ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

"INTEGRACIÓN DE UN IDS/IPS AL CONTROLADOR SDN PARA LA PREVENCIÓN Y DETECCIÓN DE ATAQUES DE SEGURIDAD (DoS) EN UN ESCENARIO DE REDES DEFINIDAS POR SOFTWARE"

TRABAJO DE TITULACIÓN:

Tipo: PROPUESTA TECNOLÓGICA.

Presentado para optar el grado académico de:

INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTOR: EDWIN MAURO MORALES DÁVILA

TUTOR: ING. ALBERTO LEOPOLDO ARELLANO AUCANCELA MsC.

Riobamba - Ecuador

2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

El Tribunal de trabajo de titulación certifica que: El trabajo de titulación: **INTEGRACIÓN DE UN IDS/IPS AL CONTROLADOR SDN PARA LA PREVENCIÓN Y DETECCIÓN DE ATAQUES DE SEGURIDAD (DoS) EN UN ESCENARIO DE REDES DEFINIDAS POR SOFTWARE,** de responsabilidad del señor Edwin Mauro Morales Dávila, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación quedando autorizado su presentación.

NOMBRE	**FIRMA**	**FECHA**
DR. JULIO SANTILLÁN **VICEDECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
ING. PATRICIO ROMERO **DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA, TELECOMUNICACIONES Y REDES**	_____	_____
ING. ALBERTO ARELLANO MsC **DIRECTOR DEL TRABAJO DE TITULACIÓN**	_____	_____
ING. OSWALDO MARTÍNEZ **MIEMBRO DEL TRABAJO DE TITULACIÓN**	_____	_____

Yo, Edwin Mauro Morales Dávila soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica de Chimborazo.

Edwin Mauro Morales Dávila

DEDICATORIA

A Dios por ser el mi guía y apoyo en los momentos difíciles de la carrera, por su infinito amor y bendiciones reflejadas a lo largo mi vida académica.

A mis padres Marlene y Noel, mi abuelo Robelio por ser mi ejemplo de superación y perseverancia, he logrado conseguir mi meta, por formarme con valores, por su amor, por su constante apoyo y por siempre confiar en mí. A mis hermanas Jessica, que en paz descanse y Lili, por su apoyo incondicional, por sus palabras de motivación en todo momento que me ha hecho crecer como persona.

A toda mi familia por confiar en mí, a mis abuelitos, tíos, primos y demás familiares, gracias por ser parte de mi vida y por permitirme se parte de su orgullo.

Edwin

AGRADECIMIENTO

A Dios por permitirme tener y disfrutar de una buena salud, por su infinito amor y bondad, por brindare sapiencia para poder asimilar con madures los momentos difíciles de la carrera y alcanzar esta meta propuesta.

A mis padres y mi abuelo por ser el pilar fundamental de mi vida, por brindarme su confianza y apoyo moral, por acompañarme en mi preparación académica, a todo su esfuerzo diario que hacen para verme cumplir mis sueños y anhelos. A mi hermana quien siempre estuvo pendiente de mi para brindarme su consejo y apoyo, quien se ha convertido en pañuelo de mis tristezas y cómplice de mis logros.

A mis amigos y compañeros, por compartir sus conocimientos, por su comprensión y ayuda constante en momentos cuando más los necesité, y en especial a TELMEL los cuales, más que amigos se han convertido en hermanos politécnicos, hemos compartido momentos inolvidables que quedaran grabados en todo mi ser.

A mi querida alma mater la Escuela Superior Politécnica Chimborazo por abrirme las puertas para formarme como profesional y a la Escuela de Ingeniería en Electrónica, Telecomunicaciones y Redes sitio que se convirtió en mi segundo hogar.

Edwin

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS.

ÍNDICE DE FIGURAS

ÍNDICE DE FOTOGRAFÍAS.

ÍNDICE DE ABREVIATURAS

SDN	Software Defined Networking (Redes Definidas por Software)
IDS	Intrusion Detection System (Sistema de Detección de Intrusos)
IPS	Intrusion Prevention System (Sistema de Prevención de Intrusos)
SNORT	Network Intrusion Detection & Prevention System (Sistema de prevención y detección de intrusos en la red)
ONF	Open Networking Foundation
DoS	Denial of Service (Ataque de Denegación de Servicio)
IP	Internet Protocol (Protocolo de Internet)
IPv4	Internet Protocol version 4 (Protocolo de Internet Versión 4)
TCP	Transmission Control Protocol (Protocolo de Control de Transmisión)
ICMP	Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet)
UDP	User Datagram Protocol (Protocolo de Datagramas de Usuario)
MAC	Media Access Control (Control de Acceso al Medio)
DHCP	Dynamic Host Configuration Protocol (Protocolo de Configuración Dinámica de Host)
DNS	Domain Name System (Sistema de Nombres de Dominio)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)
DMZ	Demilitarized Zone (Zona Desmilitarizada)

API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
NOS	Network Operating System (Sistema Operativo de Red)
HTML	HyperText Markup Language (Lenguaje de Marcas de Hipertexto)
DRDoS	Distributed Reflection Denial of Service (Denegación de Servicio de Reflexión Distribuida)
HIDS	Host Based Intrusion Detection System (Sistema de Detección de Intrusiones Basado en Host)
NIDS	Network Intrusion Detection System (Sistema de Detección de Intrusos en una Red)
TLS	Transport Layer Security (Seguridad de la Capa de Transporte)
FTP	File Transfer Protocol (Protocolo de Transferencia de Archivos)
SMB	Server Message Block (Bloque de Mensajes del Servidor)
UTP	Unshielded Twister Pair (Par Trenzado no Blindado)
VLAN	Local Area Network (Red de Área Local Virtual)
OSI	Open System Interconnection (Modelo de Interconexión de Sistemas Abiertos)

RESUMEN

El objetivo del trabajo de titulación, fue la integración de un sistema de detección y prevención de intrusos (IDS/IPS) al controlador para prevenir y detectar los ataques de seguridad (DoS). Para esto se implementó un escenario de redes definidas por software (SDN) usando el conmutador Zodiac Fx. El escenario está compuesto por un IDS/IPS Snort con reglas orientadas a la protección, que permiten analizar la red en busca de posibles ataques DoS en tiempo real, para luego registrarlos en un archivo temporal, y posteriormente con el controlador SDN bajo la plataforma Ryu, programar un algoritmo para establecer la comunicación OpenFlow entre los distintos elementos que componen la red, además adquirir la información del Snort para organizarla, clasificarla y por ultimo informar los eventos que están aconteciendo en tiempo real a un administrador de red. Se realizaron pruebas de interoperabilidad, mediante la creación de servidores: DHCP, DNS y HTTP, para comprobar la conectividad y funcionamiento de la red, se enviaron paquetes IP, con la finalidad de que el Snort detecte el flujo de datos e informe al Ryu. También se desarrollaron pruebas de rendimiento, con el estudio de dos casos; en el primero se ejecutaron ataques DoS a un sistema sin IDS/IPS, teniendo como resultado, ninguna información sobre la procedencia del flujo de datos, en el segundo se integró un sistema de detección, obteniendo información detallada, para lograr identificar intentos de intrusión, los mismos que fueron mitigados acorde a reglas previamente establecidas. Finalmente, con ayuda de las herramientas wireshark, barnyard2 y BASE, se logró determinar un porcentaje de error del 0.077. La integración del sistema de detección y prevención de intrusos ha logrado ser eficiente en cuanto a seguridad, teniendo ventaja frente a otros mecanismos de protección que funcionan aislados al controlador. Se recomienda establecer reglas puntuales para bloquear solo trafico malicioso y evitar la eliminación de información útil.

