Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN/PREVENCIÓN DE INTRUSIONES

Alumno: **Juan David Serrano Marín**

Tutor: Prof. D. Francisco Javier Sanchez-Roselly
Depto.: Ingeniería de Telecomunicación

Octubre, 2016

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN/PREVENCIÓN DE INTRUSIONES

Alumno: Juan David Serrano Marín

Tutor: Prof. D. Francisco Javier Sanchez-Roselly
Depto.: Ingeniería de Telecomunicación

Firma Alumno:

SERRANO MARIN, JUAN DAVID (FIRMA)

Firmado digitalmente por SERRANO MARIN, JUAN DAVID (FIRMA)
Fecha: 2016.10.05 19:51:38 +02'00'

Juan David Serrano Marín

Firma Vº Bº Tutor:

NOMBRE SANCHEZ-ROSELLY NAVARRO FRANCISCO JAVIER - NIF 75067070S

Digitally signed by NOMBRE SANCHEZ-ROSELLY NAVARRO FRANCISCO JAVIER - NIF 75067070S
DN: c=es, o=FNMT, ou=fnmt clase 2 ca, ou=703001774, cn=NOMBRE SANCHEZ-ROSELLY NAVARRO FRANCISCO JAVIER - NIF 75067070S
Date: 2016.10.10 14:54:45 +01'00'

Prof. D. Francisco Javier Sanchez-Roselly

ÍNDICE

1. INTRODUCCIÓN

Cada vez más, y casi sin ser conscientes de este hecho, las TIC (Tecnologías de la Información y Comunicación) están más integradas en la sociedad que nos rodea. Este hecho queda constatado sólo con observar el gran despliegue tecnológico al que se está expuesto. Sobra decir que gracias a las TIC se han facilitado muchos de los aspectos de la vida cotidiana, pero como en todo, se debe tener en cuenta lo que ello implica, y es que el hecho de esta infinidad de elementos existentes, genera una ingente cantidad de información; conversaciones, multimedia, búsquedas, posicionamiento… información de la que, por lo general, ni siquiera se llega a ser consciente. Y es que hoy día, esta información se ha vuelto más valiosa que nunca, por lo que existe un gran interés en hacerse con ella, por multitud de razones.

Es aquí donde entra en juego el importante papel de la seguridad, tratando de mantener alejada esta información de posibles manos ilegítimas, y es que de igual modo que todo en cuanto a tecnología evoluciona y avanza, la seguridad también lo hace, y sin descanso, ya que el escenario en el que se trabaja, no deja de ser un tira y afloja de defensa y ataque, en el que cada bando tiene bien definidas sus metas.

En el presente trabajo de fin de grado, se aborda la seguridad desde el punto de vista de la prevención y la detección de intrusos en redes de telecomunicación, mediante un sistema IDS (Intruder Detection System) de código abierto con características de IPS (Intruder Prevention System). Este software denominado Snort, dispone de las herramientas necesarias para la generación de alarmas y el registro de actividad sospechosa, tales como preprocesadores encargados del análisis del tráfico y el comportamiento de la red, y de un sistema de reglas contra las cuales el tráfico será probado. Todo ello totalmente configurable, en función de qué se pretenda proteger, así como de las medidas que se quieran tomar en cada caso.

A grandes rasgos, se pretende utilizar y testear el funcionamiento de dicho software en diferentes escenarios, lo más cercanos posibles a la realidad y en multitud de situaciones diversas, como son redes cableadas (Ethernet) o inalámbricas (WiFi), así como establecer mecanismos de seguridad en las capas superiores del modelo TCP/IP ante ataques previamente definidos. En las pruebas que se llevan a cabo, se adquiere, en una primera fase, el papel de la defensa preventiva, y posteriormente la del ataque, con el fin de comprobar cuán exitosa es la defensa llevada a cabo, así como del correcto funcionamiento del sistema IDS con el que se trabaja.

Cabe destacar que los escenarios elegidos, se realizan indistintamente sobre redes Ethernet o WiFi, dependiendo en cada caso del propio escenario elegido. Estos

escenarios, están formados, como mínimo, por una máquina Linux que es la que dota de la seguridad preventiva al sistema, es decir, en la que se aloja el IDS. Por otra parte, se dispone de otra máquina en la misma red, la cual actúa de atacante. En la mayoría de casos, se trata de un SBC (Single Board Computer) Raspberry Pi 2, modelo B+. En esta máquina con sistema operativo Kali Linux, se alojan las diferentes herramientas necesarias para realizar los ataques en cada escenario. Además de estas máquinas, se dispone de la infraestructura necesaria para la constitución de la red, amén de otras máquinas existentes en la misma, las cuales tienen un uso habitual, con el fin de dotar al escenario del carácter real sobre el que se quiere poner a prueba el sistema.

Estos escenarios están estrechamente ligados a los diferentes niveles del modelo TCP/IP, así pues, para la capa de aplicación, se dispone un entorno de aplicación web, sobre la cual se realizan inyecciones de código SQL (Structured Query Language), por otro lado, en otro escenario, se comprobará el funcionamiento del IDS ante un ataque por fuerza bruta a un servidor FTP (File Transfer Protocol) [1]. En el caso del nivel de Red, se disponen dos escenarios independientes sobre los que se trabaja, el primero, basado en el protocolo ARP (Address Resolution Protocol) [2], trata de abarcar la seguridad referente a un ataque ARP Spoofing. El segundo de ellos, gira en torno al protocolo ICMP (Internet Control Message Protocol) [3], centrándose en los mensajes ICMP de tipo redirección, los cuales pueden llegar a comprometer la seguridad de un sistema. Por último, y ya en el nivel de transporte, el escenario elegido es el del escaneo de puertos TCP (Transmission Control Protocol) [4], algo que suele ser uno de los primeros pasos a realizar en un ataque real, y que posteriormente da lugar a un gran abanico de métodos de comprometer la seguridad de un sistema. Sobre este último, se tienen diferentes variantes, ya que hay que tener en cuenta varios métodos de realización de los escaneos. De esta forma, se comprueba la efectividad del IDS ante cada una de las posibles variantes de escaneo elegidas, dando lugar a una comparativa de eficiencia.

2. OBJETIVOS Y METODOLOGÍA

El objetivo final de este trabajo de fin de grado es la utilización del IDS Snort para implementar un sistema para la detección y prevención de intrusiones, haciendo uso de las políticas de seguridad que se definen a tal uso. Para llegar a este objetivo final, se requiere una metodología, la cual, a medida que se vaya desarrollando, ayudará a alcanzar dicho objetivo.

Esta metodología, en primera instancia, requiere de la correcta instalación y configuración inicial general para todos los escenarios del IDS en la máquina que hace las veces de protectora del sistema. Posteriormente, y con el objetivo de la adecuación del IDS a cada escenario planteado, se realiza la programación de las reglas y preprocesadores necesarios. Todo ello, proporciona parte de la familiarización y el bagaje necesario en relación a Snort. A continuación, y con el fin de comprometer la seguridad del sistema, a la vez de comprobar el correcto funcionamiento del software Snort con posterioridad, se procede a la instalación de las herramientas de ataque necesarias en la máquina atacante (Raspberry Pi).

Con el objeto de facilitar la comprensión, ejecución y el análisis de los resultados del funcionamiento del IDS, el sistema de seguridad se ha desgranado en los diferentes ensayos que se han descrito someramente en el apartado anterior. No sin que ello signifique que las medidas tomadas en cuanto a prevención de forma independiente en cada escenario, no puedan ser tomadas conjuntamente, para dotar al sistema de una seguridad más robusta, ante todos los experimentos propuestos conjuntamente.

Por otro lado, y como objetivo añadido, cabe decir que los experimentos que se llevan a cabo, no solo tratan de ofrecer la forma de protección frente a ciertos tipos de ataque, si no que también tratan de hacer plausible el comportamiento y la eficacia real del software elegido, por lo que cada escenario, arroja unos resultados en cuanto a la fiabilidad obtenida.

3. SISTEMAS DE DETECCIÓN DE INTRUSIONES (IDS)

Los IDS o sistemas de detección de intrusiones, son herramientas de seguridad utilizadas en las redes de telecomunicaciones, que tratan de detectar posibles accesos indeseados y comportamientos sospechosos que tengan como finalidad el comprometer la seguridad del sistema que están vigilando. Por lo general, el trabajo se basa en el análisis del tráfico de la red, examinando cada paquete minuciosamente, en busca de elementos que indiquen un posible intento de ataque al sistema en forma de intrusión. Entendiendo como intrusión toda actividad no autorizada o inesperada en una red corporativa.

Dicho funcionamiento, no solo se basa en el análisis de paquetes individuales, si no que también se dispone de una base de conocimiento, en la cual se albergan las firmas (patrones) de ataques conocidos, de forma que, aunque un paquete en concreto no arroje muestras de ser considerado como de riesgo, una consecución de este tipo de paquetes sí puede llegar a serlo, y es aquí donde gana importancia este conocimiento del IDS y la información de la que disponga sobre comportamientos sospechosos. Estas firmas son las que se encargan de hacer discernir al sistema de un comportamiento entendido como normal, de un uso fraudulento. Otro método que puede utilizar un sistema de este tipo, es el de la heurística, encargada de determinar la normalidad de la actividad de la red atendiendo a diversos parámetros, de forma que si existen variaciones anómalas en alguno de estos parámetros, será entonces cuando el IDS se encargará de generar las alarmas, advirtiendo de que el estado de funcionamiento, ha dejado de ser el normal esperado.

Estas herramientas tratan de dotar al sistema de la capacidad de prevenir y alertar con suficiente antelación para tomar las medidas necesarias en el caso de que se necesiten. Por otra parte, típicamente, no suelen incluir medidas de detención concretas de los ataques que se puedan realizar, aún y así, gozan de medidas de respuesta ante estos ataques. Aunque la existencia de este tipo de medidas, dependerá concretamente del software IDS que se esté utilizando. Además, también aportan mecanismos de registro de la actividad, proporcionando una valiosa información para el posible análisis forense posterior, y la mejora del funcionamiento de la red en la que esté implantado el software.

3.1 Clasificación de los IDS [5] [6] [7]

3.1.1 En función a los métodos de detección

Aquí encontramos dos clasificaciones diferentes, basadas en el cómo el IDS realiza la detección de la intrusión. Hace referencia al tipo de procesamiento que realiza el software para determinar la existencia o no de la misma.

Detección por patrones (firmas)

En este tipo de IDS, lo que nos encontramos es en una base de reglas utilizadas para la detección de las intrusiones, de forma que el tráfico analizado, se comprobará frente a ellas. En el caso de que este tráfico esté catalogado en una regla como un ataque, el software tomará medidas y actuará en consonancia a lo que se haya dispuesto en la regla que ha provocado la detección. La principal ventaja de este tipo de sistemas, es la facilidad de uso de la que gozan, además de la de la inclusión de nuevas reglas, haciendo que estos sistemas sean sencillos de mantener actualizados. También, debido a la gran concreción de las reglas, en estos sistemas se dan un bajo número de falsos positivos, razones por las cuales este tipo de IDS están bastante extendidos. Estos sistemas requieren de poca capacidad computacional.

Como contraposición, el tener que disponer de reglas para cada tipo de ataque de forma concreta, hace que el sistema sea poco eficiente ante la existencia de nuevos ataques que todavía estén sin catalogar, lo que supone un gran esfuerzo y una actualización constante del sistema por parte del administrador. Además, el posible conocimiento de las reglas existentes por parte del intruso, hace que el sistema pueda volverse vulnerable, ya que, muchas de estas reglas son de dominio público. También cabe destacar la necesidad de dotar al sistema de medidas de seguridad en cuanto a ataques internos.

Detección por anomalías

La detección se realiza teniendo en cuenta perfiles de funcionamiento considerado normal, de forma que, posteriormente, cualquier comportamiento que se salga de este funcionamiento, será considerado como un ataque en potencia. Para el correcto funcionamiento de este tipo de IDS, es necesario de un periodo de entrenamiento del sistema, para establecer el que será el comportamiento normal y evitar en medida de lo

posible la generación de falsos positivos. Estos perfiles irán evolucionando con el fin de estar lo más adaptados posibles al comportamiento que vaya adquiriendo en el sistema.

El principal inconveniente de los sistemas que se clasifican de esta forma, es que suelen generar un elevado número de falsos positivos, debido a que cualquier actividad que se salga de lo establecido en los perfiles, se tratará como si de una intrusión fuera. Por otra parte, un posible intruso con conocimiento del sistema, puede llegar a entrenarlo para que el mismo, ante un comportamiento de intrusión, no se genere alerta alguna por considerar la misma como comportamiento normal de la red. Este tipo de sistemas, por lo general, requieren un gran coste computacional. Por otro lado, como la ventaja principal de estos sistemas, tenemos que son óptimos para la detección de nuevos ataques, ya que estos, con gran probabilidad se saldrán del comportamiento esperado.

3.1.2 En función del sistema protegido

Dicha clasificación tiene en consideración el lugar en el que el sistema de detección analiza el tráfico. Dentro de esta clasificación tenemos dos tipos, aunque también podemos encontrar un tercero, híbrido entre los dos anteriores.

Network IDS (NIDS)

Los NIDS, o IDS de red, analizan todo el tráfico de la red que protegen, normalmente en todo el dominio de colisión. Se realiza una captura del tráfico previa para su posterior análisis, el cual dependerá del tipo de IDS que se tenga. Por el hecho de que estos sistemas reciben el tráfico antes de llegar a los receptores finales, normalmente son usados como primera defensa en la red. Esta característica es una de las principales ventajas de las que dispone este tipo de software, lo que también se traduce en una ágil respuesta ante cualquier detección. Cabe decir, que debido a al propio concepto de IDS de red, los beneficiados de la seguridad que aporta este tipo de sistemas, serán todos los elementos que se encuentren en la propia red.

La principal desventaja de estos sistemas, es el gran coste computacional que requiere el análisis del tráfico, siendo a veces imperiosa la necesidad de analizar el tráfico de una forma más superficial. Hay que tener en cuenta, que dicho coste computacional será proporcional al tamaño de la red, y por ende, a la cantidad de tráfico que discurra por la misma.

Host IDS (HIDS)

Encargados de la seguridad de un terminal en concreto, su principal característica es ésta, únicamente protegen al terminal sobre el que se ejecutan. El método de trabajo que siguen es el monitoreo de eventos de la máquina en cuestión, tales como logueos, vigilancia de nuevas conexiones o el estado de los administradores, entre otros. El inconveniente más destacable de este tipo de sistemas, debido a su propia constitución, es el hecho de que únicamente protegen a un terminal, además de que son muy específicos por este hecho.

Por otro lado, la principal ventaja de la que disponen estos sistemas, es que son los más adecuados para la detección de ataques internos, ya que además de ser muy específicos, disponen de un gran abanico de herramientas y son capaces de monitorizar un elevado número de variables.

3.1.3 En función a la estructura

En ésta, se pone de manifiesto la estructura del sistema de la que dispone el IDS, pudiendo ser la misma de tipo distribuido o centralizado.

Estructura distribuida

Sistemas segregados por la red por diversidad de motivos como puede ser el aumento del rendimiento. Estos sistemas se ejecutan al mismo tiempo, y en la red existe un terminal que realiza las tareas de coordinación y gestión de los demás IDS distribuidos, así como de tomar las decisiones óptimas ante los posibles ataques. Esta arquitectura suele utilizar HIDS en los terminales. El hecho de disponer de un mayor número de máquinas, aporta al sistema una gran capacidad de cómputo, además del tratamiento de la información por parte de varios HIDS, lo que permite determinar con mayor facilidad la existencia de una intrusión. Por contraposición, cabe destacar la dificultad añadida de implantación y configuración de este tipo de sistemas.

Estructura centralizada

El funcionamiento de estos IDS viene caracterizado por el hecho de la existencia de un único terminal en el que converge todo el tráfico que discurra por la red. Este terminal es el encargado de la seguridad de la misma, y es el utilizado en los sistemas NIDS. Este

tipo de herramientas son de fácil implementación y configuración, además de disponer también de una sencilla actualización añadida al hecho de la necesidad de mantener un único terminal.

El principal inconveniente es que la seguridad de la red completa recae sobre un único equipo, lo que supone que la red puede quedar desprotegida ante la falla del mismo. Hay que tener en cuenta a la hora de la elección del terminal que este debe gozar de unas características óptimas en cuanto a capacidad de computación, ya que será el único encargado del análisis de la totalidad del tráfico.

3.1.4 En función al origen de los datos

Estado del sistema

Los datos recogidos por el IDS no son los del tráfico de red, si no que tienen proveniencia del propio sistema en el que se halla el mismo, tales como accesos a sistema de archivos, servicios, etc. Esto es algo que está enteramente ligado a los sistemas de tipo HIDS, ya que trabajan sobre el mismo sistema en el que están instalados. Los IDS de este tipo, tratan de detectar comportamientos sospechosos en la propia máquina. Por ende, las ventajas de las que gozan estos tipos de sistemas, se asemejan a las de los HIDS, así como la protección ante ataques internos y el bajo requerimiento de computación necesario.

Auditoría previa

Más relacionado con un análisis forense de lo ocurrido, este tipo de IDS, en lo que centran es en la recopilación exhausta de información una vez la intrusión ha ocurrido, con el fin de determinar las causas, posibles autores, etc. Lo que pone de manifiesto las posibles vulnerabilidades encontradas en el sistema después de su análisis. Esto proporciona una información valiosa para tomar medidas adicionales de seguridad para una protección futura. Lógicamente, estos sistemas no gozan de la inmediatez de actuación que tienen otros sistemas por el propio hecho del modo de funcionamiento que tienen, aunque sigue teniendo importancia la agilidad del sistema en cuanto a la detección de la intrusión.

Tráfico de red

Esta clasificación viene dada por el hecho de la captura de tráfico en tiempo real de la red, lo que podría compararse a una herramienta de tipo sniffer. Estos sistemas realizan una captura y análisis de los paquetes recogidos “in situ” para poder realizar las detecciones pertinentes. De forma similar a la relación vista anteriormente entre los sistemas de auditoría previa y los HIDS, estas herramientas de tráfico de red, se relacionan estrechamente con los NIDS, lo que viene a establecer unas ventajas e inconvenientes similares, como el hecho de analizar la información previamente ésta llegue a destino, y el elevado coste computacional proporcionalmente ligado al tráfico de la red existente.

3.1.5 En función de la respuesta

Determinada por la forma de actuación del software tras la detección de la intrusión.

Sistema IDS pasivo

Los sistemas pasivos, de fácil implementación y menor necesidad de recursos para su funcionamiento, se basan únicamente en la generación de alertas ante la aparición de la intrusión. Por lo general, requieren de un supervisor que las gestione, con el fin de tomar las medidas necesarias en función de la información que éstas proporcionen. Dichas alertas aportan información en cuanto al motivo de su generación, además de elementos que facilitan la interpretación de lo ocurrido. Nos encontramos ante sistemas altamente configurables, por lo que hay ser especialmente cuidadoso a la hora de realizar esta configuración, con el fin de evitar la aparición de falsas alarmas.

Sistema IDS activo

Estos sistemas, no sólo se limitan a generar alarmas como es el caso de los pasivos, si no que van más allá. Además de generar una alarma, toman medidas de respuesta ante la intrusión. Estas respuestas pueden ser de una gran variedad, tales como el cierre de conexiones, puertos, o la finalización de servicios. Estos sistemas requieren de grandes capacidades de cómputo, ya que no solo se limitan al análisis del tráfico, si no que también deben tomar decisiones, y ejecutar medidas de contención y respuesta. Normalmente, en estos sistemas, las medidas de respuesta que se configuran

no son de gran repercusión, con el fin de evitar problemas derivados de la aparición de falsos positivos.

A modo de resumen aclaratorio, la posible clasificación de los sistemas IDS sería:

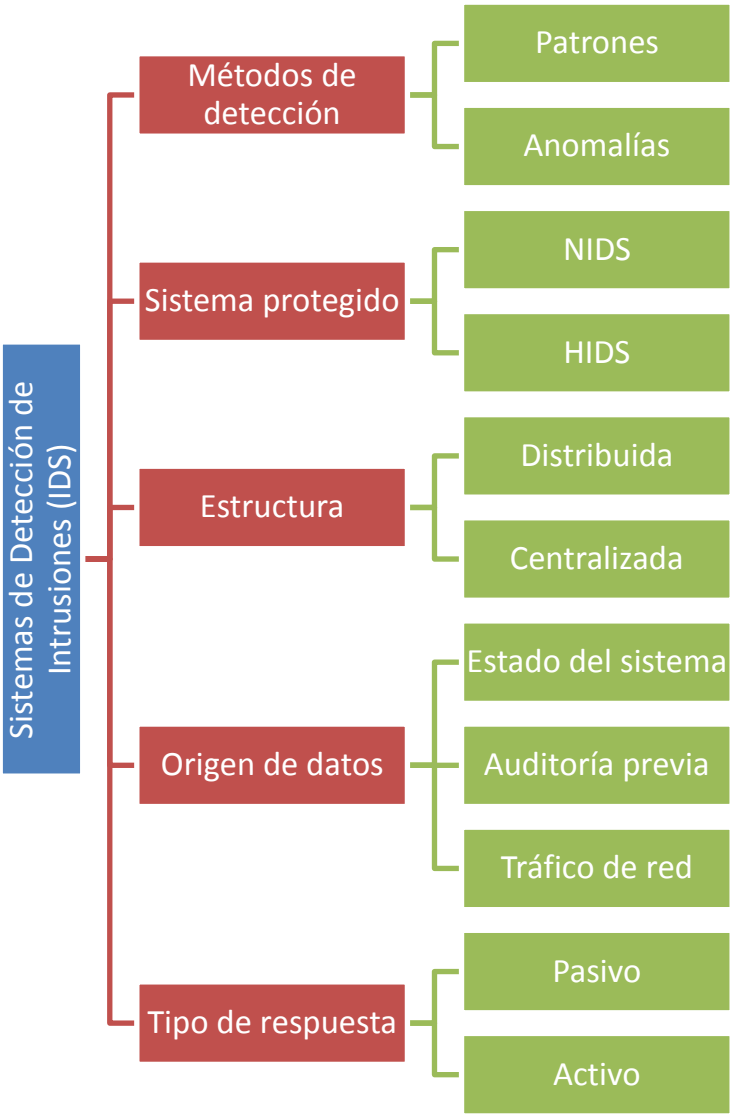

Figura 1. Clasificación de sistemas IDS

4. IDS SNORT [5] [6]

4.1 Introducción

Con origen en 1998 y por entonces llamado APE, este software se trataba de una herramienta que permitía analizar las conexiones de red por módem y como debugger para los servicios de red. Con el paso del tiempo, fue adquiriendo nuevas características, así pues, al poco tiempo se dotó al sistema con la capacidad de analizar firmas, comenzando en este momento a ser utilizado como un IDS al uso. Poco más tarde, se modificó la arquitectura de funcionamiento del sistema, haciendo que ésta fuera basada en plugins, algo que se ha mantenido hasta las últimas versiones que encontramos hoy en día.

Actualmente, Snort, se define como una herramienta de seguridad, centrada en la prevención y la detección de intrusos, con capacidades de análisis y registro del tráfico en red en tiempo real, a partir de las cuales es capaz de generar alarmas y de tomar ciertas medidas de contención ante el ataque detectado. Tanto las alarmas como las medidas de contención deben estar predefinidas en un fichero de reglas (patrones), en el cual, se determinará el tipo de acción tomada en referencia al tipo de comportamiento del que se trate.

