los
                                 BSSID leídos
             -- num_ssid: Contador que almacena el número
                    total de SSID leídos para poder recorrer
                    los arrays en cualquier momento

      Recibe: Estructura del tipo CONTENIDO_FICHERO
-------------------------------------------------------*/
struct INFO_SSID procesa_fichero_CSV(struct CONTENIDO_FICHERO contenido_del_fichero){

      //Estructura que será devuelta
      struct INFO_SSID info_ssid;
      //Variable auxiliar para almacenar el número de SSID detectados
      int cont_aux_ssid;
      //Variable auxiliar para recorrer las filas del fichero CSV
```

```
	int cont_aux_filas;
	//Variable auxiliar para recorrer las columnas del fichero CSV
	int cont_aux_columnas;
	//Variable para almacenar la columna en la que se encuentra el ESSID
	int pos_essid;
	//Variable para alamcenar la columna donde se encuentra el BSSID
	int pos_bssid;
	//Variable para alamcenar la fila en la que se encuentra la palabra ESSID
	int fil_essid;
	//Variable para almacenar la fila en la que se encuentra la palbra BSSID
	int fil_bssid;
	//Bandera para indicar el momento en el que comienza la info de los clientes
	bool info_clientes;

	//Inicializamos
	cont_aux_ssid = INICIO;
	cont_aux_filas = INICIO;
	cont_aux_columnas = INICIO;
	pos_bssid = INICIO;
	fil_bssid = INICIO;
	pos_essid = INICIO;
	fil_essid = INICIO;
	info_clientes = false;

	//Recorremos la estructura
	for  (cont_aux_filas=1;  cont_aux_filas<  contenido_del_fichero.num_frases_fichero;
cont_aux_filas++){

		for                                                    (cont_aux_columnas=0;
cont_aux_columnas<=contenido_del_fichero.palabras_por_frase[cont_aux_filas];
cont_aux_columnas++){

			/*-----------------------------------------------
				Al ser un fichero CSV, tiene una estructura
				fija en la que cada valor pertenece a una
				columna, aprovechando esto, vamos a fijar
				los valores de las columnas para obtener
				los que nos interesa
			-----------------------------------------------*/

			//Comprobamos si estamos en la posición que ocupará la información
relativa a los ESSID
			if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_filas][cont_aux_column
as], ESSID) == IGUAL){

				//Almacenamos el número de fila y columna que ocupan
				pos_essid = cont_aux_columnas;
				fil_essid = cont_aux_filas;
			}

			//Comprobamos si estamos en la posición que ocupará la información de
los BSSID
			if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_filas][cont_aux_column
as], BSSID) == IGUAL){

				//Almacenamos la columna y fila que ocupan
				pos_bssid = cont_aux_columnas;
				fil_bssid = cont_aux_filas;
			}

			//Comprobamos si vamos a pasar a ver la información relativa a los
clientes
			if
(strcmp(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_filas][cont_aux_column
as], INFO_CLIENTES) == IGUAL){

				//Cambiamos el valor de la bandera
```

```
                        info_clientes = true;
                    }

                    //Si no estamos leyendo info de clientes y estamos en una nueva fila
                    if (info_clientes == false
                        &&   (cont_aux_filas   >   fil_bssid   &&   cont_aux_filas   >
fil_essid)){

                        //Comprobamos que la columna en la que nos encontramos es
donde se almacenarán los BSSIDs
                        if (cont_aux_columnas== pos_bssid){

                            //Almacenamos el BSSID
                            info_ssid.bssid[cont_aux_ssid]                          =
strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_filas][cont_aux_columna
s]);

                        }

                        //En caso contrario, comprobamos que estamos en la columna
donde se almacenará la información del ESSID
                        else if (cont_aux_columnas==pos_essid) {

                            //Almacenamos la información del ESSID
                            info_ssid.essid[cont_aux_ssid]                          =
strdup(contenido_del_fichero.contenido_leido_del_fichero[cont_aux_filas][cont_aux_columna
s]);

                            //Aumentamos el número de SSIDs detectados
                            cont_aux_ssid++;

                        }
                    }
                }
        }

        //Almacenamos el número de SSIDs detectados en la estructura
        info_ssid.num_ssid = cont_aux_ssid;

        //Devolvemos la estructura
        return info_ssid;
}


/*-------------------------------------------------------------------
        Función que se encarga de detectar la posibilidad de SSID
        duplicado.

        Devuelve: La estructura del tipo INFO_SSID

        Recibe: Estructura del tipo INFO_SSID, el ssid y la MAC que
        queremos comprobar

        Caracterí-sitcas:
                Comprobamos los dos casos que se pueden dar para suplantar
                un SSID, mismo ESSID y distinto BSSID o viceversa
-------------------------------------------------------------------*/
struct INFO_SSID busca_SSID(struct INFO_SSID info_ssid, char *ssid, char *mac_ssid){

        //Estructura que almacenara los SSID iguales a los que se les pasa como parámetros
        struct INFO_SSID ssid_coincidentes;
        //Contador auxiliar para recorrer la estructura
        int cont_aux_ssid;
        //Contador auxiliar para almacemar el número de coincidencias
        int num_ssid_coincidentes;


        //Inicializamos
        cont_aux_ssid = INICIO;
```

```
	num_ssid_coincidentes = INICIO;


	//Recorremos la estructura
	for (cont_aux_ssid=0; cont_aux_ssid<info_ssid.num_ssid;cont_aux_ssid++){


		/*-----------------------------------------
			La primera casuística será la de
			la suplantacion del ESSID, para ello
			buscamos si el ESSID coincide y el BSSID
			es distinto
		-----------------------------------------*/
		if	((strcmp(info_ssid.essid[cont_aux_ssid],	ssid)	==	IGUAL)	&&
(strcmp(info_ssid.bssid[cont_aux_ssid], mac_ssid) != IGUAL)) {

			//Almacenamos los datos en la estructura
			ssid_coincidentes.essid[num_ssid_coincidentes]                      =
info_ssid.essid[cont_aux_ssid];
			ssid_coincidentes.bssid[num_ssid_coincidentes]                      =
info_ssid.bssid[cont_aux_ssid];
			ssid_coincidentes.coincide_BSSID[num_ssid_coincidentes] = false;
			num_ssid_coincidentes++;
		}

		/*-----------------------------------------
			El segundo caso será el de suplantar
			el BSSID, por tanto comprobamos si
			hemos leído el mismo BSSID y distinto
			ESSID
		-----------------------------------------*/
		else	if	((strcmp(info_ssid.essid[cont_aux_ssid],	ssid)	!=	IGUAL)	&&
(strcmp(info_ssid.bssid[cont_aux_ssid], mac_ssid) == IGUAL)){

			//Almacenamos los datos en la estructura
			ssid_coincidentes.essid[num_ssid_coincidentes]                      =
info_ssid.essid[cont_aux_ssid];
			ssid_coincidentes.bssid[num_ssid_coincidentes]                      =
info_ssid.bssid[cont_aux_ssid];
			ssid_coincidentes.coincide_BSSID[num_ssid_coincidentes] = true;
			num_ssid_coincidentes++;

		}
	}

	//Almacenamos el número de coincidencias que hemos encontrado
	ssid_coincidentes.num_ssid = num_ssid_coincidentes;

	//Devolvemos la estructura
	return ssid_coincidentes;
}


/*-------------------------------------------------------------
	Función que se encarga de crear una regla en IPTABLES
	si encuentra un ESSID duplicado y notifica si encuentra
	un BSSID duplicado

	Devuelve: Nada

	Recibe: Una estructura del tipo INFO_SSID con las
		coincidencias detectadas

	Características:
		Debe comprobar si la coincidencia es de
		BSSID o ESSID

			ESSID; Obtener el BSSID correspondiente
			y bloquear el tráfico mediante IPTABLES
```

```
                    BSSID; Notifica que ha detectado un BSSID duplicado
-------------------------------------------------------------*/


void bloquea_SSID (struct INFO_SSID ssid_coincidentes){

        //Variable para recorrer la estructura
        int cont_aux_ssid;
        //String para almacenar el comando que vamos a utilizar
        char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
        //Cadena para almacenar el informe de log
        char *log = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);

        //Inicializamos
        cont_aux_ssid = INICIO;

        //Recorremos la estructura
        for (cont_aux_ssid=0;cont_aux_ssid<ssid_coincidentes.num_ssid;cont_aux_ssid++){


                //Comprobamos si la coincidencia se ha dado en el ESSID
                if (ssid_coincidentes.coincide_BSSID[cont_aux_ssid] == false) {

                        /*--------------------------------------
                                Creamos la regla para cortar todo
                                el tráfico entrante de esa dirección
                                MAC de esta forma evitamos recibir
                                los beacons y así- la posibilidad de
                                establecer una conexión
                        --------------------------------------*/

                        strcpy(comando, "sudo iptables -A INPUT -m mac --mac-source ");
                        strcat(comando, ssid_coincidentes.bssid[cont_aux_ssid]);
                        strcat(comando, " -j DROP");
                        system(comando);

                        /*-----------------------------------------------------------------
                                Almacenamos el contenido en el fichero para que se pueda
                                visualizar desde la interfaz web
                        -----------------------------------------------------------------*/
                        system("sudo iptables -L -n > /home/pi/web/cgi/reglas_iptables");
                }