PALABRAS CLAVES: <TECNOLOGÍAS Y CIENCIAS DE LA INGENIERÍA>, <REDES DEFINIDAS POR SOFTWARE (SDN)>, <REDES DE COMPUTADORES>, <SISTEMAS DE DETECCIÓN Y PREVENCIÓN DE INTRUSOS>, <RENDIMIENTO>, <FLUJO DE DATOS>, <PLATAFORMAS SDN>, <SEGURIDAD DE LA INFORMACIÓN>.

ABSTRACT

The main purpose of this study was the integration of an intrusion detection and prevention system (IDS/IPS) to the controller in order to prevent, and detect security attacks (DoS). In order to make this possible, a scenario of software defined networks (SDN) was implemented by using the Zodiac Fx switch. The scenario is made of an IDS (Intrusion Detection System) / IPS (Intrusion Protection System) Snort with protection-oriented rules, which enables to analyse the network in search of possible DoS attacks in real time, and then, recording them in a temporary file, and subsequently, with the SDN controller under the Ryu platform, to program an algorithm to establish the Open Flow communication between several elements comprising the network, and also, acquire the Snort information to organize it, classify it, and finally to report the events that are happening in Real time to a network administrator. Interoperability tests were conducted by the creation of DHCP, DNS and HTTP servers. In order to verify the connectivity and operation of the network, IP packets were sent, so that Snort detects the flow of data and reports to the Ryu. Performance tests were also developed with the study of two cases: the first, where DoS attacks were executed on a system without IDS/IPS, resulting in a lack of information about the origin of the data flow. In the second case, a detection system was integrated, providing detailed information to identify intrusion attempts, which were eliminated according to rules previously established. Finally, it was determined an error percentage of 0.077 after using supporting tools like wireshark, barnyard2 and BASE. The integration of the intrusion detection and prevention system has been efficient in terms of security, taking advantage over other protection mechanisms that work isolated of the controller. It is recommended to set specific rules to block only malicious traffic and avoid the elimination of useful information.

KEY WORDS: < TECHNOLOGIES AND SCIENCES OF ENGINEERING >, < SOFTWARE DEFINED NETWORKS (SDN)>, < COMPUTER NETWORKS >, <INTRUSION DETECTION AND PREVENTION SYSTEM>, <PERFORMANCE>, <DATA FLOW>, < SDN PLATFORMS >, <SECURITY OF INFORMATION>.

INTRODUCCIÓN

ANTECEDENTES

En los últimos tiempos han surgido nuevos servicios de red que han hecho que las tecnologías usadas desde hace cincuenta años lleguen prácticamente al límite de sus capacidades. La aparición en escena del término "big data" o cantidades grandes de información, la introducción del "cloud computing" o computación en la nube, las aplicaciones en tiempo real y el hecho de que las comunicaciones modernas no sean exclusivamente entre cliente y servidor, hacen que sea necesario no solo un arreglo temporal, como ya se ha propuesto, creando nuevos protocolos, sino una solución global que afronte los problemas de las redes modernas y proponga métodos de comunicación más eficaces. (Incera y et al, 2007, p. 2)

De esta forma nace el concepto: "creación de redes definidas por software (Software Defined Networking)", las cuales proponen un cambio innovador en las redes de comunicaciones con el objetivo de aumentar su flexibilidad y minimizar sus costes. El surgimiento de nuevas arquitecturas de red, son ideales para soportar las aplicaciones que se desarrollan hoy en día, priorizando la implementación de calidad de servicio y seguridad ofrecida a los usuarios finales.

El término SDN (Software Defined Network o red definida por software) es una arquitectura de red que separa el plano del control, del plano de datos, para conseguir redes más programables, automatizables y flexibles. SDN elimina la inteligencia de las redes tradicionales del hardware, delegando las capacidades de toma de decisiones en el servidor o controlador. La tecnología desacopla la capa de datos (data layer: información real que viaja en una red) de la capa de control (control layer: la tecnología que determina que acción realizar con los datos), donde cada capa está automatizada. (Figuerola, 2013) Para la comunicación entre controlador y los elementos de red surge el protocolo OpenFlow, desarrollado por la Open Networking Foundation (ONF). Openflow es un protocolo de comunicaciones diseñado para dirigir el manejo y enrutamiento del tráfico en una red conmutada. (Velásquez, 2013, p. 1)

SDN utiliza un controlador centralizado con aplicaciones de software, donde el controlador asume las funciones más complejas como, por ejemplo: manejar la inteligencia de la red y monitorear el comportamiento de la red en tiempo real, por consiguiente SDN permite analizar patrones de tráfico para posibles problemas de seguridad como ataques de denegación de servicio, guiar paquetes sospechosos a sistemas de prevención de intrusión (IPS), modificar reglas de reenvío para bloquear tráfico, o dar privacidad a los usuarios. (Wang, 2016)

Conforme el avance de nuevas tecnologías para las arquitecturas de redes informáticas, los ataques orientados a realizar interrupción de servicios, como es el caso de los ataques de denegación de servicio (Distributed Denial of Service, DoS), siguen evolucionando, donde este tipo de ataques tratan de agotar los recursos del sistema consumiendo el ancho de banda.

Los ataques DoS se están produciendo cada vez con mayor frecuencia y, por ejemplo, según (Kaspersky) se realizan aproximadamente 450 ataques por día logrando enviar 15.8 millones de paquetes por segundo. La seguridad de todo sistema debe garantizar cuatro requisitos: autenticación, integridad, confidencialidad, disponibilidad. Un ataque de denegación de servicio es un intento de provocar la saturación o fallo de un servicio enviando trafico inservible desde uno o múltiples orígenes. Los ataques DoS representan una gran amenaza para la disponibilidad de servicios críticos, que se ven totalmente degradados a causa de estos ataques. (Ocampo y et al, 2017, p. 1)

Debido al aumento significativo de las actividades cibernéticas maliciosas, los encargados de la administración de red, tratan de aplicar protecciones como por ejemplo un DMZ (zona desmilitarizada), Firewalls, políticas internas, etc. Pero este tipo de soluciones no son 100% seguras, porque algunos ataques no son detectados por estos elementos de protección. De esta forma existen tecnologías adicionales contra estos ataques maliciosos, como por ejemplo los IDS y los IPS, los cuales son mecanismos de seguridad adicional, se caracterizan por la incorporación de mecanismos de inteligencia de amenazas, es decir bloqueo automático de páginas web, bloqueo de servidores DNS, direcciones IP de dudosa reputación, inspección profunda de paquetes, análisis de aplicaciones y archivos, etc.

FORMULACIÓN DEL PROBLEMA

¿Cómo se puede integrar un IDS/IPS al controlador SDN para la prevención y detección de ataques de seguridad (DoS) en un escenario de redes definidas por software.?

SISTEMATIZACIÓN DEL PROBLEMA

¿Cuáles son los ataques DoS maliciosos más comunes que se dan en una red SDN?
¿Cuáles son los mecanismos para detectar y prevenir ataques DoS en redes SDN?
¿Cuál es el efecto de integrar un IDS/IPS en redes definidas por software?
¿Cuáles son los IDS/IPS tradicionales que pueden comunicarse o pueden ser integrados en ambientes SDN?

