IMPLEMENTACIÓN EN UN DISPOSITIVO HARDWARE DE UN SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN RED

Memoria de Trabajo de Fin de Grado

Rodrigo Lagartera Peña

Álvaro Acosta Rodríguez

Dirigido por: María Inmaculada Pardines Lence y Marcos Sánchez-Élez Martín.

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

2019

IMPLEMENTACIÓN EN UN DISPOSITIVO HARDWARE DE UN SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN RED

Memoria de Trabajo de Fin de Grado

Rodrigo Lagartera Peña

Álvaro Acosta Rodríguez

Dirigido por: María Inmaculada Pardines Lence y Marcos Sánchez-Élez Martín.

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

2019

“La desconfianza es la madre de la seguridad”

- Aristófanes

Agradecimientos

En primer lugar, queremos agradecer a los directores de este Trabajo de Fin de Grado, Inmaculada Pardines Lence y Marcos Sánchez-Élez Martín, la paciencia y dedicación aportada para sacar este trabajo adelante.

No podemos olvidarnos tampoco de todos nuestros familiares, cuyo apoyo constante y tiempo nos ha dado la confianza para poder hacer lo que siempre quisimos hacer. Gracias por todo vuestro apoyo.

También queremos dar las gracias a todos los compañeros y compañeras que han compartido clase y horas de trabajo y estudio con nosotros. Todo vuestro apoyo, vuestra capacidad de ayudarnos en los malos momentos y vuestros consejos han sido muy importantes. Gracias.

Por último, queremos agradecer a todos esos amigos y amigas que siempre nos han dado su apoyo, sus frases de: “Tú puedes hacerlo” o “Sé que tú eres capaz” han sido más importantes de lo que pueda parecer. Gracias.

Resumen

La seguridad informática es un tema importante, que tiene como objetivo proteger la información, de forma que los datos puedan ser accedidos solo por personas autorizadas. La protección se consigue mediante el uso de uno o varios sistemas que previenen, detectan y/o actúan contra diversos tipos de amenazas. La seguridad informática abarca la protección de los sistemas contra intrusiones, la seguridad de las comunicaciones y el diseño de aplicaciones y dispositivos libres de vulnerabilidades que puedan ser aprovechadas para entrar en el sistema.

Existen distintos sistemas que previenen o alertan de posibles ataques contra la red interna de una organización o empresa. Uno de estos sistemas son los IDS (Intruders Detection System) que se situarían en una segunda línea de defensa de una red. Estos sistemas no pueden impedir un ataque, si no que su función es detectar el ataque lo más pronto posible para minimizar los posibles daños producidos.

En este trabajo se propone la implementación de un Sistema de Detección de Intrusos (IDS) en un dispositivo hardware conectado entre una red interna (perteneciente a una organización) y el exterior (internet). Este sistema funciona de forma pasiva, es decir, alerta de los ataques una vez producidos en base a un conjunto de reglas predefinidas, no es capaz de detectar ataques si no existe una regla que lo identifique ni tampoco puede evitar que estos se produzcan.

Palabras clave: Hardware, IDS, alerta, regla, protocolo, paquete, datagrama, dispositivo, máquina virtual.

Abstract

Computer security is an important issue, that has the goal of protect the information, so that data only can be accessed by authorized staff. Protection is given by one or some systems that prevent, detect or act against different types of threats. Computer security covers system protection against intrusions, communication security and the design of applications and devices free of vulnerabilities that can be exploited to enter in the system.

There are different types of systems that prevent or alert about possible attacks against a company or enterprise internal network. One of these systems are IDS (Intruders Detection System) that are in the second defense line. These systems cannot avoid an attack; however, their function is to detect the attack as soon as possible to minimize the damage produced.

This project proposes the implementation of an Intruders Detection System (IDS) in a hardware device connected between an internal network (belonging to an organization) and the outside (internet). This system works in a passive way, that is, it alerts of the attacks once produced on the basis of a ser of predefined rules. It is not capable of detecting attacks if there is no rule that identifies it, nor can it prevent them from happening.

Keywords: Hardware, IDS, alert, rule, protocol, package, datagram, device, virtual machine.

Índice general

Capítulo 1

Introducción

1.1. Historia del arte

Los grandes ataques procedentes [1] de Internet como virus, spam y explotación de vulnerabilidades software, hacen que los métodos de protección sean elementos importantes para prevenir y evitar que la información y los datos estén indefensos ante ataques. Estos métodos son: el uso de criptografía, políticas de seguridad, firewalls, IDS e IPS.

Respecto a los sistemas de detección y prevención de intrusiones, los IPS (Intrusion Prevention System) detectan, alertan y detienen las intrusiones sospechosas, mientras que los IDS (Intrusion Detection System) solo detectan y alertan de las intrusiones.

Estos sistemas pueden estar implementados en software, en este caso, se instalan y configuran en una máquina de la red, o en hardware, un dispositivo se programa para que funcione exclusivamente como IDS. Respecto a su funcionamiento, estos sistemas generan alertas cuando detectan anomalías en la red que puedan ser catalogadas como un posible ataque. Hay varios tipos de IDS, como el HIDS, basado en host, y el NIDS, basado en red.

1.2. Motivación

Actualmente, las organizaciones, empresas y particulares tienen dispositivos conectados a una red. En el momento en que un dispositivo se conecta, está expuesto a todo tipo de ataques a través de esa conexión. Los datos con los que trabaja o guarda una organización deben ser protegidos del acceso no autorizado y, de igual forma, es necesario proteger los equipos para que, en caso de ofrecer algún tipo de servicio, este siempre se encuentre disponible para los usuarios.

Para proteger lo máximo posible una red, es necesario que se implanten medidas de seguridad eficaces, tanto de prevención, como de reparación. Estas medidas deben ser capaces de asegurar todos los equipos conectados a esa red.

Los sistemas de prevención y detección pueden evitar ataques potenciales si se descubren a tiempo. Pueden actuar contra el problema, evitando o deteniendo el ataque, o simplemente avisar al administrador de la red. Las herramientas comúnmente usadas para detectar un ataque en una red son los IDS, mientras que para prevenirlos se usan los firewalls (en castellano cortafuegos).

Los cortafuegos previenen activamente de los paquetes que llegan y salen de la red del dispositivo, ya que, se configuran para controlar todo el tráfico que entra y/o sale de una organización. En base a una política de seguridad pueden dejar pasar el tráfico o no.

De igual forma, los IDS analizan los paquetes que llegan y salen del dispositivo, detectando aquellos que consideran como un posible ataque y generando alertas. Los cortafuegos suelen estar situados en el perímetro de la red de la organización,

analizando todo el tráfico que entra o sale, mientras que los IDS se sitúan a continuación, dentro de la red interna. En caso de que un intruso consiga burlar el cortafuegos, será misión del IDS detectarlo.

Lo que se propone en este Trabajo de Fin de Grado es la implementación en un dispositivo hardware de un Sistema de Detección de Intrusos basado en red (NIDS). El sistema diseñado captura y analiza paquetes de red con la finalidad de detectar si la red está siendo atacada, y en caso afirmativo, se generarán y enviarán alertas avisando de la situación.

Nuestro sistema IDS hardware estará conectado a la red interna a través de un conmutador, igual que a los equipos que está protegiendo, de forma que todo paquete que entra por un puerto del conmutador es redirigido al puerto del IDS para que pueda ser analizado. El motivo por el que se implementa en un dispositivo hardware es que este tipo de dispositivos tienen un sistema operativo reducido y no tienen instaladas otro tipo de aplicaciones, lo que implica que es menos susceptible de tener vulnerabilidades que puedan ser explotadas por un atacante. Además, es más eficiente que un IDS software corrigiendo sobre un servidor, ya que liberamos a este de la carga de trabajo que supone el análisis del tráfico de la red.

1.3. Objetivos del trabajo

Para poder llevar a cabo la captura y el análisis de paquetes en una red, en primer lugar, fue necesario estudiar y comprender el funcionamiento de aplicaciones que realizan un trabajo similar, como Snort o Suricata. Esta tarea incluyó el estudio de las reglas de Snort, cuya sintaxis sirve tanto para Snort como para Suricata y también para nuestro IDS. Estas reglas contienen firmas de ataques, es decir, contienen unos atributos que identifican el contenido que aparece en ciertos campos de un paquete cuando se produce un determinado ataque.

Posteriormente, se realizaron simulaciones de captura de paquetes para poder analizar su contenido. Los paquetes se descomponen en campos y se organizan según el protocolo usado. Una vez descompuestos los paquetes y organizados por campos, se diseñó un programa de análisis de paquetes. Para ello, primero, fue necesario vincular cada uno de los campos de un paquete con un atributo de regla. Después, comparar los campos de los paquetes capturados con los correspondientes atributos de las reglas en busca de una posible coincidencia y, por último, generar una alerta.

Por último, se instaló el IDS en el dispositivo hardware y se realizaron pruebas y simulaciones de ataques para comprobar su correcto funcionamiento.

1.4. Plan de trabajo

Para alcanzar estos objetivos, se han llevado a cabo las siguientes tareas:

1. Simulación de ataques y flujo de paquetes utilizando máquinas virtuales.
2. Estudio y clasificación de los distintos tipos de protocolos junto con los campos empleados.
3. Aprendizaje de Python como lenguaje para el procesamiento de paquetes.

4. Desarrollo de una primera versión del programa que capturaba paquetes, los descomponía y clasificaba según sus campos y según el tipo de protocolo.
5. Estudio y comprensión de las reglas Snort.
6. Creación de reglas propias basadas en la sintaxis de las reglas Snort.
7. Desarrollo del programa de análisis de paquetes, capaz de leer reglas, comparar las premisas de estas con el contenido del paquete y generar alertas cuando se produce una coincidencia.
8. Evaluación de los resultados obtenidos en una simulación a través de máquinas virtuales.
9. Instalación de los programas diseñados en el dispositivo hardware y evaluación de los resultados obtenidos en dicho dispositivo.

El plan de trabajo es orientativo y describe los pasos que se han llevado a cabo para la realización de este proyecto. Durante el desarrollo del mismo, ha sido necesario profundizar en los aspectos teóricos de los protocolos y en el desarrollo del código del programa para analizar los paquetes. La Figura 1 muestra el porcentaje de tiempo empleado en cada tarea respecto al tiempo total empleado, cada tarea está dividida en el porcentaje que ha empleado en ella cada componente del proyecto.

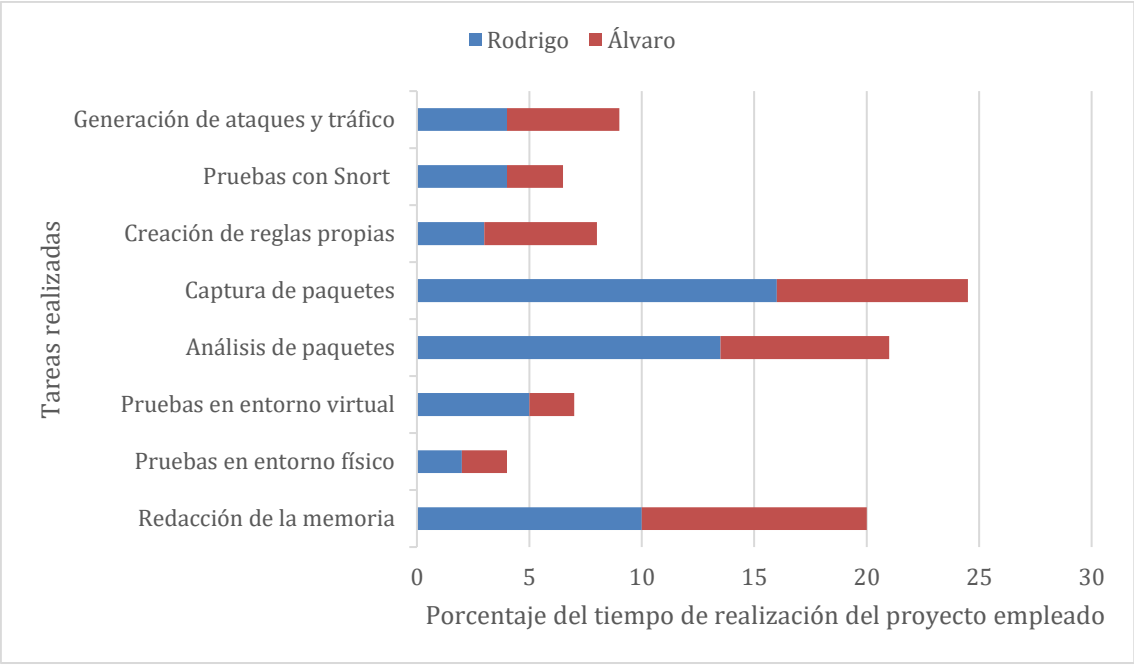


Figura 1. Porcentaje del tiempo empleado en cada tarea.

1.5. Organización de la memoria

Los capítulos en los que hemos organizado la memoria se corresponden con las distintas etapas por las que ha ido pasando nuestro trabajo. En cada uno de ellos se describen detalladamente cada una de las tareas que hemos realizado. Las secciones de cada capítulo explican en detalle los procedimientos utilizados para desarrollar cada etapa.

A continuación, se presenta brevemente el contenido de estos capítulos:

- En el **Capítulo 2** se explica cuál es el funcionamiento de un sistema de detección de intrusos, los distintos tipos de dispositivos IDS que hay y ejemplos de otros sistemas similares en los que se ha basado nuestro Trabajo de Fin de Grado.
- En el **Capítulo 3** se describen los protocolos de red, se explican sus características y cuáles son los campos en los que se divide cada paquete. También se hace referencia al tratamiento de esta información para poder usarla en este trabajo.
- En el **Capítulo 4** se enumeran los procedimientos y las herramientas que se han utilizado para el desarrollo e implementación de los programas que se han desarrollado en este trabajo.
- En el **Capítulo 5** se describen las pruebas realizadas y se evalúan los resultados de la implementación de nuestro dispositivo IDS.
- Por último, en el **Capítulo 6**, se aportan las principales conclusiones de este trabajo y se propone una serie de tareas como posible trabajo futuro.
- La normativa del Trabajo Fin de Grado de la Facultad de Informática de la Universidad Complutense de Madrid obliga además a que se realice una traducción al inglés del capítulo de introducción y del capítulo de conclusiones, dichas traducciones se presentan en forma de *Apéndice*.

Capítulo 2

Sistema de Detección de Intrusos

2.1. Definición de IDS

Un Sistema de Detección de Intrusos [2] (IDS por sus siglas en inglés) es el proceso de detección de un uso no autorizado de, o ataque sobre, un ordenador, una red o una infraestructura de telecomunicaciones. La intención básica de un IDS es detectar cualquier evento sospechoso que ocurra en una red, enviando una alerta y mostrándola como mensaje en una pantalla, mediante el envío de un e-mail, o mediante la reconfiguración de la configuración del Firewall. No es necesario que sea una intrusión para alertar, cualquier tipo de comportamiento considerado fuera de lo normal podría ser detectado.

Todos los IDS tienen tres componentes comunes: sensores, analizadores e interfaces de administrador. Los sensores colectan tráfico y los datos de actividad del usuario y lo envían al analizador, que busca actividades sospechosas. Si el analizador detecta una actividad que está programada de una manera considerada sospechosa, éste envía una alerta al administrador de la interfaz.

Los IDS se pueden dividir en dos tipos: basados en red, que monitorizan comunicaciones de red, y basados en host, que canalizan la actividad de un sistema en particular.

Los IDS pueden ser configurados para buscar ataques, analizar registros de auditorías, terminar una conexión, alertar a los administradores de que un ataque está ocurriendo, exponer técnicas hacker, ilustrar qué vulnerabilidades deben ser dirigidas, y rastrear posibles hackers individuales.

2.1.1. IDS basado en red

Un IDS basado en red [2] (NIDS por sus siglas en inglés) usa sensores, los cuales están distribuidos por todos los hosts de la red. Cada uno de ellos posee su tarjeta de interfaz de red (NIC) en modo promiscuo, lo que permite que cada NIC vigila el tráfico que tiene la dirección de red de su host, su broadcast e incluso su multicast. Los NIC capturan todo el tráfico en modo promiscuo, hacen una copia de todos los paquetes, y pasan una copia al TCP sack y otra al analizador para buscar tipos específicos de patrones.

2.1.2. IDS basado en host

Un IDS basado en host [2] (HIDS) puede estar instalado en puestos individuales y/o servidores para buscar actividades anómalas o inapropiadas. Los HIDS comprenden y monitorean el tráfico de red de un dispositivo, por lo que son usados para evitar que los usuarios eliminen archivos del sistema, reconfiguren ajustes importantes o pongan el sistema en riesgo de alguna otra manera.

2.1.3. IDS basado en firmas

Se tratan de IDS [2] que acumulan ataques específicos y cómo son llevados a cabo. Los modelos sobre cómo son llevados a cabo son desarrollados y llamados “firmas”. Cada ataque identificado tiene una firma, que es usada para detectar un ataque en proceso o determinar si ha ocurrido alguno en la red. Sin embargo, cualquier acción no reconocida como ataque, es considerada aceptable.