                //En caso contrario la coincidencia será en el BSSID
                else if (ssid_coincidentes.coincide_BSSID[cont_aux_ssid] == true){

                        /*-------------------------------------------------
                                Dado que estamos analizando el espectro y las
                                comunicaciones son en capa dos, no podemos obtener
                                la dirección IP del SSID del cuál hemos detectado
                                el BSSID duplicado, y tampoco podemos aplicar
                                la misma regla que anteriormente ya que nos
                                dejaría sin conexión a nuestro router, luego
                                la única acción que podemos tomar es notificar
                        -------------------------------------------------*/

                        //Creamos la cadena para notificar esto mediante el fichero de logs
                        sprintf(log,BSSID_DUPLICADO,  ssid_coincidentes.essid[cont_aux_ssid],
ssid_coincidentes.bssid[cont_aux_ssid]);

                        //Registramos la incidencia en el fichero
                        registra_log(log);
                }
        }

        //Liberamos la memoria
        free(comando);
```

```
	free(log);
}

/*-----------------------------------------------------
	Función que se encarga de capturar los SSID
	existentes en la red WiFi

	Recibe: Nada
	Devuelve: Nada

	Características:
		Se encarga de capturar el tráfico durante
		un tiempo, tras esto para la captura para
		generar el fichero correspondiente.
		Antes de iniciar la captura se asegura de
		que el fichero donde la vamos a almacenar
		no existía previamente
-----------------------------------------------------*/

void escanea_WiFi() {

	//Variable para almacenar el nombre del fichero
	char *nombre_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//String para almacenar los comandos auxiliares
	char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Variable auxiliar para almacenar el PID
	char *pid = (char *)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Variable auxiliar para almacenar el nombre del fichero
	char *nombre_fichero_pid = (char * )malloc(sizeof(char *)*NUM_CARACTERES_PALABRA);
	//Variable auxiliar para almacenar el nombre del fichero y la extensión
	char		*nombre_fichero_completo		=		(char
*)malloc(sizeof(char)*NUM_CARACTERES_PALABRA);
	//Variable auxiliar para saber en la línea en la que nos encontramos
	int cont_aux_linea;
	//Variable auxiliar para saber en que palabra nos encontramos
	int cont_aux_palara;
	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;

	//Inicializamos las variables
	strcpy(nombre_fichero_pid, FICHERO_PID_AIRODUMP);
	cont_aux_linea = INICIO;
	cont_aux_palara = INICIO;



	//Almacenamos el nombre del fichero
	strcpy(nombre_fichero, FICHERO_CAPTURA_WIFI_SIN_EXTENSION);


	//Comprobamos si el fichero que vamos a generar existe
	strcpy(nombre_fichero_completo, nombre_fichero);
	strcat(nombre_fichero_completo, EXTENSION_FICHERO_CAPTURA_WIFI);

	/*-----------------------------------------
		Con la función acces podemos saber si
		el fichero existe o no, el parámetro F_OK
		se le pasa para saber si existe
	-----------------------------------------*/
	if ( access(nombre_fichero_completo, F_OK) != NO_EXISTE) {

		/*-----------------------------------------
			En caso de que exista, nos interesa
			eliminarlo ya que airodump no sobreescribe
			los ficheros sino que va añadiendo una
			terminación numérica y de esta forma
			nos sería imposible seguir por qué fichero
			vamos, así que si lo eliminamos nos
			aseguramos que siempre lo escribimos con
```

```
                el mismo nombre
        ------------------------------------------*/
        //Eliminamos el fichero para la próxima captura
        strcpy(comando, "sudo rm -rf ");
        strcat(comando, nombre_fichero);
        strcat(comando, "*");
        system(comando);
}

//Creamos el comando para iniciar la captura
strcpy(comando, "{ sudo airodump-ng mon0 --output-format csv -w ");
strcat(comando, nombre_fichero);
//Redireccionamos la posible salida de errores a un ficheo auxiliar
strcat(comando, " 2>> output.txt; } &");
system(comando);

//Capturamos durante un tiempo
sleep(TIEMPO_CAPTURA);

//Eliminamos el fichero auxiliar que hemos creado anteriormente
strcpy(comando, "rm -rf output.txt");
system(comando);

//Creamos el comando para obtener el PID y redireccionarlo al fichero
strcpy(comando, "pidof airodump-ng > ");
strcat(comando, nombre_fichero_pid);
//Ejecutamos dicho comando
system(comando);


//Leemos el fichero
contenido_del_fichero = lee_fichero(nombre_fichero_pid);

//Recorremos el array devuelto para obtener el PID
for (cont_aux_linea=0;
                cont_aux_linea < contenido_del_fichero.num_frases_fichero;
                cont_aux_linea++){

        for (cont_aux_palara=0;

cont_aux_palara<=contenido_del_fichero.palabras_por_frase[cont_aux_linea];
                cont_aux_palara++) {

                //Almacenamos el PID en nuestra variable específica
                strcpy(pid,

contenido_del_fichero.contenido_leido_del_fichero[cont_aux_linea][cont_aux_palara])
;

                //Creamos el comando para parar la captura
                strcpy(comando, "sudo kill -9 ");
                strcat(comando, pid);
                //Ejecutamos dicho comando
                system(comando);
        }
}


//Liberamos memoria
free(nombre_fichero);
free(nombre_fichero_pid);
free(nombre_fichero_completo);
free(comando);
free(pid);
}
```

Por último, el código perteneciente al cuarto módulo.

Mod4_database.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 7-04-2017
Descripción: Fichero que contiene la
implementación del cuarto módulo que se
encarga de la gestión de la base de datos
de los clientes asociados
-----------------------------------------*/

#include "func_aux_mod4.h"

/*-----------------------------------------
	Función que se encarga de implementar
	las funcionalidades del módulo 4

	Recibe: Nada

	Devuelve: Nada
-----------------------------------------*/
void *mod4 () {

	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;
	//Estructura que almacenará los ficheros que almacenarán los datos para introducir
en la BBDD
	struct CONTENIDO_FICHERO lista_de_ficheros;
	//Variable auxiliar para almacenar el nombre del fichero que se crea
	char *nuevo_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_REGLAS);
	//Variable auxiliar para almacenar el nombre del fichero
	char *nombre_fichero = (char *)malloc(sizeof(char)*NUM_CARACTERES_REGLAS);
	//String para almacenar los comandos auxiliares
	char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_REGLAS);
	//Variable auxiliar para almacenar el número de líneas leídas en cada iteración
	int num_lineas_anteriores;
	//Variable auxiliar para recorrer las frases de la estrutura leida
	int cont_aux_frases;


	//Inicializamos
	num_lineas_anteriores = INICIO;
	cont_aux_frases = INICIO;
	strcpy(nombre_fichero, FICHERO_REGISTRO_CLIENTES);


	while(true){

		/*-------------------------------------------------------
			Obtenemos todos los registros almacenados en ARPWATCH
			Nos interesa únicamente las tres primeras columnas
				-- En la primera tenemos la dirección MAC
				-- En la segunda la dirección IP
				-- En la tercera el instante de tiempo en el que
					se almacenó
		-------------------------------------------------------*/
		strcpy(comando, "cat /var/lib/arpwatch/*.dat | cut -f 1,2,3 > ");
		strcat(comando, FICHERO_REGISTRO_CLIENTES);
		system(comando);

		strcpy(comando, "cat /var/lib/arpwatch/*.dat- | cut -f 1,2,3 >> ");
		strcat(comando, FICHERO_REGISTRO_CLIENTES);
		system(comando);

		//Leemos el fichero en el que hemos almacenado todos los datos
		contenido_del_fichero = lee_fichero(nombre_fichero);

		//Comprobamos que el fichero no está en el límite de líneas(NUM_FRASES)
		if (contenido_del_fichero.num_frases_fichero < LIMITE_LINEAS_LEIDAS){
```

```
                        //Comprobamos que ha aumentado el número de clientes registrado desde
la última vez que lo leímos
                        if (contenido_del_fichero.num_frases_fichero > num_lineas_anteriores)
{

                                //Cargamos los nuevos datos en la base de datos
                                inserta_en_BBDD(contenido_del_fichero, num_lineas_anteriores);

                        }
                }

                //Si estamos cerca del límite
                else {

                        //Vaciamos el fichero de registro de clientes
                        nuevo_fichero = vacia_fichero(nombre_fichero);

                        //Leemos el fichero en el que hemos almacenado todos lo datos
                        contenido_del_fichero = lee_fichero(nuevo_fichero);

                        //Comprobamos que no hemos superado el límite de líneas
                        if (contenido_del_fichero.num_frases_fichero < NUM_FRASES) {

                                //Comprobamos que ha aumentado el número de clientes
registrado desde la última vez que lo leimos
                                if         (contenido_del_fichero.num_frases_fichero         >
num_lineas_anteriores) {

                                        //Cargamos los nuevos datos en la base de datos
                                        inserta_en_BBDD(contenido_del_fichero,
num_lineas_anteriores);

                                }
                        }

                        /*-------------------------------------------------------
                                También es necesario vaciar los ficheros .dat ya
                                que si no se vacian en la próxima iteración se
                                volverá a llenar el fichero de registro ya que
                                se vuelca el contenido de todos los ficheros
                        -------------------------------------------------------*/
                        //Obtenemos todos los ficheros .dat existentes
                        system("ls /var/lib/arpwatch/*.dat > lista_ficheros");
                        system("ls /var/lib/arpwatch/*.dat- >> lista_ficheros");

                        //Leemos el fichero donde hemos almacenado todos los ficheros
                        lista_de_ficheros = lee_fichero("lista_ficheros");

                        //Recorremos todos los dicheros existentes
                        for (cont_aux_frases=0;
                                cont_aux_frases<lista_de_ficheros.num_frases_fichero;
                                cont_aux_frases++) {

                                //Vaciamos el fichero y lo almacenamos en otro auxiliar

        vacia_fichero(lista_de_ficheros.contenido_leido_del_fichero[cont_aux_frases][INICIO
]);
                        }

                        //Reiniciamos el número de líneas leídas para la próxima iteración
                        contenido_del_fichero.num_frases_fichero = INICIO;

                }

                //Almacenamos el número de líneas leidas
                num_lineas_anteriores = contenido_del_fichero.num_frases_fichero;
```

```
		//Esperamos un intervalo de tiempo hasta la próxima lectura
		sleep(INTERVALO_LECTURA);
	}

	//Liberamos memoria
	free(nuevo_fichero);
	free(nombre_fichero);
	free(comando);

	pthread_exit(NULL);
}
```