JUSTIFICACIÓN TEÓRICA

Del mismo modo que en una red tradicional, la red SDN está expuesta a varias fuentes de riesgos de seguridad, debido a la perspectiva innovadora en el diseño de su arquitectura, en donde trata de centralizar todo el control de la red en un solo elemento, esta tendencia de centralizar introduce nuevos desafíos en la seguridad.

Uno de los factores de riesgo más significativos es la posibilidad de que un ataque intente comprometer el correcto funcionamiento de un controlador de la red SDN a nivel del plano de control. Debido al diseño centralizado, el controlador de la SDN se transforma en el cerebro de la arquitectura, los intrusos pueden centrar sus esfuerzos en comprometer al controlador SDN en un intento de manipular toda la red. Si el atacante consigue ganar permisos de control, esté puede ser utilizado para administrar los dispositivos de red que controla, realizando acciones como botar o descartar todo el tráfico entrante, o bien lanzar un fuerte ataque en contra de otros objetivos.

"Las vulnerabilidades de los sistemas informáticos hacen que los ataques sean más frecuentes hacia ellos, para lo cual se implementan cada día formas más propicias para la defensa de los mismos". (Macías, 2017, p. 13)

Es por ello, que el presente proyecto, pretende brindar protección adicional al controlador SDN, mediante la integración de un IDS/IPS, estos mecanismos ayudan a detecta y prevenir actividades maliciosas basándose en la monitorización de la red para posteriormente en caso de intrusión tomar medidas necesarias para contrarrestas ataques. La integración de estos mecanismos tiene como fin proporcionar una mejora en la seguridad de la red.

JUSTIFICACIÓN APLICATIVA

La función principal de un controlador SDN es estar ubicado en un lugar centralizado en donde tendrá una visión global del estado de la red y podrá tomar decisiones de control, pudiendo actuar a la vez sobre todos los equipos de conmutación; el correcto funcionamiento dependerá de la seguridad que se le brinde para contrarrestar actividades sospechosas que atenten contra los recursos de la red. Es por ello que en este proyecto se pretende integrar un IDS/IPS para dar solución al problema planteado.

Para poder realizar la integración de un IDS/IPS, en primer lugar, se trabajará sobre un escenario físico el cual consta de una tarjeta que soporta el protocolo OpenFlow, este dispositivo simula la

función de un switch SDN, se elige esta tarjeta por que consta de los puertos necesarios para conectar los elementos de la red tanto usuarios, controlador y el IDS/IPS.

OBJETIVOS

OBJETIVO GENERAL

Integrar un IDS/IPS a un controlador SDN para la prevención y detección de ataques de seguridad (DoS) en un escenario de redes definidas por software.

OBJETIVOS ESPECÍFICOS

- Investigar los principales tipos de ataques DoS y los mecanismos de detección en un entorno de redes definidas por software.
- Desarrollar un entorno de pruebas para la verificación de un IDS/IPS en la detección de ataques de seguridad (DoS).
- Evaluar los resultados obtenidos al implementar IDS/IPS en redes definidas por software.
- Determinar los efectos al integrar un IDS/IPS en redes definidas por software.

El presente trabajo de titulación indica los procesos realizados para cumplir con los objetivos planteados. Está formado de introducción, tres capítulos, conclusiones y recomendaciones. El primer capítulo hace referencia al marco teórico y conceptos básicos. El segundo capítulo detalla el marco metodológico y el desarrollo del sistema. El tercer capítulo describe las pruebas y los resultados obtenidos.

CAPÍTULO I

1 MARCO TEÓRICO

En el presente capítulo se investigarán los conceptos básicos, que serán fundamento teórico para la realización del presente proyecto y de este modo facilitar la comprensión de los principales temas. En primer lugar, se realiza una introducción a las redes definidas por software, analizando la arquitectura y el protocolo de comunicación para este tipo de redes. Luego se detallará y comparará las plataformas para los controladores SDN. A continuación, se describe los ataques de seguridad DoS y los mecanismos para la detección y mitigación. Y por último se realiza una investigación de los IDS/IPS y los efectos que estos tienen al integrar a una red SDN.

1.1 Redes Definidas por Software (SDN)

Software Defined Networking o su similar en español redes definidas por software (SDN) es un paradigma que intenta revolucionar las redes de comunicaciones, mediante la virtualización de la red, es decir independizándola de una estructura fisica la cual únicamente tendrá la labor de reenviar el tráfico de la red, mientras que la función de gestionar y administrar se encuentra virtualizada y centralizada. (Ibáñez y et al, 2016, p. 15)

El principal objetivo del paradigma SDN es permitir que las redes sean más flexibles, escalables y eficientes mediante la separación del plano de control y el plano de datos en otras palabras, es la separación del hardware y software, en donde la función de control es encargada a un dispositivo centralizado llamado controlador, mediante el cual se podrá administrar la red, el plano de datos se lo encarga a los switch SDN.(Incera y et al, 2007, p. 20)

Al momento de separar el plano de control del plano de datos y dar estas funciones a dos dispositivos diferentes, estos necesitaran estar comunicados o interconectados para que pueda existir una red SDN, esta función se lo entrega a un protocolo llamado OpenFlow, el cual constituye la base de las redes definidas por software, este protocolo tiene sus principios en el 2007 con una colaboración de Stanford y California, en la actualidad ONF (Open Networking Foundation) es la encargada de la definición del estándar con el mayor aliado que es HP. Y tiene como objetivo estandarizar todas las tecnologías emergentes para la gestión de red y del data center. (Pérez y Marín, 2015, p. 41–63, p. 8)

1.1.1 Protocolo OpenFlow

Según la definición dada por ONF, OpenFlow es un protocolo de comunicaciones diseñado para dirigir el manejo y enrutamiento del tráfico en una red conmutada. El origen de este protocolo se remonta al año 2006, cuando se desarrolla "Ethane" la cual es una arquitectura de red lógicamente centralizada para la gestión de las políticas de seguridad de las redes empresariales. (Sánchez, 2017, p. 15)

1.1.1.1 Características de OpenFlow

OpenFlow es el primer protocolo de comunicación definido entre la capa de control y la capa de datos, permite el acceso directo y la administración de los dispositivos de red como switchs y routers tanto físicos como virtuales, debido a la falta de una interfaz que permita mover el control de la red fuera del entorno lógico, por esto se ha convertido en un estándar ideal al momento de la implementación de redes SDN, brindando las siguientes características: (Isa, 2016, p. 5)

- Separación del plano de datos y de control.
- Utilización de un protocolo estandarizado entre el controlador y un agente en la red.
- Brinda capacidad de programacion de la red desde una visión agrupada a través de una API.

El protocolo se usa entre los terminales de infraestructura (usuarios), conmutador o Switch y el software de control de la SDN (Ryu, OpgenDayLight, etc..). OpenFlow aplica el concepto de flujos para reconocer el tráfico apoyándose en reglas predefinidas por el software del controlador SDN, estas reglas pueden ser estáticas o dinámicas.

Adicionalmente, permite especificar que ruta debe tomar el tráfico a través de los dispositivos de red basados en medidas tales como los patrones de uso, aplicaciones y recursos de la nube, también brinda un control específico, lo que permite que la red no tenga problemas ante los cambios en tiempo real a nivel de aplicación, en usuarios y a nivel de sesión. OpenFlow está conformado por una API y por un conjunto de protocolos, los mismos que se dividen en dos partes: (Hervás, 2014, p. 10)

- Protocolo de conexión (actualmente en la versión 1.3): para establecer una sesión de control, define la estructura del mensaje para intercambiar las modificaciones de los flujos y obtener estadísticas, y con esta información definir la estructura fundamental del switch (puertos y tablas).