Este lenguaje de reglas implementado por Snort es flexible, de forma que permite describir con gran detalle el tráfico al que se hace referencia, con el fin de ser efectivo en cuando a la detección de las intrusiones y evitar en medida de lo posible las falsas alarmas. Por otra parte, este software pone a disposición una serie de plugins y preprocesadores, pequeños programas que, añadidos al núcleo de Snort, dotan al sistema de nuevas características. A través de los preprocesadores, se pueden detectar comportamientos anómalos o catalogados como ataques, de forma que el sistema ya no solo se basa en el análisis de los paquetes por la red, si no que es capaz de determinar si una consecución de cierto tipo de paquetes, puede dar muestras de tratarse de un ataque al sistema, algo que quizás analizando los propios paquetes individualmente no daría tales muestras.

Así pues, y atendiendo a la clasificación de IDS expuesta con anterioridad, se puede decir que Snort, actualmente, es una herramienta de seguridad, de tipo IDS, que funciona como un sistema de detección de red (NIDS), capaz de realizar su actividad en

tiempo real, y basado en reglas o patrones. De estructura centralizada y que adquiere los datos de la propia red.

4.2 Estructura del sistema y funcionamiento

Teniendo en cuenta la estructura genérica de un IDS basado en reglas, cabe decir que la arquitectura de cada sistema vendrá dada por su propia constitución a la hora de su desarrollo. En concreto, Snort, se basa en esta estructura genérica, pero presenta ciertas diferencias debido a las características añadidas de las que se ha dotado al software, tales como el decodificador, los preprocesadores y plugins, y la unión de los módulos de comparación y detección en uno solo, el motor de detección, tal y como se muestra en la figura siguiente.

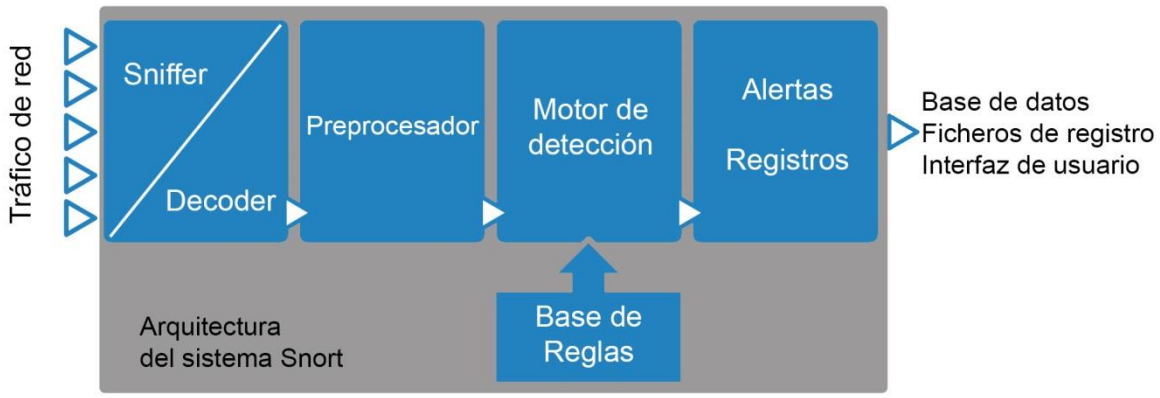

Figura 2. Arquitectura del sistema Snort

- Sniffer y Decoder

  El sniffer es el encargado de capturar el tráfico de la red (paquetes), en el caso de ejecutarse sobre sistemas Linux, utilizará la librería libpcap, mientras que en sistemas Windows, requerirá el uso de la librería winpcap. Una vez recibido el tráfico, el decoder se encarga de decodificarlo, con el fin de hacer más fácil del tratamiento de los datos por los módulos siguientes. Esta decodificación separa la información de los paquetes en cuanto al nivel de la arquitectura tcp/ip del que se trate.

- Preprocesador

  El preprocesador es el encargado de hacer un primer análisis del tráfico, pudiendo generar alertas en el caso de que detecte un comportamiento

sospechoso, además de poder proceder a la clasificación o el filtro de los mismos. Por otra parte, al realizar este análisis previo, facilita la tarea del análisis posterior por parte de otros módulos.

- Motor de detección

  Este módulo se apoya en la base de reglas, donde se encuentran todas las comprobaciones que se deben a hacer de los paquetes, así como las medidas a tomar. Dicho módulo, se encarga de hacer una comparación de los paquetes recibidos y cada una de las reglas, determinando así si alguno de los mismos es motivo de generación de alarma.

- Alertas y Registros

  Una vez determinado el resultado de la comprobación de detecciones por parte del motor de detección, este módulo es el encargado de lanzar las alertas, actualizar los registros con la información pertinente, etc. La acción a tomar vendrá dada en la propia regla que haya sido motivo de detección de la alarma.

Una vez completado este recorrido, la información obtenida finalmente, puede ser almacenada y tratada en una base de datos, manejarse a través de los ficheros de registro o ser llevada a una interfaz de usuario para una más sencilla visualización, pero esto es algo que queda fuera del alcance del propio Snort, para realizar estas tareas, se debe evaluar qué tipo de herramienta es la más propicia para trabajar conjuntamente con el software IDS.

Por otro lado, Snort dispone de 4 modos de funcionamiento diferentes. Por un lado el modo Sniffer, en el que se recoge el tráfico que discurra por una red y se va mostrando por pantalla a medida que se va capturando. Otro modo de funcionamiento que tenemos es el de Packet Logger, el cual, siendo similar al modo Sniffer, además de mostrar los paquetes capturados, permite hacer un guardado de los mismos en la propia máquina con el fin de poder realizar su análisis posterior. También tenemos el modo NIDS, el cual es el que se utiliza a lo largo de todas las pruebas que se explican en este trabajo fin de grado. En este modo de funcionamiento, Snort adquiere las características de un Sniffer, capturando el tráfico de la red en la que se encuentra y, posteriormente, lo que le da la categoría de IDS, realiza el análisis de la información recopilada, con objeto de detectar las intrusiones. Por último, el modo Inline se encargará de proporcionar a Iptables de las reglas necesarias para determinar el paso de los paquetes.

4.3 Alertas

Este IDS almacena las alertas generadas en un fichero, el cual, mediante la opción –K a la hora de ejecutarlo, tiene un formato ASCII plano, lo que permite que pueda ser consultado mediante un simple procesador de textos. Snort también permite establecer esta opción para devolver las alertas en formato pcap para su posterior interpretación en herramientas analizadoras de red, tales como Tcpdump o Wireshark.

En cuanto a los modos de alerta que puede generar, los principales son el modo Fast, también llamado de alerta rápida, el cual devolverá la mínima información necesaria, como puede ser el mensaje de alerta, su clasificación, las direcciones IP y los correspondientes puertos que estén relacionados con la propia alerta. En el modo Full, además de la información ya incluida en el modo Fast, se proporcionará la cabecera completa de los paquetes registrados. Por último, también se dispone del modo TCPdump, el cual devolverá la información de las alertas en formato binario, de esta forma, se descarga a la máquina del uso del procesamiento que requiere la codificación de las alertas, haciendo que el IDS trabaje de forma más ágil. El modo de alerta se selecciona haciendo uso de la opción –A, y durante todos las pruebas que se realizan, en el caso que atañe a este trabajo, se utilizará será el Fast, con objetivo de detectar las alertas de un modo más sencillo y visual, y por otra parte, descargar medianamente al sistema del trabajo que requiere el modo completo (Full).

Un posible ejemplo de alerta sería:

```
08/19-11:57:50.030195 [**] [1:20:8]Redirección ICMP[**] [Priority: 0] {ICMP} 192.168.1.1 -> 192.168.1.8
```

Donde:

`08/19-11:57:50.030195`

Hace referencia a la fecha y la hora a la que la alerta ha sido generada.

`[1:20:8]`

El primer dígito indica el Generator ID, el componente de Snort que genera la alerta.

El segundo número, da cuenta de qué regla generó dicha alerta.

El último dígito nos indica el número de detecciones que ha habido con dicha regla.

`Redirección ICMP`

Aquí se muestra el texto que se incluye en la regla para ser mostrado en la alerta.

`[Priority: 0]`

Indica el nivel de prioridad que se ha configurado en la regla frente a esta alerta.

`{ICMP}`

Da cuenta del protocolo que se está analizando en cada caso.

`192.168.1.1 -> 192.168.1.8`

Indica las partes que han tenido que ver en la generación de la alerta, prestando atención a los símbolos -> o <-, podremos determinar cuál de las partes es el origen y cuál el destino. En este caso concreto, podemos decir que el paquete tenía origen 192.168.1.1 y su destino era 192.168.1.8. Por otra parte, en el caso de los protocolos de transporte, también se dará cuenta de los puertos que atañen a la comunicación.

4.4 Reglas

Para lograr los objetivos que Snort tiene en cuanto a seguridad, éste hace uso de las denominadas reglas. Éstas, deben ser configuradas y almacenadas en el sistema por parte del usuario o administrador, el cual deberá crearlas para ajustarse lo máximo posible a los requerimientos de la red. Por otra parte, la página oficial de Snort pone a disposición de los usuarios conjuntos de reglas actualizadas para su uso, aunque es recomendable pulirlas y adaptarlas al sistema que vayan a proteger. Para la inclusión de estas reglas, existen dos opciones, se pueden incluir directamente en el fichero de configuración de Snort (Snort.conf), en el que convivirán con el resto de configuraciones del sistema, o, por otra parte, y la cual es la opción que se ha utilizado, en dicho fichero de configuración, se nos da la opción de hacer alusión a otro fichero externo mediante la orden include, en el cual se dispondrán las diferentes reglas propias que queramos utilizar. En nuestro caso, el fichero al que se hará alusión desde el Snort.conf se denomina reglas_propias.rules, y será aquí donde a lo largo de todo el trabajo, se configurarán las diferentes reglas y preprocesadores necesarios.

Para facilitar la comprensión de las reglas y su formato, su explicación se va a dividir en la parte de cabecera y las opciones, las cuales pueden agruparse en opciones básicas, de payload, no-payload y de post-detección.

	Elemento	Tipo / Descripción	
Cabecera	Acción de la regla	Alert	Alerta y registro del paquete
		Log	Registro del paquete
		Pass	Ignora el paquete
		Activate	Alerta y llamada a regla dinámica
		Dynamic	Llamada por Activate, se comporta como regla de tipo log
		Drop	Se deriva a iptables y se registra
		Reject	Igual que Drop, además genera un TCP reset o un ICMP port unrech (TCP o UDP)
		Sdrop	Se deriva a iptables sin registro
	Protocolo	TCP, UDP, ICMP o IP	Protocolo a analizar
	Dirección IP origen	Dirección IP, incluyendo máscara de subred en formato CIDR	
	Puerto origen	Indicación del puerto o rango de puertos de transporte que se pretenden analizar	
	Dirección IP destino	Dirección IP, incluyendo máscara de subred en formato CIDR	
	Puerto destino	Indicación del puerto o rango de puertos de transporte que se pretenden analizar	
	Dirección de la operación	Saliente (->)	Indica la dirección del tráfico al que se aplica la regla
		Entrante (<-)	
		Bidireccional (<>)	

Tabla 1. Cabecera

	Opción	Descripción
Opciones básicas (generales)	Msg	Indica el mensaje a mostrar en la alerta
	Reference	Inclusión de referencias al sistema de identificación de ataques externos
	Gid (Generator ID)	Indica el elemento de Snort que generará la alerta
	Sid	Identificador único de regla
	Rev	Indicador de revisión de la regla
	Classtype	Utilizado para la organización de las reglas en cuanto al tipo de ataque (clase)
	Priority	Nivel de prioridad de la regla
	Metadata	Adición de información adicional a la regla

Tabla 2. Opciones básicas

	Opción	Descripción
Opciones de Payload	Content	Permite buscar en el payload un contenido específico
	Nocase	Deshabilita case sensitive para las búsquedas de patrones
	Rawbytes	Permite a la regla inspeccionar el paquete previo a la decodificación
	Depth	Determina la longitud de datos del paquete que la regla debe inspeccionar
	Offset	Determina el lugar a partir del cual se realiza la inspección
	Distance	Indica la cantidad de datos que se ignoran en el payload hasta el inicio de la búsqueda
	within	Indica entre qué límites se debe realizar la búsqueda
	http_client_body	Restringe la búsqueda al cuerpo normalizado de http request
	http_cookie	Restringe la búsqueda al campo cookie header de http request
	http_header	Restringe la búsqueda al campo header de http request
	http_method	Restringe la búsqueda al fragmento method de http request

	http_uri	Restringe la búsqueda al campo normalized request URI
	fast_pattern	Modifica el contenido de las especificaciones para una regla, con el fin de poder utilizarlas en Fast Patern Matches
	Uricontent	Busca la petición normalizada del campo URI
	Urilen	Especifica la longitud de la URI
	Isdataat	Verifica que el payload tiene los datos en una localización específica
	Pcre	Permite a las reglas escribirse usando expresiones regulares (compatible con Perl)
	byte_test	Compara el campo byte con un valor específico
	Byte_jump	Permite escribir reglar para protocolos con longitud cifrada
	Ftpbounce	Permite detectar FTP Bounce Attacks
	Ans1	Decodifica un paquete o una porción en busca de codificaciones malintencionadas
	Cvs	Ayuda a la detección de Bugtraq-10384 y CVE-2004-0396

Tabla 3. Opciones de payload

	Opción	Descripción
Opciones no-Payload	Fragoffset	Permite comparar el campo IP fragment offset con un valor
	Ttl	Analiza el TTL
	Tos	Analiza el campo IP TOS con un valor específico
	Id	Analiza el campo IP ID con un valor específico
	Ipopts	Detecta si existe una opción IP concreta
	Fragbits	Analiza si la fragmentación y los bits de reserva vienen especificados en la cabecera IP
	Dsize	Comprueba el tamaño del payload
	Flags	Analiza la existencia de TCP flags
	Flow	Se asegura la dirección del tráfico
	Flowbits	Busca huellas en sesiones de transporte
	Seq	Analiza los números de secuencia TCP
	Ack	Analiza los ACK de TCP
	Window	Analiza el tamaño de ventana TCP

	Itype	Analiza el ICMP type
	Icode	Analiza el ICMP code
	Icmp_id	Analiza el ICMP ID
	icmp_seq	Analiza el número de secuencia ICMP
	Rpc	Analiza la aplicación RPC, la versión y el número de procedimiento en una petición SUNRPC CALL
	Ip_proto	Analiza la cabecera IP
	Sameip	Detecta la igualdad entre la dirección origen y destino
	Stream_size	Permite la comparación del tráfico en bytes

Tabla 4. Opciones no-payload

	Opción	Descripción
Opciones post-detección	Logto	Registro especial de los paquetes en un directorio concreto
	Session	Extrae los datos de usuario de las sesiones TCP
	Resp	Define la respuesta ante la generación de una alarma
	React	Define la reacción al tráfico que se adopta al detectar la alarma
	Tag	Permite el registro de mayor cantidad de información, además del paquete generador de la alarma
	Activates	Activa una regla en caso de la generación de la alerta
	Activates_by	Activación dinámica mediante otra regla a la que se referencia
	Count	De uso conjunto con la anterior, indica la cantidad de paquetes que se analizan con la nueva regla.

Tabla 5. Opciones post-detección

Así pues, un ejemplo de regla podría ser:

```
alert icmp 192.168.1.5 any -> 192.168.1.8 any (msg:"Redirección ICMP";
itype:5; icode:1; resp:icmp_host; sid:21; rev:1;)
```

Donde:

`alert`

Indica la generación de una alarma y el registro del paquete

`icmp`

Determina que el protocolo a analizar será ICMP

`192.168.1.5 any -> 192.168.1.8 any`

Indica el sentido del flujo de datos, de 192.168.1.5 hacia 192.168.1.8 (->), por lo que sabemos que la dirección de origen será la máquina .5 y la de destino, la .8. Por otra parte, any (cualquiera), hace referencia a los puertos, en cuyo caso, se analizarán todos.

`msg:"Redirección ICMP";`

Determina el mensaje que se mostrará en la alerta cuando ésta sea generada.

`itype:5`

Especifica que el campo type del paquete ICMP, deberá ser el 5, que se corresponde con un paquete de tipo redirección.

`icode:1`

Especifica la necesidad de analizar el campo code del paquete ICMP con el valor 1, lo que indica un ICMP de tipo redirección, de modo HOST.

`resp:icmp_host`

Determina la respuesta a tomar en caso de la activación de la regla. En este caso, icmp_host lo que realizará será enviar un ICMP de tipo HOST unrech al origen del paquete que ha generado la alarma.

`sid:21`

Indica el identificador de la regla, en este caso el 21.

`rev:1`

Indica el número de revisión de la regla, concretamente, primera revisión.

4.5 Preprocesadores

Tal y como se ha visto anteriormente, los preprocesadores son los encargados de dotar al sistema Snort de funcionalidades añadidas. Estos preprocesadores se pueden añadir a voluntad, dependiendo de la red sobre la que esté implantado el sistema y el tipo de ataques a los que el mismo esté expuesto. El código que forma un preprocesador, es

ejecutado inmediatamente después de la decodificación de los paquetes, y antes de que los mismos lleguen al motor de detección, de forma que son capaces de analizar ciertos comportamientos potencialmente peligrosos previo análisis independiente de los propios paquetes.

A continuación se listan algunos de los preprocesadores incluidos actualmente en la versión 2.9 de Snort, dando unas pinceladas de la utilidad de cada uno de ellos. En lo que concierne a los escenarios sobre los que se trabaja en este trabajo fin de grado, se han utilizado concretamente los preprocesadores ARPspoof y SfPortScan. Su configuración y funcionamiento se realizará con más detalle en cada una de las pruebas en las que hayan sido necesarios.

- ARPSpoof

  Decodifica los paquetes ARP, con objetivo de detectar ataques de tipo ARP Spoofing analizando las peticiones ARP y las inconsistencias de direcciones MAC a IP.

- DCE/RPC

  Detecta y decodifica el tráfico SMB y DCE/RPC. Ante el caso de fragmentación en capas SMB y TCP, este módulo se encarga de reensamblar los paquetes que se suministran a las reglar para que sean analizados.

- DCE/RPC 2 Preprocessor

  De igual forma que el anterior, aunque con matices, el propósito general de éste es el tratamiento de la segmentación con el fin de evitar evadir la detección por parte de las reglas mediante fragmentación en SMB y DCE/RPC.

- DNS

  Decodifica las respuestas DNS con el fin de detectar algunos xploits.

- Frag3

  El principal objetivo de este módulo es el reensamblar los fragmentos que conforman un paquete, para evitar las técnicas de evasión de Snort mediante dicha fragmentación. Orientado a fragmentación IP.

- FTP/Telnet Preprocessor

  Inspecciona los flujos de datos de TCP y Telnet. Decodifica flujo, comandos, respuestas y secuencias de escape con el fin de detectar anomalías en su uso.

- HTTP inspect

  Decodificador de HTTP para aplicaciones de usuario. Se encarga de decodificar y normalizar los campos. Puede ser evadido mediante fragmentación si no se usa en conjunto con algún otro preprocesador encargado de la misma.

- Performance monitor

  Proporciona estadísticas de ejecución del sistema de seguridad. Con dos modos de ejecución, podemos disponer de estas estadísticas en la propia consola de ejecución o ser llevadas a un fichero para su posterior tratamiento.

- RPC Decode

  Mediante un buffer de paquetes, tiene como misión la normalización de los registros fragmentados RCP en un registro único.

- SfPortScan

  Este módulo se encarga de detectar las fases previas de los ataques basados en el escaneo de puertos. Estos ataques se basan en la obtención de información del sistema, como son los puertos en uso o abiertos, para proceder con un ataque posterior concreto. El preprocesador goza de la capacidad de detección de diferentes variantes de estos escaneos.

- SMTP Preprocessor

  Decodificador SMTP, se encarga de decodificar y examinar los comandos y respuestas del protocolo con el fin de detectar posibles ataques.

- SSH

  Detecta xploits del protocolo SSH, generando una alerta en el caso de que se detecte su uso. Algunos de estos xploits son detectables una vez el intercambio de claves ya ha ocurrido, mientras que otros pueden ser detectados en fases más tempranas.

- SSL/TLS

  Para aumentar el rendimiento de Snort, éste ignora el tráfico encriptado. En el caso de querer que Snort analice dicho tráfico, este módulo se encarga de desencriptarlo, con el fin de determinar si el sistema debería analizarlo.

- Stream5

  Módulo de reensamblado TCP basado en objetivos. Analiza las sesiones TCP, así como la supervisión de conversaciones UDP e ICMP. En resumen, detecta anomalías en TCP, siendo capaz de generar hasta 8 tipos de alamas distintas en relación a estas anomalías.

4.6 Situación del IDS Snort

Debido a la versatilidad del software que se está tratando, cabe destacar la importancia de la posición que tenga la máquina que aloje el IDS dentro de la red, ya que ésta será determinante en función del uso que queramos dar a Snort y el cometido que tenga en cada caso, ya que de ello dependerá su éxito. Cabe decir que, siempre que se pretenda que Snort supervise una red o máquina la cual no sea la misma en la que se encuentra instalado, la interfaz sobre la que trabaje debe ser configurada en modo monitor o promiscuo, haciendo que todo el tráfico que sea recibido sea capaz de llegar a ser analizado, ya que de lo contrario, la propia interfaz descartará todo el tráfico del cual no sea destinataria, por lo que no llegaría información alguna al IDS.

Así pues, y en resumidas cuentas, cuanto más cercano al nivel físico se requiera que trabaje Snort, más cercano deberá situarse a la propia red a la que se intenta supervisar, mientras que si esta supervisión se plantea a niveles más altos como el nivel de red, se pueden encontrar fórmulas para que no sea necesario que dicha máquina se encuentre en la misma red, pudiendo supervisar el IDS una red lejana, si se toman las medidas necesarias. Con fin aclaratorio y para tomar conciencia de la importancia de la situación del IDS, a continuación se disponen algunos supuestos.

4.6.1 Supuesto 1: Situación en la misma red

En este supuesto, se trata de utilizar Snort para fines de supervisión del protocolo ARP [2]. Como ya sabemos, ARP es un protocolo que actúa en el nivel de red, asociando las direcciones físicas de los equipos a las direcciones de red, representativas de la posición

que tiene una máquina concreta dentro de una determinada red. Por este carácter físico del protocolo, no tiene sentido que se distribuyan los paquetes ARP más allá de la propia red que se está gestionando, por ende, si se tiene la necesidad de utilizar Snort para supervisar este tráfico, no quedará más remedio que situarlo dentro de la misma red que se pretenda supervisar. Un posible esquema para este supuesto sería el siguiente.

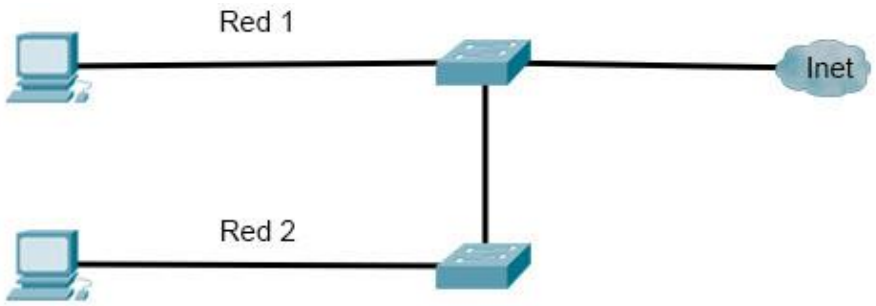

Figura 3.1. Situación en la misma red

Dadas las redes 1 y 2, siendo éstas independientes entre ellas, los paquetes ARP que se produzcan dentro de ellas, no verán el exterior por la propia naturaleza de ARP, esto es, que si en la Red 1 se genera un paquete ARP informando de una asociación MAC-IP, este paquete nunca irá más allá de la Red 1 donde se generó, es por ello, que si la intención es la de aportar seguridad mediante el IDS, éste se debe situar dentro de la propia Red 1.