Func_aux_mod4.h

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 9-03-2017
Descripción: Fichero que contiene las declaraciones
de las funciones auxiliares y constantes del
cuarto módulo
-----------------------------------------*/
#include "constantes_y_librerias.h"
#include "func_aux.h"


#ifndef func_aux_mod4
#define func_aux_mod4

/*-----------------------------------------------------------
	Función que se encarga de introducir los valores
	deseados en la base de datos

	Recibe: Estructura del tipo CONTENIDO_FICHERO y
		un entero indicando la posición en la
		que se debe comenzar a recorrer la estructura
	Devuelve: Nada
-----------------------------------------------------------*/
void inserta_en_BBDD(struct CONTENIDO_FICHERO contenido_del_fichero, int frase_inicio);



#endif
```

Func_aux_mod4.c

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 9-03-2017
Descripción: Fichero que contiene las
implementaciones de las funciones auxiliares
del cuarto módulo
-----------------------------------------*/
#include "func_aux_mod4.h"


/*-----------------------------------------------------------
	Función que se encarga de introducir los valores
	deseados en la base de datos

	Recibe: Estructura del tipo CONTENIDO_FICHERO y
		un entero indicando la posición en la
		que se debe comenzar a recorrer la estructura

	Devuelve: Nada

	Caracterísitcas:
		Mediante un bucle que comienza en la posición
```

```
		que se le indica en el segundo argumento,
		crea el comando necesario para introducir
		en la base de datos los valores leidos de
		los ficheros .dat generados por la herramienta
		ARPWATCH
-----------------------------------------------------*/
void inserta_en_BBDD(struct CONTENIDO_FICHERO contenido_del_fichero, int frase_inicio) {

	//String para almacenar el comando
	char *comando = (char *)malloc(sizeof(char)*NUM_CARACTERES_REGLAS);
	//Variables auxiliares para recorrer la estrutura leida
	int cont_aux_frases;
	int cont_aux_palabras;


	//Inicializamos
	cont_aux_frases = INICIO;
	cont_aux_palabras = INICIO;


	//Recorremos las nuevas líneas para crear el comando que vamos a cargar en la base
de datos
	for (cont_aux_frases=frase_inicio;
			cont_aux_frases<contenido_del_fichero.num_frases_fichero;
			cont_aux_frases++) {

		/*-----------------------------------------------
			Construimos el comando para introducir
			los datos
		-----------------------------------------------*/

		//Indicamos la contraseña de la base de datos
		strcpy(comando, "export PGPASSWORD='");
		strcat(comando, CONTRASENIA_BBDD);

		//Indicamos el usuario con el que nos queremos conectar
		strcat(comando, "'; psql -U ");
		strcat(comando, USUARIO_BBDD);

		//Nombre de la base de datos a la que nos queremos conectar
		strcat(comando, " -d ");
		strcat(comando, NOMBRE_BBDD);

		//Insertamos los datos en la tabla
		strcat(comando, " -c \"INSERT INTO ");
		//Nombre de la tabla dentro de esa base de datos en la que queremos insertar
los datos
		strcat(comando, NOMBRE_TABLA_BBDD);
		strcat(comando, " VALUES (");

		//Recorremos los valores obtenidos de los ficheros .dat
		for (cont_aux_palabras=0;

	cont_aux_palabras<=contenido_del_fichero.palabras_por_frase[cont_aux_frases];
				cont_aux_palabras++) {

				//Insertamos los valores que queremos almacenar en la tabla

				//Comprobamos si estamos en la posición donde se almacena la
fecha
				if (cont_aux_palabras == FECHA){

					//Si estamos en la fecha debemos realizar previamente
una conversión
					strcat(comando, "to_timestamp(");
					strcat(comando,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras]);
					strcat(comando, ")");
				}
```

```
                        else {

                                //Para la dirección MAC e IP los valores deben ir entre
comillas simples
                                strcat(comando, "'");
                                strcat(comando,
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_frases][cont_aux_palabras]);
                                strcat(comando, "', ");
                        }

                }

        //Terminamos el comando
        strcat(comando, ");\"");

        //Redireccionamos la salida
        strcat(comando, " >> /tmp/salida_bbdd.txt");
        //Redireccionamos la salida de errores, en caso de existir
        strcat(comando, " 2>> /tmp/salida_bbdd.txt");

        //Ejecutamos el comando
        system(comando);

    }

    //Liberamos memoria
    free(comando);
}
```

Se ha creado adicionalmente el fichero *elimina_semaforos.c* para eliminar todos los semáforos que se crean en la ejecución del software.

```
/*-----------------------------------------
Autor: Francisco Javier Santos
Fecha: 13-03-2017
Descripción: Fichero que elimina los semáforos
creados en el programa
-----------------------------------------*/

#include "constantes_y_librerias.h"
#include "func_aux.h"

int main () {

//Eliminamos los semáforos
    elimina_semaforo(SEM_LECTURA);
    elimina_semaforo(SEM_SNORT);
    elimina_semaforo(SEM_SID);
    elimina_semaforo(SEM_REGLAS_SNORT);

return 0;
}
```

El código desarrollado para la interfaz web, es el siguiente;

En primer lugar, el fichero *Makefile* usado para compilar el código base de la interfaz web

```
#Maefile interfaz Web

all:    borraRegla.o    procesa_borrado_regla.o    muestraRegistroReglasMITM.o
muestraReglasSnort.o

borraRegla.o: borraRegla.c /home/pi/TFG/func_aux.c /home/pi/TFG/constantes_y_librerias.h
    gcc -g -W -Wall -o borraRegla.out borraRegla.c /home/pi/TFG/func_aux.c -pthread
```

```
procesa_borrado_regla.o:          procesa_borrado_regla.c          /home/pi/TFG/func_aux.c
/home/pi/TFG/func_aux_mod2.c                                  /home/pi/TFG/func_aux_mod2.h
/home/pi/TFG/constantes_y_librerias.h
       gcc    -g    -W    -Wall    -o    procesa_borrado_regla.out    procesa_borrado_regla.c
/home/pi/TFG/func_aux.c /home/pi/TFG/func_aux_mod2.c -pthread

muestraRegistroReglasMITM.o:      muestraRegistroReglasMITM.c      /home/pi/TFG/func_aux.c
/home/pi/TFG/constantes_y_librerias.h
       gcc  -g  -W  -Wall  -o  muestraRegistroReglasMITM.out  muestraRegistroReglasMITM.c
/home/pi/TFG/func_aux.c -pthread

muestraReglasSnort.o:             muestraReglasSnort.c             /home/pi/TFG/func_aux.c
/home/pi/TFG/constantes_y_librerias.h
       gcc    -g    -W    -Wall    -o    muestraReglasSnort.out    muestraReglasSnort.c
/home/pi/TFG/func_aux.c -pthread
```