- Protocolo de configuración y administración (actualmente en la versión 1.4) basado en NETCONF9, para asignar puertos de switch físicos a un controlador en específico, define la disponibilidad (activo / en espera) y los comportamientos en el controlador, cuando se produce un fallo en la conexión.

Una SDN basada en OpenFlow, puede ser implementada en una red física o virtual, los dispositivos de red, pueden tratar a las reglas de reenvío del protocolo como se lo realiza con las configuraciones tradicionales, lo que proporciona una fácil migración a esta nueva tecnología incluso si dentro de la red se tienen equipos de diferentes distribuidores. (Sánchez, 2017, p. 14)

1.1.2 Arquitectura de redes SDN

En las redes tradicionales las arquitecturas poseen muchas limitaciones las cuales no permiten obtener el mayor beneficio a sus características de manera fácil, además su diseño no está orientado a cubrir los requerimientos que tienen los usuarios en la actualidad, la mayor desventaja de este tipo de redes es que se consume mucho tiempo a la hora de configurar o modificar un elemento de este sistema para poder brindar un nuevo servicio. (Sánchez, 2015, p. 12)

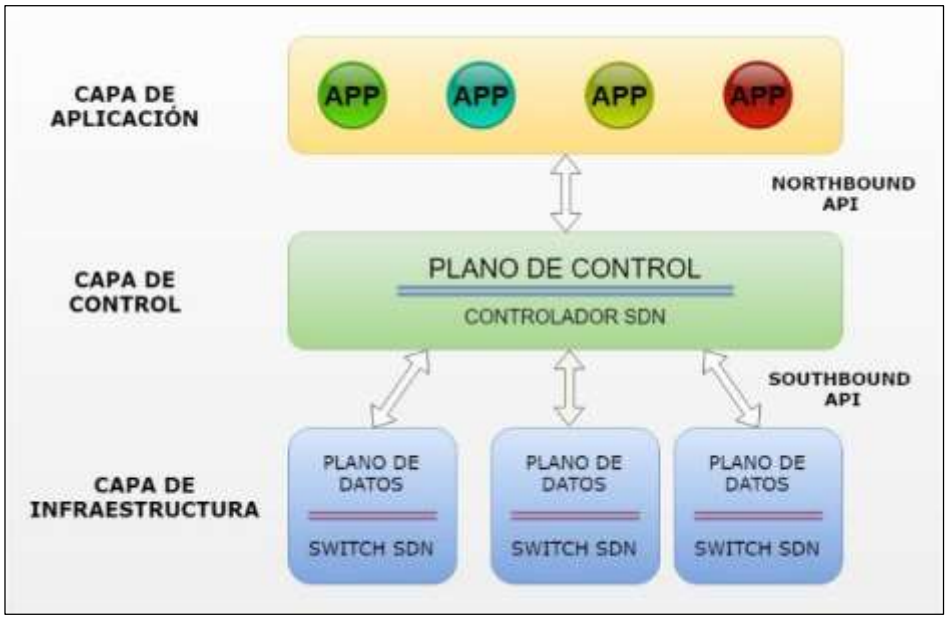


Figura 1-1: Arquitectura de las redes SDN.

Realizado por: MORALES, Edwin, 2018

De acuerdo a esta problemática, SDN es una arquitectura de red que está emergiendo, pues en esta el control está separado del reenvió, permitiendo así tener una red programable, en la figura 1-1 se puede observar la arquitectura SDN en donde la inteligencia de la red está centralizada en

el controlador y este mantiene una visión global de red, este tipo de arquitectura simplifica la operación y el diseño de la red, y además reduce el número de dispositivos en comparación con una red tradicional. (Pegado, 2015, p. 16)

1.1.2.1 Plano de Datos

El plano de datos es el encargado de dar tratamiento o procesar los paquetes que ingresan a los dispositivos de red (switch) a través de un medio físico. Cuando un paquete ingresa al dispositivo, este lo reenvía al controlador SDN para que se modifique la cabecera del paquete, luego de esto el controlador envía al dispositivo las instrucciones específicas sobre el tratamiento que se le debe dar al paquete, por ejemplo, puertos de salida, siguiente salto y finalmente el conmutador las debe aplicar. (Ibáñez y et al, 2016, p. 12)

Una de las características más importantes de los conmutadores SDN es que las instrucciones indicadas se las puede almacenar en la tabla de flujos que posee el switch, con esto, el proceso de preguntar nuevamente al controlador no se repetiría, brindando así una solución rápida y eficiente al paquete que ingresa.

1.1.2.2 Plano de control

El plano de control o específicamente controlador, es el encargado de la configuración de los nodos y de la programación del envió de los flujos de manera automática desde el controlador hacia cada uno de los nodos, teniendo en cuenta la situación en que se encuentre la red. Si este proceso se lo compara con la red tradicional es mucho más eficiente, debido a que en las antiguas redes el administrador tenía que realizar las modificaciones o ajustes en cada uno de los nodos de manera manual. (Ibáñez y et al, 2016, p. 14)

El controlador o llamado Sistema Operativo de Red (NOS), al ser centralizado o distribuido, es decir el cerebro de la red, en caso de producirse una falla provocará un impacto negativo en todo el sistema. Estas fallas generalmente se producen en la comunicación que existe entre la capa de control, datos y aplicación, en donde esta comunicación es gestionada por dos interfaces denominadas: API NorthBound y API SouthBound, como se puede observar en la figura 2-1.

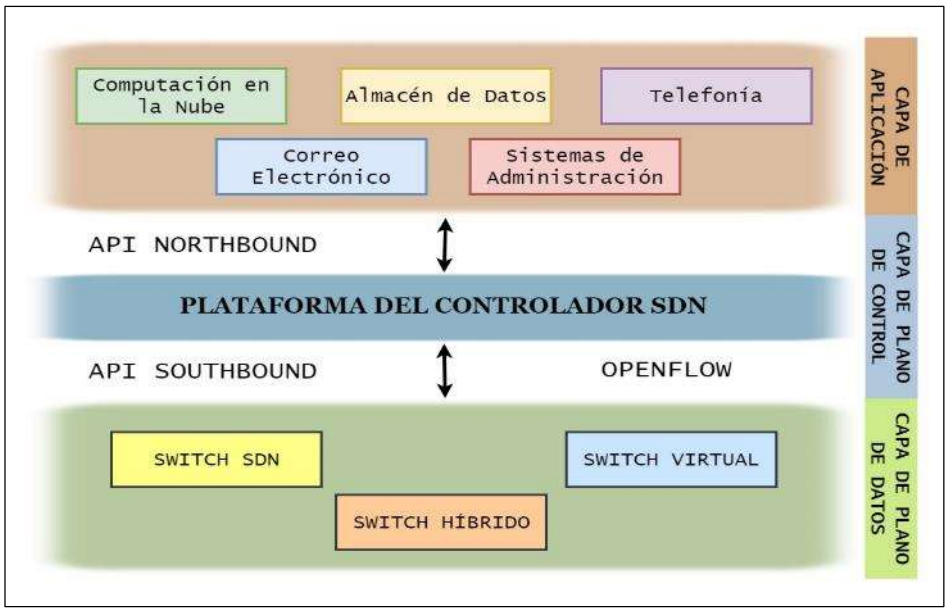


Figura 2-1: APIs NorthBound y SouthBound.