4.6.2 Supuesto 2: Situación en red distante

Ahora, suponiendo que las necesidades del sistema requieren la supervisión de una red distante, esto es, disponer de la máquina que aloje el IDS en una red independiente alejada de la red que se quiere supervisar, ¿podría Snort realizar su cometido en estas circunstancias? Para responderlo, podemos basarnos en un esquema similar al que se ha visto en el apartado anterior.

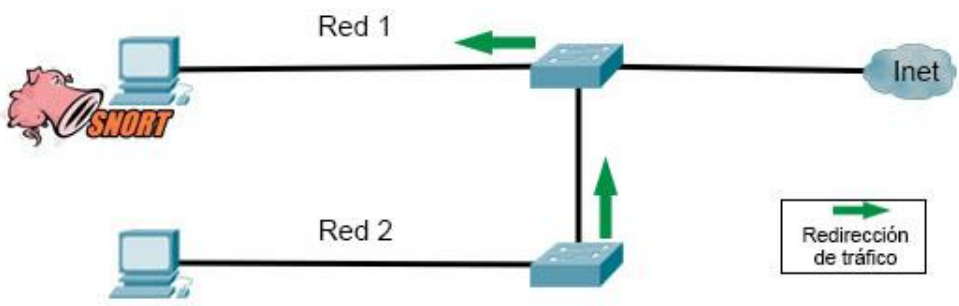

Figura 3.2. Situación en red distante

En este caso, supondremos que queremos inspeccionar y generar alarmas sobre el tráfico que se genera en la Red 2, pero nuestra máquina IDS está en la Red 1. Podríamos hacer que Snort trabajara en estas circunstancias, tan solo teniendo en cuenta que para ello, se deben hacer las configuraciones necesarias para que el IDS tenga acceso al tráfico de la red distante; en resumidas cuentas, estas configuraciones deben realizarse en los elementos intermedios de las redes que existan entre la Red 1 y la Red 2, y básicamente lo que se debe hacer es redireccionar el tráfico para que, además de llegar a su destino legítimo, llegue a la máquina que aloja Snort, con el fin de poder inspeccionar este tráfico y actuar en consecuencia. Por otra parte, debemos tener en cuenta el tipo de inspección y respecto a qué se requieren las medidas de detección, ya que, como se ha mencionado anteriormente en el caso del ARP, en el caso de tratarse de tráfico cercano al nivel físico, no quedará más remedio que situar a Snort en la misma red que se pretenda controlar.

5. ESCENARIO 1: INYECCIÓN DE CÓDIGO SQL

5.1 Introducción

En este escenario, se trata de detectar en el sistema la ocurrencia de intentos de ejecución de código malintencionado SQL. Este tipo de ataques, intentan realizar consultas no permitidas normalmente a la base de datos que utiliza una aplicación web, con el fin de hacerse con información a la que el usuario no debe tener acceso. Este tipo de ataques son efectivos por la deficiente implementación de la propia aplicación web, ya que, a la hora de su desarrollo, existen metodologías para evitar en medida de lo posible este tipo de acciones, como puede ser el filtrado de caracteres de entrada previos a la consulta a la base de datos.

La topología de la que dispondremos en este experimento será la siguiente:

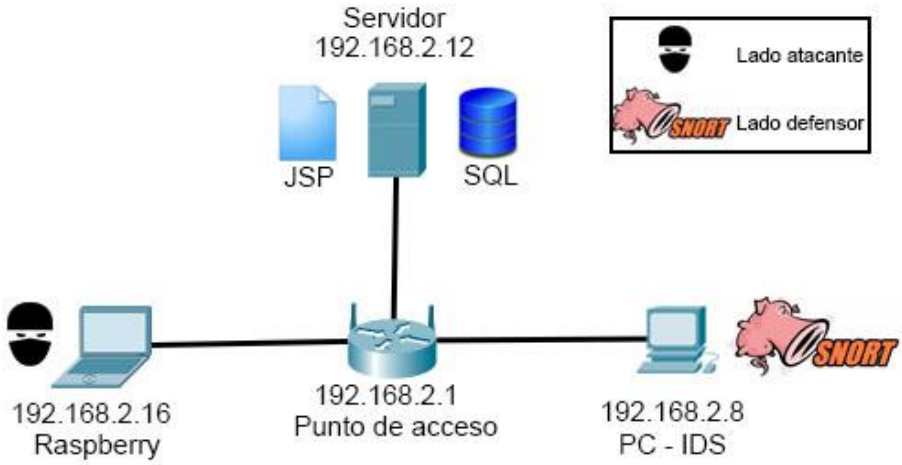

Figura 4. Topología escenario 1

Por una parte, en la dirección 192.168.2.12 se dispone de un servidor dotado de una aplicación web jsp denominada print.jsp, la cual está implementada de forma deficiente, cuyo único fin es mostrar los datos asociados a una cuenta proporcionando un id de usuario, la aplicación se encargará de hacer la consulta SQL y devolver los datos obtenidos. Además, la propia aplicación, permite mostrar en cada caso, el código inyectado, con el fin de visualizar de forma práctica el éxito de una inyección de este tipo, además de devolver también la información obtenida de forma maliciosa.

Por otra parte, bajo la dirección 192.168.2.16, nos encontramos al atacante, el cual tratará de obtener información a la que no tiene acceso directo haciendo uso de la aplicación .jsp de forma ilícita. Finalmente, en la dirección 192.168.2.8 disponemos el terminal de seguridad que ejecuta Snort, configurado en relación al ataque que debe detectar.

5.2 Ataque al sistema

Para realizar los ataques de este tipo, el intruso no requiere más que de un navegador web cualquiera, en el cual solicitará mediante la URL la ejecución de la aplicación. Esta llamada lícita, para la obtención de los datos del usuario con id=1, se realizaría mediante la URL http://192.168.2.12:8090/seguridad/print.jsp?id=1, a la que el atacante tan solo debería concatenar el código SQL de inyección que necesite, en función de la información que quiera obtener. Un ataque típico de inyección, contiene la cadena OR 1=1, lo que hace que la consulta que acompaña a esta cadena, siempre se ejecute, ya que la condición 1=1 se cumplirá en todos los casos.

Este ataque llevado a la práctica en el escenario, se podría hacer inyectando el código en la URL de la forma http://192.168.2.12:8090/seguridad/print.jsp?id=1 OR 1=1, lo que implicaría que, al cumplirse la condición 1=1, no solamente se nos devuelva la información del usuario con id=1, si no que también la del resto de usuarios, y así lo muestra la aplicación en el navegador del atacante, tras dicha inyección de código, donde podemos observar la consulta completa que se ha hecho a la base de datos, así como la información obtenida de forma ilícita de todos los usuarios.

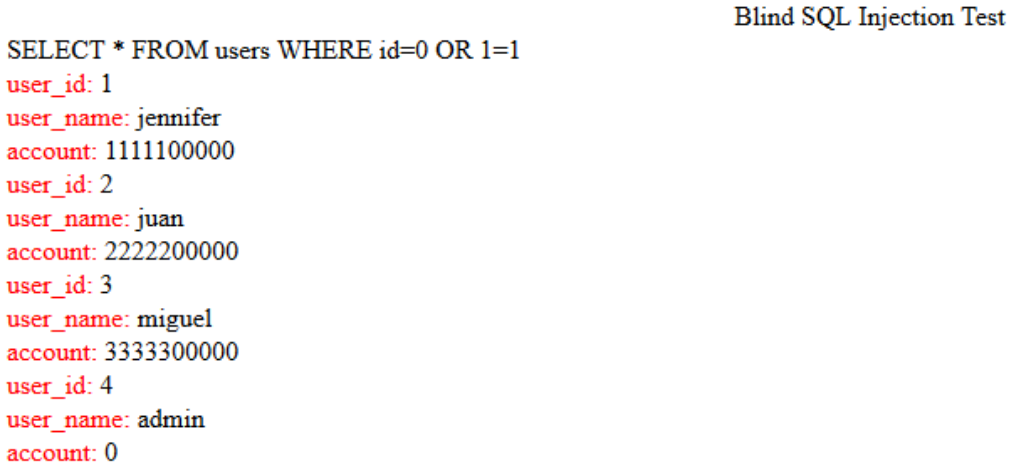

Figura 5. Resultado de inyección SQL

Tal y como hemos realizado este ataque, en la propia URL podríamos concatenar cualquier código de consultas SQL, siempre que la condición de partida se cumpla, el sistema nos devolverá la información que le hayamos solicitado mediante dicha consulta. Así pues, palabras reservadas del lenguaje SQL como UNION, DELETE o UPDATE entre muchas otras, se vuelven muy peligrosas para la integridad de la información de la base de datos.

5.3 Defensa del sistema

Tal y como se ha visto, la raíz del problema proviene de la inclusión de código no esperado en la propia URL de llamada a la aplicación, concretamente, de palabras que no esperaríamos recibir externamente, algo con lo que solo se trabajaría desde la propia aplicación de forma interna. Esta deficiente implementación de la aplicación, además, podría hacer que no seamos siquiera conscientes de que este tipo de ataques se están generando. Es por todo ello, que la configuración de la defensa mediante Snort, se ha basado en el análisis de la propia URL recibida, haciendo que se generen alarmas en función de las palabras clave recibidas, ya que Snort dispone de las herramientas necesarias para proceder al análisis del contenido de la URL.

Se han creado las reglas necesarias para la detección de estas inyecciones atendiendo a algunas de las palabras clave más usuales, además de para el caso de la condición 1=1. Por otra parte, las propias reglas se han dividido en grupos, para facilitar la detección del tipo de inyección que se está realizando, así pues, se han agrupado de la siguiente forma.

Sid	Tipo	Palabra clave
1	Condición OR 1=1	OR1=1
2	Obtención de contenido de las tablas	SELECT
3		FROM
4		WHERE
5		UNION
6	Obtención de información del sistema	DATABASE
7		VERSION
8		SCHEMA
9	Obtención de información de la sesión	USER
10		CONN_ID

11	Modificación del estado de la base de datos	DROP
12		DELETE
13		CREATE
14		UPDATE
15		ADD

Tabla 6. Agrupación de reglas

Para centrarnos en la defensa de este tipo de ataque, además, se han creado las reglas haciendo uso de las siguientes opciones:

- Content:”GET” y http_method

  Establecemos que Snort analice el campo método de HTTP, comprobando que el método sea GET.

- Content:”[x]” y http_uri

  Se configura la regla para que se analice la URI del paquete, y mediante la opción content, sustituyendo x por alguna de las palabras clave vistas anteriormente, se comprobará la existencia de la misma.

- Nocase

  Haremos que el sistema no haga distinciones entre mayúsculas y minúsculas en las ocurrencias.

Así pues, un extracto del contenido del fichero de reglas reglas_propias.rules para este escenario, es el que se muestra a continuación, en el que tan solo se muestra una regla por cada grupo de reglas visto con anterioridad, siendo las reglas faltantes similares, teniendo en cuenta el cambio de la palabra reservada buscada en la URL. Las líneas marcadas con el símbolo almohadilla (#) hacen referencia a comentarios explicativos, que el IDS ignorará en su ejecución. Cabe destacar, que en este caso no se ha tenido en cuenta el sentido de la comunicación ni las máquinas involucradas en la misma, por lo que estos ataques serán detectados en la totalidad de la red en la que Snort se encuentre.

```
#Condición OR 1=1
alert tcp any any -> any any (msg:"Inyección SQL (OR 1=1)"; content:"GET"; http_method;content:"OR 1=1"; http_uri; nocase; classtype:web-application-attack; sid:1; rev:1;)
```

```
#Obtención de contenido de las tablas mediante comandos MySQL
alert tcp any any -> any any (msg:"Inyección SQL-Información de contenido (SELECT)"; content:"GET"; http_method; content:"SELECT"; http_uri; nocase; classtype:web-application-attack; sid:2; rev:1;)

#Obtención de información del sistema
alert tcp any any -> any any (msg:"Inyección SQL-Información del sistema (database)"; content:"GET"; http_method; content:"database"; http_uri; nocase; classtype:web-application-attack; sid:6; rev:1;)

#Obtención de información de la sesión actual
alert tcp any any -> any any (msg:"Inyección SQL-Información de sesión (user)"; content:"GET"; http_method; content:"current_user"; http_uri; nocase; classtype:web-application-attack; sid:9; rev:1;)

#Modificación del estado de la base de datos
alert tcp any any -> any any (msg:"Inyección SQL-Modificación de BD (DROP)"; content:"GET"; http_method; content:"DROP"; http_uri; nocase; classtype:web-application-attack; sid:11; rev:1;)
```

Una vez creadas las reglas, procederemos al arranque de Snort desde la línea de comandos, mediante el comando Snort –A fast –K ascii –C /etc/snort/snort.conf, donde la opción –A indica el tipo de alarma que queremos generar, -K el formato en el que queremos que se registren las alarmas y –C indica el fichero de configuración que deseamos que Snort cargue en la ejecución.

5.4 Resultados de las pruebas realizadas

Después del arranque del IDS, se procede a la realización de los diferentes ataques tal y como se han descrito con anterioridad, cabe decir que se ha realizado una cantidad de 60 ataques, comprobando todas y cada una de las palabras clave que se han tenido en cuenta a la hora de la creación de las reglas, lo que finalmente se ha traducido en una detección de la totalidad de los ataques realizados. Así pues, se muestra un extracto del fichero de alertas generado por Snort en el proceso llevado a cabo.

```
08/09-11:34:09.475055  [**] [1:1:1] Inyección SQL (OR 1=1) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47108 -> 192.168.2.12:8090

08/09-11:37:17.343352  [**] [1:2:1] Inyección SQL-Información de contenido (SELECT) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47112 -> 192.168.2.12:8090

08/09-11:38:17.571071  [**] [1:11:1] Inyección SQL-Modificación de BD (DROP) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47112 -> 192.168.2.12:8090

08/09-11:38:29.812182  [**] [1:12:1] Inyección SQL-Modificación de BD (DELETE) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47112 -> 192.168.2.12:8090

08/09-11:39:53.087868  [**] [1:6:1] Inyección SQL-Información del sistema (database) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47114 -> 192.168.2.12:8090

08/09-11:40:35.643683  [**] [1:15:1] Inyección SQL-Modificación de BD (ADD) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47114 -> 192.168.2.12:8090

08/09-11:42:27.579344  [**] [1:3:1] Inyección SQL-Información de contenido (FROM) [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.16:47116 -> 192.168.2.12:8090
```

En estas alarmas generadas podemos observar a qué grupo de los expuestos anteriormente pertenecía cada ataque, así como la regla que ha hecho que se genere la alarma y la dirección del flujo de información, lo que permite establecer quién ha sido el atacante, y quién la víctima del ataque.

6. ESCENARIO 2: ARP SPOOFING

6.1 Introducción

En el siguiente escenario, se propone la configuración necesaria para la detección de posibles ataques de tipo ARP Spoofing. El protocolo ARP (Address Resolution Protocol) [2], es un protocolo de comunicaciones de nivel de red, encargado de determinar la dirección física (MAC) que se corresponde a una dirección de red (IP). Cuando una máquina pretende enviar información a otra, conociendo su dirección IP, pero no su dirección MAC, antes de proceder al envío, entra en juego este protocolo, el funcionamiento del cual se describe seguidamente.

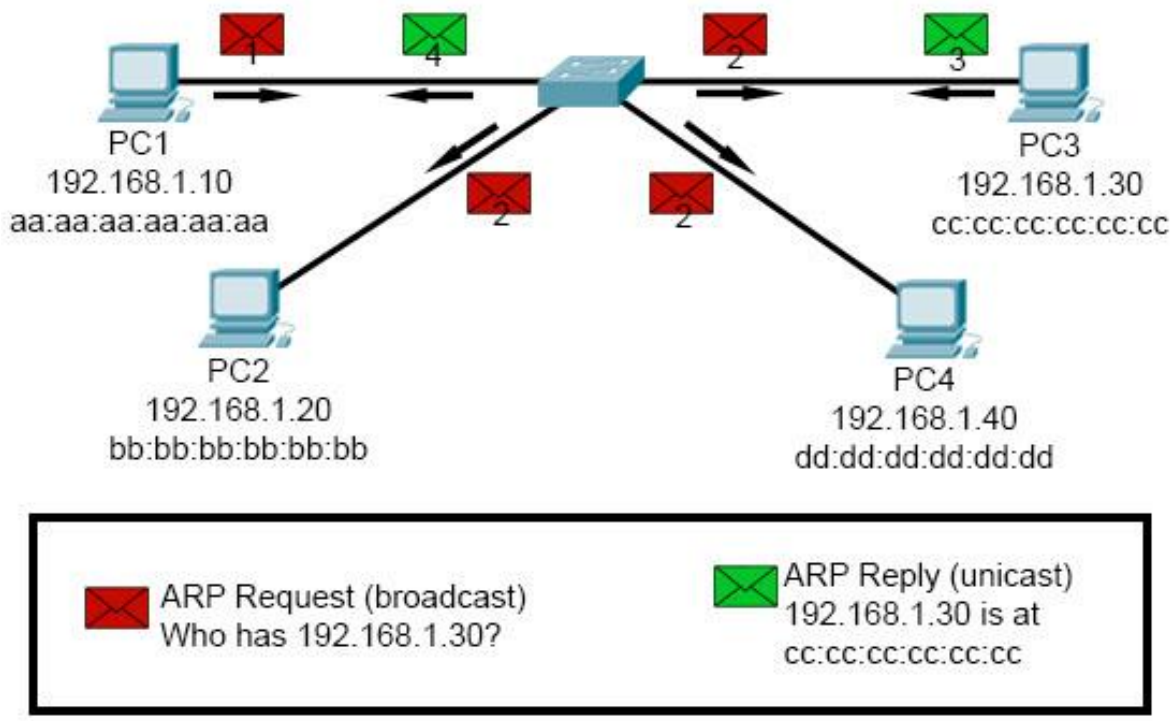

Figura 5. Funcionamiento de ARP

Suponiendo que el terminal PC1 quiera hacer el envío de información hacia la PC3 y no conozca su dirección física, la máquina origen (PC1), enviará un paquete ARP Request con dirección broadcast (por lo que llegará a todas los terminales de la red), preguntando por la dirección de MAC que pertenece a la IP conocida del PC3 (1), este paquete será retransmitido hacia todas las máquinas que conforman la red (2), de todas

ellas, tan solo responderá la máquina a la cual se corresponda dicha IP (PC3) mediante un paquete ARP Reply, informando de su dirección física (3). Este paquete tendrá como destino la máquina que envió la petición ARP Request (PC1) y será retransmitido hacia ella (4). De esta forma, la máquina origen, actualizará su tabla ARP, y ya tendrá el conocimiento necesario para realizar el envío de información a la máquina destino.

Un ataque ARP Spoofing, básicamente, consiste en generar incongruencias en las tablas ARP de emparejamiento MAC-IP que almacenan los terminales, de forma que se asocian direcciones IP que no se corresponden legítimamente con las direcciones MAC que se almacenan. Todo ello con el fin de hacer que la información que tiene como destino una máquina, realmente sea redirigida hacia otra, la cual originalmente no debería ser receptora de esta información. En resumen, mediante la suplantación de la dirección IP, una máquina se hace pasar por alguien que realmente no es. Este tipo de ataques son la antesala de otro tipo de ataques más complejos, como por ejemplo, un ataque man-in-the-middle, en el que una máquina ajena se interpone entre otras dos que se estén comunicando, de forma que esta máquina intrusa, tiene el acceso a toda la información que estén compartiendo las otras dos.

En este caso, la topología que se dispone es la que se muestra a continuación. Las pruebas realizadas, se han llevado a cabo sobre una red inalámbrica WiFi. Hay que tener en cuenta que es obligatorio que la máquina que aloja el IDS Snort esté en la misma red que se pretende supervisar, ya que el protocolo ARP [2], no alcanza más allá de la propia red sobre la que se utiliza debido a su propia constitución, ya que su cometido es el de asociar direcciones físicas con direcciones de la propia red.

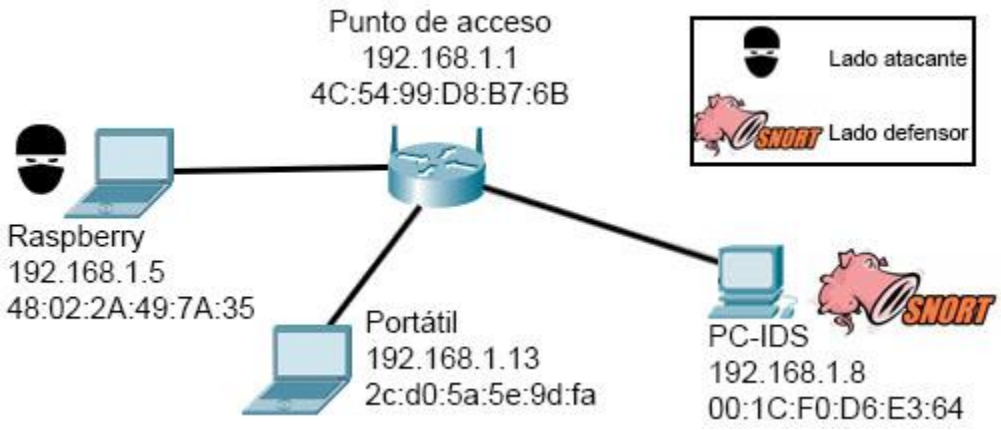

Figura 6. Topología escenario 2

Tal y como se muestra en la imagen de la topología de la red en este supuesto, disponemos de tres máquinas. Una de ellas, denominada PC-IDS será la que aloje Snort

y supervise el tráfico de la red. Por otra parte, dispondremos de la Raspberry, desde la que se ejecutarán los diversos ataques sobre nuestra tercera máquina (portátil), la cual no tendrá otro cometido que el de ser el objetivo del atacante. Estos ataques estarán centrados en hacer creer al objetivo, que la IP del punto de acceso se encuentra en la dirección física de la Raspberry, con el fin de que el objetivo envíe su tráfico al atacante.

6.2 Ataque al sistema

Para este ataque, se requiere una aplicación que permita el envío de paquetes ARP con la opción de modificar los campos del mismo, con el fin de alterar las tablas ARP con información incorrecta. Este tipo de envío de información, rompe en cierta forma con el uso normal de ARP, ya que, si tenemos en cuenta el funcionamiento normal explicado con anterioridad, en este caso, estamos forzando el envío de paquetes ARP Reply sin haber recibido ningún tipo de solicitud ARP Request, este tipo de envío de paquetes ARP se denomina ARP gratuito. En algunas ocasiones se utiliza el ARP gratuito con el fin de determinar la existencia de otra máquina en la red que ya esté utilizando la dirección IP que se pretende utilizar, normalmente esto se produce en el arranque de una interfaz de red.