El fichero hoja_estilo.css es la hoja de estilos usada para la interfaz

```
/*-----------------------------------------
       Fecha: 16-03-2017
       Autor: Francisco Javier Santos Rodríguez
       Descripción: Hoja de estilos que se
       aplicará a la interfaz web
-----------------------------------------*/


/*-------------------------------------
       Indicamos el color principal del
       fondo de la página
-------------------------------------*/
body {

       background-color: #3366ff
}

/*-------------------------------------
       Definimos el estialo para los
       distintos botones que vamos
       a crear
-------------------------------------*/
button {
       /*-------------------------------------
              En general, todos tendrán el
              mismo color de fondo, mismo
              borde(2px y con las esquinas
              redondeadas)
              Y tendrán un cambio de color
              cuando el ratón se pase por
              encima del botón
       -------------------------------------*/
       border: 2px solid #bfbfbf;
       border-radius:5px;

       position:relative;
       top: 20px ;
       left: 15%;
       float: left;

       -webkit-transition-duration: 0.5s;
       transition-duration: 0.4s;
       background-color: #8099b3;

       text-transform: capitalize;

}

button:hover {
```

```
	/*-----------------------------------
		Definimos el cambio de color de fondo del
		botón y de las letras
	-----------------------------------*/
	background-color: #bfbfbf;
	color: white;
}

.button1 {

	/*-----------------------------------
		Definimos una nueva posición para
		los botones pertenecientes a
		esta clase
	-----------------------------------*/
	position:absolute;
	top: 70px;
	left: 15%;
	float: left;

}

#button2 {
	/*-----------------------------------
		Definimos una nueva posición para
		el botón con este ID
	-----------------------------------*/
	position:relative;
	top: 20px;
	left: 2%;
	float: left;

}

/*-----------------------------------
	Definimos el tamaño y la posición
	que ocuparán en la interfaz el
	iframe
-----------------------------------*/
iframe {

	border-style: none;
	height:500px;
	width:85%;

	position: relative;
	top: 80px;
	left: 15%;
	float: left;

	background-color: #3366ff;
}


/*-----------------------------------
	Definimos el estilo para las
	diferentes clases que se han
	creado para las reglas
-----------------------------------*/
.reglas {

	position: absolute;
	top: 60px;
	left: 15%;
	float: left;
	/*-----------------------------------
		Indicamos el tipo de letra que
		queremos
	-----------------------------------*/
```

```
	text-align: left;
	text-transform: uppercase;
	font-family: "Arial", Sans-serif;
	font-size: 14px;

}
/*-------------------------------------
	Definimos el estilo con el que
	se mostrarán las reglas en la
	interfaz web
-------------------------------------*/
.info_regla {

	position: relative;
	top: 10px;
	left: 15%;

	text-align: left;
	text-transform: uppercase;
	font-family: "Arial", Sans-serif;
	font-size: 16px;
	color: white;
}

.borrar_regla {

	position: relative;
	left: 2%;
	background-color: #3399ff;
}


/*-------------------------------------
	Indicamos el estilo de las tablas
-------------------------------------*/
table {

	position: absolute;
	top: 60px;
	left: 15%;

	text-align: left;
	text-transform: uppercase;
	font-family: "Arial", Sans-serif;
	font-size: 13px;
	/*-----------------------------------
		Indicamos que queremos los bordes
		juntos para que no contenga bordes
		dobles
	-----------------------------------*/
	border-collapse: collapse;
	border: 4px solid #dad9d8;
}
/*-------------------------------------
	Indicamos la separación entre
	celdas y la colocación del texto
	en ellas
-------------------------------------*/
th, td {

	padding: 4px;
	text-align: center;
	border: 1px solid #dad9d8;

}
/*-------------------------------------
	Definimos el efecto que tendrán
	las filas (cambiar de color
	cuando se pasa el ratón por encima)
```

```
-------------------------------------*/
th, th:hover {
	background: #dad9d8;
	color: black;
}

tr:hover {

	background-color: #3399ff;
	color: white;
}

/*-------------------------------------
	Definimos la posición que queremos
	que ocupa la tabla con el registro
	de clientes
-------------------------------------*/
.tabla_clientes {
	position: absolute;
	top: 180px;
	left: 15%;
}

/*-------------------------------------
	Estilo de la lista de reglas
-------------------------------------*/
li {

	position: relative;
	top: 20px;
	margin: 4px;
	background-color: #3399ff;
}
/*-------------------------------------
	Estilo de los campos del
	formulario para filtrar los
	clientes registrados
-------------------------------------*/
.filtro {

	position: absolute;
	left: 15%;
	top: 100px;
}
/*-------------------------------------
	EStilo con el que se presentará
	el mensaje de error cuando queremos
	filtrar los clientes registrados
-------------------------------------*/
.error_sql {

	position: absolute;
	left: 15%;
	top:220px;
	color: white;
}
/*-------------------------------------
	Definimos la posición que
	ocupaŕa la imagen
-------------------------------------*/
img {
	position: absolute;
	left: 20%;
	top: 100px;
}

/*-------------------------------------
	Definimos la posición en la
	que aparecerán los mensajes de
```

```
	logs
-------------------------------------*/
.muestraLogs {

	position: absolute;
	left: 17%;
	top: 100px;
}

/*-------------------------------------
	Definimos el estilo que tendrán
	los mensajes de logs
-------------------------------------*/
.logs {

	border-left: 4px solid blue;
	border-bottom: 4px solid blue;
	background-color: #3399ff;

	width: 105%;
	height: 15px;


	text-align: left;
	text-transform: lowercase;
	font-family: "Arial", Sans-serif;
	font-size: 13px;
}
/*-------------------------------------
	Definimos el estilo con el que
	aparecerá el mensaje de fichero
	de log vacío
-------------------------------------*/
.log_vacio{

	text-align: left;
	text-transform: uppercase;
	font-family: "Arial", Sans-serif;
	font-size: 16px;

	color: white;
}
```

A continuación se detalla el código de la interfaz desarrolado en *C*

muestraRegistroReglasMITM.c

```
/*-----------------------------------------------
	Fecha: 17-03-2017
	Autor: Francisco Javier Santos Rodríguez
	Descripción: Página que se encarga de mostrar
	las reglas creadas para los ataques MITM
-----------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "/home/pi/TFG/func_aux.h"

int main () {

	//Estructura que almacenará los datos relativos al fichero
	struct CONTENIDO_FICHERO contenido_del_fichero;
	//Variables auxiliares para recorrer la estructura
	int cont_aux_lineas;
	int cont_aux_palabras_por_linea;

	//Inicializamos
```

```
	cont_aux_lineas = INICIO;
	cont_aux_palabras_por_linea = INICIO;

	printf("Content-Type:text/html\n\n");
	printf("<!DOCTYPE html>\n");

	printf("<html>\n");

		printf("<head>\n");
			/*-----------------------------------------
				Añadimos el logo en la cabecera
			-----------------------------------------*/
			printf("<link	rel=\"shortcut	icon\"	type=\"image/png\"
href=\"../cgi/logo.png\"/>\n");

			/*---------------------------------------------------
				Definimos en la cabecera el título de la página
				y la hoja de estilos que se aplicará
			---------------------------------------------------*/

			printf("<link		rel=\"stylesheet\"		type=\"text/css\"
href=\"../cgi/hoja_estilo.css\">\n");
			printf("<title>Gesti&oacuten WIPS </title>\n");

		printf("</head>\n");

		printf("<body>\n");

			//CREAMOS LA BOTONERA SUPERIOR
			printf("<p>\n");

				/*-----------------------------------------------------
					Indicamos la dirección a la que nos dirigimos si se
pulsa
					ese botón, el método con el que se realizará la
petición y
					el estilo del botón, que será en línea para que todos
aparezcan
					en una única fila en la parte superior
				-----------------------------------------------------*/
				printf("<form	method=\"get\"	action=\"../wips.html\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Inicio\n");
					printf("</button>\n");

				printf("</form>\n");

				printf("<form	method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraReglasSnort.out\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Reglas Snort\n");
					printf("</button>\n");

				printf("</form>\n");

				printf("<form	method=\"get\"
action=\"http://10.10.10.2/wips/cgi/muestraReglasIptables.html\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Reglas Iptables\n");
					printf("</button>\n");

				printf("</form>\n");
```

```
                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Clientes Registrados\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraRegistroReglasMITM.out\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Registro Reglas MITM\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Logs WIPS\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/borraRegla.out\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Borrar Reglas\n");
                                printf("</button>\n");

                        printf("</form>\n");

                printf("</p>\n");


                printf("<form   method=\"get\"   action=\"http://10.10.10.2/wips/cgi-
bin/muestraRegistroReglasMITM.out\" class=\"reglas\">\n");

                                //Leemos el fichero de las reglas creadas para los
ataques MITM
                                contenido_del_fichero                           =
lee_fichero(FICHERO_REGISTRO_REGLAS_MITM);

                                //Si no hemos leido ninguna línea, indicamos que no hay
reglas
                                if (contenido_del_fichero.num_frases_fichero == INICIO)
                                        printf("<p      class=\"info_regla\">No      hay
registro<br></p>\n");


                                else {

                                        //Creamos una tabla para mostrar los registros
                                        printf("<table>\n");

                                                //Indicamos los valores de cabecera
                                                printf("<tr>\n");
                                                        printf("<th>Hora</th>\n");
                                                        printf("<th>Accion</th>\n");
                                                        printf("<th>Protocolo</th>\n");
                                                        printf("<th>IP</th>\n");
                                                        printf("<th>Dir.                en
Origen</th>\n");
```

```
                                                printf("<th>Dir.                con Puerto</th>\n");
                                        printf("</tr>\n");

                                        //Recorremos la estructura tras haber leído el fichero
                                        for (cont_aux_lineas=0;
                                                cont_aux_lineas< contenido_del_fichero.num_frases_fichero;
                                                cont_aux_lineas++){

                                                //Para cada línea, creamos una nueva fila
                                                printf("<tr>\n");

                                                        //Dentro de cada fila metemos cada palabra en una celda distinta
                                                        for (cont_aux_palabras_por_linea=0;
        cont_aux_palabras_por_linea<=contenido_del_fichero.palabras_por_frase[cont_aux_lineas];
        cont_aux_palabras_por_linea++) {
                                                                printf("<td>%s</td>", contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][cont_aux_palabras_por_linea]);
                                                        }

                                                printf("</tr>\n");
                                        }

                                printf("</table>\n");
                        }

                        //Botón para acutalizar el registro de reglas
                        printf("<button type=\"submit\">\n");
                                printf("Actualizar registro\n");
                        printf("</button>\n");


                printf("</form>\n");
        printf("</body>\n");

        printf("</html>\n");

return 0;
}
```