Realizado por: MORALES, Edwin, 2018

- NorthBound API

La NorthBound API o también llamada interfaz API hacia el norte, proporciona una separación de red para las aplicaciones y los sistemas de gestión ubicados en la parte superior de la arquitectura SDN. Entonces una API hacia el norte es aquella que pone a las aplicaciones en control de la red. En lugar de ajustar y ajustar la infraestructura repetitivamente para que un servicio funcione correctamente, se puede configurar un algoritmo que permita a la aplicación demandar la infraestructura que necesita. (Paracuellos, 2016, p. 7)

La razón principal de la existencia de API Northbound es que los sistemas de gestión externos o las aplicaciones de red pueden desear extraer información controlando la red subyacente y partes de su comportamiento. También exponen el modelo y la funcionalidad de datos de abstracción de red universal dentro del controlador para su uso por aplicaciones de red. Se utilizan para facilitar la innovación y tener una orquestación efectiva de la red. Es necesario para la alineación de la red a las necesidades de diferentes aplicaciones y permiten la máxima utilidad de la arquitectura SDN. (Pérez y Marín, 2015, p. 41–63, p. 49)

- SouthBound API

Las SouthBound API es decir interfaz API hacia el sur facilitan el control eficiente de la red y permiten que el controlador SDN realice cambios dinámicos de acuerdo con las demandas y necesidades en tiempo real. Permite al controlador definir el comportamiento de los conmutadores

y enrutadores que se encuentra en la parte inferior de la arquitectura SDN, está en la principal diferencia entre las interfaces hacia el norte y hacia el sur. (Pérez y Marín, 2015, p. 41–63, p. 49)

1.1.2.3 Plano de aplicación

El plano de aplicación tiene como objetivo la implementación de aplicaciones de alto nivel las cuales serán usadas por los usuarios para tener el control de los servicios, como por ejemplo aplicaciones de seguridad, audio, video, optimización y gestión de la información. Los usuarios finales requieren servicios de comunicación SDN mediante la NorthBound API, a través de REST, JSON XML. (Ibáñez y et al, 2016, p. 20)

La capa de aplicación tiene como objetivo automatizar la configuración y manipulación de los servicios, permitiendo la inserción de nuevos servicios y además mejorar la toma de decisiones. Algunos ejemplos de aplicaciones son: FlowVisor, ElasticTree, OpenPipes, Flow Manager.

1.2 Plataformas para un controlador de redes definidas por software (SDN)

En la arquitectura de redes definidas por software (SDN), el elemento principal es el controlador. Pues se trata del dispositivo que toma las decisiones y luego efectúa las reglas para la red, es decir ejecuta los algoritmos o las instrucciones que llegan de las aplicaciones y luego las reenvía a los dispositivos de capa física encargados del plano de datos.

El controlador decide que procedimiento tener con los paquetes que se encuentran especificados en las entradas de las tablas de flujo, además se encarga de gestionar dichas tablas. Los controladores principalmente se diferencian entre sí, por el lenguaje de programación (Java, Python, HTML) y plataforma (OpenDayLight, Ryu, FloodLight), pero en si todos estos controladores realizan las mismas funciones, también se comunican por el mismo protocolo denominado OpenFlow. (Pérez y Marín, 2015, p. 41–63, p. 48)

En el mercado existen un gran número de plataformas para los controladores SDN, existen controladores libres basados en código abierto, también hay controladores comerciales los cuales se los puede adquirir poniéndose en contacto con las empresas proveedoras y especializadas en redes definidas por software, como se muestra en la tabla 1-1. En donde se especifica los controladores con el lenguaje de programación en la cual está basado.

Tabla 1-1: Lenguaje de programación de los diferentes controladores SDN.

Controlador	Tipo	Lenguaje de programación
APIC	Comercial pagado (Cisco)	XML, Java
VAN	Comercial pagado (HP)	Java
Open Contrail	Código Abierto (Juniper)	Java
Agile Controller	Comercial Pagado (Huawei)	Python
POX	Código Abierto	Python
Floodlight	Código abierto	Java
OpenDayLight	Código Abierto (Brocade)	Java
Ryu	Código Abierto	Python

Fuente: http://revistas.uees.edu.ec/index.php/IRR/article/view/23

Realizado por: MORALES, Edwin, 2018

Todas estas plataformas para controladores SDN mencionados en la tabla 1-1, no son opciones válidas para poder desarrollar el presente proyecto, ya que al usar la tarjeta Zodiac FX, la plataforma debe ser lo más compatible con esta tarjeta, para determinar el nivel de compatibilidad se procederá a detallar las características más importantes de cada una de las plataformas y luego realizar una comparación.

1.2.1 Plataforma OpenDayLight

OpenDayLight (ODL) es una plataforma de código abierto, robusto, modular, escalable, extensible y además tiene una infraestructura de plataforma multi-protocolo. Entre sus primordiales ventajas se puede decir que tiene una buena aceptación dentro de la industria de sistemas en la nube y además permite la integración de OpenStack, el cual es un proyecto de computación en la nube para proporcionar una infraestructura de servicios. (Ibáñez y et al, 2016, p. 18)

ODL sirve para automatizar y personalizar las redes de cualquier escala y tamaño, este proyecto OpenDayLight surge a partir del movimiento SDN, con un enfoque a la programabilidad de red. Desde un principio fue diseñado como base para soluciones comerciales que afrontan diversos casos del uso de redes existentes. ODL es promovido por una comunidad global de proveedores y usuarios.

La plataforma OpenDayLight está diseñada para que los usuarios intermedios y los proveedores puedan crear un controlador de acuerdo con sus necesidades. El diseño modular de esta plataforma permite que cualquier persona inmersa en el ecosistema ODL pueda aprovechar los servicios desarrollados por otros, además permite crear y agregar los servicios para compartir con los

demás. OpenDayLight brinda soporte para protocolos de cualquier plataforma SDN: OpenFlow, OVSDB, NETCONF, BGP. (OpenDaylight Project, 2016)

1.2.2 Plataforma Ryu

Ryu es un framework que se encuentra basado en componentes de redes definidas por software. Estos componentes proveen una API definida, la cual permite a los desarrolladores introducir nuevas aplicaciones de control. Esta plataforma admite varios protocolos para administrar dispositivos de red, como por ejemplo OpenFlow, NETCONF, OF-CONFg, etc. Los cuales sirven para manejar los dispositivos de red SDN. (Sánchez, 2017, p. 8)

Ryu es un proyecto respaldado por Nippon Telegraph and Telephone (NTT) Labs. El proyecto tiene raíces japonesas; Ryu significa "flujo" en japonés, actualmente es compatible con OpenFlow 1.0, 1.2, 1.3, 1.4, esta plataforma se encuentra desarrollado completamente en Python y todo el código está disponible gratuitamente bajo la licencia de apache 2.0, está herramienta es elegida para desarrollar el plano de control en laboratorios experimentales.

1.2.3 Plataforma FloodLight

Floodlight es una plataforma open source que está desarrollada en Java y totalmente compatible con OpenFlow 1.0, este es un proyecto que surge o nace a partir de Beacon, entre las ventajas de esta plataforma es que se puede encontrar una extensa documentación y además expone una API REST, pero su gran desventaja es que presenta una curva lenta de aprendizaje. Además, cuenta con soporte de mantenimiento por Big Switch Networks. (Ibáñez y et al, 2016, p. 18)

Floodlight cuenta con diferentes módulos los cuales permiten la implementación de muy buenas funcionalidades estándar de un controlador SDN, como, por ejemplo:

- Exponer y descubrir los estados y eventos de red (dispositivos, topología, flujos)
- Permitir la comunicación entre el controlador y los dispositivos de red con OpenFlow
- Administración de los módulos de Floodlight y de los recursos compartidos (hilos)
- Interfaz web de usuario y servidor de depuración (Jython).