Una aplicación Linux que nos permite hacerlo es arpspoof [8], la cual es una herramienta que se incluye en el paquete de instalación dsniff. Para instalarlo en nuestra raspberry, que será la máquina atacante, lo hacemos mediante la ejecución en un terminal del comando apt –get install dsniff. Una vez finalizada la instalación, ya estará listo para su uso. Para la ejecución de esta aplicación en línea de comandos, requeriremos únicamente las opciones –i, para definir la interfaz de salida del paquete; la opción –t determinará la dirección IP de la víctima. Así pues, la ejecución de la aplicación se define como:

```
Arpspoof –i [interfaz] –t [ip victima] [ip suplantada]
```

Por lo que en el ataque llevado a la práctica, se trata de hacer creer a la máquina 192.168.1.13 que el Gateway 192.168.1.1 realmente se encuentra en la dirección MAC del atacante (raspberry), esto se realiza de la forma:

```
Arpspoof –i wlan1 –t 192.168.1.13 192.168.1.1
```

Por otra parte, y a modo de curiosidad, para realizar la fase de ARP Spoofing en un ataque man-in-the-middle, deberíamos ejecutar el comando simultáneamente sobre la máquina denominada portátil (1) y sobre el punto de acceso (2), con el fin de que ambos envíen la información a la máquina del atacante, logrando así el intruso establecerse entre los dos extremos de la comunicación. Esto se realizaría, sobre una red WiFi, de forma simultánea en dos terminales, ejecutando los comandos:

```
(1) arpspoof –i wlan1 –t 192.168.1.13 192.168.1.1
(2) arpspoof –i wlan1 –t 192.168.1.1 192.168.1.13
```

Lo que daría un flujo de información como el que se muestra a continuación:

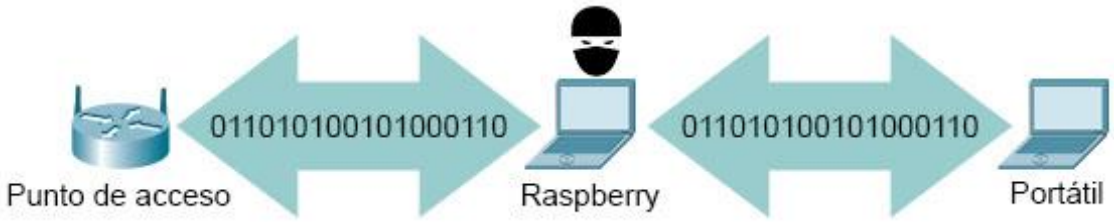

Figura 7. Man in the Middle

A partir de aquí, el atacante retransmitiría la información recibida por el punto de acceso hacia PC y viceversa, de forma que para ellos sería casi algo transparente, y él tendría además el acceso a toda la información transmitida.

6.3 Defensa del sistema

Para proceder a la configuración del sistema para su defensa, en este caso disponemos de un preprocesador dedicado exclusivamente al tratamiento de este tipo de ataque, concretamente se trata del preprocesador arpspoof, el cual ya se comentó en el apartado de preprocesadores, dentro de la presentación del IDS Snort.

La configuración de este preprocesador en nuestro fichero de reglas, pasa primeramente por su activación, algo que se realiza simplemente haciendo una instanciación del mismo como se verá a continuación. Para el uso de este preprocesador, disponemos de dos modos de funcionamiento, el que podemos denominar como normal, y el modo unicast.

Por otro lado, en nuestro fichero de reglas y con posterioridad a la activación del preprocesador en cuestión, debemos proporcionar los binomios dirección IP – dirección MAC que queramos que Snort mapee, de forma que si se detectan paquetes que intenten romper estas correspondencias de direcciones, se producirá una alarma. Estos binomios a mapear se proporcionarán mediante la orden arpspoof_detect_host: [IP MAC]. En nuestro caso y debido al tipo de ataque que vamos a utilizar, el modo de ejecución elegido será el normal, además, proporcionaremos las parejas de direcciones de todos las máquinas que conforman la red, incluida la del terminal atacante, ya que podría tratarse de un ataque que se produzca desde un terminal que pertenezca a la propia red.

Así pues, nuestro archivo de reglas reglas_propias.rules, esta vez contendrá las siguientes líneas, en el caso de querer que el preprocesador se ejecutara en modo unicast, únicamente deberíamos añadir la opción –unicast en su instanciación.

```
# Inicialización del preprocesador
preprocessor arpspoof
# Declaración de las parejas de direcciones a mapear
# Punto de acceso
preprocessor arpspoof_detect_host: 192.168.1.1 4C:54:99:D8:B7:6B
# PC-IDS
preprocessor arpspoof_detect_host: 192.168.1.8 00:1C:F0:D6:E3:64
# Portátil
preprocessor arpspoof_detect_host: 192.168.1.13 2C:D0:5A:5E:9D:FA
# Raspberry
preprocessor arpspoof_detect_host: 192.168.1.5 48:02:2A:49:7A:35
```

6.4 Resultados de las pruebas realizadas

En esta ocasión, tras las pruebas realizadas, teniendo en cuenta la ejecución de más de 60 ataques, Snort no ha sido capaz de generar ninguna alerta. Tras la comprobación de la correcta configuración, así como de la adquisición de información al respecto, y viendo que es un suceso conocido y para nada aislado, se ha procedido a la realización de pruebas adicionales, con el fin de comprobar el funcionamiento de este preprocesador en otros supuestos que se detallarán más adelante. En relación a las pruebas ya realizadas, se obtiene la siguiente captura de tráfico en Wireshark, filtrada para observar tan solo el tráfico ARP.

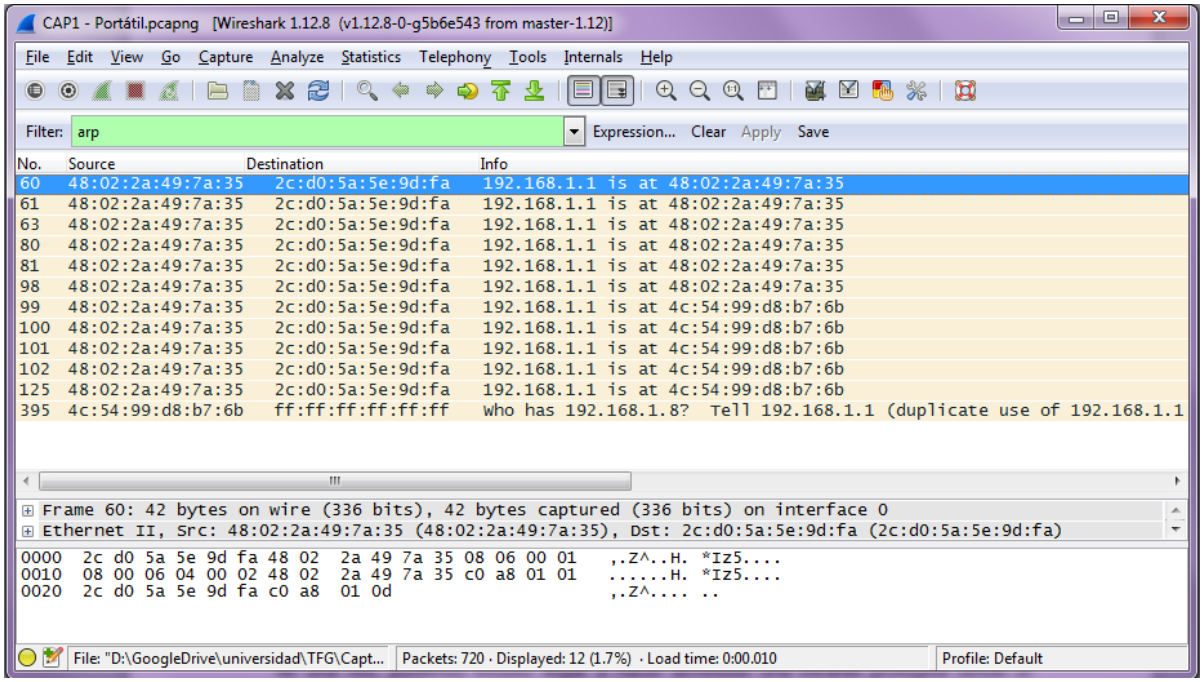

Figura 8. Captura de Wireshark escenario 2

Esta captura obtenida deja constancia de que el ataque se está realizando de forma óptima, ya que en las tramas de la 60 a la 98 (incluidas), se observa la ejecución del ataque ARP Spoofing, mediante el envío de información errónea. En estas tramas puede observarse como Raspberry realiza el envío mediante mensajes ARP Reply (gratuito), tratando de hacerse pasar por la dirección IP del punto de acceso. Por otra parte, desde las tramas 99 a la 125, se observa como la propia Raspberry vuelve a enviar dichos mensajes ARP, pero esta vez con información veraz. Esto se debe a que el ataque cesó entre las tramas 98 y 99, y la aplicación arpspoof está implementada de forma que cuando el ataque termine, vuelva a volcar la información lícita de la que se disponía antes del inicio del ataque a la red. Por último, en la última trama, tras el ataque, se puede observar cómo incluso Wireshark, es capaz de detectar que se está intentando suplantar la MAC relacionada con la dirección IP 192.168.1.1. Por lo que se puede concluir que el ataque se está realizando correctamente, algo que puede comprobarse inspeccionando, por ejemplo, la primera trama (60) de la captura.

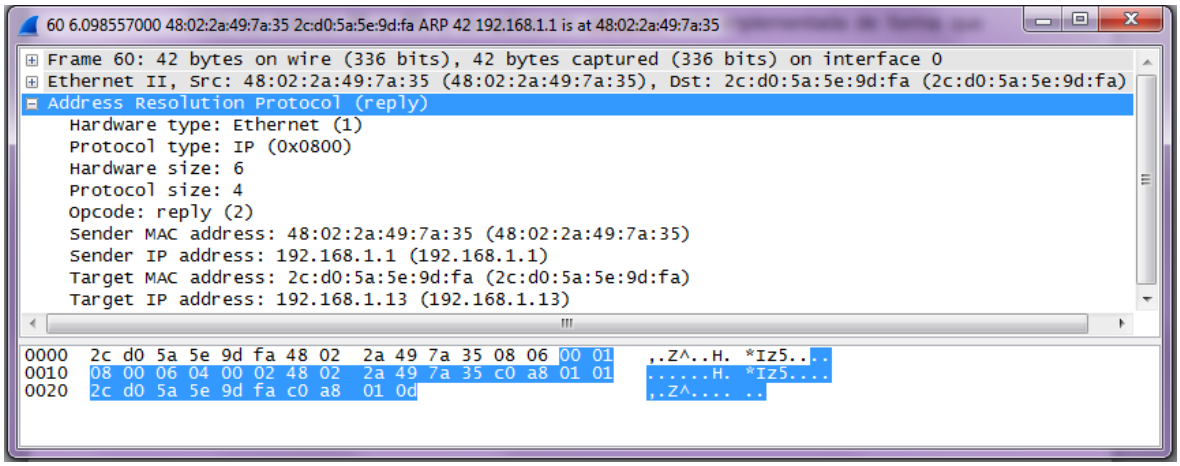

Figura 9. Trama obtenida en escenario 2

Cabe decir que, tras los resultados obtenidos y la búsqueda de solución al respecto en la propia comunidad de Snort, así como en diversos foros especializados tales como http://security.stackexchange.com, http://seclists.org o https://sourceforge.net entre otros, y viendo que en estos mismos el problema es conocido y las soluciones que se aportan no han dado resultado, se ha procedido a la notificación de este hecho en la comunidad de Snort, previo registro y aceptación, donde tras la suscripción a la lista de correos de usuarios, se puede participar en la misma, dejando consultas y siendo éstas comentadas por los propios usuarios que conforman la comunidad.

6.5 Pruebas adicionales

6.5.1 Prueba adicional 1

Con el fin de indagar más en profundidad en lo acontecido, se realizan algunas pruebas adicionales. En la primera de ellas, se va a proceder eliminando la máquina portátil del esquema de red propuesto anteriormente, de forma que ahora el ataque se realizará directamente sobre la máquina que aloja el IDS. Así pues, la nueva disposición de la red, es la que se muestra a continuación.

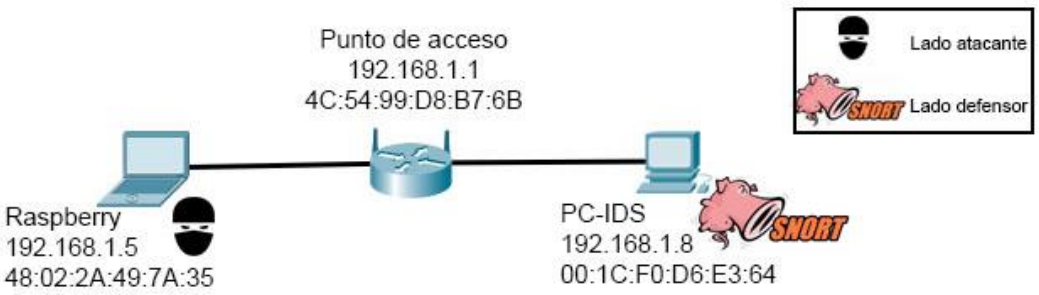

Figura 10. Topología prueba adicional 1

Tras la realización de una nueva consecución de ataques sobre PC-IDS, y teniendo en cuenta que el preprocesador ha sido configurado tanto en modo unicast, como en el modo general, los resultados que se han obtenido han sido los mismos que en el anterior caso, Snort no ha realizado detección alguna. Se puede determinar que, efectivamente y como ya se suponía, el tráfico llega sin problemas a la máquina objetivo, como da cuenta la siguiente captura de wireshark obtenida desde la propia máquina IDS, muy similar a la comentada con anterioridad, y donde el único cambio es la dirección objetivo del ataque.

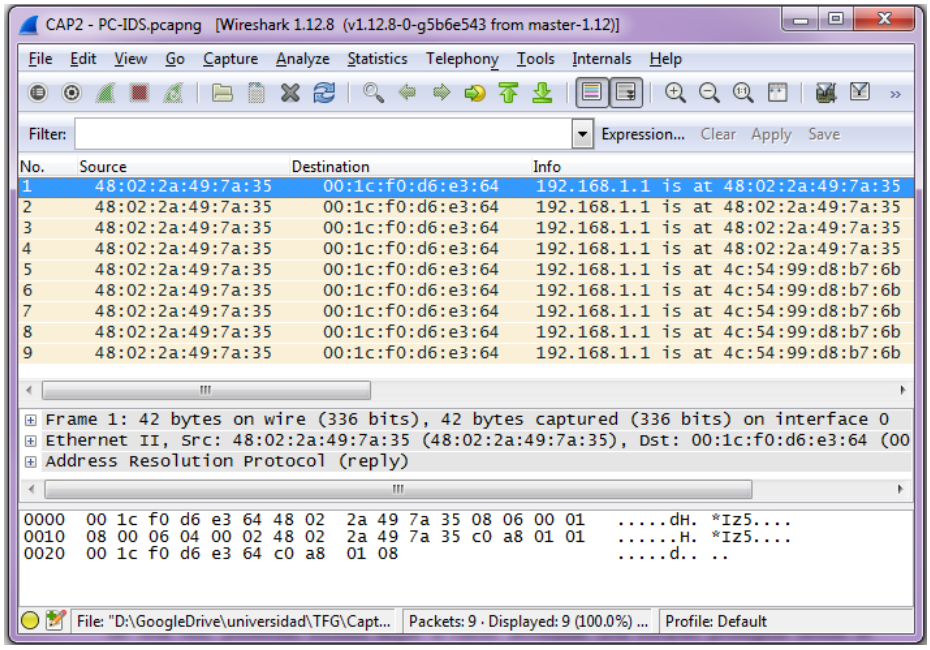

Figura 11. Captura wireshark prueba adicional 1

Donde si examinamos la primera trama con más detalle, comprobamos que el ataque se está realizando de forma correcta.

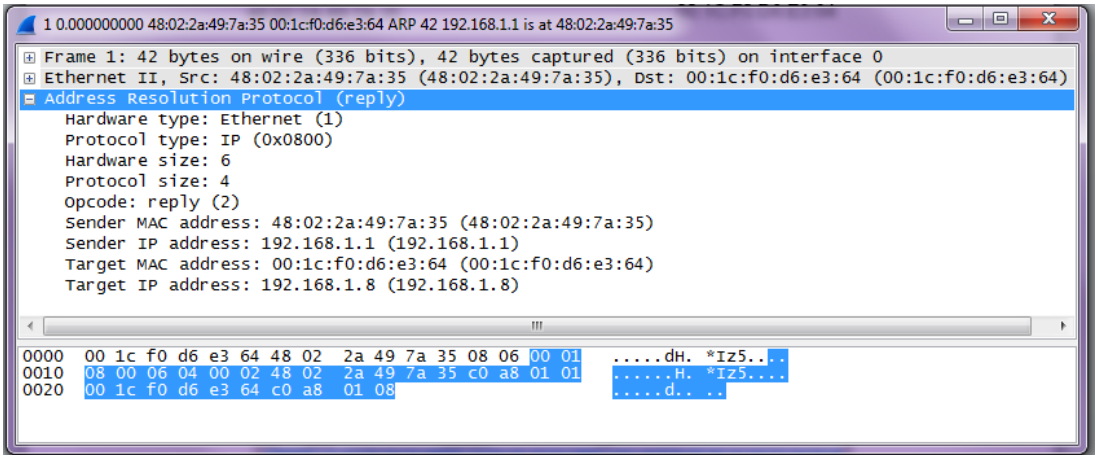

Figura 12. Trama obtenida en prueba adicional 1

6.5.2 Prueba adicional 2

Tras la prueba anterior, se procede a la realización de una más, en ella se obviará un ataque ARP Spoofing al uso, y de lo que se tratará es de hacer funcionar el protocolo ARP en la red de forma normal, con el fin de discernir si la ausencia de alarmas por parte de Snort puede venir dada por la ejecución de un ataque ineficiente.

Para ello, antes de nada, en la base de reglas del IDS, se almacenará una pareja de direcciones MAC-IP errónea, que harán referencia a la Raspberry, la cual será motivo de alarma al detectar el tráfico ARP legítimo que posteriormente se producirá, ya que no coincidirán las direcciones almacenadas en la base de reglas, con las que analizarán en los paquetes ARP. Por otro lado, desde la máquina IDS, se tratará de realizar una comunicación con dicha Raspberry, no sin antes haber vaciado la caché de direcciones ARP de PC-IDS, con el fin de que al intentar realizar la comunicación, mediante un simple ping, se fuerza a que la máquina a efectuar todo el proceso de descubrimiento de MAC en la red, y debido a la información errónea que, conscientemente, se ha proporcionado al fichero de reglas, Snort debería alertar de lo sucedido, ya que la información recibida al ser comparada con la almacenada, será incongruente.

Para empezar, lo primero que debemos realizar es el cambio en el fichero de reglas, de forma que a la pareja de direcciones de la Raspberry, le proporcionaremos una dirección MAC errónea, por lo que se modificará la línea que hace referencia a estas direcciones, quedando de la siguiente forma:

```
preprocessor arpspoof_detect_host: 192.168.1.5 aa:aa:aa:aa:aa:aa
```

Ahora, proseguimos con la caché ARP de la máquina, esta caché podemos consultarla ejecutando en un terminal el comando ip neigh, el cual nos da el siguiente resultado:

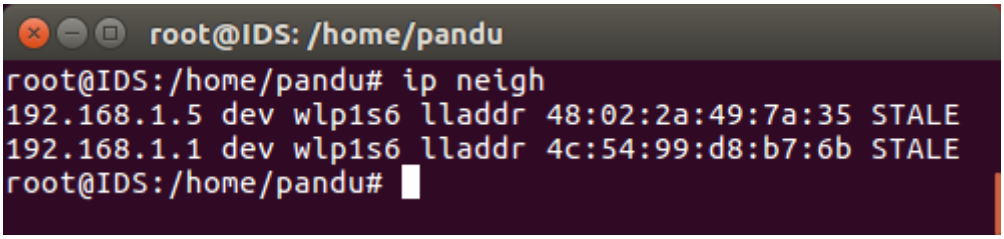

Figura 13. Consulta de caché ARP

Como se puede observar, la máquina dispone de la información necesaria para establecer comunicación con nuestra Raspberry. Ahora se procederá al vaciado de la

caché, privando a la máquina de esta información, lo que se realiza mediante el comando ip neigh flush dev wlp1s6 , teniendo en cuenta que el argumento dev hace referencia al dispositivo de red al que se quiere hacer alusión. Así pues, tras ejecutar este comando, y nuevamente consultando la caché ARP, se observa como ha sido eliminada la información.

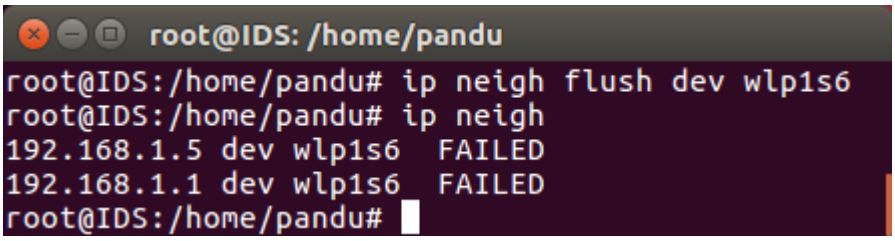

Figura 14. Vaciado y verificación de caché ARP

Y como paso final, podemos proceder a la ejecución del ping con destino 192.168.1.5 y origen 192.168.1.8, el cual queda registrado en la siguiente captura de tráfico.

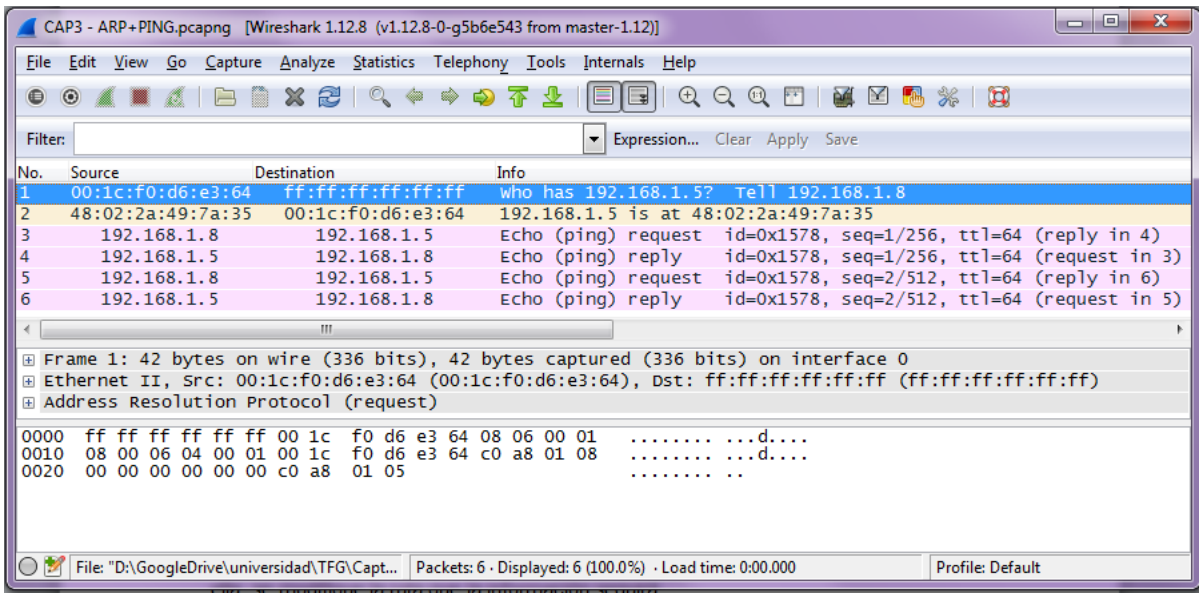

Figura 15. Captura de Wireshark prueba adicional 2

Como puede observarse en la primera trama, al no disponer de la dirección física a la que se hace referencia mediante la IP solicitada por haber eliminado la información de la tabla ARP, la máquina se ve obligada, previo al envío del ping, a realizar una consulta ARP Request con dirección broadcast, con el fin de determinar a qué máquina concreta hace referencia dicha IP. En el momento de que la Raspberry recibe este mensaje, ya que ella tiene la dirección por la que se está preguntando en el ARP Request, se encarga

de responder como puede verse en la segunda trama, mediante un ARP Reply con dirección destino unicast a la máquina solicitante, lo que hará que se actualice la información de su tabla ARP, y por consiguiente, que pueda proseguirse con la realización del ping, tal y como muestran las tramas restantes.