muestraReglasSnort.c

```
/*-----------------------------------------------
        Fecha: 18-03-2017
        Autor: Francisco Javier Santos Rodríguez
        Descripción: Página que muestra las reglas
        definidas en Snort
-----------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "/home/pi/TFG/func_aux.h"

int main () {

        //Estructura que almacenará los datos relativos al fichero
        struct CONTENIDO_FICHERO contenido_del_fichero;
        //Variables auxiliares para recorrer la estructura
```

```
	int cont_aux_lineas;
	int cont_aux_palabras_por_linea;

	//Inicializamos
	cont_aux_lineas = INICIO;
	cont_aux_palabras_por_linea = INICIO;

	printf("Content-Type:text/html\n\n");
	printf("<!DOCTYPE html>\n");

	printf("<html>\n");

		printf("<head>\n");

			/*-----------------------------------------
				Añadimos el logo en la cabecera
			-----------------------------------------*/
			printf("<link	rel=\"shortcut	icon\"	type=\"image/png\"
href=\"../cgi/logo.png\"/>\n");

			/*---------------------------------------------------
				Definimos en la cabecera el título de la página
				y la hoja de estilos que se aplicará
			---------------------------------------------------*/
			printf("<link		rel=\"stylesheet\"		type=\"text/css\"
href=\"../cgi/hoja_estilo.css\">\n");
			printf("<title>Gesti&oacuten WIPS </title>\n");
		printf("</head>\n");

		printf("<body>\n");

			//CREAMOS LA BOTONERA SUPERIOR
			printf("<p>\n");

				/*-------------------------------------------------------
					Indicamos la dirección a la que nos dirigimos si se
pulsa
					ese botón, el método con el que se realizará la
petición y
					el estilo del botón, que será en línea para que todos
aparezcan
					en una única fila en la parte superior
				-------------------------------------------------------*/
				printf("<form	method=\"get\"	action=\"../wips.html\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Inicio\n");
					printf("</button>\n");

				printf("</form>\n");

				printf("<form	method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraReglasSnort.out\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Reglas Snort\n");
					printf("</button>\n");

				printf("</form>\n");

				printf("<form	method=\"get\"
action=\"http://10.10.10.2/wips/cgi/muestraReglasIptables.html\"
style=\"display:inline\">\n");

					printf("<button type=\"submit\">\n");
						printf("Reglas Iptables\n");
					printf("</button>\n");
```

```
                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Clientes Registrados\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraRegistroReglasMITM.out\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Registro Reglas MITM\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Logs WIPS\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                   method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/borraRegla.out\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Borrar Reglas\n");
                                printf("</button>\n");

                        printf("</form>\n");

                printf("</p>\n");


                printf("<form   method=\"get\"   action=\"http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out\" class=\"reglas\">\n");

                                //Leemos el contenido del fichero
                                contenido_del_fichero                                   =
lee_fichero(FICHERO_REGLAS_SNORT);

                                //Creamos una lista no ordenada
                                printf("<ul>\n");

                                //Recorremos los elementos leidos del ficheros
                                for (cont_aux_lineas=0;
                                        cont_aux_lineas<
contenido_del_fichero.num_frases_fichero;
                                        cont_aux_lineas++){

                                                //Insertamos cada elementos en un punto
nuevo
                                                printf("<li>\n");
                                                for (cont_aux_palabras_por_linea=0;

      cont_aux_palabras_por_linea<=contenido_del_fichero.palabras_por_frase[cont_aux_line
as];
                                                                cont_aux_palabras_por_linea++) {
```

```
                                                        printf("%s                          ",
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][cont_aux_palabras_por_
linea]);
                                                }
                                                printf("</li>\n");

                                        printf("<br>\n");
                                }

                                printf("</ul>\n");

                                //En el caso que no hayamos leido ninguna regla, lo
indicamos
                                if (contenido_del_fichero.num_frases_fichero == INICIO)
                                        printf("<p      class=\"info_regla\">No      hay
reglas<br></p>\n");

                                //Botón que permite actualizar las reglas
                                printf("<button type=\"submit\" id=\"button2\">\n");
                                        printf("Actualizar reglas\n");
                                printf("</button>\n");


                        printf("</form>\n");
                printf("</body>\n");

        printf("</html>\n");

return 0;
}
```

procesa_borrado_regla.c

```
/*-----------------------------------------------
        Fecha: 18-03-2017
        Autor: Francisco Javier Santos Rodríguez
        Descripción: Página que muestra las reglas
        definidas en Snort
-----------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "/home/pi/TFG/func_aux.h"

int main () {

        //Estructura que almacenará los datos relativos al fichero
        struct CONTENIDO_FICHERO contenido_del_fichero;
        //Variables auxiliares para recorrer la estructura
        int cont_aux_lineas;
        int cont_aux_palabras_por_linea;

        //Inicializamos
        cont_aux_lineas = INICIO;
        cont_aux_palabras_por_linea = INICIO;

        printf("Content-Type:text/html\n\n");
        printf("<!DOCTYPE html>\n");

        printf("<html>\n");

                printf("<head>\n");

                        /*----------------------------------------
                                Añadimos el logo en la cabecera
                        ----------------------------------------*/
                        printf("<link      rel=\"shortcut      icon\"      type=\"image/png\"
href=\"../cgi/logo.png\"/>\n");
```

```
                        /*-------------------------------------------------------
                               Definimos en la cabecera el título de la página
                               y la hoja de estilos que se aplicará
                        -------------------------------------------------------*/
                        printf("<link          rel=\"stylesheet\"          type=\"text/css\"
href=\"../cgi/hoja_estilo.css\">\n");
                        printf("<title>Gesti&oacuten WIPS </title>\n");
                printf("</head>\n");

                printf("<body>\n");

                        //CREAMOS LA BOTONERA SUPERIOR
                        printf("<p>\n");

                               /*-------------------------------------------------------
                                      Indicamos la dirección a la que nos dirigimos si se
pulsa
                                      ese botón, el método con el que se realizará la
petición y
                                      el estilo del botón, que será en línea para que todos
aparezcan
                                      en una única fila en la parte superior
                               -------------------------------------------------------*/
                               printf("<form      method=\"get\"      action=\"../wips.html\"
style=\"display:inline\">\n");

                                      printf("<button type=\"submit\">\n");
                                             printf("Inicio\n");
                                      printf("</button>\n");

                               printf("</form>\n");

                               printf("<form                                        method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraReglasSnort.out\"
style=\"display:inline\">\n");

                                      printf("<button type=\"submit\">\n");
                                             printf("Reglas Snort\n");
                                      printf("</button>\n");

                               printf("</form>\n");

                               printf("<form                                        method=\"get\"
action=\"http://10.10.10.2/wips/cgi/muestraReglasIptables.html\"
style=\"display:inline\">\n");

                                      printf("<button type=\"submit\">\n");
                                             printf("Reglas Iptables\n");
                                      printf("</button>\n");

                               printf("</form>\n");

                               printf("<form                                        method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp\"
style=\"display:inline\">\n");

                                      printf("<button type=\"submit\">\n");
                                             printf("Clientes Registrados\n");
                                      printf("</button>\n");

                               printf("</form>\n");

                               printf("<form                                        method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraRegistroReglasMITM.out\"
style=\"display:inline\">\n");

                                      printf("<button type=\"submit\">\n");
                                             printf("Registro Reglas MITM\n");
```

```
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                    method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Logs WIPS\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                    method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/borraRegla.out\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Borrar Reglas\n");
                                printf("</button>\n");

                        printf("</form>\n");

                printf("</p>\n");


                printf("<form   method=\"get\"   action=\"http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out\" class=\"reglas\">\n");

                                //Leemos el contenido del fichero
                                contenido_del_fichero                                =
lee_fichero(FICHERO_REGLAS_SNORT);

                                //Creamos una lista no ordenada
                                printf("<ul>\n");

                                //Recorremos los elementos leidos del ficheros
                                for (cont_aux_lineas=0;
                                        cont_aux_lineas<
contenido_del_fichero.num_frases_fichero;
                                        cont_aux_lineas++){

                                                //Insertamos cada elementos en un punto
nuevo
                                                printf("<li>\n");
                                                for (cont_aux_palabras_por_linea=0;

        cont_aux_palabras_por_linea<=contenido_del_fichero.palabras_por_frase[cont_aux_line
as];
                                                                cont_aux_palabras_por_linea++) {

                                                                printf("%s                       ",
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][cont_aux_palabras_por_
linea]);
                                                }
                                                printf("</li>\n");

                                        printf("<br>\n");
                                }

                                printf("</ul>\n");

                                //En el caso que no hayamos leido ninguna regla, lo
indicamos
                                if (contenido_del_fichero.num_frases_fichero == INICIO)
                                        printf("<p      class=\"info_regla\">No      hay
reglas<br></p>\n");

                                //Botón que permite actualizar las reglas
                                printf("<button type=\"submit\" id=\"button2\">\n");
```

```
                                        printf("Actualizar reglas\n");
                                    printf("</button>\n");


                        printf("</form>\n");
                printf("</body>\n");

        printf("</html>\n");

return 0;
}
```