Estas funcionalidades han hecho de Floodlight uno de los controladores más conveniente para realizar la experimentación con redes SDN, y también para despliegues reales.

1.3 Ataques de denegación de servicios (DoS).

Los ataques de denegación son aquellos que causan una interrupción o suspensión de uno o varios servicios, esto se consigue mediante el consumo excesivo de uno o varios recursos que tiene el servidor o de los elementos de la red, estos recursos pueden ser: el ancho de banda de la conectividad, Ciclos de CPU o sobre procesamiento, abarrotamiento de la memoria y llenar de información innecesaria el disco, las bases de datos o las tablas internas de un elemento de red.

Normalmente, este efecto se consigue mediante algunas técnicas de envió masivo de tráfico informático, este proceso se lo denomina flooding, al cual el usuario atacado o víctima no puede responder por que el tráfico es demasiado grande en cantidad o tamaño, por lo que este inmediatamente deja de dar servicio, otro modo de conseguir este efecto es explotar las vulnerabilidades de los sistemas operativos o de las aplicaciones. (Macías, 2017, p. 14)

En la actualidad, existen tipos de ataques sencillos los cuales son ejecutados por un solo atacante con origen conocido, en donde este ataque tiene una incidencia menor ya que es fácil de detectar y de mitigar, ya que los servidores y dispositivos pueden soportar la carga para que el servicio no se vea afectado, existe una variante a este tipo de ataque el cual es muy complejo ya que el ataque puede realizarse desde diferentes orígenes al mismo tiempo, denominado ataques de denegación de servicio distribuido (DDoS). (Macías, 2017, p. 33)

Existen otros tipos de ataques para la denegación de servicio, en donde estos no necesitan del envío de información, en cambio estos intentan suministrar paquetes capaces de explotar alguna vulnerabilidad del equipo de la víctima, un ejemplo de este tipo de ataque es el ping de la muerte.

1.3.1 Clasificación de los ataques de denegación de servicio (DoS).

A continuación, se presentará una clasificación de los principales ataques de denegación de servicio que son más relevantes o tienen mayor impacto en una arquitectura SDN, en base a esto se ha establecido varios conjuntos de ataques: reflexión, amplificación y por inundación (son los que tienen un mayor porcentaje de uso según Prolexic Technologies).

1.3.1.1 Denegación de servicio basada en inundación.

Una denegación de servido mediante inundación trata de inyectar o suministrar una gran cantidad de tráfico, este ha sido un tema de interés debido a la facilidad de ejecución y la extensión de su impacto, existen diversas estrategias para conseguir un ataque por inundación de manera eficaz,

una de estas consiste en la generación de trafico de manera constante y uniforme, y la otra consiste en inyectar el tráfico de manera evolutiva. (Paracuellos, 2016, p. 33). Los ataques de inundación cuando operan en la capa de red, explotan funcionalidades de los protocolos de comunicación de las redes, como TCP, UDP, ICMP, DNS.

- Inundación por SYN

Este tipo de ataques explota las vulnerabilidades del protocolo TCP, pues este es utilizado para establecer la conexión y envío del flujo de datos, al ser un protocolo que está orientado a la conexión, el servidor tiene como obligación dar respuesta a una petición SYN sin importar la autenticidad del remitente, para dar respuesta el servidor debe almacenar las peticiones en la memoria y esperar la confirmación por parte del destinatario para establecer el enlace. (Gastón, 2016, p. 2)

Un atacante es capaz de beneficiarse de esta situación para remitir una gran cantidad de peticiones SYN al servidor sin tener ninguna intención de establecer conexión (intentando conectarse a IPs inexistentes), de este modo se formarían conexiones abiertas, llegando a consumir memoria hasta un punto donde el servidor no pueda atender más peticiones, rechazando el acceso a los usuarios e interrumpiendo el servicio, este resultado se lo conoce como denegación de servicio. (Paracuellos, 2016, p. 33)

- Inundación por ICMP

El protocolo ICMP se lo usa para el envío de mensajes que ayudan a identificar errores en la red, al ser mensajes que ayudan a verificar la existencia de errores, son enviados constantemente y además tienen prioridad de procesamiento en el conmutador, por ende, al ser un protocolo que siempre está en uso, un ataque de inundación por ICMP genera una gran cantidad de tráfico, el cual consume el ancho de banda disponible, esta acción origina una denegación de servicio. (Gastón, 2016, p. 2)

Existen variantes de este tipo de ataques como por ejemplo el SMURF, el cual usa mensajes ICMPv4 de difusión y así obligar a varios hosts a responder, si este mensaje es enviado de manera constante y rápida, genera una denegación de servicio debido a que la víctima se encuentra ocupada procesando el tráfico. Otro tipo de ataque es el ping de la muerte el cual usa los mensajes ICMP (echo y request/reply), mediante estos, es posible generar paquetes con una extensión demasiado grande (más de 64Kb), de modo la víctima no los puede procesar generando así una denegación de servicio.

1.3.1.2 Denegación de servicio basada en reflexión.

Surge por la necesidad de los atacantes para ocultar el origen de la intrusión, se los denomina ataques de denegación de servicio distribuida y reflejada (DRDoS), estos tratan de aprovechar alguna vulnerabilidad de terceros para obligarlos a generar trafico malicioso hacia una víctima. Un ejemplo de ataque por reflexión son los smurfing, los cuales son una variante de los ataques de inundación SYN, estos ataques usan los elementos intermedios de la red, para reemplazar la dirección de origen en los paquetes la con la dirección de la víctima. (Paracuellos, 2016, p. 34)

1.3.1.3 Denegación de servicio basado en amplificación.

El ataque basado en amplificación consiste en ejecutar peticiones a terceros para que estos generen paquetes de respuesta con un mayor tamaño que el de las peticiones, y luego falsificar las direcciones de retorno de tal manera que las respuestas envés de regresar al origen, sean enviadas a la víctima. El elemento de la red que se usa para conseguir una amplificación son los servidores DNS, este ataque se lo denomina amplificación DNS. (Paracuellos, 2016, p. 34)

1.3.2 Mecanismo de defensa ante ataques DoS.

Existen diferentes técnicas para la defensa ante un intento de ataques de denegación de servicio, se los puede clasificar dependiendo la situación en que se encuentre el proceso de intrusión, estas situaciones se las puede clasificar como: prevención, detección y mitigación. A continuación, se detallará las técnicas para cada uno.

1.3.2.1 Prevención.

Los mecanismos para la prevención de los ataques DoS son aquellos que actúan antes de que una violación a la seguridad suceda, independientemente de esta función. El objetivo es minimizar al máximo el daño que pueden causar los atacantes. Los DoS son difíciles de detener e incluso detectar, para poder prevenir este tipo de ataques se usan los mismos principios existentes para cualquier violación de seguridad. (Paracuellos, 2016, p. 35)

- Monitorización y comprensión de la plataforma.