Después de este proceso, y esperando que Snort genere las alarmas pertinentes, al no coincidir la información de la que dispone en el fichero de reglas, con la información que está discurriendo por la red, debiendo supervisar la misma, el resultado vuelve a ser una vez más negativo para el IDS, ya que en ningún momento se produce ningún tipo de alarma. Así pues, y tras todas las pruebas realizadas referentes al ARP Spoofing, Snort no ha sido capaz de detectar estos ataques.

7. ESCENARIO 3: ICMP REDIRECT

7.1 Introducción

En el estudio de este escenario, se propone adoptar medidas de seguridad en cuanto a los mensajes ICMP (Internet Control Message Protocol) [3] de tipo redirección, ya que este tipo de mensajes, pueden dar lugar a ataques que comprometan la intergridad de una red, pudiendo incluso llegar a hacer accesible una intranet protegida desde el exterior. Básicamente, estos mensajes ICMP, dotan a los puntos de acceso de la capacidad de informar a los hosts de la existencia de rutas mejores para el envío de sus paquetes, lo que provoca en los hosts la actualización de la tabla de enrutamiento, y, con ella, se modifique la ruta que la información seguirá.

El uso malintencionado de la redirección ICMP, está basada en la modificación de estas tablas de rutas no por el hecho legítimo de proporcionar una ruta mejor, si no para hacer llegar la información a un lugar donde ésta no debería terminar, lo que podría traducirse en un acceso a información confidencial si hacemos que una máquina redirija la información hacia las manos ajenas que la codician, o por ejemplo, lograr el acceso a un servidor de una intranet al que normalmente no tendríamos acceso desde el exterior, haciendo que nuestra información sea redirigida por una máquina intermedia perteneciente a la intranet hacia el propio servidor.

Por algunas razones como estas, normalmente se deshabilita el uso de estos tipos de mensajes ICMP, aún y así, en la red de la que se dispone para este escenario, dotaremos a nuestro IDS de las capacidades de detección y alarma de los diferentes tipos de mensajes de redirección existentes que se definen en la RFC 792 [3], además, esta vez, no solo se programarán las reglas necesarias para su detección, si no que a las propias reglas se las dotará de medidas de respuesta cuando una detección de este tipo se produzca. Esta vez, la topología de red sobre la que se trabaja, se dispone como se muestra a continuación.

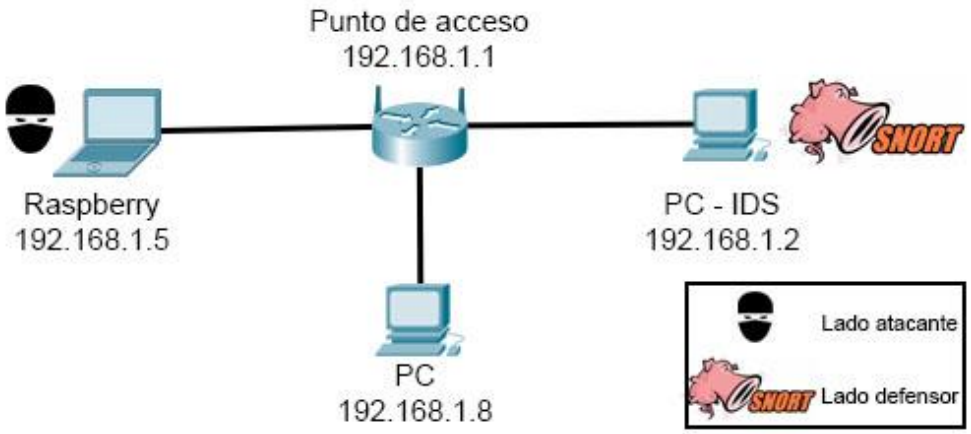

Figura 16. Topología escenario 3

Como se muestra en la figura, se sigue manteniendo como equipo atacante bajo la dirección 192.168.1.5, el SBC Raspberry. En otro punto de la red se tiene la máquina en la que se aloja del IDS Snort con dirección 192.168.1.2, la cual tendrá las reglas necesarias para la detección de los paquetes de redirección. Por otra parte, en este escenario se dispone un terminal que hará las veces de objetivo, bajo la dirección 192.168.1.8, esta máquina será la destinataria de los paquetes ICMP de redirección que enviará el atacante. Para dar conectividad bajo una misma red a todas estas máquinas, se instala un punto de acceso WiFi, cuya dirección de puerta de acceso es la 192.168.1.1.

Teniendo en cuenta la RFC 792 [3] a la que se ha hecho mención con anterioridad, concretamente en el apartado de los mensajes de redirección, vemos que en el campo type de los mismos, debe constar el número 5 para ser considerado de tipo redirección. Algo que también debe ser tenido en cuenta, es que para los mensajes de redirección, existen diferentes códigos identificativos que dependerán del tipo de mensaje de redirección del que se trate. Estos tipos de redirección, se codifican en el campo código del mensaje ICMP mediante un valor numérico, pudiendo ser el mismo:

0 Redirección para la red.
1 Redirección para el host.
2 Redirección para servicio y red.
3 Redirección para servicio y host.

Por lo que dentro de los propios paquetes de redirección ICMP, sería interesante tener en cuenta también de qué tipo de redirección se trata, para así poder acotar mejor el ataque que se está realizando. Toda esta información será tenida en cuenta a la hora de la programación de las reglas del IDS, ya que, tal y como se verá en el apartado de la

defensa de este escenario, Snort dispone de las herramientas necesarias para la diferenciación no sólo del tipo de mensaje ICMP del que se trata, si no también de qué tipo de redirección se está llevando a cabo.

7.2 Ataque al sistema

En este caso, para proceder a comprometer la seguridad del sistema, lo que se tratará de hacer es modificar la tabla de enrutamiento de la máquina objetivo (192.168.1.8), haciendo uso de los mensajes ICMP redirect. A este terminal se le envían paquetes informando de que debe redirigir el tráfico con destino a 192.168.1.1 a través de 192.168.1.5, lo que supone el envío de la información directamente al atacante. Se enviarán mensajes de redirección de los tipos vistos con anterioridad, con el fin de realizar posteriormente una detección concretizada de cada uno de los tipos.

Para realizar estas operaciones se requiere de una aplicación capaz de realizar el envío de mensajes ICMP, siendo capaz de incluir dentro de los campos del mensaje la información que deseemos. La herramienta elegida para este fin es la aplicación de consola llamada icmpush [9], la cual instalaremos en nuestra máquina atacante mediante el comando apt-get install icmpush, ejecutado desde un terminal con permisos de administrador. Una vez instalada la misma, ya podemos proceder a su utilización.

Esta aplicación proporciona un amplio abanico de posibilidades en cuanto al envío de mensajes ICMP, no solo los de tipo redirección, si no de casi cualquier tipo, tal y como se observa en el estudio de su manual, aún y así, para los fines que se pretenden, únicamente se utilizará para los de tipo ICMP redirect. Para utilizar esta aplicación, como la mayoría de aplicaciones de consola, basta con llamarla haciendo uso de su nombre, e incluir las opciones que se requieran para el uso que queramos darle. En nuestro caso, estas opciones serán:

- -red

  Indica que el mensaje ICMP enviado será de tipo redirección (type:5)

- -sp [IP]

  Dirección IP de spoofing, es decir, a qué dirección se está suplantando como emisor del mensaje, dicho de otra forma, en nombre de quién se envía el mensaje.

- -dest [IP]

  Dirección IP destinataria de la redirección. A qué máquina irá ahora la información.

- -c [host|net|serv-net|serv-host]

  Únicamente para mensajes de tipo redirección, se refiere al código de tipo ICMP redirect de los dispuestos anteriormente. Esta opción irá acompañada de dicho tipo, pudiendo ser este: host, net, serv-net o serv-host.

Teniendo todo ello en cuenta, la ejecución genérica de la aplicación para este propósito, sería tal que:

```
icmpush –red –sp [IP Spoof] –dest [IP dest] –c [host|net|serv-net|serv-host] [IP Target]
```

Y, concretamente para la topología que se dispone, la llamada a la aplicación se haría de la forma:

```
icmpush –red –sp 192.168.1.1 –dest 192.168.1.5 –c host 192.168.1.8
```

Lo que generará un mensaje de tipo redirección, con la dirección del punto de acceso como origen (192.168.1.1), y la víctima como receptor (192.168.1.8), informando de que debe redirigir su tráfico hacia la dirección del atacante (192.168.1.5).

Para posteriores ejecuciones y variar los tipos de mensajes de redirección, iremos variando la opción –c con las diferentes variantes que se han dispuesto con anterioridad, tal y como se muestra en la imagen a continuación.

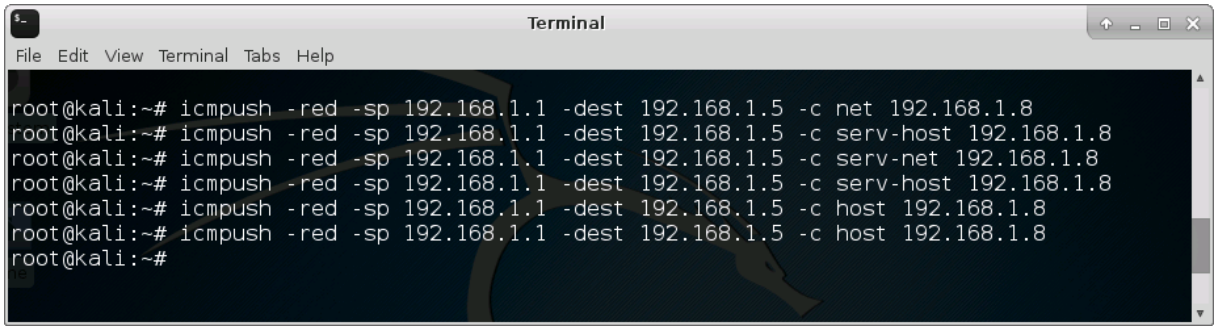

Figura 17. Ataques icmpush

7.3 Defensa del sistema

En este caso, la defensa se volverá a realizar en base a reglas programadas que analicen el contenido de los mensajes ICMP, ya que Snort cuenta con habilidad sobrada en cuanto a la inspección de paquetes y mensajes, y en este caso, no se trata de detectar un comportamiento anómalo que requiera de un preprocesador. ICMP tiene los campos bien definidos en la RFC 792 [3], junto con los valores esperados, por lo que únicamente debemos inspeccionar los mensajes recibidos en búsqueda de un mensaje ICMP, en el cual en su campo type se incluya el valor 5. Además, las reglas se programan para discernir también de qué tipo de paquete de redirección se trata, por lo que Snort deberá inspeccionar también el campo code del mensaje ICMP redirect. Con tal de probar la capacidad de Snort para diferenciar los tipos de redirección, se crearán las reglas para cada uno de los tipos de redirección salvo uno, concretamente para el tipo 1, redirección para el host, el cual no será tenido en cuenta. Aún y así, se enviarán mensajes ICMP de este tipo, debiendo el IDS no realizar acción alguna, ya que no se tiene en cuenta en la base de reglas.

Las opciones especiales utilizadas en la generación de estas reglas para la detección serán:

- itype:[int]

  Da cuenta el tipo de mensaje ICMP en base al que se quiere generar la alarma, en nuestro caso, y atendiendo directamente a la RFC 792 [3], sabemos que el tipo redirección está codificado como 5.

- icode:[int]

  Mediante esta opción, diferenciaremos el código de tipo de redirección tal y como indica la RFC, como ya se vio anteriormente, estos códigos pueden ser:

  0. Redirección para la red.
  1. Redirección para el host.
  2. Redirección para servicio y red.
  3. Redirección para servicio y host.

Por otra parte, y para dotar de mayor versatilidad al IDS, esta vez, en las reglas se utiliza la opción resp, la cual le proporciona a la regla la capacidad de dar una respuesta

ante una detección. Desde el punto de vista de la prevención en cuanto a la seguridad, se puede pensar que sería interesante que, ante un ataque de este tipo, Snort, al detectar un intento de redirección de tráfico, fuera capaz de contrarrestar la acción con el simple hecho de enviar un nuevo mensaje ICMP redirect que incluya la información correcta, forzando a la víctima del ataque a reactualizar sus tablas de encaminamiento, lo que detendría la acción del atacante. Por el momento, es algo que Snort no permite, por lo que tenemos que ceñirnos a las reacciones que el IDS ya implementa. Estas posibles acciones que pueden llevarse a cabo mediante la utilización de la opción resp, quedan registradas en la siguiente tabla.

Respuesta	Descripción
rst_snd	Envía un paquete TCP reset al origen
rst_rcv	Envía un paquete TCP reset al destino
rst_all	Envía un paquete TCP reset a origen y destino
icmp_net	Envía un mensaje ICMP de red inalcanzable al origen
icmp_host	Envía un mensaje ICMP de host inalcanzable al origen
icmp_port	Envía un mensaje ICMP de puerto inalcanzable al origen
icmp_all	Envía todos los mensajes ICMP descritos anteriormente al origen

Tabla 7. Argumentos de la opción resp

Concretamente, en las reglas que se programan para la defensa, se utiliza la respuesta icmp_host para la detección del código 3, y la respuesta icmp_net para los códigos 0 y 2. Tal y como se ha dicho anteriormente, el tipo 1, no se tendrá en cuenta a la hora de generar las alarmas. Así pues, el contenido del fichero de reglas es el que se muestra a continuación.

```
#Reglas para la detección de mensajes de redirección ICMP
alert icmp any any -> any any (msg:"Redirección ICMP - icode=0, Destino red"; itype:5; icode:0; resp:icmp_net; sid:20; rev:1;)

alert icmp any any -> any any (msg:"Redirección ICMP - icode=2, Destino servicio y red"; itype:5; icode:2; resp:icmp_net; sid:22; rev:1;)
alert icmp any any -> any any (msg:"Redirección ICMP - icode=3, Destino servicio y host"; itype:5; icode:3; resp:icmp_host; sid:23; rev:1;)
```

Una vez finalizada la inclusion de las reglas en el sistema, podemos proceder a iniciar el IDS de la misma forma que se ha hecho con anterioridad, mediante el comando Snort –A fast –K ascii –C /etc/snort/snort.conf en un terminal de la máquina que aloja el propio IDS.

7.4 Resultados de las pruebas realizadas

Tras la inicialización del IDS, se procede al envío de los mensajes de redirección siguiendo el proceso que se ha descrito en el apartado pertinente. Han sido realizados 60 envíos de mensajes de redirección, concretamente 15 de cada tipo (código) alternos , y todos ellos sin excepción han sido detectados por Snort, generando las alarmas pertinentes y las respuestas configuradas en la base de reglas. A continuación se puede observar un extracto del fichero de alarmas generado por Snort.

```
08/19-11:57:50.030195 [**] [1:473:4] ICMP redirect net [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 192.168.1.1 -> 192.168.1.8

08/19-11:57:50.030195 [**] [1:20:1] Redirección ICMP - icode=0, Destino red [**] [Priority: 0] {ICMP} 192.168.1.1 -> 192.168.1.8

08/19-11:58:05.971027 [**] [1:473:4] ICMP redirect net [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 192.168.1.1 -> 192.168.1.8

08/19-11:58:32.592282 [**] [1:20:1] Redirección ICMP - icode=0, Destino red [**] [Priority: 0] {ICMP} 192.168.1.1 -> 192.168.1.8

08/19-11:58:44.119618 [**] [1:436:6] ICMP Redirect for TOS and Host [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.1 -> 192.168.1.8

08/19-11:58:44.119618 [**] [1:23:1] Redirección ICMP - icode=3, Destino servicio y host [**] [Priority: 0] {ICMP} 192.168.1.1 -> 192.168.1.8
```

Como se puede observar, por cada detección, se han generado dos alertas, siendo, de cada pareja, la primera generada por el propio sistema, y la segunda por las reglas que se han incluido manualmente, lo que da cuenta de que el sistema, con la configuración por defecto, ya está en disposición de alertar ante la detección de los mensajes ICMP de tipo redirección, aunque con nuestras propias reglas, tenemos un mayor control del sistema.

También podemos observar cómo, efectivamente, pese a haber generado mensajes de redirección de tipo 1 (redirección para host), Snort no ha generado ninguna alerta en función a las reglas incluidas, aunque sí una más de las generadas por el propio sistema, por lo que podemos determinar que el IDS realmente es capaz de diferenciar los mensajes de redirección incluso por su código. Algo que también llama la atención si nos centramos en las direcciones de origen y destino en las alertas generadas, es que, concretamente, en la dirección de origen, se hace referencia a la dirección del punto de acceso, algo que no es así, ya que realmente se han generado estos mensajes por parte del atacante situado en la dirección 192.168.1.5, lo que ocurre, es que de forma malintencionada, al generarlos, en el campo de dirección de origen, se ha suplantado la dirección de dicho punto de acceso.

Por otro lado, y para probar el correcto funcionamiento de las medidas de respuesta de las que se dotó al sistema, se tiene la siguiente captura de tráfico en whireshark.

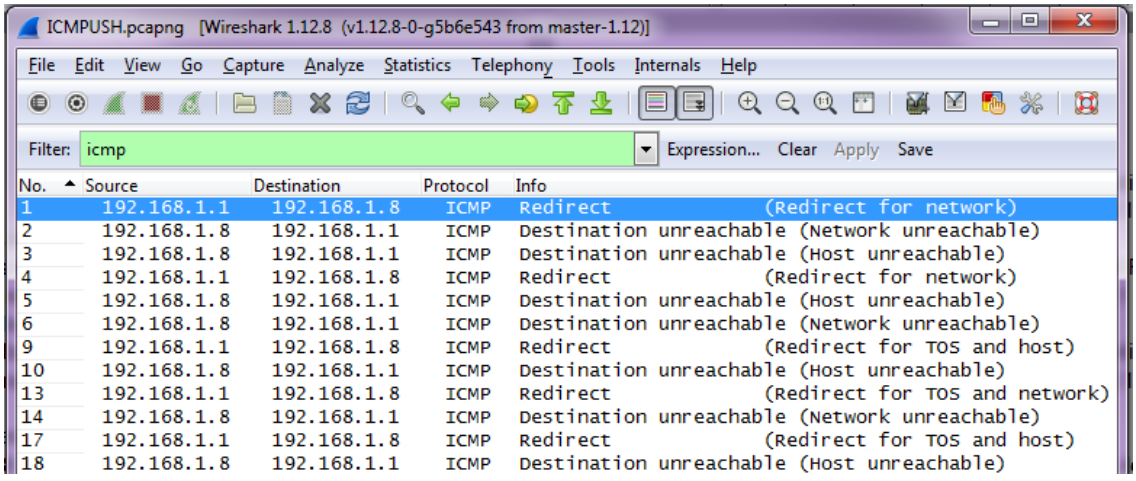

Figura 18. Captura de wireshark escenario 3

Tal y como se muestra en la imagen, podemos comprobar que ante cada envío de mensaje de redirección con origen 192.168.1.1 hacia el destino 192.168.1.8, se han ido generando los paquetes de respuesta que se habían configurado en las reglas en sentido contrario. Por otra parte, aunque sabemos que estos mensajes han sido generados por el atacante, que, recordemos, tiene la dirección 192.168.1.5, sin embargo, en la cabecera

de los paquetes recibidos, el origen del mismo es el 192.168.1.1, por lo que el spoffing de dirección IP por parte de la máquina atacante, se está realizando de forma correcta.

También, si nos centramos en uno de los mensajes de redirección en concreto, podemos observar que la información que se quiere transmitir hacia 192.168.1.8 es que debe redireccionar los envíos de información hacia la 192.168.1.5, tal y como se observa a continuación.

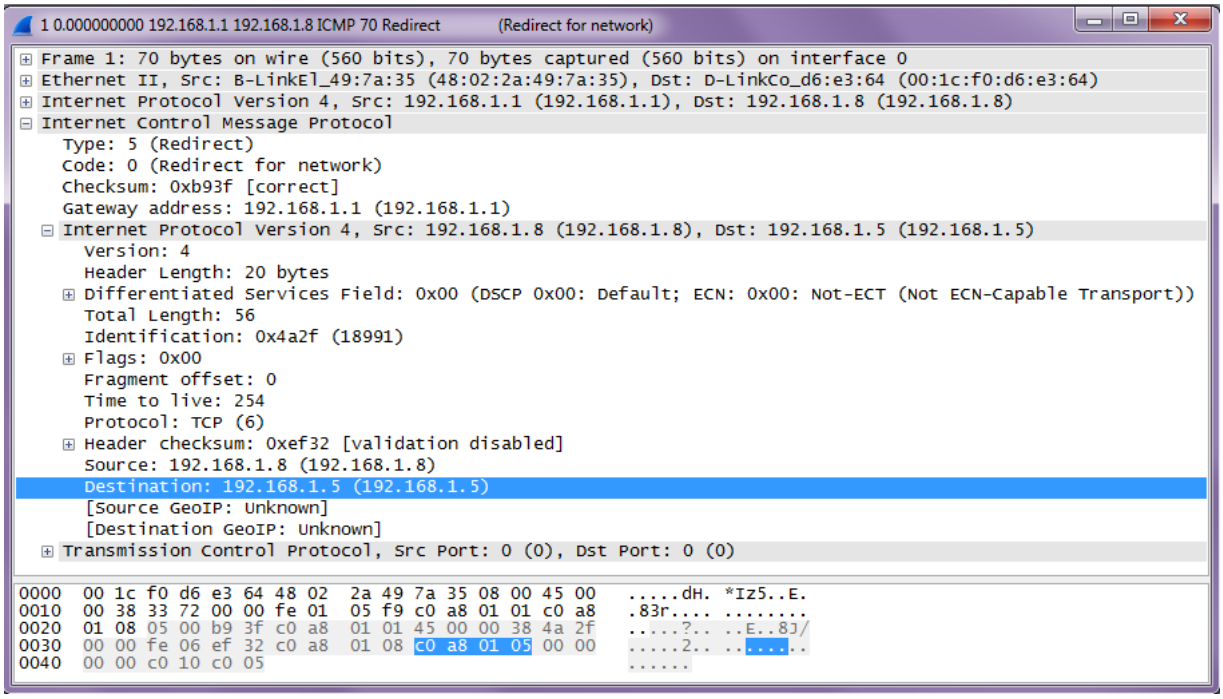

Figura 19. Trama obtenida en escenario 3

8. ESCENARIO 4: ESCANEO DE PUERTOS TCP

8.1 Introducción

El siguiente supuesto, trata de abarcar un tipo actuación que no es considerada en sí un ataque, aunque se utiliza para la recopilación de información, la cual se puede utilizar a posteriori para realizar un ataque en toda regla. Así pues, el escaneo de puertos propiamente dicho, estando situado en el nivel de transporte, de lo que trata es de intentar detectar en una máquina, qué puertos están siendo utilizados o abiertos, incluso tratar de averiguar qué servicios son los que están haciendo uso de los mismos, con tal de buscar una vía de acceso a la máquina por parte del intruso. Una vez realizado el escaneo y obtenida la información, ésta puede dar una idea de la mejor forma de atacar una máquina, lo que hará que el atacante se decante por un cierto tipo de ataque u otro, intentando aprovechar las debilidades que el escaneo de puertos haya sacado a la luz.

Para ser consciente de la diferencia entre los modos de escaneo de puertos del protocolo TCP (Transmission Control Protocol) [4] que se llevan a cabo en este escenario, hay que tener presente el esquema de establecimiento de una nueva conexión TCP, mediante la negociación a tres vías, ya que, dependiendo del escaneo que se realice, se avanzará en mayor o menor medida dentro de esta negociación, la cual se muestra a continuación.