borraRegla.c

```
/*-----------------------------------------------
        Fecha: 17-03-2017
        Autor: Francisco Javier Santos Rodríguez
        Descripción: Página que se encarga de mostrar
        las reglas definidas en Snort y mediante
        un botón selecciona cual queremos eliminar
-----------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "/home/pi/TFG/func_aux.h"

int main () {

        //Estructura que almacenará los datos relativos al fichero
        struct CONTENIDO_FICHERO contenido_del_fichero;
        //Variables auxiliares para recorrer la estructura
        int cont_aux_lineas;
        int cont_aux_palabras_por_linea;
        //Variable para indicar el ID de cada regla
        int id_regla;

        //Inicializamos
        cont_aux_lineas = INICIO;
        cont_aux_palabras_por_linea = INICIO;
        id_regla = INICIO;

        printf("Content-Type:text/html\n\n");
        printf("<!DOCTYPE html>\n");

        printf("<html>\n");

                printf("<head>\n");
                        /*-----------------------------------------
                                Añadimos el logo en la cabecera
                        -----------------------------------------*/
                        printf("<link      rel=\"shortcut      icon\"      type=\"image/png\"
href=\"../cgi/logo.png\"/>\n");
                        /*---------------------------------------------------------
                                Definimos en la cabecera el título de la página
                                y la hoja de estilos que se aplicará
                        ---------------------------------------------------------*/
                        printf("<link           rel=\"stylesheet\"           type=\"text/css\"
href=\"../cgi/hoja_estilo.css\">\n");
                        printf("<title>Gesti&oacuten WIPS </title>\n");

                printf("</head>\n");

                printf("<body>\n");
                        //CREAMOS LA BOTONERA SUPERIOR
                        printf("<p>\n");

                                /*-------------------------------------------------
```

```
                                Indicamos la dirección a la que nos dirigimos si se
pulsa
                                ese botón, el método con el que se realizará la
petición y
                                el estilo del botón, que será en línea para que todos
aparezcan
                                en una única fila en la parte superior
                        ----------------------------------------------------*/
                        printf("<form      method=\"get\"      action=\"../wips.html\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Inicio\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraReglasSnort.out\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Reglas Snort\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/cgi/muestraReglasIptables.html\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Reglas Iptables\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Clientes Registrados\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/muestraRegistroReglasMITM.out\"
style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Registro Reglas MITM\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp\" style=\"display:inline\">\n");

                                printf("<button type=\"submit\">\n");
                                        printf("Logs WIPS\n");
                                printf("</button>\n");

                        printf("</form>\n");

                        printf("<form                                          method=\"get\"
action=\"http://10.10.10.2/wips/cgi-bin/borraRegla.out\" style=\"display:inline\">\n");
```

```
                                    printf("<button type=\"submit\">\n");
                                          printf("Borrar Reglas\n");
                                    printf("</button>\n");

                              printf("</form>\n");

                        printf("</p>\n");


                        printf("<form   method=\"get\"   action=\"http://10.10.10.2/wips/cgi-
bin/procesa_borrado_regla.out\" class=\"reglas\">\n");

                                    //Leemos el fichero de reglas de Snort
                                    contenido_del_fichero                                    =
lee_fichero(FICHERO_REGLAS_SNORT);

                                    //Recorremos la estructura que se devuelve al leer el
fichero
                                    for (cont_aux_lineas=0;
                                          cont_aux_lineas<
contenido_del_fichero.num_frases_fichero;
                                          cont_aux_lineas++){

                                          //Creamos un nuevo párrafo por cada regla
                                          printf("<p class=\"borrar_regla\">\n");

                                          //Incluimos un botón de tipo radio para poder
seleccionar la regla que queremos eliminar
                                          printf("<input    type=\"radio\"    name=\"regla\"
id=\"%d\" value=\"%d\">\n", id_regla, id_regla);
                                          id_regla++;

                                                //Imprimimos la regla
                                                for (cont_aux_palabras_por_linea=0;

      cont_aux_palabras_por_linea<=contenido_del_fichero.palabras_por_frase[cont_aux_line
as];
                                                      cont_aux_palabras_por_linea++) {

                                                      printf("%s                          ",
contenido_del_fichero.contenido_leido_del_fichero[cont_aux_lineas][cont_aux_palabras_por_
linea]);
                                                }

                                          printf("<br>\n");
                                          printf("</p>\n");
                                    }
                                    //Si no se ha leido ninguna regla, lo indicamos
                                    if (contenido_del_fichero.num_frases_fichero == INICIO)
                                          printf("<p      class=\"info_regla\">No      hay
reglas<br></p>\n");

                                    //Botón  para  enviar  la  petición  de  la  regla  que
deseamos elimianr
                                    printf("<button type=\"submit\" id=\"button2\">\n");
                                          printf("Borrar regla\n");
                                    printf("</button>\n");


                        printf("</form>\n");
                  printf("</body>\n");

      printf("</html>\n");

return 0;
}
```

Por último, se detalla el código desarrollado en *JSP* y *HTML*

muestraClientesRegistrados.jsp

```
<!--
	Fecha: 19-03-2017
	Autor: Francisco Javier Santos Rodríguez
	Descripción: Página que se encarga de mostrar
	la información de la BBDD de los clientes registrados
	y permite filtrarlos
-->

<!-- Indicamos que vamos a introducir código en Java -->
<%@ page language="java" import="java.sql.*, java.io.*, java.util.*"
contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<%
	try {

		Class.forName("org.postgresql.Driver").newInstance();

		//Indicamos los parámetros para conectarnos a la BBDD
		String url = "jdbc:postgresql://localhost:5432/clientes_asociados";
		String user = "clientes";
		String pass = "clientes";

		//Establecemos la conexión
		Connection conn = DriverManager.getConnection(url,user,pass);

		//Creamos el objeto para enviar sentencias
		Statement st = conn.createStatement();

%>

<!DOCTYPE html>
<html>

	<head>

		<!--
			Añadimos el logo en la cabecera
		-->
		<link rel="shortcut icon" type="image/png" href="../cgi/logo.png"/>
		<!--
			Definimos en la cabecera el título de la página
			y la hoja de estilos que se aplicará
		-->
		<link rel="stylesheet" type="text/css" href="../cgi/hoja_estilo.css">
		<title> Gesti&oacuten WIPS </title>

	</head>

	<body>
			<!--
				CREAMOS LA BOTONERA SUPERIOR
			-->
			<p>

			<!--
				Indicamos la dirección a la que nos dirigimos si se pulsa
				ese botón, el método con el que se realizará la petición y
				el estilo del botón, que será en línea para que todos
aparezcan
				en una única fila en la parte superior
			-->
			<form method="get" action="../wips.html" style="display:inline">

				<button type="submit">
					Inicio
```

```
                        </button>
                    </form>

                    <form        method="get"        action="http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out" style="display:inline">

                        <button type="submit">
                            Reglas Snort
                        </button>

                    </form>

                    <form                                                    method="get"
action="http://10.10.10.2/wips/cgi/muestraReglasIptables.html" style="display:inline">

                        <button type="submit">
                            Reglas Iptables
                        </button>

                    </form>

                    <form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">

                        <button type="submit">
                            Clientes Registrados
                        </button>

                    </form>

                    <form        method="get"        action="http://10.10.10.2/wips/cgi-
bin/muestraRegistroReglasMITM.out" style="display:inline">

                        <button type="submit">
                            Registro Reglas MITM
                        </button>

                    </form>

                    <form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp" style="display:inline">

                        <button type="submit">
                            Logs WIPS
                        </button>

                    </form>

                    <form        method="get"        action="http://10.10.10.2/wips/cgi-
bin/borraRegla.out" style="display:inline">

                        <button type="submit">
                            Borrar Reglas
                        </button>

                    </form>

            </p>


            <form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">


            <!--
                Creamos la tabla donde se motrará la información de los clientes
            -->
```

```
<table class="tabla_clientes">
	<tr>
		<!--
			Indicamos los valores de cabecera
		-->
		<th>
			Direccion MAC
		</th>
		<th>
			Direccion IP
		</th>
		<th>
			Fecha registro
		</th>
	</tr>

	<%
		/*-------------------------------------------------
			Obtenemos los parámetros del formulario que
			nos permite filtrar la información
		-------------------------------------------------*/

		String mac = request.getParameter("macaddr");
		String ip = request.getParameter("ip");
		//Creamos la primera parte de la consulta
		String sentencia = "SELECT * from datos_clientes ";


		if (mac != null && ip != null ) {

			//Comprobamos si hemos introducido algún valor
			if (ip.length() > 0 || mac.length()>0) {

				//Añadimos la condición
				sentencia+="where ";
			}

			//Si queremos filtrar por la MAC
			if (mac.length() > 0) {
				/*----------------------------------------
--

					Añadimos la condición para mostrar
solo

					los que contienen esa dirección MAC
				----------------------------------------
-*/

				sentencia+= "dir_mac='" + mac + "'";
			}

			//Si hemos introducido valores en ambos campos
			if (ip.length() > 0 && mac.length()>0){

				/*--------------------------------
					Añadimos "and" para indicar
					que vamos a introducir otra
					condición
				--------------------------------*/
				sentencia+= " and ";
			}

			//Si hemos introducio algún valor en la IP
			if (ip.length() > 0) {

				//Añadimos esa condición
				sentencia+= "dir_ip ='" + ip + "'";
			}

		}
		//Añadimos el terminador
```

```
                                sentencia+=";";

                                //Realizamos la consulta y la almacenamos
                                ResultSet rs = st.executeQuery(sentencia);

                                //Recorremos el resultado obtenido
                                while (rs.next()){

                                        //Incluimos en una fila nueva cada resultado
                                        out.println("<tr>");

                                                //Incluimos en cada celda el valor de cada
columnna
                                                out.println("<td>"  +  rs.getObject(1)  +
"</td>");
                                                out.println("<td>"  +  rs.getObject(2)  +
"</td>");
                                                out.println("<td>"  +  rs.getObject(3)  +
"</td>");

                                        out.println("</tr>");
                                }
                                        //Cerramos la conexión
                                        rs.close();
                                        st.close();
                                        conn.close();

                                } catch (SQLException e) {

                                                //En  caso  de  error,  imprimimos  el
siguiente mensaje de error
                                                out.println("<p  class  =  \"error_sql\">La
consulta realizada es erronea, presione <b>Actualizar registro</b> para ver todos los
registros</p>");

                                        }
                        %>
                </table>

                <!--
                        Botón para actualizar el registro de clientes
                -->
                <button type="submit" class="button1">
                        Actualizar registro
                </button>



        </form>



        <!--
                CREAMOS EL FORMULARIO QUE NOS PERMITIRÁ FILTRAR LA INFORMACIÓN DE LA
BBDD EN FUNCIÓN DE LOS DATOS DE LOS CLIENTES
        -->
        <form                                                           method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp">

                <!--
                        Creamos los campos a rellenar del formulario para filtrar la
información
                -->
                <p class="filtro">

                        Direccion MAC:
                        <input type="text" name="macaddr">

                        Direccion IP:
```

```
                        <input type="text" name="ip">

                    </p>

                    <!--
                        Botón con el que enviaremos los datos de la consulta
                    -->
                    <button                 type="submit"                 class="button1"
style="display;inline;position:absolute;left: 61%;top:116px;" ">
                        Filtar
                    </button>


            </form>

      </body>

</html>
```