2.1.4. IDS basado en anomalías

Este tipo de IDS [2] es un sistema basado en el comportamiento en el tráfico de red o en la actividad de los usuarios. Este tipo de sistema no usa firmas predefinidas, si no que utiliza un modo de aprendizaje para construir un perfil del entorno de actividades comunes. Esto permite que, si se detecta una anomalía en alguna actividad del entorno, se envía una alerta.

2.2. Reglas

El funcionamiento de los NIDS basados en firmas consiste en capturar el tráfico de la red y comprobar si los atributos de los paquetes capturados coinciden con la firma de algún ataque, en este caso, será necesario generar una alerta. Las firmas están definidas en reglas y consisten en ciertos patrones que aparecen en distintos campos del paquete cuando hay un ataque. Existen conjuntos de reglas que se pueden descargar de la red o bien el administrador de la red puede definir sus propias reglas. Es conveniente que el personal de seguridad revise y complete las reglas ofrecidas, centrándose en las vulnerabilidades del sistema.

Algunas de las reglas que se han utilizado para la realización de este trabajo son reglas incluidas en el IDS Snort para protocolos como IP, ICMP, UDP y TCP, otras han sido diseñadas por nosotros usando patrones de ataques y testeo conocidos, como detección de ataques Ping Flood, detección de escaneo de puertos y TCP DoS, entre otros.

2.3. Alertas

La funcionalidad de un IDS no es detener un ataque, si no informar de su detección cuando se cumplen las premisas de una firma predefinida.

Cuando, en el análisis de paquetes, un paquete capturado tiene atributos que coinciden con estas premisas, se emite una alerta. Las alertas informan de que se ha detectado un posible ataque y aportan detalles del mismo.

Las alertas emitidas en este trabajo aparecen en la consola de ejecución del programa cuando se detectan coincidencias. El mensaje que aparece aporta información acerca del posible ataque, pero no aporta detalles como direcciones IP, MAC, puertos, tiempo de vida o tamaño del paquete.

2.4. Ejemplos de NIDS

El sistema de detección de intrusos implementado en este trabajo, con las reglas utilizadas y los programas de captura y análisis de paquetes está inspirado en el funcionamiento de los IDS Snort [3] y Suricata [4].

2.4.1. Snort

Snort es un NIDS open source que cuenta con un lenguaje de creación de reglas en el que se pueden definir los patrones que se utilizarán a la hora de monitorizar un sistema. Además, ofrece una serie de reglas y filtros predefinidos que se pueden ajustar. Snort puede funcionar como sniffer, registro de paquetes o como un NIDS. Cuando un paquete coincide con algún patrón de las reglas de configuración, se logea para registrar el lugar y el momento en el que se produjo el ataque.

La arquitectura en la que está basado Snort [5] consiste en un sistema compuesto por un sniffer, preprocesadores, el motor de detección y la salida. El sniffer colecta el tráfico e identifica cada estructura de paquete. Tras pasar por el sniffer, los paquetes son enviados al preprocesador, que determina qué tipo de paquetes o comportamiento se está tratando, para ello, posee protocolos definidos que ayudan a clasificar la información. Posteriormente, esta información pasa el motor de detección, que compara cada paquete con cada regla predefinida, y, si existe coincidencia con el contenido de esta, son enviadas a la salida. En la salida, se registran y/o envían alertas basadas en las acciones de las reglas. En la Figura 2 [6], se puede observar esta arquitectura:

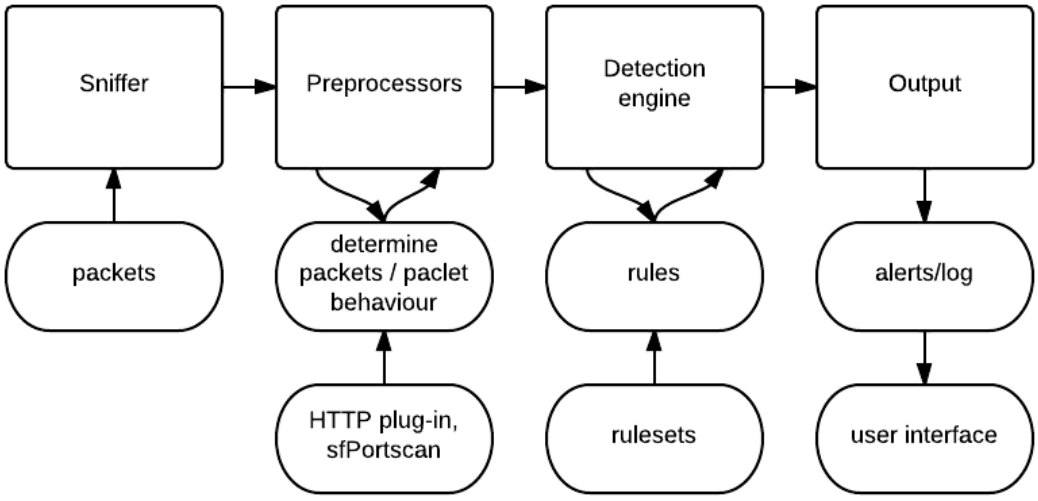


Figura 2. Arquitectura de Snort.

2.4.2. Suricata

Suricata es un IDS open source rápido y robusto. Puede funcionar como IPS, monitor de seguridad de una red (NSM) y procesador offline de paquetes capturados. Suricata inspecciona el tráfico de red usando extensas reglas y un lenguaje de firma, junto con un potente script detector de amenazas.

Una de las características más importantes de Suricata [7] es que permite la ejecución de varios procesos o subprocesos de forma simultánea, siendo posible procesar una gran cantidad de paquetes de forma simultánea, aumentando así el rendimiento. Este IDS utiliza, también, reglas que son independientes del puerto que use un protocolo, ya que éste es automáticamente detectado, por esa razón las reglasSnort son compatibles con Suricata.

Capítulo 3
Protocolos

Para la realización de este trabajo se han usado distintos protocolos comunes en paquetes de red. Estos protocolos juegan un papel muy importante en el tráfico de red, así como en los sistemas de seguridad, por lo que la mayoría de los ataques explotan vulnerabilidades existentes en los protocolos, para ello utilizan los campos de los protocolos más comunes, y es necesario analizar estos campos.

Una gran parte del tráfico en internet es tráfico TCP/IP, que pertenece al modelo OSI [8], y es uno de los protocolos más usados. OSI es un modelo formado por 7 capas cuya finalidad es facilitar la comunicación entre sistemas. En cada capa, el paquete se encapsula con un determinado protocolo, de modo que el nuevo paquete tendrá una cabecera, propia del protocolo, y un campo datos, que se corresponde con el paquete que le pasa la capa de arriba.

En la figura 3 [9] explicativo de los protocolos empleados en TCP/IP y el modelo OSI:

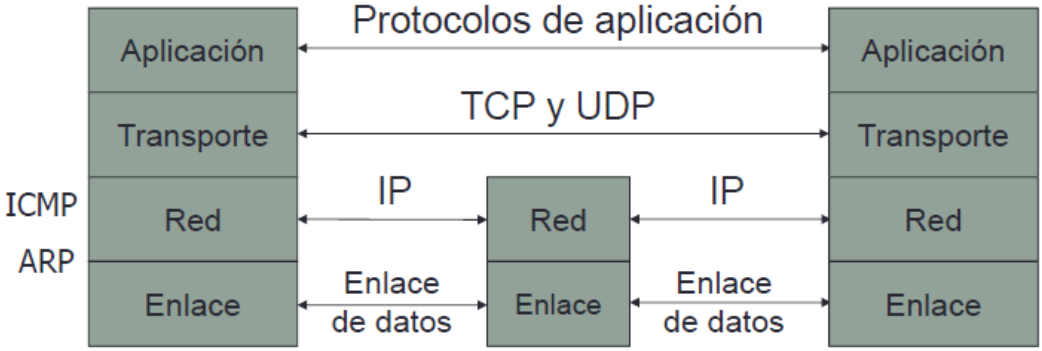


Figura 3. Modelo OSI y TCP/IP.

Estos protocolos de red tienen vulnerabilidades que pueden ser explotadas por un atacante. Los ataques que se basan en explotar las debilidades de un determinado protocolo tienen unas características comunes, que se traducen por la utilización de determinados valores en campos concretos del paquete. Cada paquete contiene varios protocolos encapsulados, esto es debido a que, en el modelo OSI, cada capa añade la cabecera al campo datos y lo pasa a la capa de abajo, que añade a la vez cabecera y lo pasa a la de abajo.

3.1. Definición de protocolo y arquitectura TCP/IP

Un protocolo [10] de red es un grupo estándar de reglas que determina como se comunican los sistemas mediante redes. Dos sistemas pueden usar el mismo protocolo para comunicarse y entenderse entre ellos a pesar de las diferencias. Un protocolo define qué información es comunicada, cómo y cuándo es comunicada.

El protocolo TCP/IP [11] es un protocolo jerárquico, del modelo OSI, formado de módulos interactivos, los cuales proporcionan una funcionalidad específica. Sin embargo, estos módulos no son necesariamente independientes. Mientras que el modelo OSI especifica qué funciones pertenecen a cada una de sus capas, las capas del protocolo TCP/IP contienen protocolos relativamente independientes que pueden ser mezclados y emparejados dependiendo de las necesidades del sistema.

En la capa de red TCP/IP, el principal protocolo definido es el IP y en la capa de transporte se definen protocolos como el UDP y el TCP. En este proyecto se tratan ambos protocolos.

3.2. Protocolo IP

El protocolo IP [12] es parte de la capa de Internet del conjunto de protocolos TCP/IP. Es un protocolo de confianza y no orientado a conexión, cuyo objetivo es realizar las entregas con la mayor efectividad posible, pero sin garantías.

Este protocolo utiliza direcciones numéricas denominadas direcciones IPv4 compuestas por cuatro números enteros (4 bytes) entre 0 y 255, y escritos en el formato xxx.xxx.xxx.xxx.

En IPv6, el protocolo de Internet fue modificado extensamente para acomodar el crecimiento imprevisto de Internet. El formato y la longitud de la dirección IP fueron cambiados junto con el formato del paquete.

El organismo a cargo de asignar direcciones públicas de IP, es decir, direcciones IP para los equipos conectados directamente a la red pública de Internet, es el ICANN.

El protocolo IP proporciona un servicio básico de entrega de paquetes sobre el que se construyen las redes TCP/IP, no es un protocolo orientado a conexión, tampoco realiza detección ni recuperación de paquetes perdido o erróneos. Este protocolo no garantiza que los paquetes lleguen en orden ni la detección de paquetes duplicados.

Se determina el destinatario del mensaje mediante la dirección IP del equipo, la máscara de subred y la pasarela determinada, que permite al protocolo saber a qué equipo enviar un datagrama, si el equipo de destino no se encuentra en la red de área local.

Las funciones básicas que presenta el protocolo IP son, el direccionamiento, la fragmentación y reensamblaje de paquetes, es decir, división del paquete en fragmentos de un tamaño aceptable por la red para luego volver a juntarlo, encaminamiento de datagramas, encaminado de paquetes atendiendo a información de tablas de rutas y por último la construcción de tablas de rutas mediante dos mecanismos, manual, llamado routing estático, o también por routing dinámico, por medio de RIP, OSPF, BGP.

3.2.1. Formato del datagrama IPv4

A continuación, la figura 4 muestra el formato del datagrama IPv4 [13]:

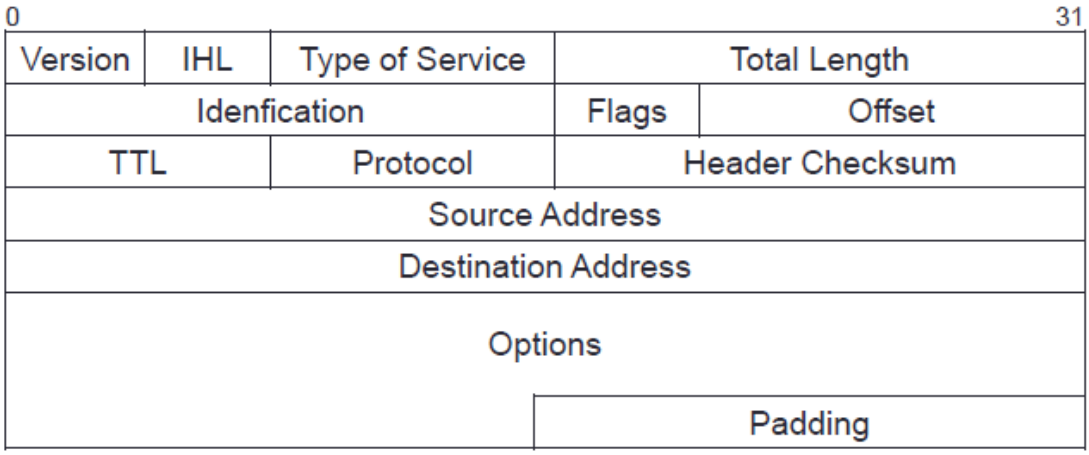


Figura 4. Datagrama IP.

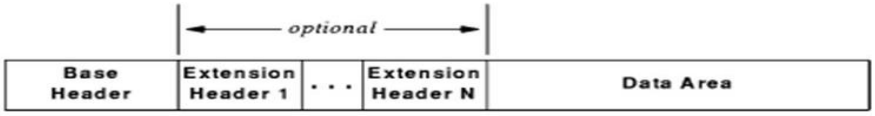
- **Versión**: Los protocolos evolucionan y cambian con el tiempo. Por esto, es conveniente saber con qué versión se ha generado un datagrama.
- **Longitud**: Es la longitud de la cabecera medida en palabras de 32 bits. Puesto que este campo tiene 4 bits la longitud máxima de la cabecera es de 64 octetos.
- **Servicio**: Lo rellena quien envía el datagrama. Su utilidad actual es muy escasa, pero irá aumentando en la medida en que se empleen diferentes tipos de tráfico. Su formato es:
    - PRIO: Se utiliza en casos de congestión.
    - D: Dar prioridad al retardo.
    - T: Dar prioridad al throughput.
    - R: Dar prioridad a la fiabilidad.
    - C: Dar prioridad al coste.
- **Longitud total:** Longitud del datagrama (cabecera + datos) en bytes.
- **Longitud máxima del datagrama**: 64 Kbytes.
- **Identificador:** Número de 16 bits que identifica al datagrama.
- **Flags:**
    - MF (More Fragments): si está a 1 indica que no es el último fragmento.
    - DF (Don't Fragment): si es 1 prohíbe la fragmentación.
- **Desplazamiento del fragmento:** Número secuencia del fragmento (8 bytes)
- **Tiempo de vida** (TTL, *Time To Live*): Número de encaminadores que puede atravesar el paquete. Cuando TTL = 0 el paquete debe ser descartado.
- **Protocolo**
  Protocolo de la capa superior al que deben entregarse los datos:
    - 1: Internet Control Message Protocol (ICMP)

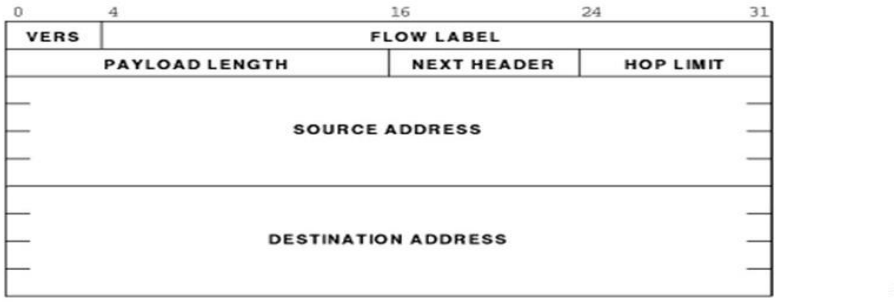
- 2: Internet Group Management Protocol (IGMP)
- 6: Transmission Control Protocol (TCP)
- 8: Exterior Gateway Protocol (EGP)
- 17: User Datagram Protocol (UDP)
- 41: IP Version 6 (IPv6)
- 89: Open Shortest Path First (OSPF)

- **Checksum:** Suma de control de la cabecera.
- **Direcciones IP origen y destino**: Identifican al host emisor y receptor del paquete.
- **Opciones:** Campo opcional, con opciones especiales. Encaminamiento de origen, sello de ruta, sello de tiempo. Tamaño máximo del campo opciones: 10 palabras.

3.2.2. Formato del datagrama IPv6

A continuación, en la figura 4ª se muestra el formato del datagrama IPv6[44]:





- **Version**: en este caso la versión es IPv6, longitud de 4 bits
- **Traffic Class** distingue diferentes requisitos de entrega del datagrama, es similar al campo DS de IPv4, su longitud es de 8 bits.
- **Flow Label**: Etiqueta el paquete como perteneciente a un flujo para mejorar el procesamiento realizado por los encaminadores de la red.
- .**Payload Length**: Longitud sin contar la cabecera, hasta un máximo de 64 Kbytes.
- **Hop Limit**: Similar al campo TTL de IPv4, su longitud es de 8 bits.
- **Source/Destination Address**: Direcciones origen y destino
- **Next Header**: Define la siguiente cabecera en el datagrama, encapsulada en la sección de datos.

3.2.3. Vulnerabilidades del protocolo IP

La vulnerabilidad más conocida del protocolo IP [14] es la falta de integridad y autenticidad de origen. Se pueden falsificar mensajes IP usando, por ejemplo, una dirección de origen arbitraria *(IP address spoofing*).