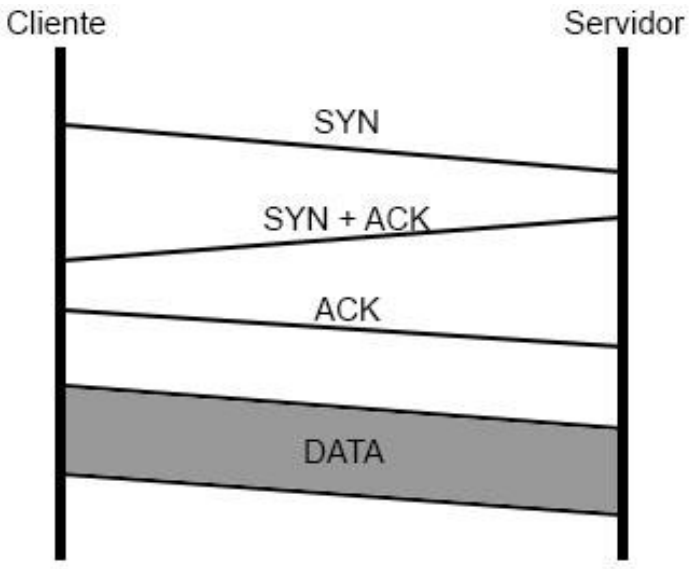

Figura 20. Establecimiento de conexión TCP

Tal y como se observa, el cliente solicitaría una nueva conexión a un puerto TCP al servidor mediante un mensaje SYN, el cual, en caso de que el puerto solicitado estuviera habilitado para su uso, respondería mediante un SYN+ACK para proseguir con la nueva conexión y aceptándola, llegado este momento, el cliente remitiría también un ACK hacia el servidor, haciéndole saber que se ha dado por enterado de la nueva conexión que él mismo solicitó, y se iniciará la transferencia de datos. Dependiendo del tipo de escaneo de puertos que se lleve a cabo, este esquema de negociación a tres vías se realizará completamente para cada puerto, o, como se verá más adelante, se romperá en un determinado momento.

En este escenario, el atacante realiza una serie de escaneos dirigidos a una máquina de la red, estos escaneos serán de diferentes tipos, los cuales se explicarán con más detalle más adelante. Por otra parte, la topología de red de la que se dispone es la misma que la del escenario anterior, tal y como se muestra en la figura.

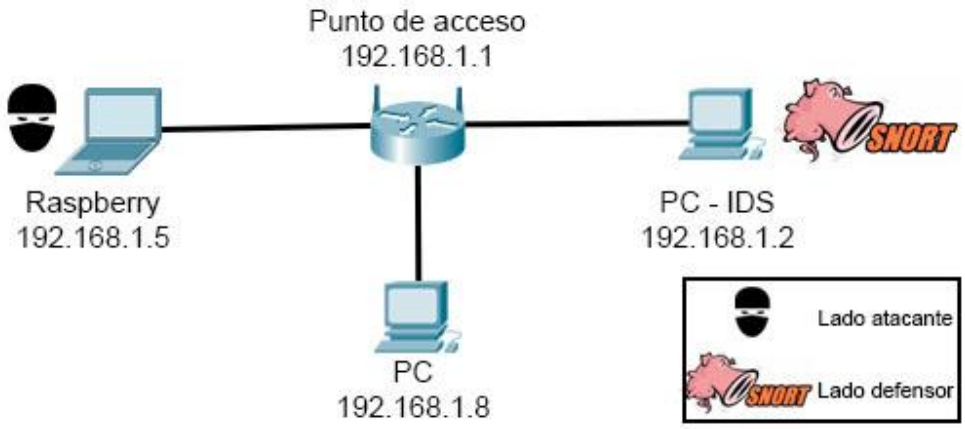

Figura 21. Topología escenario 4

En general, el encargado de garantizar la conectividad entre todos los elementos de la red será el punto de acceso WiFi con dirección 192.168.1.1, mientras que por una parte, y con dirección 192.168.1.2, tendremos el terminal que aloja el IDS Snort. Por otra parte, se dispone el equipo que hará las veces de atacante, en este caso, el encargado de realizar los escaneos de puertos. Tal y como se muestra en la imagen anterior, se utiliza un SBC con dirección IP 192.168.1.5, y por último, dispondremos en la red una tercera máquina, la cual será el objeto de los escaneos. Esta máquina, con dirección 192.168.1.8, únicamente será la receptora de las actuaciones del atacante.

8.2 Ataque al sistema

Para comprometer la seguridad del sistema, en este caso, no se necesita de ninguna aplicación externa a instalar, ya que la propia distribución Kali Linux de la que disponemos en la Raspberry, incorpora la herramienta necesaria para el escaneo de puertos, debido a que es una actividad muy usual en cuanto a seguridad se refiere. La herramienta de la que disponemos se denomina nmap [10], una aplicación en línea de comandos, la cual nos permite realizar diferentes tipos de escaneos de puertos. Concretamente, para probar la eficacia del IDS, se realizarán tres tipos de escaneos diferentes utilizando la herramienta nmap. Estos escaneos serán, por un lado, escaneos simples rápidos, escaneos sigilosos y escaneos con fragmentación.

- Escaneo simple rápido

Este tipo de escaneo de puertos es el que nmap ejecuta normalmente, por cada puerto se realiza una conexión TCP completa, por lo que es probable que el servidor almacene información respecto a esa conexión. Con el fin de intentar evadir la detección, nmap nos da la opción de espaciar el escaneo de cada puerto con el argumento –T, este argumento puede ir acompañado de un número, el cual dará cuenta del tiempo entre los escaneos, consta de 6 niveles, siendo 0 el mayor latencia. Cuanto más espaciados estén los escaneos, más fácil será evadir la detección, pero por otro lado, más tiempo tardaremos en obtener la información. En concreto, en las pruebas llevadas a cabo, se ha utilizado el argumento –T3, una latencia considerable, pero que no llega a hacer la espera de obtención de la información demasiado prolongada. Así pues, el comando a utilizar en este tipo de escaneo será:

```
nmap –T3 [IP Objetivo]
```

Y el ejemplo práctico llevado a cabo, también haciendo uso de la opción –A para intentar obtener mayor cantidad de información, como el sistema operativo, la versión, etc:

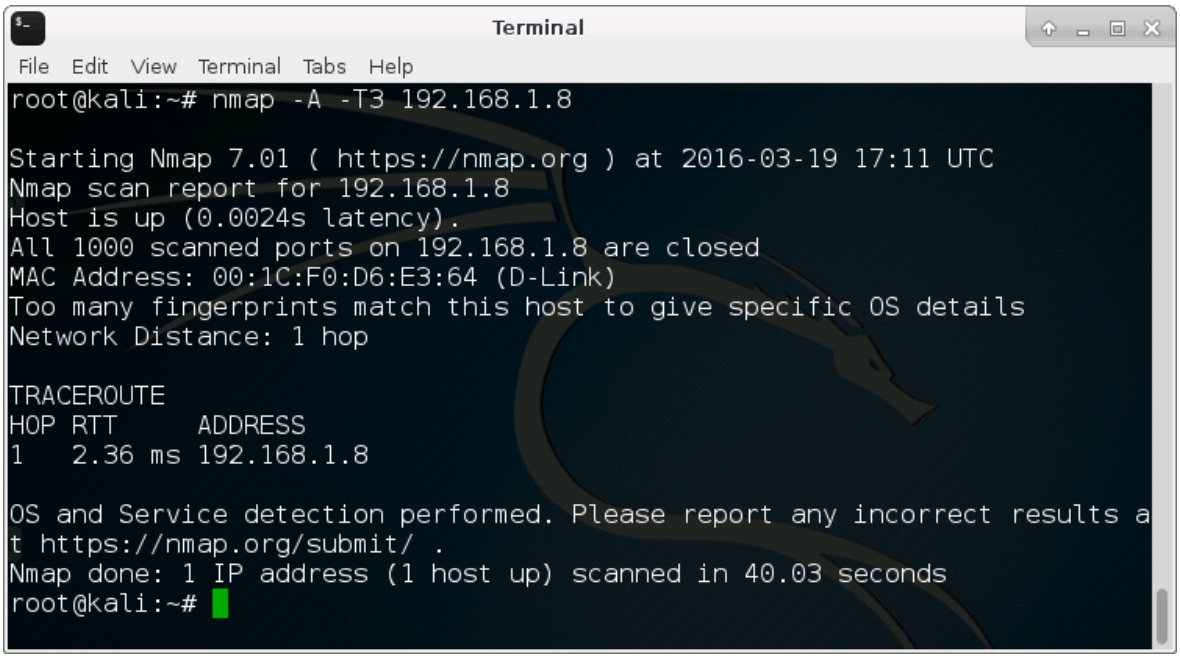

Figura 22. Escaneo de puertos T3

- Escaneo sigiloso

En este tipo de escaneo, lo que trata es de pasar lo más desapercibido posible. Al realizar este sondeo, lo que se realiza es el envío de un paquete SYN al puerto a comprobar, con lo que, si en ese puerto existe algún servicio activo, el sistema proseguirá con el envío hacia atrás de otro paquete SYN-ACK. En este momento, la máquina atacante debería proseguir con el envío de un ACK, pero no lo realiza, lo que supone romper con la secuencia de conexión, y que la misma no quede registrada en los logs. De esta forma, el sondeo puede pasar desapercibido con mayor facilidad, aunque también se obtendrá una menor cantidad de información. Para la ejecución de nmap haciendo uso de este tipo de escaneo, tan solo debemos usar la opción –PS, por lo que el comando a introducir desde el terminal será:

```
nmap –PS [IP Objetivo]
```

Lo que se traduce en la práctica dentro del escenario propuesto como:

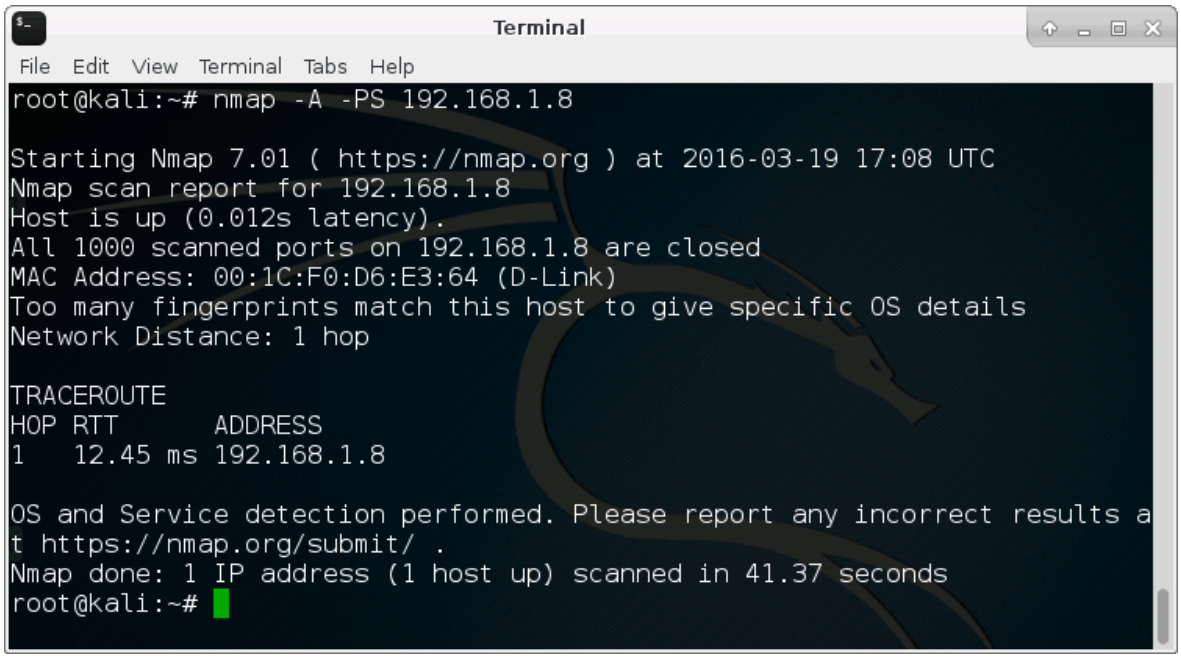

Figura 23. Escaneo de puertos sigiloso

- Escaneo con fragmentación

Este último tipo de sondeo que se llevará a cabo está especialmente indicado para la evasión de las detecciones de escaneos por parte de firewalls e IDS. El funcionamiento es el mismo que el de un escaneo normal, propiamente dicho, pero de lo que se trata es en trocear la información que se envía en fragmentos, de forma que se dificulta enormemente la tarea de la detección, ya que antes del análisis de los paquetes recibidos, éstos tienen que reensamblarse, y debido a la gran cantidad de paquetes que pueden ser enviados, se puede sobrepasar el tamaño del buffer, haciendo que algunos paquetes se pierdan, no puedan reensamblarse, y por ende, no puedan ser analizados, por lo que el escaneo pasaría desapercibido.

Para que nmap fragmente la información, podemos hacer uso de la opción –f, aunque para un mayor control, la opción --mtu nos permitirá, además de la fragmentación, establecer el tamaño en bytes que queremos que tengan los fragmentos, siempre con un mínimo de 8, y múltiplo de este valor. En nuestro caso, para dificultar al máximo la tarea de Snort e intentar sobrecargar al sistema, el tamaño utilizado para la fragmentación será el mínimo posible, es decir, 8 bytes. La estructura general del comando a utilizar para el escaneo sería:

nmap --mtu [Tamaño (Bytes)] [IP Objetivo]

Y adaptándolo al ataque que se utiliza descrito anteriormente:

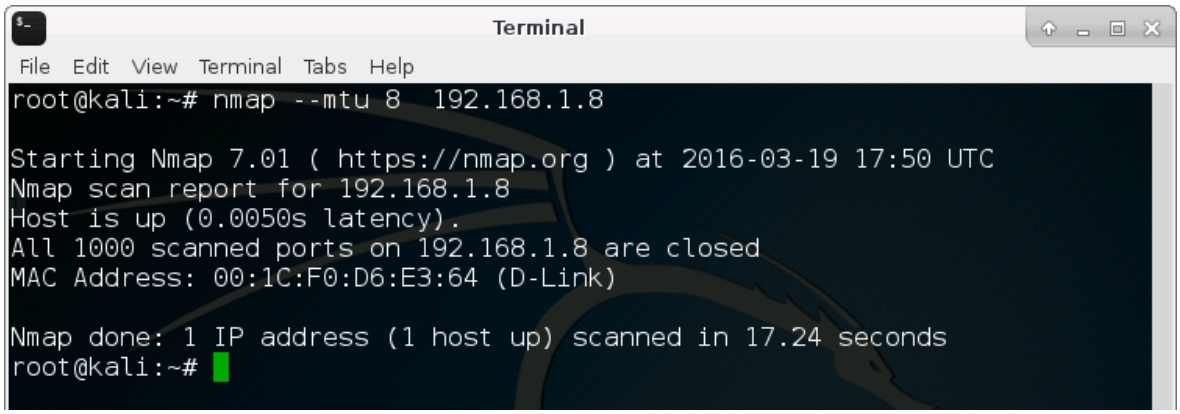

Figura 24. Escaneo de puertos con fragmentación

8.3 Defensa del sistema

Para el supuesto que nos concierne, tal y como sucede para ARP Spoofing, Snort dispone de un preprocesador dedicado a la detección de los escaneos de puertos, llamado sfPortscan, por lo que el uso de este preprocesador es la opción utilizada en cuanto a establecer la seguridad del sistema ante los sondeos. Para la activación de este preprocesador, bastará instanciarlo en el fichero de reglas, configurando sus opciones atendiendo a las necesidades del sistema. A continuación se enumeran las opciones que tendremos en cuenta a la hora de configurar este preprocesador.

- proto [TCP|UDP|ICMP|ip_proto|all]

  Hace referencia al protocolo concreto que Snort debe supervisar.

- logfile [ruta]

  Este preprocesador, además de generar alarmas en el fichero que Snort dispone a tal uso, nos brinda la opción de aportar una ruta, donde se generará un fichero de log, que incluirá una mayor información acerca del ataque detectado, al margen de la propia alarma generada.

- scan_type [portscan|portsweep|decoy_portscan|distributed_portscan|all]

  Esta opción da cuenta del tipo de escaneo del que queremos alertar en el sistema. En nuestro caso, y debido a los diferentes sondeos que se van a realizar, utilizaremos el argumento all, con el fin de dotar de mayor seguridad.

- memcap [int]

  Establece el tamaño en bytes del buffer de memoria del que dispondrá el preprocesador para llevar a cabo sus tareas, por ejemplo, para hacer frente a ataques de fragmentación.

- sense_level [low|médium|high]

  Configuramos el nivel de sensibilidad que se requiere que tenga el preprocesador, si utilizamos el argumento high, dispondremos de una mayor facilidad de detección, lo que también provocará la aparición falsos positivos.

Así pues, para nuestro cometido, estableceremos que el IDS inspeccione todos los protocolos, de tal forma que englobará la detección de escaneo de puertos tanto TCP como UDP, por otra parte, se creará un fichero de log en la ruta /alertas_escaneos.txt. Tal y como se ha comentado con anterioridad, se tratará de tener cubiertos todos los tipos de escaneos a los que el preprocesador puede hacer frente, con un nivel de sensibilidad alto, y una memoria del buffer considerada aceptable. Por lo que el fichero de reglas, para la activación del preprocesador, contendrá las líneas que se muestran a continuación.

```
Preprocessor sfportscan: proto {all} logfile {/alertas_escaneos.txt} scan_type {all} memcap {100000} sense_level {high}
```

Una vez añadida la línea anterior al fichero de reglas y guardado el mismo, ya se puede proceder al arranque de la aplicación IDS y por consiguiente, a las pruebas de escaneo propuestas.

8.4 Resultados de las pruebas realizadas

Tras la realización de 60 escaneos de cada tipo, ante la detección de los sondeos, Snort muestra el mismo tipo de información, independientemente del tipo de sondeo de puertos que se haya realizado, sin que podamos discernir entre un tipo de escaneo u otro. Un ejemplo de esta información que arroja el IDS en el fichero de log, por cada una de las detecciones realizadas, es la que se muestra seguidamente, donde se puede

observar, entre otras, la fecha y hora de la ocurrencia, las direcciones de origen y destino, el número de conexiones realizadas y los puertos en los que se ha detectado el escaneo.

```
Time: 07/16-11:04:41.248937
event_ref: 0
192.168.1.5 -> 192.168.1.8 (portscan) TCP Portscan
Priority Count: 9
Connection Count: 10
IP Count: 1
Scanner IP Range: 192.168.1.5:192.168.1.5
Port/Proto Count: 10
Port/Proto Range: 22:8888
```

Por otra parte, en el fichero de alertas de Snort, también se nos da el aviso, aunque este no incluye la misma cantidad de información que en el archivo de log, únicamente trata de dar cuenta de lo ocurrido. Algunas de las alarmas generadas por los sondeos realizados, pueden verse en las líneas siguientes.

```
07/16-11:04:44.597876 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.5:42447 -> 192.168.1.8:1

07/16-11:04:46.037941 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.5:42447 -> 192.168.1.8:1

07/16-11:12:14.711182 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.5:60521 -> 192.168.1.8:1

07/16-11:12:16.234943 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.5:60521 -> 192.168.1.8:1
```

Cabe destacar, que en el caso de los escaneos con fragmentación, Snort también genera alarmas en el caso de que no le haya sido posible reensamblar los fragmentos,

por lo que da cuenta de que existe información no analizada. Así pues, cuando esto suceda, en el fichero de alarmas, nos encontraremos con alarmas del siguiente tipo.

```
07/16-11:58:47.691330  [**] [1:410:5] ICMP Fragment Reassembly Time Exceeded [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.5

07/16-11:58:47.691352  [**] [1:410:5] ICMP Fragment Reassembly Time Exceeded [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.5

07/16-11:58:47.691357  [**] [1:410:5] ICMP Fragment Reassembly Time Exceeded [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.5

07/16-11:58:47.691361  [**] [1:410:5] ICMP Fragment Reassembly Time Exceeded [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.5

07/16-11:58:47.691367  [**] [1:410:5] ICMP Fragment Reassembly Time Exceeded [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.5
```

Finalmente, los resultados obtenidos en cuanto a los diferentes tipos de escaneos de puertos probados y sus detecciones, se recogen en la siguiente tabla de resultados.

Escaneo	Repeticiones	Detecciones	Efectividad
Simple rápido	60	60	100%
Sigiloso	60	60	100%
Fragmentación	60	46	76.67%

Tabla 7. Resultados de detección 1

Como puede observarse, la detección de los escaneos es total en los de tipo simple y sigiloso, pero Snort no tiene la misma fiabilidad en cuanto a la detección de escaneos con fragmentación. Cabe decir que para intentar aumentar el porcentaje de efectividad, podría aumentarse el buffer de memoria utilizado por el preprocesador en caso de que la memoria fuera insuficiente, aunque también, podrían usarse otros preprocesadores

conjuntamente con sfportscan, encargados del reensamblaje previo de los paquetes recibidos, tales como el Stream5, centrado en TCP, o el preprocesador Frag3, más enfocado a IP.

Con el fin de aumentar la eficacia de detección en el caso del escaneo con fragmentación, se ha procedido a la configuración del preprocesador Stream5 (centrado en el reensamblaje TCP), haciendo que trabaje conjuntamente con el preprocesador sfPortscan, de forma que a medida que lleguen los fragmentos, Stream5 se encargará de ensamblarlos, lo que hará que a sfPortscan le lleguen los mensajes TCP completos, por lo que no tendrá que utilizar parte de su capacidad de procesamiento y tiempo en esta tarea, dedicándose enteramente a la detección de escaneo de puertos.

La configuración de Stream5 se realiza como la de cualquier otro preprocesador. En el archivo de reglas, se debe hacer alusión al preprocesador, conjuntamente con las opciones que se requieran y los valores asignados a tales opciones. Las opciones de este preprocesador, pueden dividirse en opciones generales (para todos los protocolos), y una serie de opciones centradas en el propio protocolo, ya sea TCP, UDP o ICMP.

- Track_tcp [yes|no]
  Habilita el seguimiento de las sesiones TCP.
- Max_tcp [int]
  Establece el número máximo de sesiones TCP a supervisar.
- Memcap [int]
  Cantidad de memoria (bytes) de almacenamiento para TCP.
- Track_udp [yes|no]
  Habilita el seguimiento de UDP.
- Max_ud [int]
  Establece el número máximo de conexiones UDP a supervisar.
- Track_icmp [yes|no]
  Habilita el seguimiento de ICMP.
- Max_icmp [int]
  Establece el número máximo de flujos ICMP a supervisar.

Así pues, en el archivo de reglas, para configurar este preprocesador, se añadirá la línea siguiente (además de la existente para el procesador sfPortScan).

```
Preprocessor stream5_global: track_tcp yes, max_tcp 5000, memcap
8388, track_udp no, track_icmp no;
```

Tras la activación del preprocesador Stream5 tal y como se ha mencionado, se ha procedido de nuevo a las pruebas realizadas con anterioridad, reiniciando el IDS Snort y volviendo a realizar otra serie de escaneos de fragmentación con tal de evaluar el cambio de comportamiento en cuanto a detecciones del propio Software. El resultado arrojado después de estas nuevas pruebas, es que se ha conseguido alcanzar el 100% de detecciones de escaneos con fragmentación, fruto de combinar ambos preprocesadores. Así pues, la nueva tabla de resultados obtenidos tras la utilización de ambos preprocesadores es la que se dispone a continuación.