Por obvias razones el conocimiento del servicio que se presta en la red es fundamental para poder organizar tareas que ayuden a la prevención de un ataque de seguridad, para ello se debe obtener información primordial como, por ejemplo, número de peticiones en una línea temporal,

localización de los clientes que obtienen el servicio, flujos y puertos de conexiones, ancho de banda usado, valores medios de la memoria, CPU, etc. Estos parámetros se podrán comparar para posteriormente detectar un ataque o comportamiento anormal. (Macías, 2017, p. 34)

La mayor parte de esta información se la obtiene luego de haber realizado una correcta monitorización de la red, este sondeo tiene una doble función, la primera es obtener la información de los parámetros descritos anteriormente y la segunda recoger los datos que ya se encuentran procesados, para luego determinar si el sistema se encuentra bajo algún tipo de ataque malicioso. Cabe recalcar que este sondeo se lo debe realizar periódicamente.

- Diseño de la plataforma y organización de procedimientos

Resulta imposible optimizar o crear un buen diseño de la plataforma, si no se tiene la información de los parámetros descritos en la sección anterior. Estos datos son el punto de partida para poder determinar cómo va a estar compuesta la plataforma (tamaño, topología, dispositivos) para brindar el servicio, por ejemplo, si los parámetros indican que el uso de la CPU es alto, se debe tomar la decisión de incluir dispositivos con velocidades de procesamiento altos, para poder dar tratamiento rápido y eficaz a la información. (Macías, 2017, p. 38)

Una parte del diseño muy importante para evitar los ataques de seguridad es la implementación de políticas hacia el flujo de tráfico, pero este proceso de implementar políticas debe ser minucioso. Muchas veces estas medidas no son suficientes y para ello es necesario implementar algunos dispositivos adiciones que tendrán la función de contrarrestar las violaciones de seguridad como por ejemplo los firewalls de nivel 7, los IDS/IPS, DMZ. Todo esto con el objetivo de minimizar los ataques DoS.

1.3.2.2 Detección.

Realizar una detección oportuna de los ataques de seguridad es de vital importancia, con base a esto deben actuar los componentes defensivos, en la mayoría de los casos la detección recae sobre terceros como, por ejemplo, equipos adicionales en la plataforma (firewall), o a su vez la utilización de programas que brindan el servicio de detección y limpieza (IDS), también existen métodos complejos en los que el administrador de red desarrolla una solución propia mediante la programación de scripts y configurando cada uno de los elementos de la red.

Para el rastreo de ataques DoS se consideran dos paradigmas que están presente en los sistemas de detección de intrusos: reconocimiento de firmas y las anomalías. El primero, reconocimiento

de firmas está basado en la identificación a los patrones de ataques conocidos previamente. El sistema de detección mediante patrones tiene una gran desventaja ya que no permite la detección de nuevas amenazas, porque se basa solo en patrones realizados anteriormente. (Paracuellos, 2016, p. 35)

Debido a esto gran parte de la comunidad investigadora ha preferido el desarrollo de sistemas que se basan en el estudio de reconocimiento de anomalías, este se basa en analizar el comportamiento habitual del sistema, con el fin de determinar las acciones que difieran con el comportamiento normal, para este tipo de detecciones se han propuesto diferentes técnicas como, los modelos de probabilidad basados en Markov, la lógica difusa, la teoría de caos o el estudio de la variación de la entropía. (Paracuellos, 2016, p. 36)

1.3.2.3 Identificación del origen.

Después de haber identificado que el sistema esta siento víctima de un ataque de denegación de servicio, el siguiente paso a tomar es identificar el origen, es decir se trata de averiguar de dónde proviene el ataque. Este proceso casi en todas las ocasiones es muy complicado, ya que el culpable tiene a su disposición diferentes métodos para ocultar su origen, estos pueden variar desde procesos sencillos de suplantación de identidad hasta realizar ataques desde IPs invalidas, Por lo tanto, obtener la dirección de origen es una tarea que muy pocas ocasiones se consigue.

Algunos fabricantes han desarrollado técnicas o algoritmos que permiten identificar el origen, por ejemplo: mediante la medición de "rates" se analiza el número de peticiones y tráfico generado por el cliente, esto sirve para identificar los ataques tipo flooding; otro ejemplo es la clasificación de los clientes por país de procedencia, con sus rangos de IPs públicas, y por último un análisis exhaustivo del comportamiento de cada uno de los clientes para identificar si se repiten patrones similares de tráfico. (Macías, 2017, p. 25)

1.3.2.4 Mitigación.

Luego de conocer los orígenes y el modo en el que se está produciendo el ataque de seguridad, se debe proceder a realizar acciones para mitigar el impacto negativo que se produce sobre el servicio. Una mitigación eficiente se puede producir en el mismo equipo o infraestructura que ha detectado o reenviado trafico malicioso, este proceso también lo pueden realizar otros dispositivos que se encuentren dentro de la plataforma.

Para los ataques de denegación de servicio (DoS), es necesario desplegar una serie de medidas que ayuda a reducir el daño, generalmente consiste en incrementar los recursos físicos, y una actualización de las políticas de cifrado y las listas de acceso. Todo esto con el fin de restauras los servicios de la plataforma que se encuentran comprometidos. (Paracuellos, 2016, p. 23)

La arquitectura SDN brinda una facilidad muy importante para erradicar este tipo de ataques, ya que permite desactivar los conmutadores que se encuentren comprometidos y reemplazarlos por unos nuevos para que realicen la misma función, además SDN tiene un mecanismo apto para eliminar flujos de datos perjudiciales en tiempo real, y bloquear la comunicación con los usuarios que se encuentren desarrollando un proceso de intrusión.

1.4 Introducción a los sistemas de detección y prevención de intrusos.

En este apartado se explicará la importancia y los efectos que tienen los sistemas de detección y prevención ante la presencia de ataques de denegación de servicio DoS, estos sistemas son uno de los mecanismos para la defensa más usados por parte de los administradores de red para minimizar el riesgo de los ataques que están dirigidos hacia los bienes informáticos.

1.4.1 Sistema de detección de intrusos (IDS)

Un sistema de detección de intrusos (IDS) o Intruder Detection System, es una herramienta que monitoriza el tráfico que transita por la red, de este modo se puede identificar los ataques o amenazas que puedan violentar la seguridad y el desempeño de la plataforma. La eficacia de los IDS se basa en la capacidad de la búsqueda y análisis de los patrones de ataques de seguridad que se hayan realizado anteriormente. (Martinez, 2010, p. 12)

Para que un IDS, el cual es ajeno a la red, se lo pueda integrar de manera eficaz y logre funcionar correctamente en conjunto con la plataforma, para realizar la función de detectar posibles intrusiones o amenazas, esta herramienta de seguridad debe estar compuesta de los algunos elementos:

- Fuentes de recolección de datos: su principal objetivo es conseguir todos los paramentos de los paquetes o información que transita por la red, para poder ejecutar el proceso de detección, para lograr estos parámetros se puede asociar con una base de datos.
- Reglas de contenido de datos: las reglas son patrones que se han definido por parte del administrador de red o por una comunidad en el internet para poder detectar las anomalías en el sistema.

- Filtros: este elemento del IDS compara los parámetros obtenidos de los datos con las reglas insertadas.
- Detectores de eventos extraños en el tráfico de la red: este elemento permite que el IDS desempeñe su función de detectar intrusos o amenazas, para que en un futuro afecte a la red.
- Dispositivo generador de alarmas e informes: una de las funciones del IDS es informar al administrador de la red de posibles amenazas, que puede afectar el funcionamiento y desempeño correcto de los elementos en la red.