3.2.4. Posibles ataques

Se pueden realizar ataques como el DoS reflejado, acceso no autorizado y el anterior mencionado, IP address spoofing. Como contramedidas, es recomendada la filtración de entrada y salida y el protocolo de seguridad IPsec.

3.3. Protocolo TCP

TCP [15] (que significa Protocolo de Control de Transmisión) es uno de los principales protocolos de la capa de transporte del modelo TCP/IP. Es un protocolo que usa puertos orientado a conexión, es decir, crea una conexión virtual entre dos máquinas para enviar datos, permitiendo una conexión proceso-proceso. Las principales características del protocolo TCP son que permite comenzar y finalizar la comunicación, es capaz de detectar los segmentos de datos perdidos o erróneos y los vuelva a transmitir, si hay un segmento duplicado lo descarta, hace una entrega ordenada de paquetes, pues ordena los segmentos en el destino, establece tres fases en la comunicación, establecimiento, transferencia y cierre, la unidad de transferencia es el segmento TCP y por último realiza control de flujo y errores gracias al método de la ventana deslizante.

3.3.1. Formato del datagrama

En la figura 5 se muestra el formato del datagrama TCP [16].

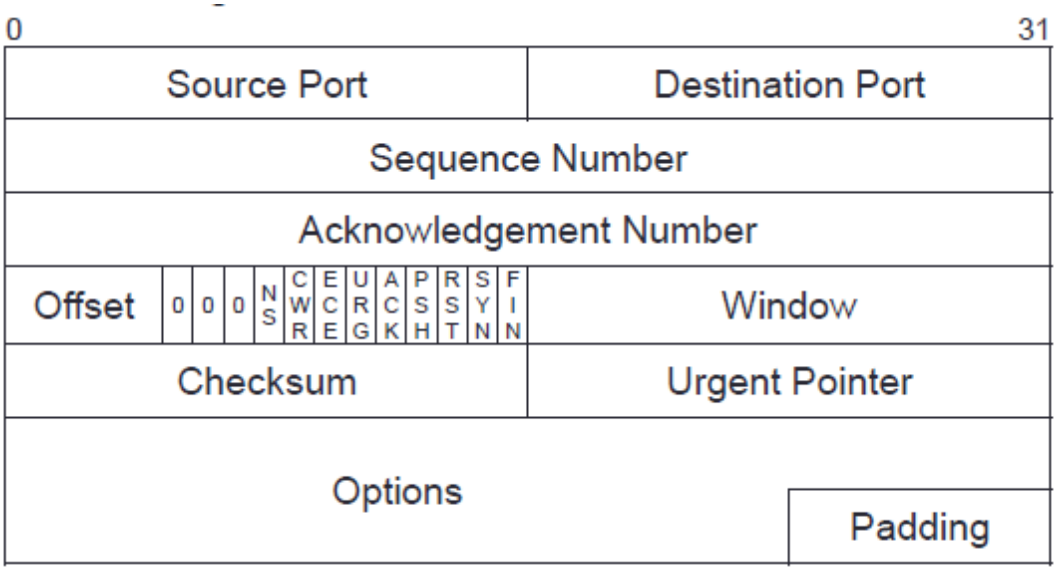


Figura 5. Datagrama TCP.

- **Puerto origen y destino:** Identifican los extremos de la conexión.

- **Número de secuencia:** Se usa para determinar la posición del primer byte del segmento con respecto al mensaje original.
- **Número de confirmación:** Indica el número de secuencia del siguiente byte que se espera recibir.
- **Longitud de la cabecera:** Medida en palabras de 32 bits.
- **Tamaño de ventana:** Permite anunciar el tamaño de la ventana de recepción durante la conexión TCP. El valor del campo ventana indica la cantidad de bytes que el receptor es capaz de aceptar.
- **Flags:**

    - **URG**: indica si el segmento transporta datos urgentes al principio.
    - **ACK:** indica si el segmento lleva un número de confirmación valido.
    - **PSH**: indica si los datos deben ser pasados inmediatamente a la aplicación. Si no se activa, los datos se pueden almacenar en un buffer de recepción y estos se pasan a la aplicación cuando el buffer se llena.
    - **RST**: utilizado para abortar una conexión.
    - **SYN**: utilizado en el establecimiento de la conexión. Ambos extremos deben sincronizar los números de secuencia iniciales de la transmisión.
    - **FIN**: utilizado en la finalización de la conexión.

- **Checksum:** Se calcula como el complemento a 1 de la suma en complemento a 1 de todas las palabras de 16 bits que forman parte de:

    - El segmento TCP (datos más cabecera TCP).
    - Una pseudo-cabecera formada por información de la cabecera IP: dirección IP fuente, dirección. IP destino, campo Protocolo y longitud del segmento TCP.

- **Opciones:** Permite negociar parámetros adicionales de la conexión

    - Tamaño máximo del segmento (MSS): Si no se negocia el MSS al inicio de una conexión, se establece un valor por defecto de 536 bytes.
    - Factor de escala de ventana: Permite trabajar con ventanas mayores de 2^{16} bytes.
    - Sello de tiempo (timestamp): Permite introducir la hora de envío de un segmento.
    - Opción de confirmación selectiva (Selective ACK): Permite confirmar segmentos fuera de orden.

3.3.2. Vulnerabilidades en TCP

El protocolo TCP posee ciertas vulnerabilidades [17] como el acuerdo TCP, que, al consumir recursos, puede provocar una saturación del sistema o impedir nuevas conexiones. Otra vulnerabilidad conocida es que no proporciona integridad ni

autenticidad de origen, ya que se pueden suplantar, controlar o reiniciar conexiones obteniendo o adivinando su estado. La revelación de información en TCP permite identificar al SO y los puertos abiertos de los equipos, esto es debido a respuestas a peticiones poco convencionales o diferencias de comportamiento.

3.3.3. Posibles ataques

Los ataques [18] más comunes a este tipo de protocolo son:

- Inundación TCP SYN: Consiste en el envío de una gran cantidad de mensajes SYN por parte del atacante, lo que provoca una denegación de servicio. El mecanismo SYN cookies es la contramedida más eficaz, ya que codifica de forma segura el estado de conexión en el número de secuencia del servidor del mensaje SYN-ACK, recuperando y verificando la llegada del ACK final. Una variante de este ataque es el TCP Connection Flooding.
- Finalización de conexiones TCP: Un atacante obtiene, mediante escucha, los números de secuencia de una conexión existente y envía un paquete RST para cerrarla, lo que provoca una denegación de servicio.
- Suplantación de conexiones TCP: El atacante crea una conexión TCP en nombre de una dirección IP origen falsa. La mayoría de pilas TCP generan números de secuencia aleatorios para prevenir estos ataques.
- Secuestro de conexiones TCP: El atacante obtiene los números de secuencia y toma el control de la conexión, que normalmente conlleva una denegación de servicio.
- Exploración de redes y puertos: Consiste en el envío de peticiones a un rango de direcciones o puertos para encontrar máquinas activas o puertos abiertos, respectivamente.

Las medidas de prevención usadas para este protocolo son los cortafuegos, IDS, SYN cookies, números de secuencia aleatorios y seguridad de transporte (SSL/TLS).

3.4. Protocolo UDP

Es un protocolo [19] de transporte no orientado a conexión, por lo que no garantiza un servicio de extremo a extremo fiable. Divide los mensajes en datagramas (son llamados así los paquetes de este protocolo), pero estos van sin numerar. El receptor no envía una confirmación de recepción de los mensajes. No se garantiza la recepción ordenada de los datagramas, ni se retransmiten los datagramas perdidos o erróneos, ni se eliminan los datagramas duplicados.

3.4.1. Formato del datagrama UDP

La figura 6 muestra el formato del datagrama UDP [20]:

Figura 6. Datagrama UDP.

- **Puerto origen y destino:** Identifican a las aplicaciones que se comunican.
- **Longitud UDP:** Longitud total en bytes del datagrama UDP (incluyendo datos y cabecera).
- **Checksum:** Se utiliza para comprobar si se han producido errores en la transmisión. Se calcula como el complemento a 1 de la suma en complemento a 1 de todas las palabras de 16 bits que forman parte de:
    - El datagrama UDP (datos + cabecera UDP).
    - Una pseudo-cabecera formada por información de la cabecera IP: dirección IP fuente, dirección. IP destino, campo Protocolo y longitud UDP.

3.4.2. Vulnerabilidades del protocolo UDP

El protocolo UDP [21] como bien sabemos es un protocolo no orientado a conexión lo que hace que sea muy susceptible de spoofing. Al no comprobar la dirección IP origen, podemos poner la dirección IP de la víctima como origen de peticiones maliciosas, en un ataque por ejemplo por reflexión, llamado también UDP Relfection Flood.

3.4.3. Posibles ataques

Teniendo en cuenta las vulnerabilidades de este protocolo podemos encontrar ataques [18] como:

- IP Spoofing, el atacante puede suplantar la dirección IP, ya que UDP no lo verifica.
- Ataques por amplificación, el atacante realiza una pequeña solicitud la cual genera significativamente una mayor respuesta.
- UDP Flood, el atacante puede enviar un gran número de falsas peticiones a máquinas vulnerables las cuales responden a las peticiones.
- UDP Reflection Flood, en primer lugar, haciendo uso de un gran envío de peticiones maliciosas; segundo, el uso de la dirección IP falseada, que hace que la dirección origen es la dirección IP víctima, la cual es el objetivo del ataque donde se envían todas las respuestas.

Desde el lado de la víctima este tipo de ataques son casi imposibles mitigar, podemos prevenirlos haciendo uso de IP Tables limitando el tráfico UDP, realizar buenas configuraciones de red.

3.5. Protocolo ARP

El protocolo ARP [22] (Address Resolution Protocol), se utiliza para conocer la dirección física correspondiente a una determinada dirección IP.

Cada equipo conectado a la red tiene una dirección física (MAC) de 48 bits. Éste es un número único establecido en la fábrica en el momento de fabricación de la tarjeta. Sin embargo, la comunicación en Internet no utiliza directamente este número (ya que las direcciones de los equipos deberían cambiarse cada vez que se cambia la tarjeta de interfaz de red), sino que utiliza una dirección lógica asignada por un organismo: la dirección IP.

Para que las direcciones físicas se puedan corresponder con las direcciones lógicas (direcciones IP), el protocolo ARP interroga a los equipos de la red para averiguar sus direcciones físicas y luego guarda toda esa información en la denominada tabla ARP.

Cuando un equipo se quiere comunicar con otro, consulta la tabla ARP. Si la dirección IP requerida no se encuentra en la tabla, el protocolo ARP envía una solicitud a toda la red, preguntando por la dirección MAC asociada a la dirección IP buscada. La máquina que tenga esta dirección IP responderá con un mensaje ARP en el que devolverá su dirección MAC, que se almacenará en la tabla ARP de la máquina que ha hecho la solicitud, a continuación, podrá establecerse la comunicación. La información en las tablas ARP se guarda por un tiempo limitado.

3.5.1. Formato del datagrama ARP

En la figura 7 se muestra el formato del datagrama ARP [23]:

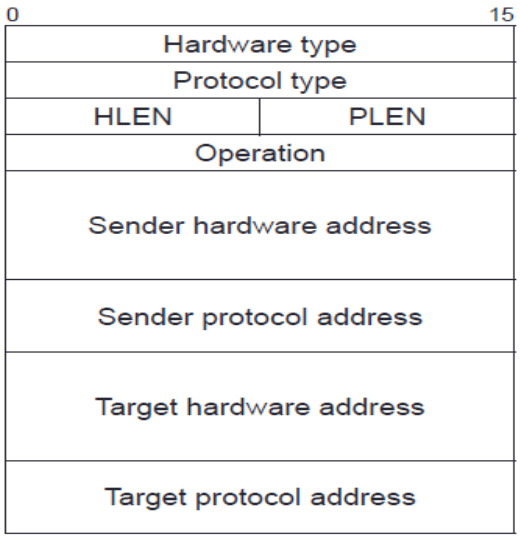


Figura 7. Datagrama ARP.

El Protocolo de resolución de direcciones utiliza un formato simple de mensaje que contiene una solicitud de resolución de dirección o una respuesta ARP.

- **Tipo de hardware (HTYPE)**: Este campo especifica el tipo de protocolo de red.
- **Tipo de protocolo (PTYPE)**: Este campo especifica el protocolo de interconexión de las redes para las que se destina la petición ARP. (IPv4 o IPv6)
- **Longitud Hardware (HLEN)**: Longitud (en octetos) de una dirección hardware. El tamaño de las direcciones Ethernet es 6 bytes.
- **Longitud del Protocolo (PLEN)**: Longitud (en octetos) de las direcciones utilizadas en el protocolo de capa superior. (El protocolo de capa superior especificado en PTYPE.)
- **Operación Especifica** (la operación que el emisor está realizando): 1 para la petición, 2 para la respuesta.
- **Dirección de hardware del remitente (SHA)**: Dirección física de la máquina emisora.
- **Remitente dirección de protocolo (SPA)**: Dirección IP de la máquina.
- **Dirección de hardware de destino (THA)**: Dirección MAC de la máquina receptora. Este campo está a cero en las solicitudes, ya que corresponde a la información buscada.
- **Target dirección de protocolo (TPA)**: Dirección IP de la máquina receptora.

3.5.2. Vulnerabilidades del protocolo ARP

ARP [24] es un protocolo que no proporciona integridad ni autenticidad de origen, por tanto, se pueden enviar mensajes falsos para insertar entradas maliciosas en la cache ARP, dando lugar a ataques como ARP spoofing o ARP cache poisoning.

3.5.3. Posibles ataques

Teniendo en cuenta las vulnerabilidades de dicho protocolo los posibles ataques [18] a los que está expuesto podrían ser:

- Envenenamiento de cache, por medio de mensajes ARP falsos enviados por el atacante en una LAN.
- Man in the Middle, el cual tiene que ver con un ataque ARP Spoofing, en este tipo de ataques, se envían paquetes ARP falsos, con el fin de infiltrarse en la comunicación entre dos sistemas con el objetivo de espiar o manipular el tráfico de datos, mediante la asociación de la dirección MAC del atacante con la dirección IP de la víctima.
- Denegación de servicio (DoS), llegando a impedir el normal funcionamiento de la red, mediante la asociación de la dirección IP víctima a una dirección MAC inexistente.

Como medidas de prevención ante posibles ataques podrían ser, tablas ARP estáticas, seguridad MAC, escucha DHCP o incluso un IDS.

3.6. Protocolo ICMP

El protocolo ICMP [25], Internet Control Messaging Protocol (Protocolo de Mensajes de Control de Internet), es un protocolo de la capa de red que sirve para informar de sucesos que han ocurrido en la red. Permite a los nodos intermedios enviar mensajes de control a los equipos que enviaron la información.

3.6.1. Funciones básicas del protocolo ICMP

Solo informa de errores a la máquina origen del datagrama, ya que es la única fuente fiable conocida, pues el destino puede existir o estar apagado en ciertos momentos; de este modo se evita sobrecargar las redes con tráfico innecesario. El protocolo ICMP sólo informa, no corrige errores.

Los mensajes ICMP que se pueden enviar pueden ser:

- **Confirmación de host** (Echo Request / Echo Response) Los mensajes de eco de ICMP se pueden utilizar para determinar si un nodo está funcionando. El equipo local envía una petición de eco (Echo Request) de ICMP a un nodo. El nodo que recibe el mensaje de eco responde mediante la respuesta de eco (Echo Response) de ICMP. Estos mensajes de eco son la base del funcionamiento de los comandos ping y traceroute.
- **Destino inalcanzable** (Destination Unreachable): Se puede usar el mensaje de destino inalcanzable de ICMP para notificar a un equipo que el destino o servicio es inalcanzable. Cuando un equipo recibe un paquete que no puede encaminar, puede enviar un mensaje ICMP de destino inalcanzable al equipo que origina el paquete. Este mensaje de destino inalcanzable tendrá asociado un código que indica el motivo por el cual el paquete no pudo ser entregado. Entre los códigos de destino inalcanzable se encuentran:
    - 0 -> red inalcanzable.
    - 1 -> host inalcanzable.
    - 2 -> protocolo inalcanzable.
    - 3 -> puerto inalcanzable.
- **Tiempo excedido** (Time Exceeded): En la cabecera IP existe el campo TTL (Time To Live) que permite limitar el número de encaminadores que atravesará un paquete para llegar a su destino. Cuando un router recibe un paquete, disminuye el valor del campo TTL del paquete en una unidad. Si el nuevo valor del campo TTL es cero, el router descarta el paquete. El router puede enviar un mensaje ICMP de tiempo excedido para indicar a la máquina de origen que no se ha podido enviar un paquete debido a que el campo TTL del paquete ha expirado.
- **Redireccionamiento de ruta** (Redirect): Un encaminador puede usar un mensaje de redireccionamiento de ICMP para notificar a los equipos de una red que existe una mejor ruta disponible para alcanzar un determinado destino. Cuando un router recibe un paquete para el cual tiene una ruta cuyo próximo

salto está conectado a la misma interfaz por la que ha recibido el paquete, el router puede enviar un mensaje de redireccionamiento de ICMP al equipo origen informándole acerca del próximo salto en una ruta de la tabla de encaminamiento.