Escaneo	Repeticiones	Detecciones	Efectividad
Simple rápido	60	60	100%
Sigiloso	60	60	100%
Fragmentación	60	60	100%

Tabla 8. Resultados de detección 2

9. ESCENARIO 5: FUERZA BRUTA A FTP

9.1 Introducción

Como último escenario de pruebas, se ha optado por la experimentación sobre el nivel de aplicación, teniendo en cuenta alguno de los protocolos existentes, así pues, el protocolo elegido para las pruebas es FTP (File Transfer Protocol) [1], el cual, como su nombre indica, realiza las operaciones necesarias para la transferencia de archivos entre sistemas. Este protocolo se apoya en TCP [4] como protocolo de transporte, estableciendo una conexión en el puerto 21 de TCP para el control de la conexión, y otra en el puerto 20 para la transferencia de archivos propiamente dicha. Estos puertos son los que se tendrán en cuenta en la fase de defensa del sistema.

Así pues, un cliente que desee establecer una conexión mediante FTP con un servidor, deberá realizar una conexión TCP al puerto 21, en el que el servidor responderá, solicitando al cliente las credenciales de acceso al sistema de archivos. Tras este proceso de autenticación, cliente y servidor negociarán las condiciones de la conexión que se iniciará normalmente sobre el puerto 20 del servidor. Hay que tener en cuenta de que en caso de un intento de autenticación fallido, el servidor responderá con un mensaje de tipo 530 (autenticación fallida), indicando el código 5xx que no se recibe respuesta por parte del servidor, y x3x que el problema acontecido tiene relación con la autenticación.

En estas pruebas, tratará de ponerse a prueba a Snort ante un ataque de tipo fuerza bruta a un servidor FTP, tratando de realizar un acceso ilícito al mismo. Este tipo de ataques son muy rudimentarios a la par que eficaces, ya que tan solo requieren de tiempo para lograr su cometido, siempre y cuando no se tomen medidas para contrarrestar la acción atacante. Así pues, el ataque en sí, no es otro que intentar autenticarse ante el servidor probando una y otra vez, variando las credenciales de acceso para testear todas las combinaciones posibles hasta que se produzca el acceso. La topología de red que se dispondrá para la realización de las pruebas en este escenario queda reflejada en la siguiente figura.

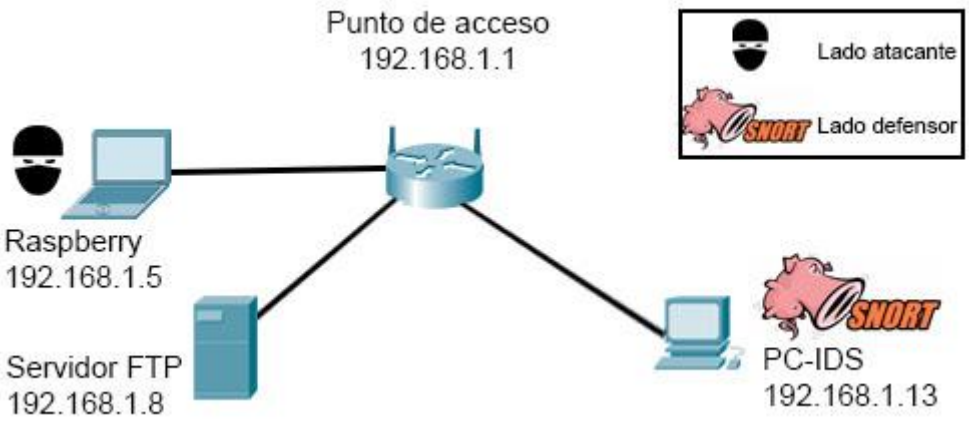

Figura 25. Topología escenario 5

Como puede observarse, por una parte, tenemos el servidor FTP, el cual será objeto del ataque por fuerza bruta. Por otra parte, se dispone la Raspberry en la que se situará el atacante, el cual mediante un software destinado a tal fin, se encargará de realizar intentos de autenticación frente al servidor FTP mediante fuerza bruta. Finalmente, en la máquina restante, se alojará el IDS, el cual se encargará de analizar todo el tráfico que discurra por la red, generando las alarmas necesarias apoyándose en las reglas que se definirán más adelante.

9.2 Ataque al sistema

Tal y como se ha mencionado con anterioridad, en la máquina atacante se dispone de un software que permite el intento de autenticación masivo frente al servidor FTP. Este programa que se utiliza se denomina Hydra [11], el cual ya está incluido en la suite de auditoría de seguridad Kali Linux de la que ya se dispone en la Raspberry. Hydra es una aplicación de línea de comandos que permite realizar ataques por fuerza bruta a diferentes protocolos, tales como FTP, SSH, TELNET o IMAP entre otros.

En este caso, las opciones utilizadas en la ejecución de la aplicación serán, primeramente, la opción –l, la cual permite definir un usuario o fichero de nombres de usuario con los que se iniciará sesión, y la opción –x, mediante la cual hydra irá generando las diferentes contraseñas que testeará en cada intento de autenticación. A esta opción, se le debe proporcionar una longitud mínima de clave, así como una máxima, además de un patrón de clave, el cual en nuestro caso, será “a1”, con el fin de que en la generación de las contraseñas, éstas se generen teniendo en cuenta caracteres alfanuméricos. Así pues, para la ejecución del ataque, el comando genérico a usar será el siguiente.

```
hydra –l [usuario|fichero] –x [min:max:patrón] [víctima]
```

Y, concretamente, el ataque que se ejecutará desde el terminal de Raspberry:

```
hydra –l admin –x 1:3:a1 ftp://192.168.1.8
```

Desarrollando este comando, lo que el software realizará será un ataque por fuerza bruta a la dirección 192.168.1.13 al puerto TCP 21 (FTP de control), intentando autenticarse con el nombre de usuario admin, y probando todas las combinaciones posibles de claves alfanuméricas con longitud de uno a tres caracteres.

9.3 Defensa del sistema

Para proceder a la configuración del sistema de defensa, debemos tener en cuenta que, en este caso, no siempre que un usuario no logre autenticarse se tratará de un posible ataque al servidor, ya que el mismo, puede simplemente haber olvidado sus credenciales de acceso, o haber producido algún error en la escritura de las mismas. Es por ello, que hay que ser capaz de discernir entre una serie de intentos fallidos de autenticación que pueden producirse de forma aislada, de una consecución de los mismos con el fin de acceder al sistema de forma ilícita, lo que sería el ataque en sí.

En este caso, en lo que se va a centrar la configuración de la defensa del sistema, será en el flujo de información saliente del servidor FTP, ya que, debido a que se trata de un protocolo conocido y bien definido, sabemos de antemano que, en caso de producirse un intento de autenticación fallido, tal y como se ha dicho anteriormente, el servidor contestará al cliente mediante un mensaje de código 530; es por ello, que para detectar estas autenticaciones fallidas, tan solo debemos hacer que Snort supervise el payload de los paquetes, en busca de este código. Además, como ya se ha visto en otros casos, la regla a utilizar se podrá centrar en un tipo de protocolo y puerto concreto, algo que puede hacerse sin problemas, ya que sabemos que, por parte del servidor, el protocolo de transporte utilizado será TCP en su puerto 21, ya que este es el canal de control.

El principal problema de esta detección viene dado por la necesidad de diferenciar un ataque por fuerza bruta, de un error aislado de autenticación, con la principal diferencia entre ellos de que el ataque realizará una gran cantidad de intentos en un espacio de tiempo muy reducido, mientras que en el caso de los errores, se producirán

en menor medida, y mucho más espaciados en el tiempo. Para ayudar a diferenciar estos dos casos, en la inclusión de la regla, además de las ya conocidas, disponemos de la opción threshold, la cual nos permite cierta flexibilidad a la hora de generar la alarma, estableciendo límites temporales de detección y recuento de detecciones entre otras opciones o argumentos que se utilizarán en la regla, y que se definen a continuación.

- Type [limit | threshold | both]

  Establece el momento en el que se generará la alerta. En el caso de limit, ésta se generará la primera vez que surja la detección dentro del tiempo establecido. Para threshold, se generará si se sobrepasa el número de detecciones establecidas dentro del tiempo de supervisión. Y both es una combinación de las dos anteriores, generando la alarma en cualquiera de los dos casos.

- Track [by_dst | by_src]

  Proporciona el tipo de seguimiento que quiere hacerse del tráfico, esto es, si se pretende que se realice una supervisión en función de la dirección de destino (by_dest) o, por el contrario, en función a la dirección de origen (by_src).

- Count [int]

  Define el número de detecciones que deben producirse antes de que se genere la alarma.
- Seconds [int]

  Define el tiempo en segundos que se mantiene el seguimiento desde la primera detección.

Así pues, mediante la combinación de estos argumentos, se puede hacer que la propia regla solo genere una alarma si se produce una cantidad de detecciones controladas, dentro de un tiempo establecido, además, pudiendo controlar si estas detecciones se hacen en función a una dirección de destino concreta, que en el caso de tratarse de un ataque por fuerza bruta, será así.

Teniendo en cuenta todo lo visto anteriormente, y haciendo uso en este caso además de la opción threshold que se ha comentado, la única regla que se ha proporcionado al sistema para poder detectar este tipo de ataque, es la que se muestra a continuación.

```
alert tcp 192.168.1.8 21 -> any any (msg:”Ataque FTP fuerza bruta”; content:”530”; classtype: unsuccessful-user; threshold: type threshold, track by_dst, count 5, seconds 30; sid:30; rev: 1;)
```

De esta regla se obtiene que, además de las opciones como sid, rev y classtype, que ya se han comentado en otros escenarios y que tienen como función establecer un identificador de regla, un número de revisión y un tipo de ataque tipificado por Snort, que el IDS controlará el flujo saliente desde el servidor FTP hacia cualquier otra máquina en cualquier otro puerto, inspeccionando el puerto 21 de TCP (conexión de control de FTP), en búsqueda de la cadena 530 dentro del paquete, lo que se correspondería con el mensaje de tipo autenticación fallida proporcionado por el servidor. Por otra parte, y teniendo en cuenta los argumentos proporcionados a la opción threshold, se realizará un seguimiento de estas detecciones en función de la dirección de destino, y sólo se generará una alerta por parte de Snort en el caso de que se produzcan 5 intentos de autenticación fallidos en un periodo de 30 segundos, algo que sería poco factible si se tratara de una persona física intentando autenticarse de forma errónea, pero que si se trata de un software realizando un ataque por fuerza bruta, es muy probable que se sobrepasen estos intentos.

Por otro lado, hay que tener en cuenta que el seguimiento se realiza en función de la dirección de destino, ya que si se hiciera teniendo en cuenta la dirección de origen (servidor FTP), podrían generarse falsos positivos, teniendo en cuenta que podría darse el caso de que una gran cantidad de usuarios intentara establecer conexión con el servidor, pudiendo muchos de ellos hacerlo de forma errónea.

9.4 Resultados de las pruebas realizadas

Tras activar el sistema de defensa, y dejar a la Raspberry un tiempo prudencial ejecutando ataques al servidor FTP, cabe decir que Snort se ha comportado de la forma esperada, generando las alertas en relación a la configuración de reglas realizada, las cuales pueden observarse en el propio fichero de alertas, y del cual se extraen las líneas que se muestran a continuación.

```
09/22-14:09:54.442944  [**] [1:50:1] Ataque FTP fuerza bruta [**] [Classification: Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.1.8:21 -> 192.168.1.5:57829
```

```
09/22-14:09:57.750380  [**] [1:50:1] Ataque FTP fuerza bruta [**]
[Classification: Unsuccessful User Privilege Gain] [Priority: 1] {TCP}
192.168.1.8:21 -> 192.168.1.5:57831

09/22-14:09:58.879574  [**] [1:50:1] Ataque FTP fuerza bruta [**]
[Classification: Unsuccessful User Privilege Gain] [Priority: 1] {TCP}
192.168.1.8:21 -> 192.168.1.5:57824
```

Tal y como puede observarse, queda patente que las detecciones se están realizando de forma correcta, además de poder comprobar el sentido del flujo de información que ha provocado la alerta, donde incluso podemos observar a qué puerto de la máquina atacante se están realizando las conexiones. Para tener un mayor detalle del tráfico que se ha producido en la red, se dispone de la siguiente captura de tráfico en wireshark, la cual ha sido filtrada mostrando únicamente las tramas relacionadas con un único ataque al servidor FTP, ya que de mostrar la captura completa, debido a la ingente cantidad de tráfico que se produce por cada intento de autenticación (establecimiento de conexión TCP por cada puerto, ACKs correspondientes, envío de cada una de las claves, etc.) sería contraproducente, pretendiendo simplificar el ejemplo.

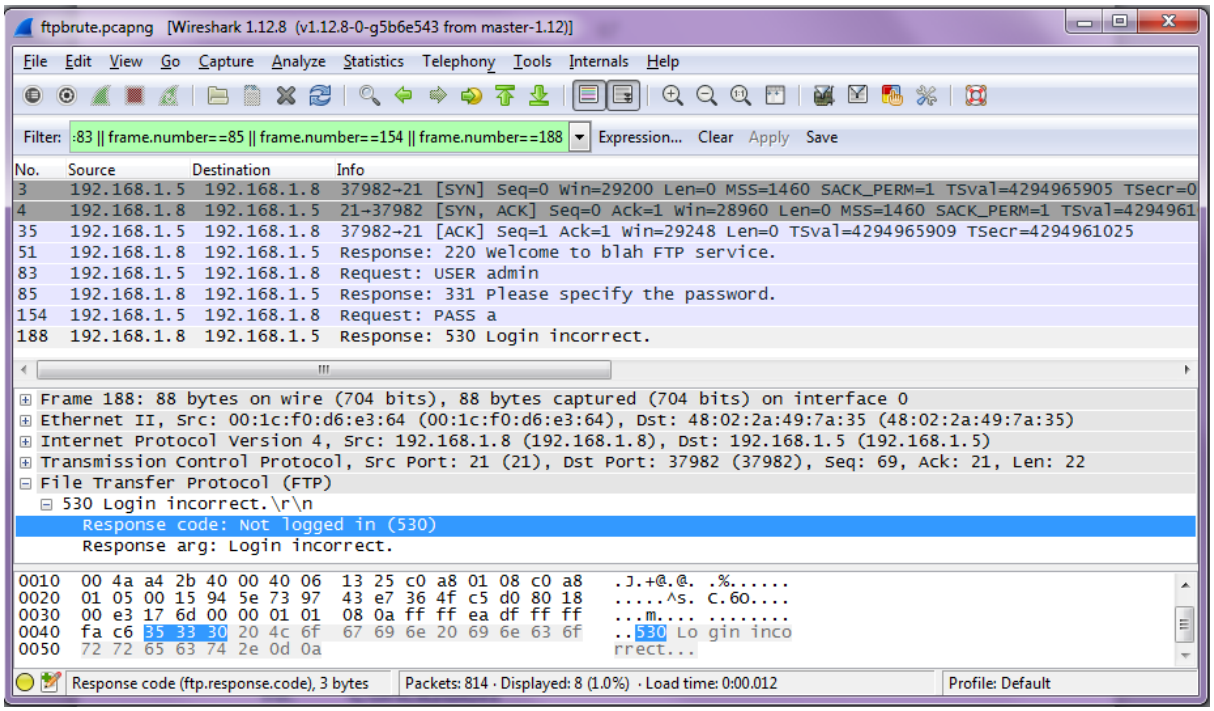

Figura 26. Captura de wireshark escenario 5

Por una parte, las tres primeras tramas que se observan (3, 4 y 35), hacen referencia al establecimiento de conexión TCP, desde el atacante en el puerto 37982 hacia el puerto 21 del servidor. Concretamente estas tres tramas dan cuenta de la negociación a tres vías de TCP, siendo la trama 3 la solicitud de nueva conexión al servidor mediante un mensaje de tipo SYN, la trama 4 la respuesta del servidor con otro mensaje SYN y el ACK correspondiente a la trama 3, y por último, la trama 35, donde el servidor envía el ACK, confirmando la recepción del mensaje anterior.

Una vez establecida la conexión TCP, el servidor FTP envía el mensaje de bienvenida al cliente, tal y como se observa en la trama 51. Es aquí donde comienza el proceso de autenticación por parte del cliente, en la trama 83, donde el cliente intenta identificarse con nombre de usuario admin, ante el cual, el servidor responde solicitando la contraseña para tal usuario, como puede verse en la trama 85. Posteriormente, el atacante prueba con la primera contraseña generada por el patrón establecido, concretamente sólo con la letra “a” como contraseña, lo que se puede ver en la trama 154. Debido a que esta contraseña es errónea, el proceso de autenticación no se verá completado, por lo que el servidor responde con el mensaje 530 de autenticación incorrecta, el cual es el que sirve a Snort para determinar un posible ataque por fuerza bruta, siempre que el servidor se vea obligado a enviar una consecución de mensajes 530 dentro de los periodos y cantidades establecidas en la regla.

En el caso de querer dotar al sistema de medidas preventivas, no solo de detección, una buena práctica sería añadir la opción resp a la regla configurada, con un argumento rst_dst como se vio en el apartado 7.3 de este trabajo, con el fin de, además de generar la propia alarma, hacer que Snort envíe un mensaje TCP de reseteo de conexión al origen.

10. CONCLUSIÓN

Viendo los resultados obtenidos tras todas las pruebas realizadas, se puede concluir que el IDS Snort es una herramienta realmente útil y efectiva en la gran mayoría de situaciones analizadas, alcanzando una detección total salvo en el caso del ARP Spoofing, donde no ha sido capaz de realizar detección alguna, pese a la intensa búsqueda de solución. Por otra parte, hay que tener en cuenta, que el éxito de Snort recae en su correcta configuración, su conocimiento y el de los protocolos que se pretenden supervisar, y en saber adaptarlo a la situación que deba supervisar, tal y como se ha podido ver en el escenario 4, donde no se alcanzaba una detección total en el caso del escaneo de puertos con fragmentación, algo que se ha solventado rápidamente tan solo con el hecho de incluir un nuevo preprocesador en la configuración del sistema, pasando de una detección de en torno al 76% al 100%.

Por otra parte, algo que también hay que tener muy en cuenta a la hora de utilizar este IDS, como en cualquiera, es su posición en la red, ya que ésta está íntimamente ligada a la supervisión que Snort será capaz de realizar, ya que podría darse el caso de que simplemente por la situación elegida de la máquina que almacene el IDS, éste no pueda realizar las tareas de detección. Otro detalle a tener muy en cuenta es la configuración del dispositivo de red, el cual debe ser configurado en modo monitor o promiscuo siempre que se pretendan dar los servicios de seguridad a algo que no sea la propia máquina IDS, salvando el tráfico de broadcast.

Al margen de esto, Snort está dotado de una gran versatilidad, pudiendo actuar como IDS en una red sin necesidad de un terminal con gran capacidad de procesamiento, aunque ello irá íntimamente ligado a la cantidad de tráfico que deba supervisar. Además, está capacitado para inspeccionar cualquier nivel de la torre de protocolos, ya que trabaja como un sniffer, inspeccionando al detalle toda la información recibida, tal y como también puede verse en el escenario 3, donde queda patente la capacidad del software de discernir dentro del mensaje ICMP de tipo redirección, de qué tipo de redirección se trata, inspeccionando campos, posiciones o incluso bytes concretos.

Como punto negativo a destacar, además del mal funcionamiento detectado en el caso del preprocesador arpspoof en el escenario 2, como se ha comentado en el ejemplo de la redirección ICMP, podría dotarse al sistema de la posibilidad de envío de información a medida, más relacionada con la prevención que con la detección, lo que daría la posibilidad de resarcir las problemáticas surgidas de un ataque de redirección

ICMP de forma automática, sin limitarse solamente a finalizar conexiones o el envío de determinados tipos de mensajes ICMP hacia atrás, que a fin de cuentas, no solucionan los problemas generados en las tablas de encaminamiento de las máquinas tras recibir los mensajes de redirección.

Para finalizar, y en líneas generales, cabe decir que este software se comporta de forma más que aceptable, proporcionando soluciones a todos los escenarios elegidos, sin olvidar de que se trata de un IDS de software libre, bastante extendido, el cual dispone de una gran comunidad, con gran cantidad de información relacionada, ya que no se trata de una aplicación relativamente nueva, aunque sí actualizada, y dispone de un largo recorrido temporal en el que los usuarios han ido poniendo de manifiesto las deficiencias encontradas, lo que ha ido dando lugar al buen funcionamiento de Snort.

11. LÍNEAS FUTURAS DE EXPERIMENTACIÓN

Tras el trabajo realizado y durante la elaboración del mismo, se han considerado suficientemente interesantes como para tenerlas en cuenta en experimentaciones futuras las cuestiones que se mencionan a continuación.

Por una parte, dada la lógica de que el coste computacional requerido por la máquina que aloje Snort va estrechamente ligada al tráfico que circule por la red, puede ser interesante el estudio de este nivel de dependencia, teniendo en cuenta unos recursos computacionales concretos y viendo cómo se van viendo afectados a medida de que el tráfico que discurre por la red aumenta, para comprobar también en qué medida repercute en la eficiencia de Snort en cuanto a las detecciones que es capaz de realizar.

Por otra parte, derivado de la cuestión anterior, suponiendo la utilización de Snort en redes de mayor tamaño, y por consecuente, con mayor tráfico, también se puede tomar como una línea de experimentación el hecho de la combinación del uso del IDS con otros elementos de seguridad en las redes de telecomunicación, tales como firewalls, de forma que podría crearse un primer nivel de seguridad mediante el filtrado previo de del tráfico por parte del firewall, antes de que el mismo llegue a Snort, con el fin de descargar de trabajo al IDS, optimizando así la utilización de los recursos computacionales existentes.

También, teniendo en cuenta que este trabajo fin de grado se ha centrado únicamente en un solo software IDS, puede ser de utilidad comprobar el funcionamiento de otros IDS existentes en el mercado, ya sea para generar conocimiento sobre otro software concreto, o con el fin de obtener una comparativa respecto a resultados obtenidos ante los mismos supuestos.

Por último, debido a la gran versatilidad de Snort y a la variedad de situaciones a las que puede someterse, se podría seguir estudiando su uso y comportamiento en multitud de escenarios diferentes a los que ya se han propuesto en este trabajo.

ANEXO I – INSTALACIÓN DE SNORT

A continuación se detalla el proceso de instalación del IDS Snort en la máquina que actuará de sistema de defensa en los escenarios propuestos. Este terminal es un PC de sobremesa convencional, dotado con un sistema operativo Linux, concretamente la distribución Ubuntu Desktop, en su versión 16.04. Esta instalación se realiza bajo línea de comandos, siempre en modo superusuario, y consta de los siguientes pasos:

1. En un terminal, escribimos el comando de instalación del paquete.

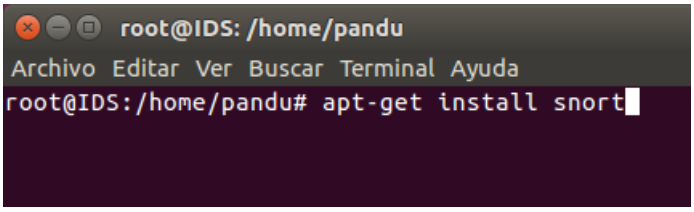

Figura 27. Instalación 1

2. Tras pulsar intro, se recopilarán los paquetes necesarios para su instalación, y antes de proseguir, se nos preguntará si deseamos continuar con la instalación del software elegido, pulsamos “s” e intro para iniciar la instalación.