muestraLogsWIPS.jsp

```
<!--
      Fecha: 19-03-2017
      Autor: Francisco Javier Santos Rodríguez
      Descripción: Página que se encarga de mostrar
      la información de la BBDD de los clientes registrados
      y permite filtrarlos
-->

<!-- Indicamos que vamos a introducir código en Java -->
<%@     page     language="java"     import="java.sql.*,     java.io.*,     java.util.*"
contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<%
      try {

            Class.forName("org.postgresql.Driver").newInstance();

            //Indicamos los parámetros para conectarnos a la BBDD
            String url = "jdbc:postgresql://localhost:5432/clientes_asociados";
            String user = "clientes";
            String pass = "clientes";

            //Establecemos la conexión
            Connection conn = DriverManager.getConnection(url,user,pass);

            //Creamos el objeto para enviar sentencias
            Statement st = conn.createStatement();

%>

<!DOCTYPE html>
<html>

      <head>

            <!--
                  Añadimos el logo en la cabecera
            -->
            <link rel="shortcut icon" type="image/png" href="../cgi/logo.png"/>
            <!--
                  Definimos en la cabecera el título de la página
                  y la hoja de estilos que se aplicará
            -->
            <link rel="stylesheet" type="text/css" href="../cgi/hoja_estilo.css">
            <title> Gesti&oacuten WIPS </title>
```

```
	</head>

	<body>
			<!--
				CREAMOS LA BOTONERA SUPERIOR
			-->
			<p>

			<!--
				Indicamos la dirección a la que nos dirigimos si se pulsa
				ese botón, el método con el que se realizará la petición y
				el estilo del botón, que será en línea para que todos
aparezcan
				en una única fila  en la parte superior
			-->
			<form method="get" action="../wips.html" style="display:inline">

				<button type="submit">
					Inicio

				</button>
			</form>

			<form		method="get"		action="http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out" style="display:inline">

				<button type="submit">
					Reglas Snort
				</button>

			</form>

			<form                                                          method="get"
action="http://10.10.10.2/wips/cgi/muestraReglasIptables.html" style="display:inline">

				<button type="submit">
					Reglas Iptables
				</button>

			</form>

			<form                                                          method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">

				<button type="submit">
					Clientes Registrados
				</button>

			</form>

			<form		method="get"		action="http://10.10.10.2/wips/cgi-
bin/muestraRegistroReglasMITM.out" style="display:inline">

				<button type="submit">
					Registro Reglas MITM
				</button>

			</form>

			<form                                                          method="get"
action="http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp" style="display:inline">

				<button type="submit">
					Logs WIPS
				</button>

			</form>
```

```
                    <form        method="get"        action="http://10.10.10.2/wips/cgi-
bin/borraRegla.out" style="display:inline">

                          <button type="submit">
                                Borrar Reglas
                          </button>

                    </form>

             </p>


             <form                                                          method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">


             <!--
                    Creamos la tabla donde se motrará la información de los clientes
             -->
                    <table class="tabla_clientes">
                          <tr>
                                <!--
                                      Indicamos los valores de cabecera
                                -->
                                <th>
                                      Direccion MAC
                                </th>
                                <th>
                                      Direccion IP
                                </th>
                                <th>
                                      Fecha registro
                                </th>
                          </tr>

                          <%
                                /*-------------------------------------------------
                                      Obtenemos los parámetros del formulario que
                                      nos permite filtrar la información
                                -------------------------------------------------*/

                                String mac = request.getParameter("macaddr");
                                String ip = request.getParameter("ip");
                                //Creamos la primera parte de la consulta
                                String sentencia = "SELECT * from datos_clientes ";


                                if (mac != null && ip != null ) {

                                      //Comprobamos si hemos introducido algún valor
                                      if (ip.length() > 0 || mac.length()>0) {

                                            //Añadimos la condición
                                            sentencia+="where ";
                                      }

                                      //Si queremos filtrar por la MAC
                                      if (mac.length() > 0) {
                                            /*---------------------------------------
--
                                                  Añadimos la condición para mostrar
solo
                                                  los que contienen esa dirección MAC
                                            ---------------------------------------
-*/
                                            sentencia+= "dir_mac='" + mac + "'";
                                      }
```

```
                                        //Si hemos introducido valores en ambos campos
                                        if (ip.length() > 0 && mac.length()>0){

                                                /*--------------------------------
                                                        Añadimos "and" para indicar
                                                        que vamos a introducir otra
                                                        condición
                                                --------------------------------*/
                                                sentencia+= " and ";
                                        }

                                        //Si hemos introducio algún valor en la IP
                                        if (ip.length() > 0) {

                                                //Añadimos esa condición
                                                sentencia+= "dir_ip ='" + ip + "'";
                                        }

                                }
                                //Añadimos el terminador
                                sentencia+=";";

                                //Realizamos la consulta y la almacenamos
                                ResultSet rs = st.executeQuery(sentencia);

                                //Recorremos el resultado obtenido
                                while (rs.next()){

                                        //Incluimos en una fila nueva cada resultado
                                        out.println("<tr>");

                                                //Incluimos en cada celda el valor de cada
columnna
                                                out.println("<td>"   +   rs.getObject(1)   +
"</td>");
                                                out.println("<td>"   +   rs.getObject(2)   +
"</td>");
                                                out.println("<td>"   +   rs.getObject(3)   +
"</td>");


                                        out.println("</tr>");
                                }
                                        //Cerramos la conexión
                                        rs.close();
                                        st.close();
                                        conn.close();

                                } catch (SQLException e) {

                                                //En   caso   de   error,   imprimimos   el
siguiente mensaje de error
                                                out.println("<p  class  =  \"error_sql\">La
consulta realizada es erronea, presione <b>Actualizar registro</b> para ver todos los
registros</p>");

                                        }
                                %>
                        </table>

                        <!--
                                Botón para actualizar el registro de clientes
                        -->
                        <button type="submit" class="button1">
                                Actualizar registro
                        </button>



                </form>
```

```
		<!--
			CREAMOS EL FORMULARIO QUE NOS PERMITIRÁ FILTRAR LA INFORMACIÓN DE LA
BBDD EN FUNCIÓN DE LOS DATOS DE LOS CLIENTES
		-->
		<form                                                          method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp">

			<!--
				Creamos los campos a rellenar del formulario para filtrar la
información
			-->
			<p class="filtro">

				Direccion MAC:
				<input type="text" name="macaddr">

				Direccion IP:
				<input type="text" name="ip">

			</p>

			<!--
				Botón con el que enviaremos los datos de la consulta
			-->
			<button                type="submit"                class="button1"
style="display;inline;position:absolute;left: 61%;top:116px;" ">
				Filtar
			</button>


		</form>

	</body>

</html>
```