- **Disminución de velocidad en origen** (Source Quench): Este mensaje es útil para implementar un mecanismo básico de control de flujo. El mensaje de disminución de velocidad en origen de ICMP puede usarse para informar al origen que deje de enviar paquetes por un tiempo. Cuando un equipo recibe un mensaje de disminución de velocidad en origen de ICMP, lo informa a la capa de transporte. El host de origen puede utilizar el mecanismo de control de flujo de TCP para adaptar la transmisión.

3.6.2. Formato del datagrama ICMP

En la figura 8 se muestra el formato del datagrama ICMP [26]:

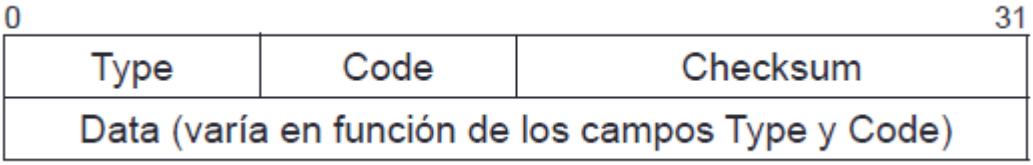


Figura 8. Datagrama ICMP.

- **Tipo:** Determina el tipo de mensaje ICMP.
- **Código:** Especifica un subtipo dentro de un tipo dado, en caso de haberlo**.**
- **Checksum:** Mecanismo de comprobación de errores. Calculado a partir de la cabecera ICMP más el campo datos, con un valor de cero para este campo. El algoritmo de suma de comprobación viene especificado por los estándares (RFC 1071).
- **Resto de la cabecera:** Puede variar en función del tipo y código del mensaje ICMP. Posee dos campos importantes para el análisis, como son el id y la secuencia ICMP.

Tipo ICMP	Nombre del tipo	Código	Descripción
1	Echo Reply		Respuesta a un ping de red para comprobar la accesibilidad.
3	Destination Unreachable	0–15	Mensaje ICMP que informa acerca de que el mensaje no ha podido ser entregado.
5	Redirect Message	0–3	Mensaje en el que se indica que un paquete debe ser direccionado para la red indicada (0), para la máquina escogida (1), para el servicio especificado y para la

			red (2) o para la máquina y el servicio especificados (3).
8	Echo Request		Ping de red.
9	Router Advertisement		Lo utilizan los encaminadores para comunicarse con sus encaminadores vecinos y poder construir tablas de encaminamiento .
11	Time Exceeded	0 o 1	Informe de estado que o bien indica que el tiempo de vida (Time to Live, TTL) de un paquete (0) o el tiempo de espera para el ensamblaje de paquetes IP (1) ha expirado.
13	Timestamp		Dota al paquete IP de una marca de tiempo que se corresponde con el momento del envío y que es de utilidad para la sincronización de dos ordenadores.
14	Timestamp Reply		Mensaje de respuesta a una petición de marca de tiempo enviado por el destinatario tras la recepción de esta.
30	Traceroute		Se utiliza para el seguimiento de la ruta de un paquete de datos en la red.

3.6.3. Vulnerabilidades del protocolo

Es un protocolo [27] que no proporciona integridad ni autenticidad de origen, por tanto, se pueden generar mensajes de control falsos, como ICMP redirect, ICMP Time Exceeded. También tenemos que tener en cuenta que las respuestas ICMP consumen recursos de nuestros sistemas, por tanto, consumen ancho de banda entrante y saliente, al no proporcionar autenticidad se puede dar la posibilidad de responder a peticiones maliciosas.

3.6.4. Posibles ataques

Teniendo en cuenta las vulnerabilidades de este protocolo los posibles ataques [18] a los que está expuesto podrían ser:

- ICMP Ping Flooding, es decir, un atacante envía multitud de paquetes ICMP Echo Request a la víctima, usando la opción –f del comando ping, el atacante consigue enviar mayor número de paquetes sin esperar respuesta.
- Denegación de servicio (DoS), inundando la red de mensajes e impidiendo el normal funcionamiento de la misma, tiene éxito si el atacante tiene un mayor ancho de banda que la víctima, el atacante espera a que la víctima responda con mensajes ICMP Echo Reply, consumiendo ancho de banda saliente y entrante.

Como posibles medidas para hacer frente a este tipo de ataques pueden ser, el uso de IP Tables, filtrando así el acceso a nuestra solo a tráfico deseado, un buen Firewall, limitar la capacidad de ancho de banda de nuestra red, el uso de un IDS.

Capítulo 4
Implementación

En este capítulo se describen los pasos y procedimientos que se han seguido para la realización de este proyecto, así como la estructura que tienen las funciones del programa y cómo se han implementado.

La implementación de nuestro proyecto se puede calificar de estática, ya que, en primer lugar, se genera tráfico entrante en la red, que es capturado con la herramienta Wireshark. Cuando se obtiene una traza de tráfico lo suficientemente extensa para ser procesada, esos datos se exportan en formato byte y se almacenan en un archivo. Posteriormente, el IDS desarrollado lee este archivo, obtiene los paquetes, los descompone según los protocolos que utilice y los analiza comparando los campos del paquete con las reglas preestablecidas. Por último, el sistema genera una alerta si se produce una coincidencia como resultado de esta comparación.

En las siguientes secciones se explica cómo se ha seleccionado el componente hardware utilizado, cómo se ha realizado la instalación del sistema operativo y configuración del entorno, cómo se ha desarrollado el código del programa utilizado y, la definición de las reglas que determinan un posible ataque. Finalmente, se describe el funcionamiento e implementación del sistema IDS propuesto.

4.1. Elección del componente hardware

Se han analizado diferentes plataformas hardware para seleccionar aquella que más se ajuste a las necesidades requeridas en la implementación de este proyecto.

La principal característica analizada ha sido la capacidad de procesamiento. Se necesita un dispositivo que pueda procesar a una velocidad adecuada la captura, la descomposición y el análisis de todo el tráfico de la red a la que esté conectado.

Para realizar este estudio, se seleccionaron tres dispositivos que se ajustarban a los requerimientos exigidos. Los componentes mencionados son:

- Odroid XU4 [28]
- Arduino Uno [29]
- Raspberry Pi 3 B+ [30]

A continuación, se procede a describir las características más importantes de cada uno de ellos:

4.1.1. Odroid XU4

Odroid forma parte de una familia de ordenadores monoprocesador creados por *HandKernel*. Su principal característica es que es un componente que funciona con software libre.

En cuanto a sistemas operativos, es compatible con cualquier distribución de *Linux* y también *Android*. Teóricamente, también es compatible con *Windows 10*, pero no está confirmado.

Las características de Odroid XU4 son:

- Procesador Samsung Exynos 5224 de 8 núcleos.
- Frecuencia de reloj de 2GHz.
- GPU Mali-T628 MP6.
- RAM de 2 GB del tipo DDR3.
- Almacenamiento Flash integrado en la placa con opción de incorporar una tarjeta de almacenamiento externo MicroSD.
- Dos puertos USB 3.0 y un puerto USB 2.0.
- Tarjeta de red Gigabit Ethernet 10/100/1000.
- Salida HDMI.
- Botón de encendido de la placa.
- Precio: alrededor de 100€

4.1.2. Arduino Uno

Arduino es una plataforma de desarrollo de código abierto, basada en hardware y software libre y creada para el desarrollo de dispositivos digitales y electrónicos que puedan detectar y controlar objetos del mundo real. Se encarga de facilitar el uso de la electrónica y programación de sistemas.

Las placas de Arduino cuentan con diversos microprocesadores y microcontroladores. En cuanto al sistema operativo en multiplataforma es compatible con *Windows*, *MacOs* y *GNU/Linux*.

Las características de Arduino Uno son:

- Microcontrolador ATmega328.
- Velocidad de reloj de 16 MHz.
- SRAM de 2 KB.
- Memoria Flash de 32 KB.
- EEPROM de 1 KB.
- 14 pines de entrada/salida digital y 6 salidas PWM.
- Voltaje operativo de 5V y entrada de 7-12V.
- Precio: Alrededor de 20€.

4.1.3. Raspberry Pi 3 B+

Raspberry Pi es un ordenador de placa única, también llamado ordenador de placa simple, de bajo coste desarrollado en el Reino Unido por la empresa Raspberry Pi. Tiene un software de código abierto y su sistema operativo oficial es una versión adaptada de Debian llamada *Raspbian*. A pesar de ser multiplataforma, sus dimensiones de placa son muy reducidas, alrededor de 85 x 54 mm.

Se selecciona el modelo B+, ya que, a diferencia de otros modelos, posee conexión Ethernet, Wifi y dos puertos USB.

Las características de Raspberry Pi 3 modelo B+ son:

- Procesador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC.
- Frecuencia de reloj de 1,4 GHz.

- GPU una VideoCore IV de 400 MHz.
- SDRAM de 1GB.
- Conectividad de red Wifi de tipo IEEE 802.11.b/g/n/ac y Bluetooth 4.2, BLE.
- Capacidad de memoria de almacenamiento externa MicroSD de al menos 4GB.
- Salida HDMI, salida RCA, entrada tipo Jack y video compuesto.
- 40 pines GPIO y 4 puertos USB 2.0.
- Alimentación tipo micro USB.
- Precio: 36,50€.

4.1.4. Conclusiones

Arduino no es un dispositivo que pueda encajar en nuestro trabajo, ya que, aunque se puede utilizar para la implementación de multitud de pequeños sistemas, sus especificaciones técnicas no son suficientes para hacer frente a la envergadura de este proyecto, ya que la memoria RAM no es lo bastante grande, la velocidad de reloj no es la idónea para procesar todos los datos y el espacio de almacenamiento es insuficiente para albergar el sistema operativo.

Por el contrario, ODROID XU4 cumple con creces las expectativas para implementar el IDS. Además, ha sido utilizado previamente con éxito en un sistema similar. Sin embargo, su precio es elevado.

Por último, Raspberry Pi 3 modelo B+ tiene las características técnicas requeridas: frecuencia de reloj, memoria RAM, posibilidad de albergar almacenamiento externo, gran capacidad de conectividad y compatibilidad del sistema operativo con otros equipos y programas. El producto también tiene un precio asequible. Por todas estas razones Raspberry Pi 3 modelo B+ parece la mejor opción para ser utilizado como el dispositivo hardware que albergará el IDS que vamos a implementar en este Trabajo de Fin de Grado.

4.2. Instalación del Sistema Operativo

Disponemos de un equipo que consta de una Raspberry Pi 3 modelo B+ y una tarjeta microSD con una capacidad de almacenamiento de 8Gb en el que es necesario instalar un sistema operativo. Nos decantamos por Raspbian, un sistema operativo que consiste en una distribución de la versión 9 de Debian [31] que dispone de un kernel de Linux [32], versión 4.14, de tipo monolítico.

El primer paso en su instalación es insertar la tarjeta externa microSD en un ordenador, en nuestro caso un MacBook Pro, y después utilizar una herramienta nativa de la que dispone este ordenador, que permite formatear tarjetas de memorias externa, discos, etc. Se selecciona la tarjeta de memoria en cuestión y se formatea con un formato NTFS. Una vez formateada, está lista para ser usada.

Posteriormente, desde la página oficial de Raspberry Pi [33], se descarga la última versión de Raspbian [34] para ser utilizada como el sistema operativo de nuestro dispositivo hardware, en este caso se selecciona la versión Debian Stretch with Raspberry Pi Desktop.

Una vez descargada la imagen .iso es necesario instalar esta imagen en la tarjeta de memoria. Tras investigar sobre posibles programas compatibles con el sistema operativo MacOS Mojave, que nos permitiesen hacer esto, decidimos utilizar BalenaEtcher [35], una aplicación para montar la imagen .iso en la tarjeta con una interfaz muy sencilla, que indica de forma guiada los pasos a seguir.

El proceso finaliza con la instalación del sistema operativo en la Raspberry Pi. Para ello, se inserta la tarjeta de almacenamiento, se conecta el ratón, el teclado y la pantalla y se enciende el dispositivo, conectándolo la fuente de alimentación. La primera vez que se usa el sistema operativo, se inicia una instalación gráfica y guiada, donde se configura el idioma, teclado, wifi, el nombre de usuario y contraseña (en nuestro caso, no hemos rellenado el campo contraseña). Tras completar la configuración, automáticamente se termina la instalación.

4.3. Instalación de programas usados

A continuación, se describen los programas que hemos instalado y su utilidad en este trabajo.

- **Wireshark** [36]: Es un analizador de protocolos open-source. Su principal objetivo es el análisis de tráfico, pero además es una excelente aplicación para el estudio de las comunicaciones y para la resolución de problemas de red. Wireshark implementa una amplia gama de filtros que facilitan la definición de criterios de búsqueda para los protocolos soportados actualmente y todo ello por medio de una interfaz sencilla e intuitiva que permite desglosar por capas cada uno de los paquetes capturados.

  Gracias a que Wireshark comprende la estructura de los protocolos, podemos visualizar los campos de cada una de las cabeceras y capas que componen los paquetes monitorizados, haciendo así más fácil el análisis del tráfico en la red.

  La instalación de Wireshark en nuestro componente hardware con sistema operativo Raspbian, se ha realizado desde un terminal utilizando una serie de comandos básicos:

  ```
  $sudo apt-get update
  $sudo apt-get install wireshark
  ```

  Con esto ya tendríamos nuestra herramienta lista para funcionar y escuchar en cualquier interfaz de red. En nuestro caso, siempre escucharemos por la interfaz de red 0 (eth0).

- **Jupyter-Notebook** [37]: Es un entorno interactivo web de ejecución de código, que en este trabajo es usado para desarrollar un programa capaz de tratar los datos obtenidos y exportados desde la herramienta Wireshark.

  Jupyter-notebook se lanza desde un terminal de nuestro sistema operativo Raspbian. Al escribir el comando **jupyter-notebook** automáticamente se abre una pestaña en el navegador en la que se muestra el directorio donde están los archivos a tratar.

  Para instalar esta herramienta ha sido necesario descargar los componentes de python3: matplotlib, scipy y pip, ejecutando los siguientes comandos:

$**sudo apt-get update**

$**sudo apt-get install python3-matplotlib, python3-scipy**

$**sudo install --upgrade pip**

Una vez instaladas las anteriores herramientas, es necesario reiniciar el sistema. Posteriormente, para finalizar la ejecución de la herramienta se ejecuta:

$**sudo pip install jupyter**

4.4. Tratamiento de los mensajes capturados

En este apartado se explica el funcionamiento del programa que hemos desarrollado en Python [38] para implementar un NIDS. Este IDS se encarga de leer y descomponer los paquetes capturados en un archivo. En particular, se centra en la parte de las cabeceras de los diferentes protocolos de internet, para así poder analizar en profundidad toda la información transmitida. El análisis está diseñado para tratar cinco protocolos de red: TCP, IP, ARP, ICMP, y UDP.

Para poder realizar correctamente esta tarea fue necesario: capturar los paquetes, utilizando las funcionalidades de Wireshark, y descomponer y analizar esos paquetes mediante el IDS, desarrollado en Python en este TFG.

Los paquetes generados por Wireshark contienen, además del paquete que es transmitido, información acerca del número de bytes que ocupa y el número de paquete generado por este programa. La clase **paquete** tiene atributos como el identificador, donde se guarda el número que proporciona Wireshark, el número de bytes, el tiempo de llegada, direcciones MAC de origen y destino y cada uno de los campos de los protocolos que se tratan en este proyecto.

Es necesario indicar que se ha creado una clase, llamada **paquete**, con su correspondiente constructor, que contiene los elementos que se obtienen al descomponer el paquete.

Además, para poder realizar todo este tratamiento, se ha diseñado la estructura de datos y funciones que se muestran en la figura 9. Según esta estructura, una vez leído el paquete, el contenido del campo datos y de los distintos campos de la cabecera, se almacena en una lista llamada **matriz**. Para recorrer esta lista se utiliza un contador, (llamado **contador)** que se incrementa con cada campo que se asigna a una variable, es decir, las posiciones que ocupe un campo determinado en **matriz** son las unidades que se sumarán al valor de **contador**.