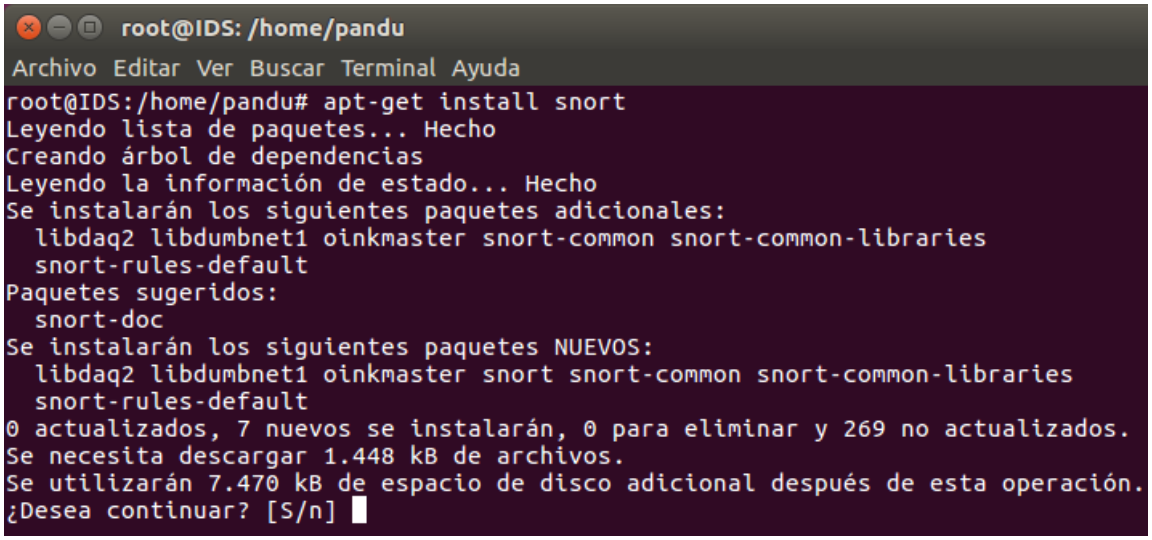

Figura 28. Instalación 2

3. Antes de seguir con la instalación, y para disponer de los datos que necesitemos previamente a la configuración de Snort, en un terminal nuevo haremos uso del comando ifconfig, para disponer del nombre de la interfaz de red sobre la que queremos que el IDS se ejecute, ya que es un dato que se requerirá en la instalación.

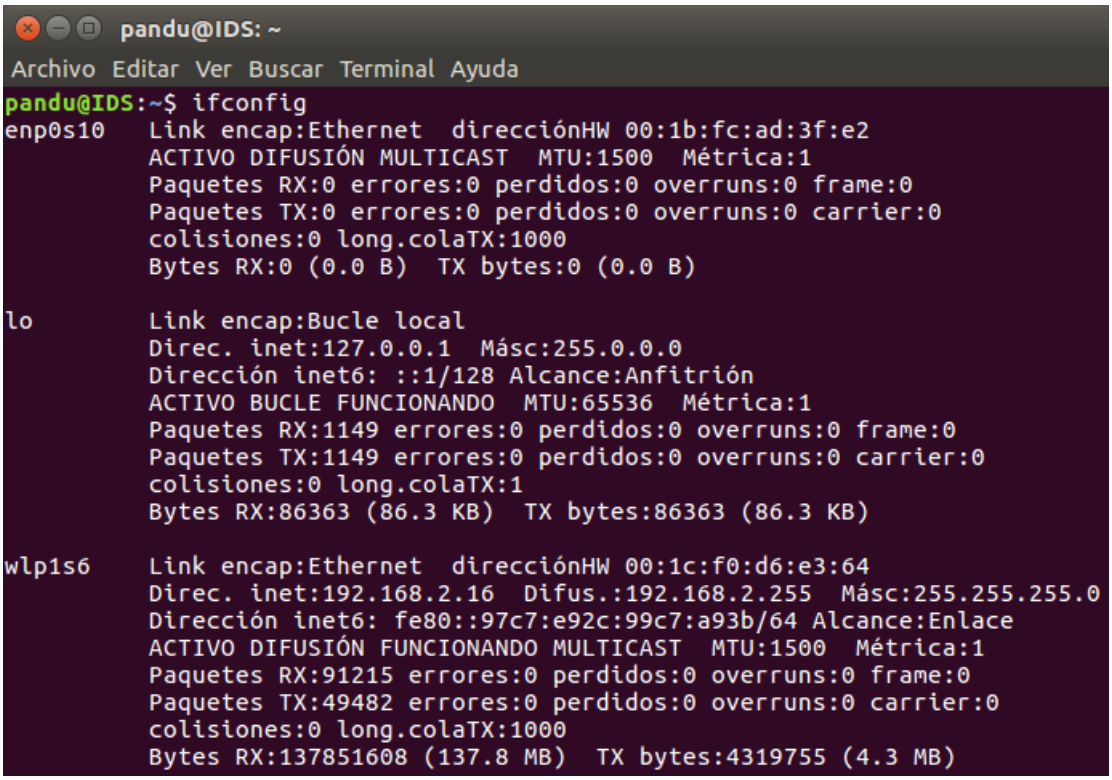

Figura 29. Instalación 3

4. Una vez anotada la interfaz (en nuestro caso utilizaremos la interfaz inalámbrica denominada wlp1s6), podemos cerrar este terminal, y volver al de la instalación, en el que se nos solicitará el nombre de dicha interfaz.

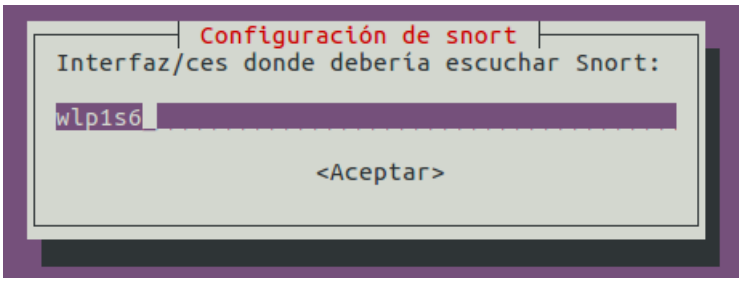

Figura 30. Instalación 4

5. En el siguiente paso, se nos solicita la dirección de red en la que el IDS trabajará, debe ser una dirección de red, en formato CIDR, lo que implica la inclusión de la máscara de subred. En el caso que nos atañe, la red es la 192.168.1.0/24.

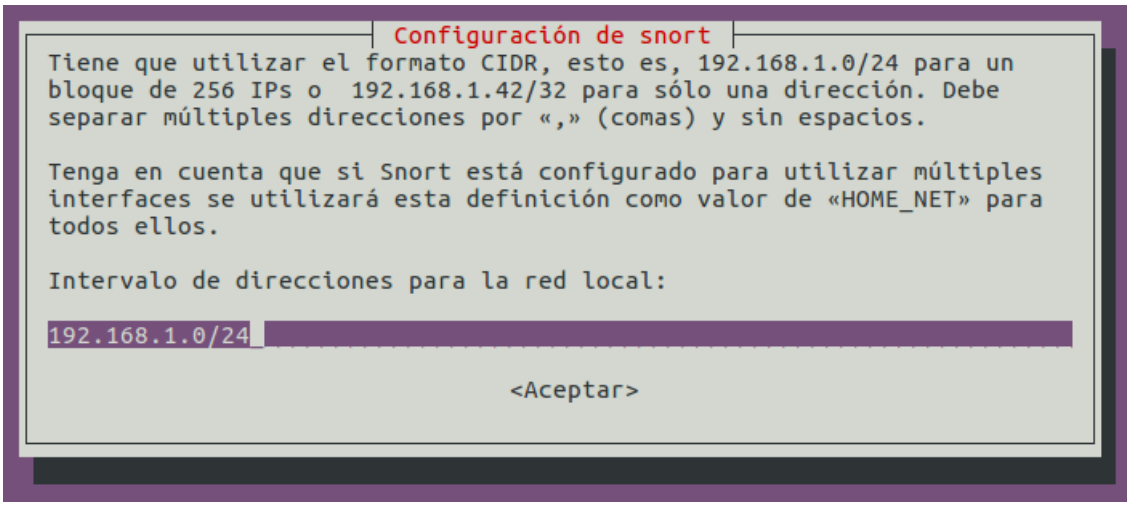

Figura 31. Instalación 5

6. Con los pasos seguidos hasta el momento, ya se habría realizado la configuración básica obligatoria de Snort tras su instalación para empezar a utilizarlo. Pero para proceder a una configuración más fina, aunque tengamos que volver a reincidir en cierta información, haremos uso del comando de configuración del paquete, el cual nos permite dicha configuración.

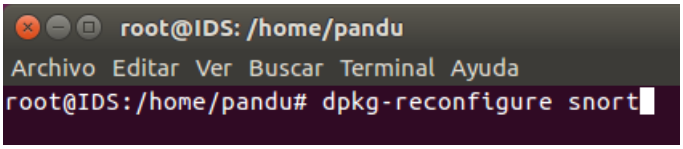

Figura 32. Instalación 6

7. Una vez realizado el paso anterior, volvemos a realizar la configuración del software, esta vez teniendo acceso a un mayor número de opciones. La primera será la elección del modo de ejecución. Para las pruebas que se realizan, lo más óptimo será la elección del arranque manual por consola.

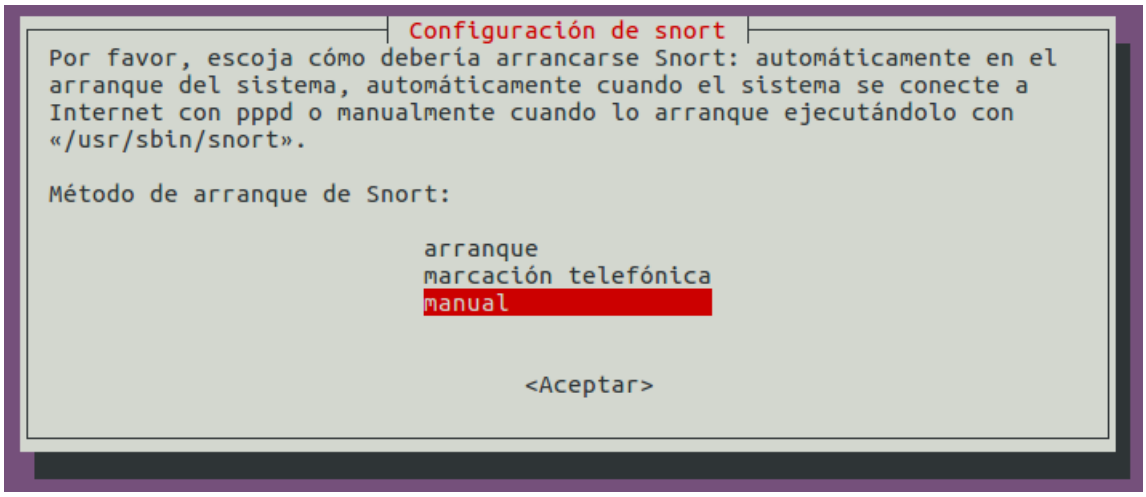

Figura 33. Instalación 7

8. En el siguiente paso, se nos vuelve a solicitar la interfaz de trabajo del IDS, aunque esta vez se nos proporciona algo más de información detallada. Volvemos a introducir la interfaz deseada.

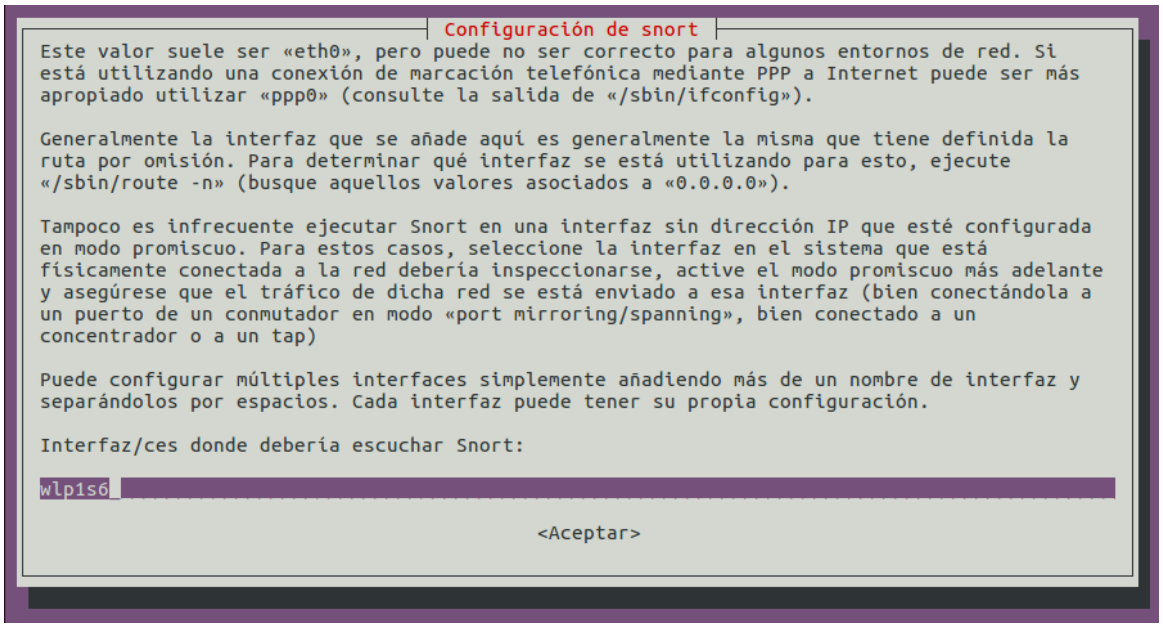

Figura 34. Instalación 8

9. Una vez más, se vuelve a solicitar la dirección de red que será supervisada por el sistema. La indicamos de nuevo.

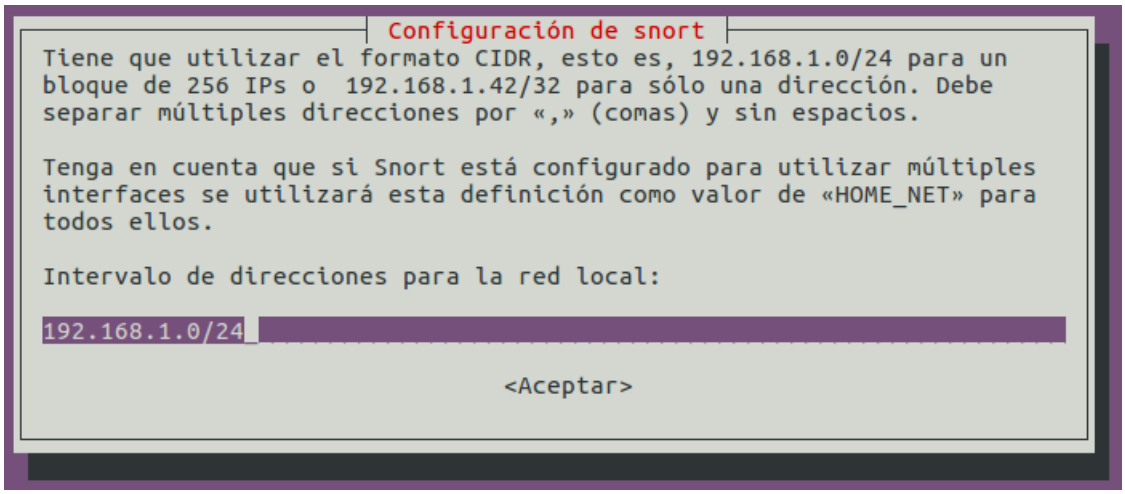

Figura 35. Instalación 9

10. Este nuevo paso, nos permite definir si deseamos deshabilitar que Snort se ejecute en modo promíscuo, como queremos que se analice el tráfico de la totalidad de la red, no deshabilitamos el modo promíscuo.

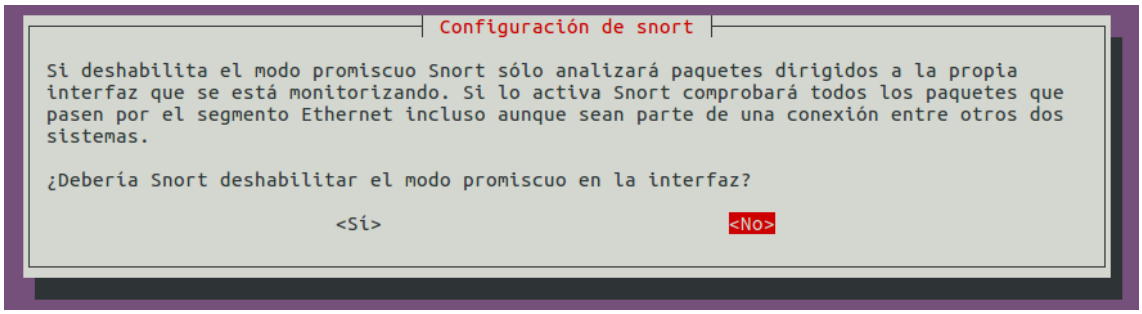

Figura 36. Instalación 10

11. A continuación, el programa de configuración, nos brinda la opción de inicializar Snort con unas opciones predeterminadas siempre que el sistema se inicie. En nuestro caso, dichas opciones se irán cambiando dependiendo del escenario y el tipo de ejecución que deseemos, por lo que dejamos el campo en blanco y pasamos al siguiente paso.

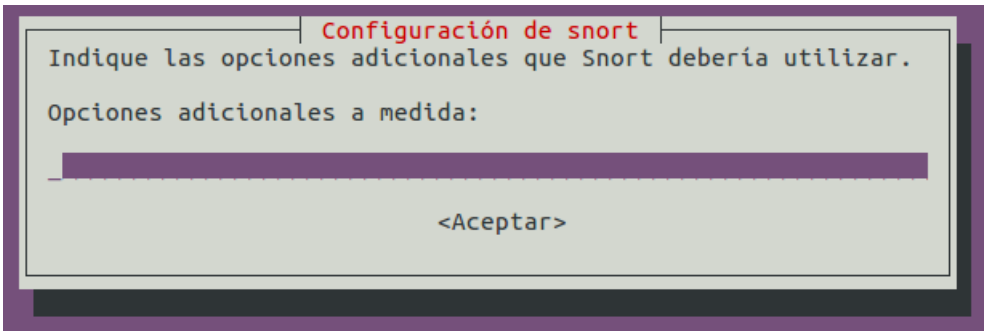

Figura 37. Instalación 11

12. Snort dispone de la funcionalidad de enviar diariamente por correo electrónico, unos resúmenes de los registros que se hayan generado. En el siguiente paso, podemos activar a conveniencia esta funcionalidad. Para el trabajo que se va a desempeñar, no es algo necesario, por lo que se ha desactivado dicha opción.

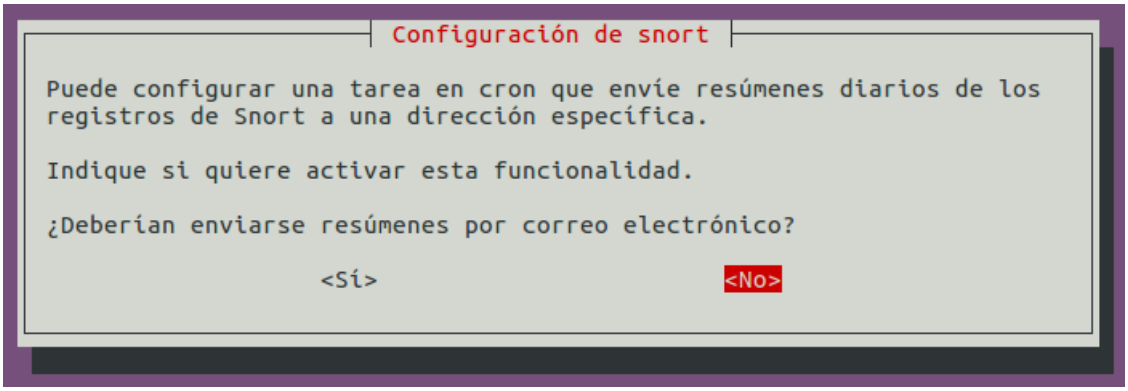

Figura 38. Instalación 12

13. Para finalizar, se nos muestra un aviso dando la información de que para que los cambios en la configuración realizada surjan efecto, deberemos reiniciar el servicio de snort.

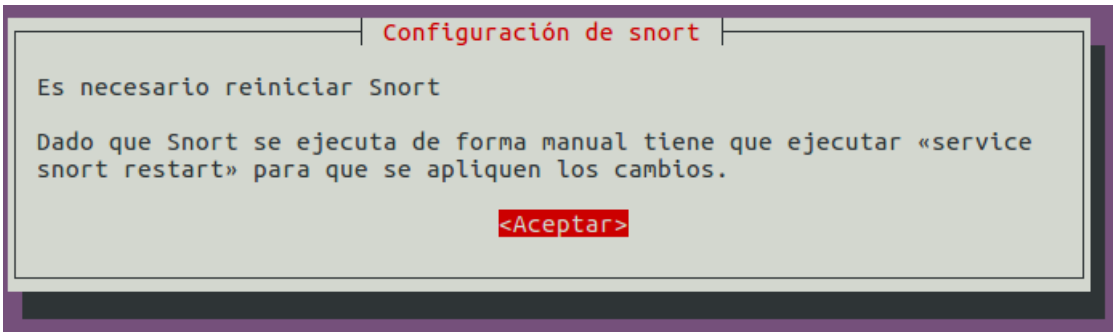

Figura 39. Instalación 13

14. Por último, y tal y como se solicitaba en el aviso anterior, procedemos al reinicio del servicio mediante el comando proporcionado. Con esto, podemos dar por finalizada la instalación de Snort en nuestro terminal.

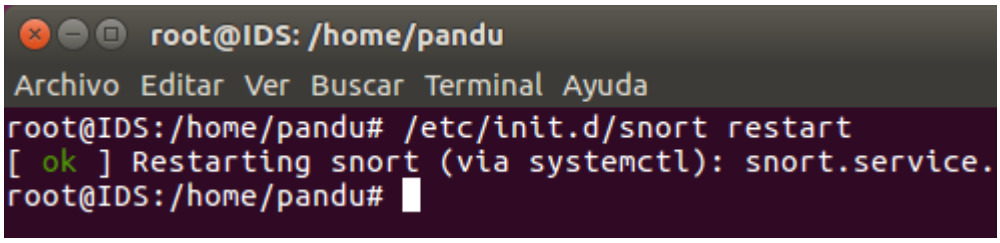

Figura 40. Instalación 14

12. REFERENCIAS BIBLIOGRÁFICAS

1. **J. Postel, J. Reynolds.** RFC 959 (FTP) File Transfer Protocol. 1985.

2. **Plumber, David C.** *RFC 826 (ARP) An Ethernet Address Resolution Protocol.* 1982.

3. **J.Postel.** *RFC 792 (ICMP) Internet Control Message Protocol.* 1981.

4. **IETF.** RFC 793 (TCP) Transmission Control Protocol. 1981.

5. **Rehman, Rafeeq Ur.** *Intrusion Detection Systems With Snort.* USA : Prentice Hall PTR, 2003.

6. **Jay Beale, Andrew R. Baker, Brian Coswell, Mike Poor.** *Snort 2.1 Intrusion Detection.* Rockland (USA) : Syngress Publishing Inc, 2004.

7. **Howlett, Tony.** *Open Source Security Tools.* USA : Prentice Hall, 2004.

8. Arpspoof Application Manual. s.l. : https://linux.die.net/man/8/arpspoof, 2016.

9. ICMPush Application Manual. s.l. : https://linux.die.net/man/8/icmpush, 2016.

10. Nmap Application Manual. s.l. : https://nmap.org/book/man.html, 2016.

11. Hydra Application Manual. s.l. : http://tools.kali.org/password-attacks/hydra, 2016.

12. **Jay Beale, Andrew R. Baker, Joel Esler.** *Snort IDS and IPS Toolkit.* Burlington : Syngress Publishing Inc., 2007.

13. **Project, The Snort.** *SNORT users manual v2.9.8.3.* 2015.

14. **Beggs, Robert W.** *Mastering Kali Linux for Advanced Penetration Testing.* s.l. : Packt Publishing, 2014.