muestraReglasIptables.html

```
<!--
	Fecha: 16-03-2017
	Autor: Francisco Javier Santos Rodríguez
	Descripción: Página que se encarga de
	mostrar las reglas definidas en IPTABLES
-->

<!DOCTYPE html>
<html>

	<head>
		<!--
			Añadimos el logo en la cabecera
		-->
		<link rel="icon" type="image/png" href="../cgi/logo.png"/>
		<!--
			Definimos en la cabecera el título de la página
			y la hoja de estilos que se aplicará
		-->
		<link rel="stylesheet" type="text/css" href="../cgi/hoja_estilo.css">
		<title> Gesti&oacuten WIPS </title>
	</head>

	<body>
		<!--
			CREAMOS LA BOTONERA SUPERIOR
```

```
		-->
		<p>
		<!--
			Indicamos la dirección a la que nos dirigimos si se pulsa
			ese botón, el método con el que se realizará la petición y
			el estilo del botón, que será en línea para que todos aparezcan
			en una única fila en la parte superior
		-->
			<form method="get" action="../wips.html" style="display:inline">

				<button type="submit">
					Inicio
				</button>

			</form>

			<form		method="get"		action="http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out" style="display:inline">

				<button type="submit">
					Reglas Snort
				</button>

			</form>

			<form                                                    method="get"
action="http://10.10.10.2/wips/cgi/muestraReglasIptables.html" style="display:inline">

				<button type="submit">
					Reglas Iptables
				</button>

			</form>

			<form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">

				<button type="submit">
					Clientes Registrados
				</button>

			</form>

			<form		method="get"		action="http://10.10.10.2/wips/cgi-
bin/muestraRegistroReglasMITM.out" style="display:inline">

				<button type="submit">
					Registro Reglas MITM
				</button>

			</form>

			<form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp" style="display:inline">

				<button type="submit">
					Logs WIPS
				</button>

			</form>

			<form		method="get"		action="http://10.10.10.2/wips/cgi-
bin/borraRegla.out" style="display:inline">

				<button type="submit">
					Borrar Reglas
				</button>
```

```
                </form>

            </p>

            <p>

                <form                                                    method="get"
action="http://10.10.10.2/wips/cgi/muestraReglasIptables.html" style="display:inline">

                    <!-- Botón para actualizar el registro de reglas de
iptables -->

                    <button type="submit" class="button1">
                        Actualizar reglas
                    </button>

                </form>

            </p>

            <div>

                <p>

                    <iframe src="reglas_iptables">
                    </iframe>

                </p>

            </div>

        </body>

</html>
```

wips.html

```
<!--
        Fecha: 16-03-2017
        Autor: Francisco Javier Santos Rodríguez
        Descripción: Página principal de la
        interfaz web
-->

<!DOCTYPE html>
<html>

        <head>
            <!--
                Añadimos el logo en la cabecera
            -->
            <link rel="icon" type="image/png" href="cgi/logo.png"/>
            <!--
                Definimos en la cabecera el título de la página
                y la hoja de estilos que se aplicará
            -->
            <link rel="stylesheet" type="text/css" href="cgi/hoja_estilo.css">
            <title> Gesti&oacuten WIPS </title>
        </head>

        <body>
            <!--
                CREAMOS LA BOTONERA SUPERIOR
            -->
            <p>
            <!--
                Indicamos la dirección a la que nos dirigimos si se pulsa
```

```
                ese botón, el método con el que se realizará la petición y
                el estilo del botón, que será en línea para que todos aparezcan
                en una única fila en la parte superior
        -->
                <form method="get" action="wips.html" style="display:inline">

                        <button type="submit">
                                Inicio
                        </button>

                </form>

                <form         method="get"         action="http://10.10.10.2/wips/cgi-
bin/muestraReglasSnort.out" style="display:inline">

                        <button type="submit">
                                Reglas Snort
                        </button>

                </form>

                <form                                                    method="get"
action="http://10.10.10.2/wips/cgi/muestraReglasIptables.html" style="display:inline">

                        <button type="submit">
                                Reglas Iptables
                        </button>

                </form>

                <form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraClientesRegistrados.jsp"
style="display:inline">

                        <button type="submit">
                                Clientes Registrados
                        </button>

                </form>

                <form         method="get"         action="http://10.10.10.2/wips/cgi-
bin/muestraRegistroReglasMITM.out" style="display:inline">

                        <button type="submit">
                                Registro Reglas MITM
                        </button>

                </form>

                <form                                                    method="get"
action="http://10.10.10.2/wips/jsp/muestraLogsWIPS.jsp" style="display:inline">

                        <button type="submit">
                                Logs WIPS
                        </button>

                </form>

                <form         method="get"         action="http://10.10.10.2/wips/cgi-
bin/borraRegla.out" style="display:inline">

                        <button type="submit">
                                Borrar Reglas
                        </button>

                </form>

        </p>
```

```
		<!--
			Insertamos la imagen principal
		-->
		<div>
			<img src="cgi/logo.png" />
		</div>
	</body>

</html>
```

REFERENCIAS

[1] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Man-in-the-middle_attack. [Último acceso: 06 Abirl 2017].

[2] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Address_Resolution_Protocol. [Último acceso: 06 Abril 2017].

[3] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Ataque_de_denegaci%C3%B3n_de_servicio. [Último acceso: 06 Abril 2017].

[4] «Cisco,» [En línea]. Available: http://www.cisco.com/c/en/us/products/collateral/wireless/adaptive-wireless-ips-software/data_sheet_c78-501388.html. [Último acceso: 04 Abril 2017].

[5] «OpenWIPS-ng,» [En línea]. Available: http://www.openwips-ng.org/index.html. [Último acceso: 04 Abril 2017].

[6] «Fossbytes,» [En línea]. Available: https://fossbytes.com/hacking-techniques/. [Último acceso: 04 Abril 2017].

[7] «Incibe,» [En línea]. Available: https://www.incibe.es/protege-tu-empresa/blog/internet-of-things-ciberseguridad. [Último acceso: 04 Abril 2017].

[8] Sophos, «redestelecom,» [En línea]. Available: http://www.redestelecom.es/internet/noticias/1087983001903/dispositivo-conectado-susceptible-hackeado.1.html. [Último acceso: 04 Abril 2017].

[9] «Colombia Digital,» [En línea]. Available: https://colombiadigital.net/actualidad/noticias/item/8712-9-inventos-de-internet-de-las-cosas-para-su-hogar.html. [Último acceso: 04 Abril 2017].

[10] «Microsoft,» [En línea]. Available: https://news.microsoft.com/es-xl/16-increibles-estadisticas-que-pronostican-el-futuro-de-internet-de-las-cosas/#sm.00000sobvzch99ct2ulrx3s4xmh41. [Último acceso: 04 Abril 2017].

[11] «Snort,» [En línea]. Available: https://www.snort.org/. [Último acceso: 06 Abril 2017].

[12] «Aircrack-ng,» [En línea]. Available: https://www.aircrack-ng.org/. [Último acceso: 07 Abril 2017].

[13] «Airmon-ng,» [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=airmon-ng. [Último acceso: 07 Abril 2017].

[14] «Airodump-ng,» [En línea]. Available: https://www.aircrack-ng.org/doku.php?id=airodump-ng. [Último acceso: 07 Abril 2017].

[15] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Arpwatch. [Último acceso: 07 Abril 2016].

[16] «Arpwatch,» [En línea]. Available: http://linuxcommand.org/man_pages/arpwatch8.html. [Último acceso: 07 Abril 2017].

[17] «Netfilter,» [En línea]. Available: https://www.netfilter.org/projects/iptables/index.html. [Último acceso: 07 Abril 2017].

[18] «Wikipedia-iptables,» [En línea]. Available: https://es.wikipedia.org/wiki/Netfilter. [Último acceso: 07 Abril 2017].

[19] «Wikipedia-Imagen Tablas,» [En línea]. Available: https://es.wikipedia.org/wiki/Netfilter#/media/File:Netfilter-packet-flow.svg. [Último acceso: 07 Abril 2017].

[20] «Apache,» [En línea]. Available: https://httpd.apache.org/. [Último acceso: 07 Abril 2017].

[21] «W3Techs,» [En línea]. Available: https://w3techs.com/technologies/overview/web_server/all. [Último acceso: 07 Abril 2017].

[22] «Tomcat,» [En línea]. Available: http://tomcat.apache.org/. [Último acceso: 07 Abril 2017].

[23] D. T. US, «Fundamentos de aplicaciones y servicios telemáticos,» de *Práctica 06*, Sevilla, 2015, pp. 9-10.

[24] D. T. US, «Fundamentos de aplicaciones y servicios telemáticos,» de *Práctica 06*, Sevilla, 2015, p. 34.

[25] D. T. US, «Fundamentos de aplicaciones y servicios telemáticos,» de *Práctica 07*, Sevilla, 2015, p. 18.

[26] «PostgreSQL,» [En línea]. Available: https://www.postgresql.org/about/. [Último acceso: 07 Abril 2017].

[27] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Estructura_de_descomposici%C3%B3n_del_trabajo. [Último acceso: 12 Abril 2017].

[28] «Snort,» [En línea]. Available: https://www.snort.org/faq/readme-daq. [Último acceso: 12 Abril 2017].

[29] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol. [Último acceso: 22 Abril 2017].

[30] «Rules Snort,» [En línea]. Available: https://www.snort.org/downloads/#rule-downloads. [Último acceso: 20 Febrero 2017].

[31] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo_OSI. [Último acceso: 06 Aril 2017].