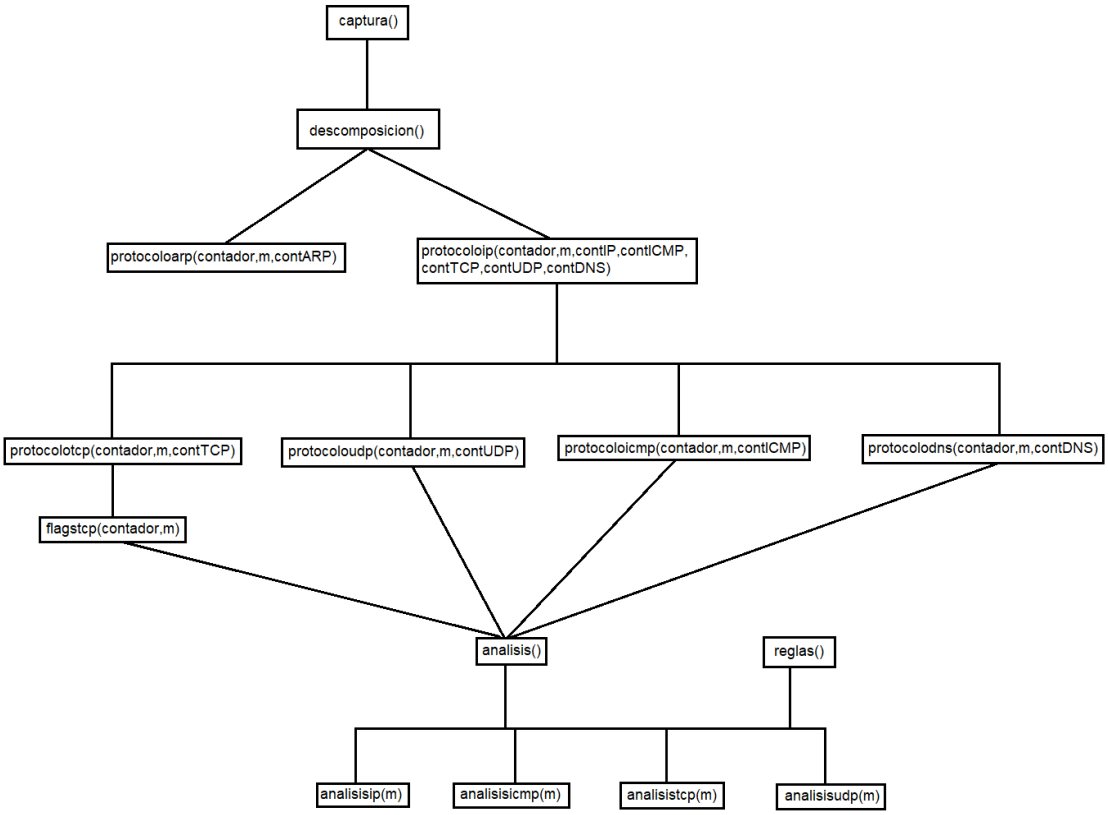


Figura 9. Esquema del programa.

4.4.1. Obtención de paquetes

Esta es una de las principales partes de un IDS. Consiste en la captura de todos los paquetes que el dispositivo detecta en la red y que, después, serán tratados.

Se usa la herramienta Wireshark para capturar los paquetes y exportarlos en formato byte para poder ser almacenados en un archivo que leerá el programa IDS. Este archivo será abierto por el programa, que leerá el contenido línea a línea.

Los paquetes capturados están organizados según la estructura de datos explicada en el apartado anterior. El IDS ha sido programado para que sea capaz de diferenciar los distintos paquetes almacenados en el archivo y almacene cada uno en una lista.

Durante la lectura de los paquetes, en la función **captura ()**, se separan las partes proporcionadas por Wireshark, como el identificador del paquete, el número de bytes que ocupa y el contenido del paquete, este último contiene los bytes que son utilizados para el análisis, y es guardado en una lista para facilitar su posterior tratamiento. También, se realiza una captura de los segundos que el programa tarda en realizar este proceso, ya que posteriormente este tiempo será utilizado en el análisis. La figura 10 contiene el esquema de esta función.

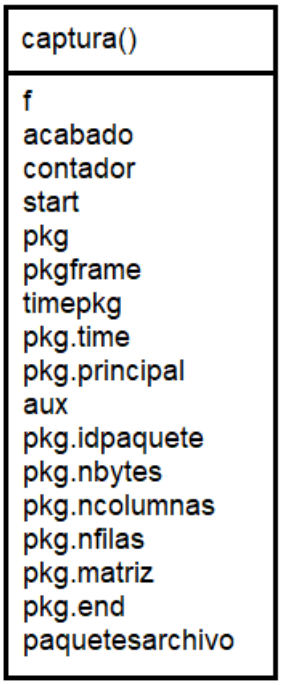


Figura 10. Función captura.

4.4.2. Descomposición de paquetes

Los paquetes capturados se descomponen en la función **descomposicion ()** según la estructura explicada anteriormente. Mediante **contador**, se accede a las posiciones de **matriz** y se van tratando y asignando las partes correspondientes. En primer lugar, se asignan las primeras variables de la clase **paquete** con las direcciones MAC origen y destino. A continuación, se establece el campo **tipo**, en el que se especifica el protocolo al que pertenecen los datos contenidos en el paquete. Dependiendo del tipo de protocolo se aplica la función o funciones de análisis correspondiente. La figura 11 contiene el esquema de la función.

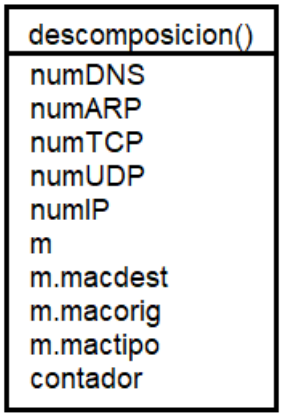


Figura 11. Función descomposicion.

Al entrar en la función **descomposición ()**, se activa el flag indicador del protocolo que se está usando y se incrementa en una unidad el contador asociado a dicho protocolo. Este contador se usará posteriormente, en el análisis, para tener un registro del número de paquetes registrados de cada protocolo.

4.4.2.1. Datagrama ARP

Los datagramas ARP se tratan con la función **protocoloarp (contador, m, numARP)**. En esta función el paquete se descompone en los siguientes campos, que son guardados en variables: tipo de hardware usado, tipo de protocolo ARP utilizado, tamaño del hardware, tamaño del protocolo, código de operación, dirección MAC origen, dirección IP origen, dirección MAC objetivo, dirección IP objetivo y el relleno. Estos campos se obtienen de las posiciones correspondientes de la lista **matriz**. Una vez terminada la descomposición, se actualiza el contador, sumando la longitud de este datagrama. La figura 12 contiene el esquema de la función **protocoloarp**.

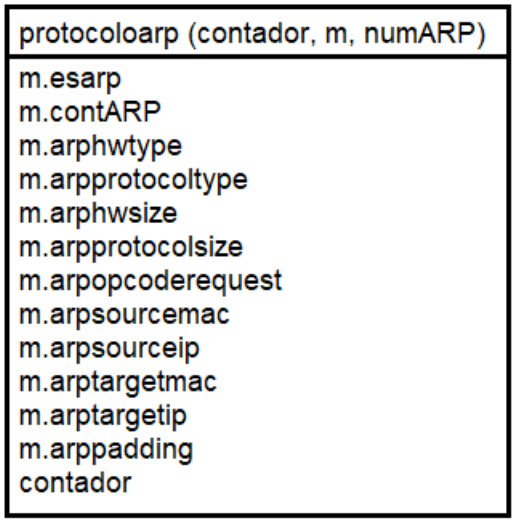


Figura 12. Función protocoloarp.

El protocolo ARP se ha incluido con la idea de implementar un IDS lo más completo posible. Sin embargo, en la versión desarrollada en este trabajo no se han recogido reglas para analizar los paquetes que contengan este tipo de datagramas, ya que nos hemos centrado en el análisis de otros protocolos vinculados a las reglas Snort. Los ataques relacionados con el protocolo ARP son tratados por Snort mediante preprocesadores y no se basan en el análisis de reglas. La detección de este tipo de ataques queda pendiente de desarrollar como trabajo futuro.

4.4.2.2. Datagrama IP

A la hora de tratar los datagramas IP es necesario tener en cuenta las dos versiones que soporta el protocolo, tanto IPv4 como IPv6, cada una con sus formatos y tamaños establecidos. La figura 4.4.2.2. contiene el esquema de la función **protocoloip**

(contador, m, numIP, numICMP, numTCP, numUDP, numDNS), que descompone este tipo de datagramas. Tiene como parámetros: contador, matriz, y los contadores IP, ICMP, TCP, UDP y DNS. Esta función detecta cuál es la versión del protocolo IP contenida en el paquete y, dependiendo de la versión, se obtendrán distintos tipos de variables. La figura 13 contiene el esquema de la función.

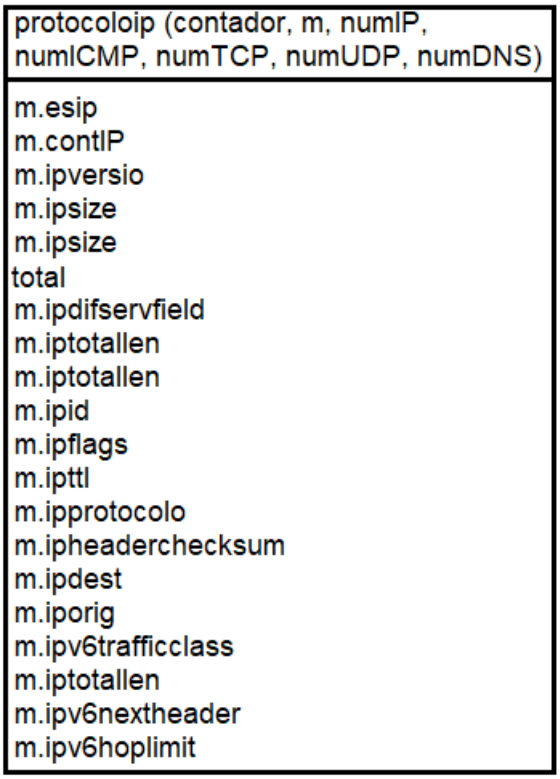


Figura 13. Función protocoloip

Si se trata de la versión 4, los campos tratados y almacenados en variables para su posterior análisis son: el tamaño de la cabecera del datagrama, el número de identificación IP, los flags que pueda tener activos, el tiempo de vida del paquete, el tipo de protocolo del mensaje contenido en el campo datos, tamaño total del datagrama IP, el checksum de la cabecera, la dirección IP origen y la dirección IP destino.

Para la versión 6, se usan los campos: clase de tráfico, próxima cabecera, el límite de saltos (hop limit), la dirección IP origen y la dirección IP destino. Estos campos se almacenan en variables para su posterior análisis.

La función implementada solo es capaz de descomponer datagramas IP versión 4. Dependiendo del protocolo contenido en el campo datos, desde ella se llamará a otras funciones de descomposición de datagramas, que se describen en las siguientes secciones.

Para el análisis de este protocolo se usa la función **analisisip(m)**, que compara las variables tiempo de vida, tamaño del datagrama y protocolo IP con el valor de estos atributos en las reglas Snort.

4.4.2.3. Datagrama ICMP

La función que trata los datagramas ICMP es **protocoloicmp (contador, m, numICMP)**. En este caso, los campos que son almacenados en variables son: tipo, código, checksum, id y secuencia. Los dos últimos campos son extraídos del campo datos ya que son necesarios para el análisis. La figura 14 contiene el esquema de la función.

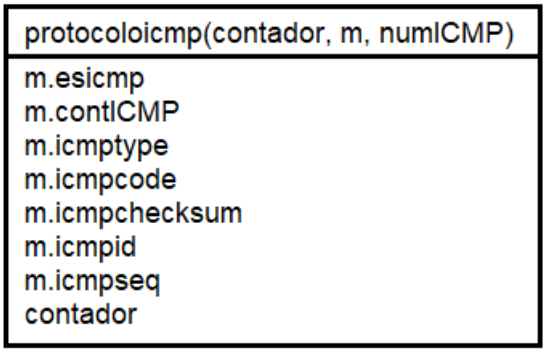


Figura 14. Función protocoloicmp.

Todas las variables anteriores son importantes para realizar el análisis, ya que las utiliza la función **analisisicmp(m)** al realizar las comparaciones con las reglas Snort. También se incluyen en esta función otros campos calculados anteriormente como el tiempo de vida y el tamaño de datagrama IP.

4.4.2.4. Datagrama TCP

Los datagramas TCP se descomponen con la función **protocolotcp (contador, m, numTCP)**. Esta función se encarga de obtener todos los campos del datagrama, como los puertos de origen y destino, el número de secuencia, la longitud de cabecera del datagrama y los flags, que son almacenados en variables. La figura 15 contiene el esquema de la función **protocolotcp**.

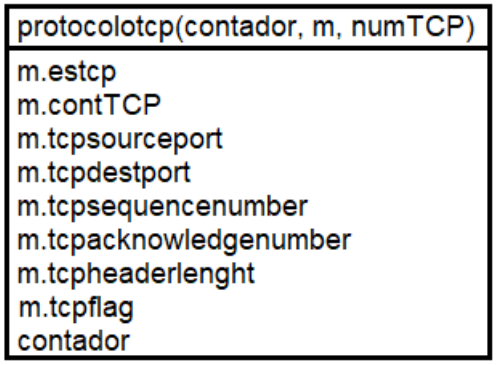


Figura 15. Función protocolotcp.

Es necesario indicar que se ha tenido que crear otra función llamada **flagstcp (contador, m)**, en la que se identifican puedan diferenciar los flags TCP que están activos, ya que, dependiendo de estos flags, el datagrama puede tener unos campos u otros, como el tamaño de ventana, checksum, el sello de tiempo, la opción sack y la opción no operación, entre otros. Todos ellos se almacenan como variables. La figura 16 muestra el esquema de esta función.

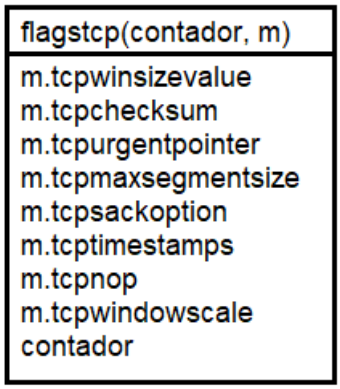


Figura 16. Función flagstcp.

Las variables más importantes son los puertos, los flags y el número de paquetes TCP, que son usadas en la función **analisistcp (m)** para compararlas con el contenido de las reglas destinadas a detectar ataques asociados al protocolo TCP.

4.4.2.5. Datagrama UDP

Los datagramas UDP son tratados con la función **protocoloudp (contador, m, numUDP)**. Los campos que se guardan en variables son: el puerto origen, puerto destino, la longitud del datagrama y el checksum (suma de comprobación). La figura 17 contiene el esquema de la función.

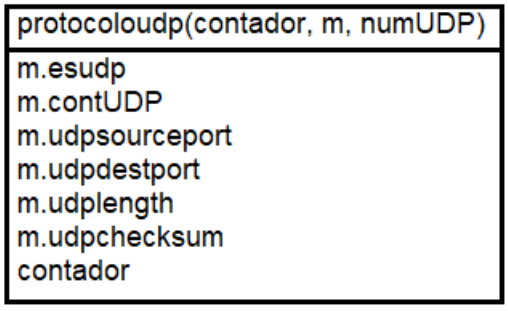


Figura 17. Función protocoloudp.

Respecto al análisis, las variables a tener en cuenta son los puertos origen y destino, el número límite de paquetes con datagrama UDP detectados y la hora en que el paquete fue registrado. La función que realiza el análisis de este tipo de datagramas es **analisisudp (m).**

4.4.2.6. Mensaje DNS

La función que descompone los mensajes DNS es **protocolodns (contador, m, numDNS)**. Los campos que se descomponen en variables son: el id del mensaje, los flags DNS, contador de preguntas, contador de respuestas, la autoridad, el campo adicional, el nombre de consulta, el tipo de consulta y la clase de consulta. La figura 18 contiene el esquema de la función.

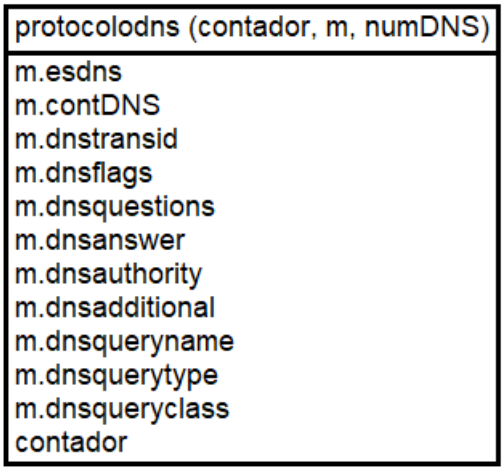


Figura 18. Función protocolodns.

Respecto al análisis, a pesar de que Snort tiene reglas para detectar ataques DNS, por cuestiones de tiempo, estos no han sido estudiados en este trabajo. Se ha dejado preparado el IDS para que en un trabajo futuro la incorporación de la identificación de este tipo de mensajes sea inmediata.

4.5. Reglas Snort

Después de la etapa de descomposición, se procede a la lectura de todas las reglas predefinidas mediante la función **reglas ()**. Esta función lee el contenido de los archivos de reglas proporcionados (en las pruebas realizadas hemos usado el archivo "local.rules") y guarda las reglas en listas, en función del protocolo al que va dirigido el ataque que detectan. Existen cuatro protocolos en las reglas que se han utilizado: IP, ICMP, TCP y UDP.

En el programa IDS, las listas aportadas para almacenar estas reglas tienen los nombres de **tcprules, udprules, iprules** e **icmprules**. Una vez terminada la lectura, se procede al análisis de paquetes. La figura 19 contiene el esquema de la función.

Figura 19. Función reglas.

Las reglas utilizadas en este proyecto han sido utilizadas con el mismo formato que las reglas que proporciona Snort. El formato [39] de estas consta de una cabecera y del campo opciones. En la cabecera, se especifica la acción que realizan (alert, log, drop o sdrop), el protocolo usado, las direcciones IP, los puertos y el sentido de la regla (unidireccional o bidireccional). En el campo opciones, se proporciona información de la regla con **msg** (alerta que se emite), **sid** (identificación de regla) y **rev** (número de revisión), se busca en las cabeceras de los protocolos con **ipopts, icmp_id, icmp_seq** y **flags**.

A continuación, se muestran algunas de las reglas utilizadas:

- *alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; sid:10000001; rev:001; classtype:icmp-event;)*
- *alert icmp any any -> 192.168.1.1/24 any (msg:"PING Flooding"; ttl:100; dsize: > 200; sid:10000001; rev:1;)*
- *alert icmp any any -> $HOME_NET any (msg:"Possible Nmap ping sweep"; sid:10000002; rev:001; classtype:web-application-attack; dsize:0;)*
- *alert tcp any any -> $HOME_NET any (msg:"TCP Port Scanning"; sid:10000003; rev:001; classtype:web-application-attack; detection_filter:track by_src, count 30, seconds 60;)*
- *alert tcp any any -> $HOME_NET any (msg:"Nmap XMAS Tree Scan"; sid:10000005; rev:001; flags:FPU;classtype:web-application-attack;)*
- *alert udp any any -> 192.168.1.3 any (msg:"UDP flood attack detected"; threshold: type threshold, track by_dst, count 10, seconds 60; sid: 5000003; rev:1;)*

4.6. Análisis y generación de alertas

Una vez terminada la descomposición del paquete y la lectura de reglas se procede al análisis, mediante la función **análisis ()**. Esta función recorre la lista donde están todos los paquetes y llama a las funciones correspondientes para analizar cada uno de ellos, dependiendo del protocolo que se use. El análisis consiste en la comparación de los campos de un datagrama con las premisas establecidas en cada una de las reglas

predefinidas, por lo que es necesario comparar cada datagrama con cada regla. La figura 20 contiene el esquema de la función.

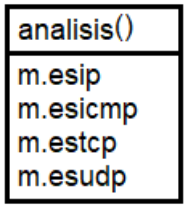


Figura 20. Función analisis.

En todas las funciones de análisis, el parámetro de entrada **m** es el paquete a analizar y se recorre en un bucle para comparar su contenido con todas las reglas del protocolo. En cada iteración del bucle, se descompone la regla correspondiente y se extraen sus atributos, como el mensaje de alerta, que se guarda en la variable **msg.** Además, independientemente del protocolo, se almacenan las variables **dirorigen, puertoorigen, dirdest y puertodest,** que se corresponden con las direcciones y puertos de origen y de destino.

Para el análisis de datagramas IP, las variables obtenidas de las reglas son **ttlip** (tiempo de vida), **dsizeip** (tamaño del datagrama) y **protocoloip, listaprotocolos** y **operadorprotip** (correspondiente al protocolo o protocolos usados en el campo datos del datagrama y al operador en caso de que exista). Posteriormente, se comparan estas variables con las que hay en el paquete y, si coinciden, se envía un mensaje de alerta. La figura 21 muestra la estructura de la función **analisisip(m)**.

Figura 21. Función analisisip.

En el análisis ICMP se toman las variables **ttl** (tiempo de vida), **dsize** y **dsizeoperator** (tamaño del paquete y operador, en caso de existir), **itype** (tipo ICMP), **icode** e **icodeoperator** (código ICMP y operador, en caso de existir), **idicmp** (identificador ICMP) y **seqicmp** (secuencia ICMP). De igual manera, se genera una alerta cuando

estas variables coinciden con los valores correspondientes del paquete. La figura 22 muestra la estructura de la función **analisisicmp(m)**.

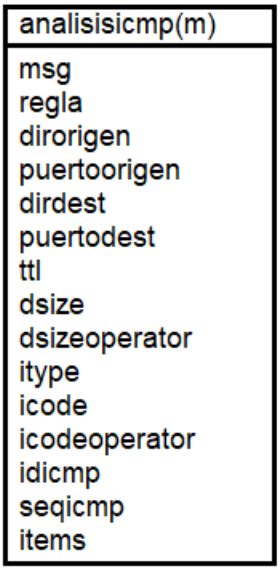


Figura 22. Función analisisicmp.

Respecto al análisis TCP, se aportan las variables **flags, count** y **seg** para los flags TCP, el número de paquetes límite y número de segundos mínimo para impedir ataques SYN flood. Esta función cuenta con una serie de booleanos que tienen en cuenta si hay operadores en los puertos y direcciones, si hay varios puertos y si hay coincidencia en las direcciones de origen y destino. Además, se usan otros booleanos para alertar sobre la posibilidad de que se esté produciendo un ataque TCP SYN flood. La figura 23 muestra la estructura de la función **analisistcp(m)**.

Figura 23. Función analisistcp.

En el análisis UDP, existen las variables **count** y **seg** para establecer un mínimo de paquetes recibidos dentro de un determinado periodo de tiempo para disparar una alerta sobre ataques UDP flood. También posee unos booleanos para alertar sobre este último ataque. La figura 24 muestra la estructura de la función **analisisudp(m)**.

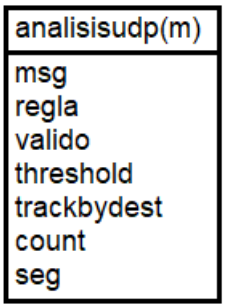


Figura 24. Función analisisudp.

Capítulo 5

Resultados

Una vez explicada la implementación, se procede a presentar los resultados de este proyecto. El objetivo de este capítulo es mostrar el correcto funcionamiento de nuestro IDS.

Esta parte del proyecto la hemos realizado en dos fases:

- Pruebas en un entorno virtual: Pruebas realizadas en máquinas virtuales (MVs) simulando el dispositivo hardware, para prevenir posibles complicaciones y daños en el dispositivo. Explicaremos cómo hemos simulado los ataques y las posibles alertas generadas.
- Pruebas en un entorno físico: Se han llevado a cabo de la misma forma que las realizadas en un entorno virtual, pero haciendo uso del dispositivo hardware.

El código desarrollado está expuesto, junto con las reglas usadas, en un repositorio público de Github [40], al que se puede acceder a través del siguiente enlace:

https://github.com/rodrigolagartera/TFG-Python

Estas dos fases de pruebas serán explicadas a continuación, exponiendo los puntos más relevantes de cada una de ellas.

5.1. Pruebas en un entorno virtual

Para poder llevar a cabo el proceso de experimentación y análisis de resultados, se realizan las pruebas pertinentes en máquinas virtuales que simulen el comportamiento del dispositivo hardware empleado.

En esta sección se describen los pasos realizados para llevar a cabo las pruebas, como la instalación y configuración de las máquinas virtuales, el tipo de tráfico generado en la red virtual, la captura de este tráfico y las pruebas realizadas con Snort y con nuestro IDS.

5.1.1. Creación e instalación de la máquina virtual Raspbian

Raspbian es la distribución de Debian específicamente diseñada para la Raspberry Pi y utilizada en este proyecto.

Para esta parte del proyecto, se ha utilizado Virtual Box [41], un programa diseñado para la creación y emulación de sistemas operativos a partir de una imagen. Hemos utilizado la versión 6.0, la más actual.

Para poder recrear el sistema operativo Raspbian en Virtual Box, fue necesario descargar una imagen .iso de Raspbian de la página web oficial de Raspberry. Fue

seleccionada la versión 4.9, que es una distribución de noviembre de 2018, la más actualizada y estable que se puede utilizar.

5.1.2. Configuración de la máquina virtual

En la opción de “Configuración”, se seleccionan las siguientes características:

- En el apartado **General**, en la pestaña “Avanzado”, se indica la carpeta donde se almacenarán las capturas de pantalla que realicemos (se guardarán en esa carpeta del ordenador físico). En las opciones de copiar en el portapapeles y arrastrar y soltar, se selecciona “Bidireccional”, con lo que conseguiremos copiar y pegar elementos desde el ordenador físico a la máquina virtual y viceversa, acciones que nos facilitarán en cierto modo el trabajo.
- En el apartado **Almacenamiento**, en la sección de “Controlador: IDE”, se selecciona la imagen iso descargada. Este será nuestro disco duro virtual en el cual almacenaremos todos los programas y funcionalidades de nuestro proyecto.
- En el apartado **Red**, habilitamos el Adaptador 1, seleccionamos la opción de “Adaptador puente” en “Conectado a”. En los demás adaptadores se configuran las redes internas que se han establecido para conectar las MVs. Este apartado es muy importante ya que debemos hacer una buena elección de configuración de red. Normalmente el primer adaptador, está conectado a la propia red del ordenador físico, los demás adaptadores se usan en nuestro caso para poder conectar la máquina virtual a distintas redes.

Una vez realizada toda la configuración y arrancada la máquina virtual, lo primero, que tenemos que hacer es instalar el sistema operativo Raspbian, para lo que elegimos una instalación gráfica. En la sección de partición del disco seleccionamos la opción “All files in one partition”, que nos permitirá tener todos los archivos en una única partición, una vez aplicados los cambios se procederá a la instalación del Grub de arranque.

Por último, la instalación se hará de forma automática y una vez terminada aparecerá la interfaz de Raspbian, en la que podemos usar un asistente de configuración del sistema con ajustes básicos. Para terminar, el sistema operativo se actualiza a la última versión.

5.1.3. Arquitectura del entorno virtual

La arquitectura usada para recrear el modelo se basa en una topología de red multipunto, que consta de tres máquinas conectadas mediante un cable, todas ellas dentro de la misma red (ver figura 25). En primer lugar, hay una máquina atacante (Kalilinux) que será la encargada de generar tráfico malicioso y enviarlo a la máquina Víctima. Otra hará el papel de víctima, y, por último, la tercera máquina tendrá desplegado el NIDS, por lo que escuchará todo el tráfico de la red y hará uso de un

conjunto de reglas predeterminadas, así como de reglas creadas por nosotros (como administradores de la red que se quiere proteger), para generar una alerta en el sistema cuando el host atacante envíe tráfico malicioso a la máquina víctima (el ordenador girado).

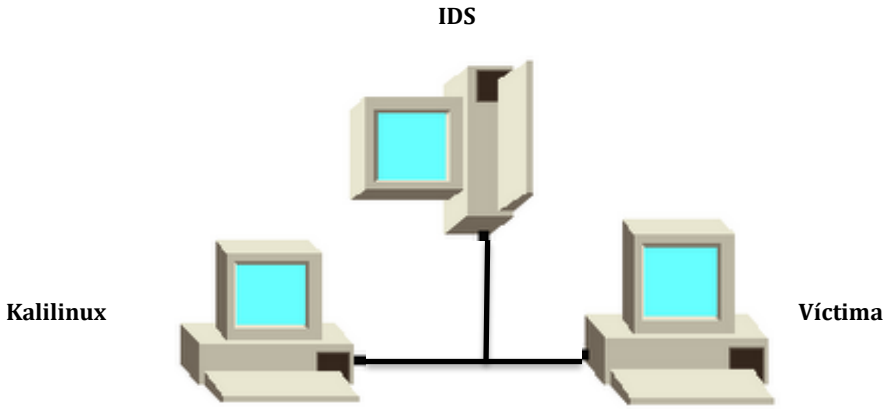


Figura 25. Arquitectura del entorno virtual

Las máquinas Víctima e IDS utilizan los sistemas operativos Debian y Raspbian, respectivamente. Las máquinas estarán conectadas a internet y a la red interna 192.168.1.0/24. Sus direcciones IP, serán, 192.168.1.2 y 192.1681.3, respectivamente.

Como máquina atacante o emisora, utilizamos una máquina virtual con sistema operativo Kali Linux [42], conectada a la misma red y a internet y con dirección IP 192.168.1.1.

En cada una de las máquinas se levantan las interfaces de red eth1, para poder conectar las máquinas a la red interna. Mediante un comando ping se comprueba la conexión entre ellas.

5.1.4. Máquina virtual de Raspbian

Para esa parte, ha sido necesaria la práctica 4.3 de la asignatura de Redes y Seguridad [43], en la que se hace uso del IDS Snort para alertar sobre la presencia de paquetes anómalos que pudieran tener relación con algún tipo de ataque en la red.

Para poder generar las alertas, que se mostrarán por pantalla, desde la máquina atacante realizamos una serie de ataques a la máquina Víctima que pueden ser detectados con el conjunto de reglas utilizado. Si se produce alguna coincidencia entre el contenido de los paquetes de red capturados y los atributos de las reglas se mostrará una alerta por la pantalla del sistema, alertando así, que la red está siendo atacada.

A continuación, se detalla cómo se ha generado ese tráfico malicioso, utilizando comandos y herramientas conocidas:

- Escaneo de puertos [43]: El escaneo de puertos se realiza para detectar qué puertos están abiertos buscando vulnerabilidades del sistema para poder atacarlo. El formato del comando es **nmap –"opción" "dir.ip destino"**. Cuando tenemos disponemos privilegios de súper usuario (root), podemos realizar también los siguientes escaneos, jugando con los flags de TCP, la opción –sN, escaneo Null, no fija ningún bit de la cabecera por tanto es cero, la opción –sF, conseguimos fijar el flag TCP FIN, por último la opción –sX, fijando los bits de los flags FIN, PSH URG.

  Realizando un escaneo de puertos con Nmap, se pueden observar todas las solicitudes de mensajes enviados que se encargan de comprobar uno a uno los puertos.

  Si se realiza un ataque de escaneo de puertos con el flag TCP FIN activo, nuestro IDS lo detectará y generará el siguiente mensaje:

```
Paquete 1 :
Alertas IP:
Alertas TCP: ! - TCP scan detected
192.168.1.1 -> 192.168.1.2
```

- Ataque ICMP Ping Flooding [43]: consiste en enviar una gran cantidad de paquetes ICMP Echo Request en un intervalo muy pequeño de tiempo consiguiendo así consumir muchos recursos de la máquina víctima y produciendo anormalidades en el correcto funcionamiento del sistema. Para reproducir este ataque desde la máquina atacante Kali Linux ejecutamos el comando **ping 192.168.1.2 –t –l 6550**, de este modo se envían 6550 bytes a la máquina víctima cada vez que se manda un ICMP Echo Request. Un ping normal tiene un tamaño de 32 bytes, el objetivo al aumentar el tamaño del paquete es colapsar la máquina y conseguir una denegación de servicio, en un determinado momento los mensajes no son respondidos debido a la gran cantidad de solicitudes que se envían.

  En un ataque ICMP Ping Flooding aparece un mensaje del tipo:

```
Paquete 1 :
Alertas IP:
Alertas ICMP: ICMP Ping Flooding detected
192.168.1.1 -> 192.168.1.2
```

- Ataques ICMP Ping Flooding [43] con IP spoofing: es similar al ataque anterior, pero utilizando direcciones IP origen aleatorias. Para reproducirlo vamos a usar la herramienta hping3. En la máquina Kali Linux (atacante) ejecutamos el comando **hping3 –icmp –flood –rand-source 192.168.1.2**,

donde la dirección IP 192.168.1.2 es la dirección de la máquina Víctima, conseguiremos así inundar de tráfico esta máquina.

En un ataque de este tipo el mensaje generado es muy parecido al anterior:

```
Paquete 1 :
Alertas IP: IP Spoofing detected
Alertas ICMP: ICMP Ping Floofind detected
192.168.1.1 -> 192.168.1.2
```

- TCP SYN Flooding [43]: es otro ataque de denegación de servicio, pero que explota las vulnerabilidades del protocolo TCP. Solo funciona si la máquina atacada tiene desactivado el mecanismo de SYN cookies. Para poder reproducirlo desde la máquina atacante de Kali Linux ejecutamos el comando **hping3 –S –flood –p 80 –rand-source 192.168.1.2**, donde la opción –S sirve para activar el flag TCP SYN, 80 es el puerto TCP destino y 192.168.1.2 la dirección IP de la máquina víctima.

  En un ataque TCP SYN Flooding aparece el siguiente mensaje:

```
Paquete 1 :
Alertas IP: IP Spoofing detected
Alertas TCP: TCP DoS detected
192.168.1.1 -> 192.168.1.2
```

La captura de los paquetes generados por los ataques anteriores ha sido posible gracias a la herramienta Wireshark instalada en una de las máquinas de la red, que ha estado escuchando todo el tráfico por la interfaz eth1. Para cada ataque se ha capturado una cantidad de paquetes lo suficientemente grande para que se pueda reconocer el ataque, con unos veinte o treinta paquetes ha sido suficiente.

En resumen, desde la máquina virtual atacante se lanzan ataques Ping Flooding, TCP SYN Flood y de escaneo de puertos, entre otros. Los paquetes generados se capturan y se utilizan como entrada al IDS, para ser analizados en busca de firmas de posibles ataques contenidas en el conjunto de reglas usado. En ese caso, se generará una alerta que será mostrada por pantalla.

5.1.5. Creación de reglas propias

Nuestro IDS detecta la presencia de posibles intrusos en la red mediante el uso de unas reglas predeterminadas (reglas Snort) que tienen la siguiente estructura (ver figura 26):

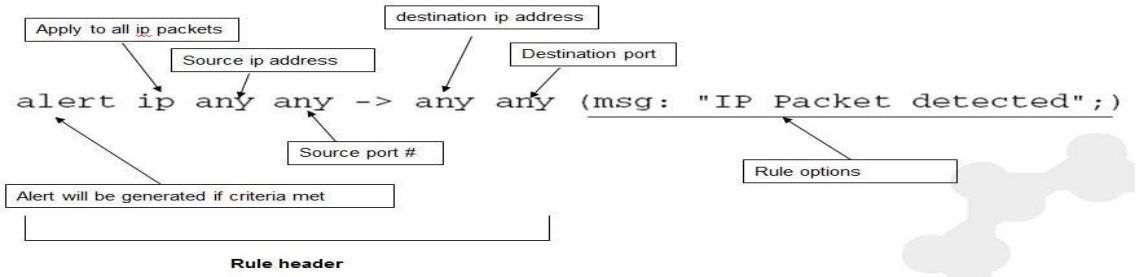


Figura 26. Estructura de una regla Snort

Estas reglas están ubicadas en el directorio **/etc/snort/rules**. El repositorio de reglas de Snort es extenso y variado, por lo que para probar el funcionamiento del IDS implementado en los casos descritos se han creado una serie de reglas siguiendo la sintaxis mencionada anteriormente y se han guardado en el directorio **/etc/snort/rules** dentro del fichero /**local.rules**. Se han añadido reglas capaces de detectar los ataques más comunes, escaneo de puertos, pings y tráfico común conocido.

5.1.6. Pruebas

Tras las pruebas realizadas con Snort, se pudo desarrollar el programa para la captura y análisis de paquetes. Tras implementarlo, fue necesario probarlo con el mismo tráfico generado para Snort.

Se generaron pings, tanto en IPV4 como en IPv6, escaneo de puertos, conexiones TCP y UDP, ataques de Ping Flooding y TCP SYN Flood, como se ha explicado anteriormente.

Primero, se ejecutaron los comandos necesarios para generar el tráfico en la red desde la máquina virtual atacante.

En la máquina Raspbian, con el programa Wireshark en modo escucha, se capturaron todos los paquetes necesarios y estos fueron exportados en formato “C arrays” a un fichero con el nombre del ataque o tráfico generado. Este formato nos da la información dentro de un array de char (de tamaño igual a lo que ocupa el paquete), por tanto, podríamos tratarlo como una matriz de datos, aunque finalmente nos decantamos por hacer un array con un contador, por motivos de eficiencia.

Una vez capturado el tráfico, se procedió a la lectura de estos archivos por parte del programa de análisis de paquetes para poder obtener las cabeceras de los paquetes y descomponerlas en campos para poder ser tratados y analizados en busca de coincidencias con alguna de las reglas definidas.

Como conclusión, se pudo observar que el IDS que hemos desarrollado en este TFG generaba los mensajes establecidos como alertas cuando los campos de las cabeceras de los paquetes coincidían con las especificaciones de alguna de las reglas. Estas alertas, no eran tan detalladas como las generadas por Snort, ya que el programa se encargaba de generarlas así. Sin embargo, se pudo comprobar que las alertas generadas por el programa coincidían con las que generaba Snort.

Nuestro IDS es capaz de alertar de los ataques para los que está programado, dejando las correspondientes alertas en el fichero alert establecido. Nuestro programa de análisis de paquetes está preparado para analizar todo tipo de tráfico proveniente de los protocolos explicados en el capítulo tres. Sin embargo, algunos de los ataques que reciba no harán saltar las alertas, ya que por motivos de falta de tiempo no se ha llegado a crear el tipo de reglas que hagan saltar esas alertas.

5.2. Arquitectura en un entorno Físico

La arquitectura que hemos usado en el entorno real es sencilla, un ordenador atacante, del tipo MSI (en concreto el ordenador de uno de los que han realizado el proyecto), conectado mediante un cable Ethernet a al dispositivo Raspberry Pi (ver figura 27). La idea era simular una red similar a la virtual, de modo que el dispositivo Raspberry Pi estuviese escuchando y analizando todo el tráfico malicioso que una máquina atacante enviase a través de la red a una máquina víctima, todo funcionando de una forma dinámica y no estática, como finalmente se ha implementado en este proyecto. Hemos generado una serie de paquetes (explicados anteriormente) con tráfico malicioso y los hemos almacenado en la Raspberry Pi. Una vez examinados esos paquetes con nuestro programa de análisis de paquetes, si hay una coincidencia con alguna de las reglas predeterminadas o alguna de las creadas por nosotros se genera una alerta y deja constancia, mediante un mensaje por pantalla, de la existencia de un posible ataque contra un dispositivo de la red que se pretende proteger.

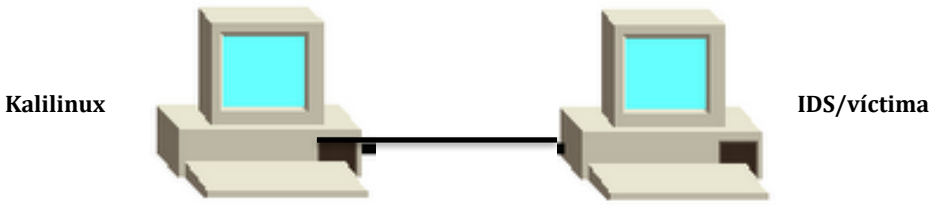


Figura 27. Arquitectura del entorno físico

5.2.1. Pruebas en entorno físico

Tras realizar pruebas para la comprobación del correcto funcionamiento del IDS en un entorno virtual, se procedió a realizar las mismas pruebas el programa desarrollado corriendo en el dispositivo hardware. Para ello, se emplearon dos máquinas, una emisora y otra receptora, conectadas mediante un cable Ethernet.

Como dispositivo atacante, se empleó un ordenador MSI con sistema operativo Windows 10, este ordenador serviría para emitir los distintos tipos de ataques y emisiones de tráfico en la red.

Como dispositivo receptor/víctima, se empleó la Raspberry Pi anteriormente mencionada con el sistema operativo Raspbian, con el programa IDS, la herramienta Wireshark.

El procedimiento para realizar las pruebas en el dispositivo físico fue el mismo que el empleado en el entorno virtual.

Desde la supuesta máquina atacante, el ordenador MSI, se lanzan pings, se realizan conexiones TCP, escaneo de puertos, ataques, ICMP Ping Flooding, TCP SYN Flood, etc, usando los comandos explicados en detalle en secciones anteriores. Este tráfico es capturado en la Raspberry por medio del programa Whireshark.

Una vez escuchada una cantidad suficiente de paquetes (unos veinte) se para la captura y esos paquetes se exportan en formato “C arrays” para ser analizados por el programa de análisis de paquetes, una vez analizados los paquetes se procederá a la separación de los campos de las cabeceras, para así buscar coincidencias con alguna de las reglas Snort predefinidas o alguna de las creadas por nosotros. Si hay alguna coincidencia saltará la alerta en el sistema de la Raspberry y se mostrará por la pantalla del dispositivo. Los mensajes de alertas resultantes son exactamente los mismos que los mostrados en el entorno virtual.

5.3. Ejemplo completo

En este apartado vamos a explicar en detalle las distintas fases de un ejemplo de simulación completo en el entorno virtual de la Figura 5.3: la generación del tráfico malicioso provocado por un cierto ataque, el estudio de ese tráfico con nuestra herramienta de análisis de paquetes, la búsqueda de posibles coincidencias con alguna de las reglas predeterminadas o reglas propias, y por último, la generación de una determinada alerta si hay una coincidencia.

Hemos elegido como ejemplo ilustrativo el ataque de escaneo de puertos con el flag TCP FIN activado.

En el entorno virtual usado tenemos la máquina Kali Linux que actúa como atacante (tiene un sistema operativo Kali Linux) y la dirección IP 192.168.1.1; una máquina víctima (con sistema operativo Raspbian) y la dirección IP 192.168.1.2; y por último, otra máquina que contiene el NIDS (con sistema operativo Raspbian) que escuchará el tráfico y alertará de lo ocurrido en la red.

Desde el terminal de la máquina atacante ejecutamos el comando **nmap –sF 192.168.1.0/24**. La máquina que contiene el NIDS tiene abierta la herramienta Wireshark para comenzar la captura de paquetes en el momento que se genere tráfico en la red. Una vez lanzado el comando comienza a capturar paquetes hasta que tenemos una cantidad suficiente y se para la captura.

El siguiente paso es alimentar nuestro programa de análisis de paquetes con la captura realizada con la herramienta Wireshark. Los datos de la captura son exportados al formato de C arrays y almacenados en el ordenador. El programa comienza leyendo el fichero con el contenido de la captura, separa los bytes leídos y

los guarda en una estructura de datos llamada **paquete**. Esa estructura es común para todos los paquetes, pero en función de las cabeceras de los protocolos se rellenarán unos campos u otros. En nuestro caso, como el ataque que estamos simulando usa el protocolo TCP se rellenará toda la estructura perteneciente al protocolo TCP, separamos todos los elementos de la cabecera y almacenamos todos los campos en nuestra estructura (este programa se explica en detalle en el capítulo 4).

Una vez descompuesto el paquete TCP, es hora de pasar a buscar coincidencias con las reglas. Guardamos todas las reglas en un fichero llamado **local.rules**, almacenado en el directorio **/etc/snort/**, incluimos tanto las reglas predeterminadas de Snort como las creadas por nosotros mismos. Lo primero que hacemos es obtener todas las reglas del fichero que hacen referencia a ataques TCP, vamos analizando cada regla que encontremos en el archivo con los campos del paquete objeto de estudio en busca de posibles coincidencias.

En este caso la coincidencia la encuentra en la regla:

```
alert tcp any any -> $HOME_NET any (msg:"TCP Port
Scanning";sid:10000003;rev:001;classtype:web-
application-attack; detection_filter:track by_src,
count 30, seconds 60;)
```

Una vez encontrada la coincidencia se generará una alerta, en nuestro caso:

```
Paquete 1 :
Alertas IP:
Alertas TCP: ! - TCP scan detected
192.168.1.1 -> 192.168.1.2
```

Esta alerta se mostrará en la pantalla del sistema, para que el administrador pueda ver lo ocurrido. Esta es la explicación de cómo salta una alerta desde que se genera un posible ataque hasta su aviso.

5.4. Resultados experimentales

Snort cuenta con un conjunto de reglas muy grande, reglas de todo tipo, que pueden desde un simple ping, una conexión por ftp, un host inalcanzable, un envenenamiento de las tablas ARP hasta alertar de un posible ataque de denegación de servicio (Dos). Como es evidente, por motivos de falta de tiempo no hemos podido tratar todo tipo paquetes que cumplan una determinada regla la cual haga saltar una posible alerta. Un paquete está compuesto por una cabecera y un campo datos, nosotros en este

proyecto solo nos hemos fijado en la cabecera del paquete. Nos hemos centrado en los protocolos TCP, ICMP, IP, pero nuestra herramienta está programada para analizar los demás protocolos de red. Además, hemos desarrollado reglas propias, basadas en la sintaxis de Snort, al ser una reproducción la visualización de las alertas no es exactamente igual que la que muestra Snort, nosotros mostramos el tipo de alerta, y algunos campos de interés como direcciones origen y destino.

A continuación, se detalla una tabla con los posibles ataques que nuestra herramienta es capaz de detectar.

PROTOCOLO	ATAQUE/ALERTA
TCP	**Suspicious IP address**
	Nmap XMAS Tree Scan
	Possible TCP DoS
	Possible SCAN FIN
	TCP Port Scanning
ICMP	**PING Flooding**
	ICMP test detected
	Possible Nmap ping sweep
	Posible Ping of death
	Large ICMP packet
	ICMP ping flooding attempt ip spoofing
	Possible ICMP ping flooding
Total	**12 ataques detectados**

Capítulo 6
Conclusiones y trabajo futuro

En esta sección, se exponen las conclusiones obtenidas tras realizar este proyecto, así como las explicaciones de los resultados. También, se aporta cuál podría ser el trabajo futuro, en el que se exponen una serie de opciones para ampliar este proyecto y dotarlo de más funcionalidades.

6.1. Conclusiones

El principal objetivo de este proyecto es emular el funcionamiento de un Sistema de Detección de Intrusos basado en red (NIDS) con un flujo de red normal y capaz de detectar ataques comunes. Para ello se han tomado de referencia otros NIDS conocidos como Snort y Suricata.

Durante el desarrollo de este trabajo, hemos podido estudiar y comprender cómo funciona un NIDS, que es un sistema capaz de detectar un posible ataque a una red en función de los paquetes que entran en dicha red.

En el funcionamiento de este tipo de sistemas se distinguen distintas etapas: la captura de paquetes que circulan por la red, su descomposición en campos, el análisis de estos paquetes que implica la comparación de sus campos con los atributos de una serie de reglas que son capaces de detectar ataques conocidos; y, por último, la generación de alertas en el caso de que el contenido de los paquetes analizados coincida con alguna regla.

En este Trabajo de Fin de Grado, hemos capturado el tráfico de la red utilizando, en un primer momento, la herramienta Wireshark; después, los paquetes capturados se han exportado en un formato legible y se han grabado en un fichero, que se ha enviado al IDS para que fuesen analizados.

También hemos desarrollado un programa capaz de analizar los paquetes capturados en una red. Para lo que ha sido fundamental descomponer los paquetes y detectar el tipo de protocolo de red transportado en ellos. Hemos creado unas cuantas reglas, basadas en la sintaxis de las reglas Snort, que nos permiten detectar una serie de ataques básicos. Estos ataques se aprovechan de las vulnerabilidades de protocolos, como IP, ICMP, TCP o UDP, por lo que para detectarlos solo es necesario comprobar el contenido de las cabeceras de cada uno de los protocolos. Por último, hemos definido el mensaje que se muestra cuando una alerta es generada ante la detección de un posible ataque.

Hemos realizado pruebas para comprobar la viabilidad de nuestro sistema en un entorno virtual y en un entorno físico. Tras las pruebas realizadas, los resultados fueron satisfactorios, ya que el programa fue capaz de alertar de los ataques que se estaban realizando.

Es destacable mencionar que asignaturas del Grado en Ingeniería Informática como Redes, Redes y Seguridad y Ampliación de Sistemas Operativos y Redes fueron

fundamentales para el entendimiento de los protocolos de red, el funcionamiento de los sistemas de seguridad, el análisis y la captura del tráfico de red.

6.2. Trabajo futuro

El objetivo de este trabajo era desarrollar un NIDS basado en reglas similar a Snort. La cantidad de reglas que tiene esta herramienta es muy amplia, el trabajo que implica crear una herramienta similar, que detecte los mismos ataques que detecta Snort, implica una cantidad de tiempo del que no se dispone en un TFG. Por lo tanto, hemos desarrollado un programa capaz de detectar ciertos tipos de ataques, pero hemos creado una estructura para que este proyecto se vaya ampliando de forma relativamente sencilla hasta convertirse en un IDS capaz de detectar cualquier ataque conocido.

A lo largo del desarrollo de este proyecto, se ha podido observar que existen distintas formas de capturar los paquetes que llegan al dispositivo, así como es posible realizar un análisis más profundo de los paquetes.

Se propone, por tanto, como trabajo futuro una forma más dinámica de capturar mensajes y enviarlos al programa de análisis que permita una mayor velocidad a la hora de analizar los paquetes de la red. Lo que implicaría la implementación de un programa demonio que permita la captura y envío automático de paquetes al programa IDS. De este modo, se evita guardar el tráfico en un archivo y su posterior lectura, lo que garantizaría su funcionamiento en tiempo real. El programa de análisis de paquetes de nuestro IDS permanecería en un bucle a la espera de la recepción de los paquetes enviados por el programa demonio.

Otra propuesta de trabajo futuro es realizar un análisis más exhaustivo de los paquetes, añadiendo, por ejemplo, el estudio del contenido del campo “datos”, que, junto al análisis de la cabecera, permitiría detectar un mayor número de ataques. Así mismo, es interesante la opción de incorporar más protocolos para analizar.

Es necesario destacar que las alertas generadas se emiten por consola durante la ejecución del programa de análisis, una implementación posible para mejorar la comunicación con el usuario sería clasificar las alertas según su grado de importancia y emitir estos mensajes de una manera más distinguible.

Por último, aunque la funcionalidad de un IDS es alertar, es interesante la posibilidad de implementar un sistema que permita detener un paquete que haya sido clasificado como ataque.

Appendix A - Introduction

History of art

The enormous attacks [1] form the Internet as viruses, spam, and software vulnerabilities exploitation make protection methods important elements to prevent and to avoid information and data from being defenseless against attacks. These methods are: cryptography, security politics, firewalls, IDS and IPS.

About intrusion detection and prevention systems, IPS (Intrusion Prevention System) detect, alert and stop suspicious intrusions, whereas IDS (Intrusion Detection System) only detect and alert.

These systems can be software implemented, in this case, they are installed and configured in a network host, or hardware implemented, where a device in specialized to work connected in a network, and hardware implemented, where a device is specialized to work exclusively as an IDS. About their performance, these systems generate alerts when they detect network anomalies that can be classified as a possible attack. There are some types of IDS, as HIDS, host based, or NIDS, network based.

Motivation

Nowadays, organizations, companies and particulars have network connected devices. The moment a device is connected, it is exposed to all types of attacks through this connection. Data used and stored must be protected form not authorized access and, at the same way, it is necessary to protect the devices for, in case they offer any kind of service, being always available for users.

To protect a network the maximum possible, efficient security measurement must be implanted, both prevention and detection. These measurements must be able to ensure all devices connected to this network.

Prevention and detection systems can avoid potentials attacks if they are found on time. These measurements can act against the problem, avoiding or stopping the attack, or warn the network administrator. The common tools used to detect an attack in a network are IDS, whereas firewalls are used to prevent them.

Firewalls prevent actively from incoming and outgoing from device packages, detecting those that are considered as threats and sending alerts. Based on a security policy, they can let traffic pass or not.

Similarly, IDS analyze incoming and outgoing network packages from the device, detecting those considered as a possible attack and generating alerts. The difference with firewalls is that these are usually located in the network organization perimeter, analyzing all the traffic that goes in and out, meanwhile IDS are located after firewalls on the internal network. In case of an intruder gets through the firewall, the IDS must detect it and warn the network administrator.

The main objective of this Final Degree Project is the hardware implementation of a Network Intruders Detection System (NIDS). The designed system captures and analyzes network packages with the goal of detect if the network is being attacked, and

if an attack is detected, alerts will be generated and delivered warning about the situation.

Our IDS hardware system will be connected through a switch internal network, same as the devices it is protecting, so that every package that enters through a port of the switch is redirected to the IDS to be analyzed. The reason of being implemented in a hardware device is that this type of devices has a reduced operative system and they do not have installed another type of applications, what it means to be less susceptible of having vulnerabilities that can be exploded for an attacker. Additionally, it is more efficient to have an IDS software correcting on a server, as it is released form workload that is the analysis of the network traffic.

Project goals

To be able to carry out the capture and the analysis of the network packages, at first, it was necessary to watch and to understand the performance of applications that make similar jobs, like Snort or Suricata. This task included the Snort rules study, whose syntax is used by Suricata and also by our IDS. These rules contain attack signatures, it means, they have attributes that identify the content that appears in certain fields of a package when a specific attack occurs.

Subsequently, package capture simulations were carried out to analyze the content and distribution of the different package fields. The packages are decomposed in fields and organized according to the protocol used.

Once packages are decomposed and organized by fields, a package analysis program was designed. For that, at first, it was necessary to link each package field with a rule attribute. Later, these package fields were compared to their correspondent rule attribute to search for matches and, ultimately, to generate an alert.

Finally, the IDS were installed in the hardware device and tests and simulations were made to check its correct performance.

Work plan

In order to reach these objectives, the next tasks have been made:

1. Attacks and package streaming simulation using virtual machines.
2. Study and classification of the different types of protocols and attributes used.
3. Python learning as a package processing language.
4. Development of a first program version that captured and decomposed packages and, after this, it classified them according to their fields and protocols.
5. Reading and understanding of Snort rules.
6. Creation of own rules based on Snort rules syntax.
7. Development of the analysis program allowing to read rules, compare their premises with the package content and generate alerts when a match is produced.
8. Results obtained evaluation in a simulation through the virtual machines.
9. Installation and evaluation of the results obtained in the hardware device.

The work plan is indicative and describes the steps taken to the project realization. During the development of this project, it has been necessary to delve into the theoretical aspects of the protocols and in the code development of the IDS program. The Figure 1 shows the percentage of the total time spent of each task, divided in the respective percentage of each component of the project.

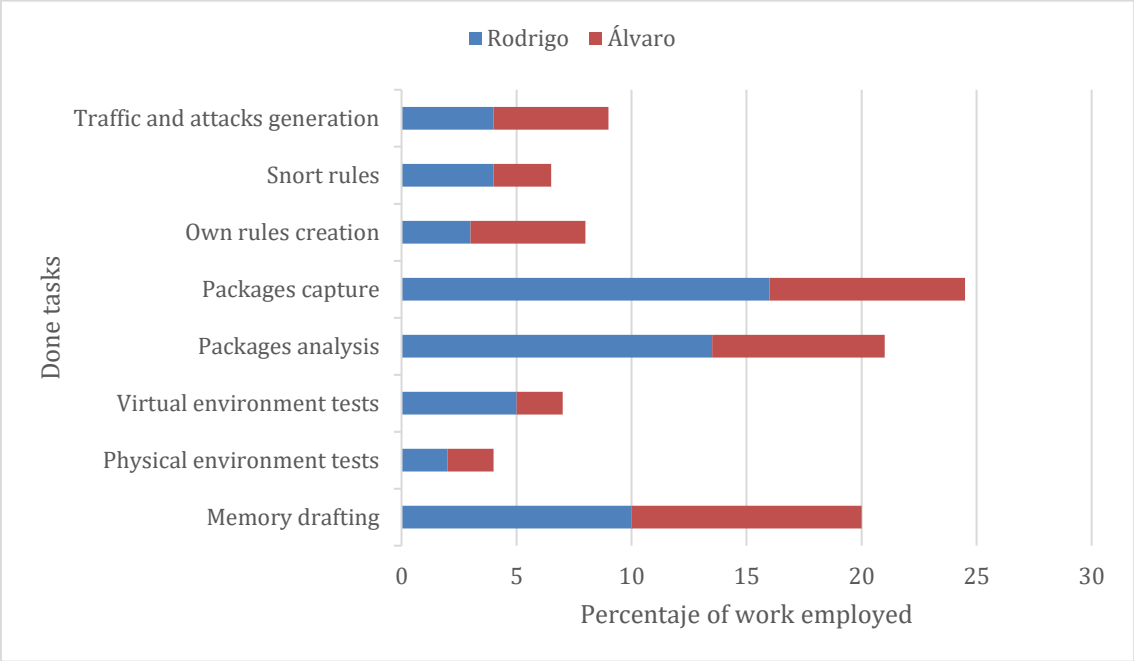


Figure 1. Percentage of time employed on the project

Memory organization

The chapters in which we have organized the memory correspond to the different stages our work has passed. In each one of them it is described in detail each task done. The sections of each chapter explain in detail the different procedures used to develop each stage.

Coming up next, it is briefly presented the following chapters content:

- **Chapter 2** is about what is the operation of an intrusion detection system, the different types of IDS devices and examples of other similar systems on which our Final Degree Project is based.
- **Chapter 3** describe network protocols used, with their properties explanation and the fields in which each package is divided. It is also referenced the treatment of this information to use it in this Project.
- **Chapter 4** explain procedures and tools used to the programs development and implementation of them.
- **Chapter 5** reviews the program implementation results and the tests made.
- **Chapter 6** brings the conclusions of the project members are contributed and proposes a series of tasks as a possible future work.

- Final Degree Project regulation of the IT Faculty of the Complutense University of Madrid obliges also to make a translation to English of introduction chapter and conclusions chapter, these translations are presented as *Appendix.*

Appendix B – Conclusions and future work

In this section, it is exposed the obtained conclusions after performing the project, the results are also explained. As well, it is provided the future work, where is exposed a series of options to extend this project and provide it with more features.

En este apartado, se proporcionan las conclusiones obtenidas tras realizar el proyecto, así como las explicaciones de los resultados. También, se aporta el trabajo futuro, en el que se exponen una serie de opciones para ampliar este proyecto y dotarlo de más funcionalidades.

Conclusions

The main objective of this project is to emulate the performance of a Network Intrusion Detection System (NIDS) with a regular network stream and capable of detecting common attacks. For this, other NIDS known as Snort and Suricata have been taken as reference.

During the development of this work, we have been able to study and understand how a NIDS works, which is a system capable of detecting a possible attack on a network depending on the packets that enter that network.

In the performance of this type of systems, different stages are distinguished: the capture of packets that circulate through the network, their decomposition in fields, the analysis of these packets that implies the comparison of their fields with the attributes of a series of rules that are capable of detecting known attacks; and, finally, the generation of alerts in the case that the content of the analyzed packets coincides with some rule.

In this Final Degree Project, we have captured network traffic using, at first, the Wireshark tool; then, the captured packages have been exported in a readable format and recorded in a file, which has been sent to the IDS for analysis.

We have also developed a program capable of analyzing the packets captured in a network. It has been essential to decompose the packages and detect the type of network protocol transported in them. We have created a few rules, based on the syntax of Snort rules, that allow us to detect a series of basic attacks. These attacks take advantage of vulnerabilities in protocols such as IP, ICMP, TCP or UDP, so to detect them it is only necessary to check the content of the headers of each protocol. Finally, we have defined the message that is displayed when an alert is generated on detection of a possible attack.

We have carried out tests to check the viability of our system in a virtual environment and in a physical environment. After the tests carried out, the results were satisfactory, as the program was able to alert of the attacks that were being carried out.

It is remarkable to mention that subjects of the Degree in Computer Engineering such as Networks, Networks and Security and Expansion of Operating Systems and Networks were fundamental for the understanding of network protocols, the functioning of security systems, the analysis and capture of network traffic.

Future work

The aim of this work was to develop a rule-based NIDS similar to Snort. The number of rules that this tool has is very extensive, the work that involves creating a similar tool, which detects the same attacks that Snort detects, implies an amount of time that is not available in a TFG. Therefore, we have developed a program capable of detecting certain types of attacks, but we have created a structure for this project to expand relatively easily to become an IDS capable of detecting any known attack.

During the development of this project, it has been observed that there are different ways to capture the packets that arrive at the device, as well as it is possible to carry out a deeper analysis of the packets.

Therefore, a more dynamic way of capturing messages and sending them to the analysis program is proposed as a future work, allowing a higher speed when analyzing network packets. This would imply the implementation of a daemon program that allows the automatic capture and sending of packets to the IDS program. In this way, it is avoided to save the traffic in a file and its later reading, which would guarantee its operation in real time. The package analysis program of our IDS would remain in a loop waiting for the reception of the packages sent by the daemon program.

Another proposal for future work is to carry out a more exhaustive analysis of the packages, adding, for example, the study of the content of the "data" field, which, together with the analysis of the header, would make it possible to detect a greater number of attacks. The option of incorporating more protocols for analysis is also interesting.

It is necessary to emphasize that the generated alerts are emitted by console during the execution of the analysis program, a possible implementation to improve the communication with the user would be to classify the alerts according to their degree of importance and emit these messages in a more distinguishable way.

Finally, although the functionality of an IDS is to alert, it is interesting the possibility of implementing a system that allows to stop a package that has been classified as an attack.

Bibliografía

[1] Tamer AbuHmed, Abedelaziz Mohaisen y DaeHun Nyang, <<Introduction>>, en *Deep Packet Inspection for Intrusion Detection Systems: A Survey*. Information Security Research Laboratory, Inha University, Incheon 402-751, Korea.

[2]: Shon Harris y Fernando Maymí, <<Intrusion Detection Systems>>, en *CISSP. All-in-One Examn Guide.* Mc Graw Hill, (2016). P.822-824.

[3]: OpenWebinars.net. (2019). <<Qué es Snort>>. [online] Available at https://openwebinars.net/blog/que-es-snort/

[4]: Suricata. (2019). <<Suricata>>. [online] Available at: https://suricata-ids.org/

[5]: Victor Truica. (2019). <<Understanding the Snort architecture>> [online] Available at: https://truica-victor.com/snort-architecture/

[6]: Victor Truica. (2019). <<Snort architecture image>> [online] Available at: https://truica-victor.com/wp-content/uploads/2014/03/snort-arch-.png

[7]: Seguridad y Redes. (2019). <<Entendiendo y configurando Suricata. Parte I>> [online] Available at: https://seguridadyredes.wordpress.com/2011/02/22/ids-ips-suricata-entendiendo-y-configurando-suricata-parte-i/

[8]: Behrouz A. Forouzan, <<The OSI model>>, en *Data communications and Networking*. Fourth Edition. Mc Graw Hill, (2007). P.29

[9]: Inmaculada Pardines Lence, << Arquitectura de protocolos TCP/IP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.5

[10]: Behrouz A. Forouzan, <<Protocols and Standards>>, en *Data communications and Networking*. Fourth Edition. Mc Graw Hill, (2007). P.19

[11]: Behrouz A. Forouzan, <<TCP/IP Protocol Suite>>, en *Data communications and Networking*. Fourth Edition. Mc Graw Hill, (2007). P.42

[12]: Behrouz A. Forouzan, <<IPv4>>, en *Data communications and Networking*. Fourth Edition. Mc Graw Hill, (2007). P.582-597

[13]: Inmaculada Pardines Lence, << Internet Protocol (IP)>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.9

[14]: Inmaculada Pardines Lence, << Vulnerabilidades en IP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.21

[15]: Behrouz A. Forouzan, << TCP >>, en *Data communications and Networking*. Fourth Edition. Mc Graw Hill, (2007). P.715

[16]: Inmaculada Pardines Lence, << Transmission Control Protocol (TCP)>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.17

[17]: Inmaculada Pardines Lence, << Vulnerabilidades en TCP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.25

[18]: Inmaculada Pardines Lence, << Ataques a protocolos de red>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.27-41

[19]: Rafael Moreno Vozmediano, Inmaculada Pardines Lence, Carlos González Calvo, Julio Septién del Castillo, Eduardo Huedo Cuesta, Alberto del Barrio García, Rubén Santiago Montero y Juan Carlos Fabero Jiménez. <<El protocolo UDP>>, en *Tema 5. La capa de transporte. Protocolos TCP y UDP.* Transparencias de la asignatura “Redes” del curso (2017/2018). P. 4,11-14.

[20]: Inmaculada Pardines Lence, << User Datagram Protocol (UDP)>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.15

[21]: Inmaculada Pardines Lence, << Vulnerabilidades en UDP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.25

[22]: Rafael Moreno Vozmediano, Inmaculada Pardines Lence, Carlos González Calvo, Julio Septién del Castillo, Eduardo Huedo Cuesta, Alberto del Barrio García, Rubén Santiago Montero y Juan Carlos Fabero Jiménez. <<El protocolo ARP >>, en *Tema 4. La capa de red. Protocolo IP.* Transparencias de la asignatura “Redes” del curso (2017/2018). P. 32-40.

[23]: Inmaculada Pardines Lence, << Address Resolution Protocol (ARP)>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.11

[24]: Inmaculada Pardines Lence, << Vulnerabilidades en ARP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.21

[25]: Rafael Moreno Vozmediano, Inmaculada Pardines Lence, Carlos González Calvo, Julio Septién del Castillo, Eduardo Huedo Cuesta, Alberto del Barrio García, Rubén Santiago Montero y Juan Carlos Fabero Jiménez. <<El protocolo ICMP >>, en *Tema 4. La capa de red. Protocolo IP.* Transparencias de la asignatura “Redes” del curso (2017/2018). P. 54-63

[26]: Inmaculada Pardines Lence, << Internet Control Message Protocol (ICMP) >>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.13

[27]: Inmaculada Pardines Lence, << Vulnerabilidades en ICMP>>, en *3.1. Vulnerabilidades en protocolos de red y ataques.* Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.23

[28]: RedesZone. (2019). <<Odroid-XU4, un mini-ordenador compatible con Linux y Android>>. [online] Available at: https://www.redeszone.net/2015/07/18/odroid-xu4-un-mini-ordenador-compatible-con-linux-y-android/

[29]: PlusElectric. (2019). <<Arduino Uno: Especificaciones y características>>. [online] Available at: https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/

[30]: Pastor, J. (2019). <<Raspberry Pi 3 Model B+, análisis: más potencia y mejor WiFi para un miniPC que sigue asombrando>>. [online] Xataka.com. Available at: https://www.xataka.com/ordenadores/raspberry-pi-3-model-b-analisis-mas-potencia-y-mejor-wifi-para-un-minipc-que-sigue-asombrando

[31]: Debian. (2019). [online] Available at: Debian https://www.debian.org/index.es.html

[32]: linux.org. (2019). [online] Available at: https://www.linux.org/

[33]: Raspberrypi.org. (2019). [online] Available at: https://www.raspberrypi.org

[34]: Raspbian. (2019). <<Raspbian>>. [online] Available at: https://www.raspberrypi.org/downloads/raspbian/

[35]: BalenaEtcher. (2019). [online] Available at: https://www.balena.io/etcher/

[36]: Wireshark.org. (2019). <<Wireshark Go Deep>>. [online] Available at: https://www.wireshark.com.

[37]: Jupyter.org. (2019). *Project Jupyter*. [online] Available at: https://jupyter.org.

[38]: Python. (2019). [online] Available at: https://www.python.org/

[39] Inmaculada Pardines Lence, <<Reglas>>, en *3.3.IDS*. Transparencias de la asignatura “Redes y Seguridad” del curso (2018/2019). P.23-25.

[40]: GitHub. (2019). *Build software better, together*. [online] Available at: https://github.com/.

[41] Seguridad y Redes. (2019). Snort. Búsqueda de patrones. Reglas de contenido. [online] Available at: https://seguridadyredes.wordpress.com/2008/06/12/snort-busqueda-de-patrones-reglas-de-contenido/

[42] Kali.org. (2019). [online] Available at: https://www.kali.org/downloads/.

[43] Román, S. (2019). Conceptos básicos de vulnerabilidades de red y firewalls. [online] Cv4.ucm.es. Available at: https://cv4.ucm.es/moodle/pluginfile.php/5139864/mod_resource/content/3/4.2_Avanzadavuln_red_FW.pdf

[44]Eduardo Huedo Cuesta << El protocolo IPv6>> en *1.4. El protocolo IPv6.* Transparencias de la asignatura “Ampliación de sistemas Operativos y Redes ” del curso (2018/2019).