La utilidad de un IDS puede ser evaluada considerando la probabilidad del sistema en detectar una amenaza y la probabilidad de no cometer errores en la emisión de las alarmas, la gestión y revisión de estos dos parámetros es una tarea muy pesada y multiplica el trabajo para los administradores de los sistemas. (Gimenez, 2008, p. 6)

1.4.1.1 Clasificación de los IDS.

Dependiendo de la función que cumplen los IDS, existen varios enfoques que permiten su clasificación, el más importante se basa en "función de los sistemas que vigilan", es decir realizan un análisis de las actividades del host o una máquina, buscando posibles amenazas, debido a esto surge una subdivisión: IDS basados en red y IDS basados en host.

- IDS basados en red.

También denominados NIDS, estos monitorean los paquetes que transitan por toda la red, buscando indicios de un ataque hacia algún elemento de la red. Los IDS pueden situarse en cualquier terminal o en un elemento por donde pase la mayoría del trafico proveniente del exterior como, la principal ventaja de este tipo de red es que analiza todos los hosts al mismo tiempo, y esta es la principal diferencia con la otra clasificación. (Martinez, 2010, p. 13)

Para analizar el tráfico se lo hace mediante un sniffer el cual es un hardware o software, el cual es capaz de censar el tráfico, para que posteriormente el IDS realice su función, esta captura y análisis del tráfico es en tiempo real y para ello el IDS debe trabajar a nivel del protocolo TCP/IP o capa de trasporte y como punto adicional en la capa de aplicación.

- IDS basados en host.

También llamados HIDS, estos fueron los primeros IDS en ser desarrollados e implementados, operan en base a la información recogida desde una computadora, al ser datos de un solo

dispositivo, permite que el IDS pueda analizar las actividades producidas con una gran precisión, llegando a determinar con exactitud los procesos y usuarios que están involucrados en un ataque de denegación de servicio.

Debido a la tendencia actual del uso de conexiones encriptadas, nace un incuestionable interés en la mejora de la seguridad de los sistemas, porque los sistemas que realizan un monitoreo de la red completa disponen de muy poca información para reconocer el trafico malicioso del aceptable, esta característica permite que los HIDS tengan una gran ventaja ante los NIDS. (Martinez, 2010, p. 62).

1.4.2 Sistema de prevención de intrusos (IPS)

Un sistema de prevención funciona de igual manera que un IDS, la diferencia radica en que el IDS envía alertas al administrador de red cuando se haya detectado una posible violación de seguridad, mientras que el sistema de prevención de intrusos establece políticas de seguridad automáticamente para proteger el dispositivo de red, en otras palabras, se realiza una alerta antes de la detección de un ataque, mientras que el IDS protege luego de que se haya detectado el ataque.

El objetivo de los IPS es lograr la automatización en la respuesta ante la presencia de una violación de seguridad, y además minimizar o intentar anular los efectos negativos del intento de intrusión. En general existen cuatro estrategias para una respuesta activa basadas en red, cada una de estas se usa en las diferentes capas de red. (Gimenez, 2008, p. 20)

- **Enlace de datos:** el IPS automáticamente deshabilita el puerto por donde se está realizando el ataque.
- **Red:** en esta capa el IPS modifica las políticas que maneja el dispositivo controlador, generalmente se lo realiza con un bloqueo de la dirección IP del atacante.
- **Transporte:** realiza una supervisión de los paquetes, mediante la generación de mensajes resets TCP hacia los atacantes que usan el protocolo TCP y mensajes de Port Unreachable para los de ICMP y UDP.
- **Aplicación**: intenta modificar o alterar los paquetes que provienen de un ataque malicioso.

1.4.3 Efectos de integrar un sistema de detección a una red SDN.

Agregar un IDS/IPS a una red cualquiera ayuda a la detección y prevención de los ataques que se suscitan a diario, una red SDN como se explicó anteriormente, se la puede encontrar casi siempre

en un data center de empresas importantes, si asumimos esto, se puede decir que el tráfico que pasa por la red SDN es valioso y delicado.

Si se presenta un ataque a una red definida por software los daños pueden ser muy graves, para dar solución a este problema existen tres mecanismos: un firewall para la prevención, un sistema de detección de intrusos para la detección y copias de seguridad para la recuperación. Si se combina las funciones de detección y un controlador en un solo equipo, es de gran ayuda para cualquier administrador de red.

1.4.4 SNORT

Snort (www.snort.org), es una de las aplicaciones más conocidas de libre distribución, que es capaz de efectuar un análisis en tiempo real del tráfico, y además tener un registro de todos los paquetes que transitan por la red, también puede realizar análisis de protocolos, y coincidencias para poder detectar una gran variedad de ataques, como: (Cruz, 2017)

- Desbordamiento de búfer
- Escaneos de puertos sigilosos
- Ataques CGI
- Análisis SMB
- Intentos de fingerprinting de un sistema operativo.

Esta herramienta se la dio a conocer en 1998 por Martin Roesch, es considerado como la herramienta de seguridad más antigua del mundo. Este software tiene grandes ventajas debido a la fácil implementación y a la hora de brindar soporte, ya que posee la comunidad más grande, esta comunidad o desarrolladores mantienen a la plataforma Snort actualizada, para contrarrestar los ataques más recientes que han surgido. (Cruz, 2017)

Snort es un software gratuito y de código abierto, es decir se puede modificar para adaptar el software a las necesidades de la red, la arquitectura está basado en capas, en donde los paquetes van pasando por cada una de estas, a continuación, en la figura 3-1 se detalla todas las capas. (Viñes, 2005, p. 42)

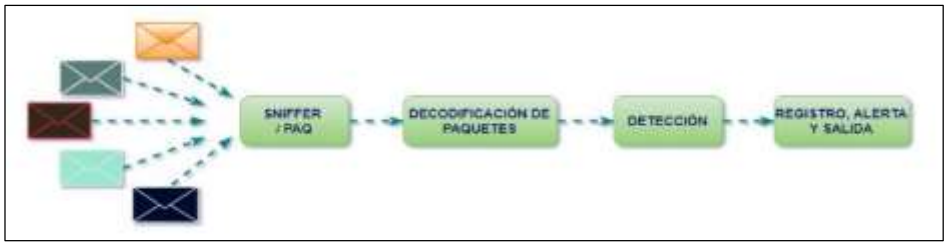


Figura 3-1: Arquitectura de Snort

Realizado por: MORALES, Edwin, 2018

- **Sniffer:** Mediante la librería libpcap se leen los paquetes que pasan por la red, para luego ser enviados a la decodificación.
- **Decodificación de paquetes:** es la encargada de decodificar y guardar los paquetes en una memoria interna para tener acceso.
- **Detección:** se toma los paquetes decodificados y se los compara con las reglas internas definidas en el Snort, cuando el paquete coincide con alguna regla pasa a la siguiente capa.
- **Registro y alerta:** si el paquete ha sido detectado como malicioso se ejecuta una serie de comandos para erradicarlo.

1.4.5 SURICATA

Suricata (https://suricata-ids.org/) es un proyecto de código abierto para un sistema de prevención y detección de intrusos (IDS/IPS), fue desarrollado por Open Information Security Foundation o en sus siglas OISF. La primera versión disponible para descarga gratuita surgió en el 2010, este IDS poseía características muy buenas las cuales cubrían fallas de los demás sistemas de detección y prevención de intrusos que estaban en el mercado, algunas de estas características son: (Astudillo y et al, 2011, p. 15).

- **Multi-Threading:** esta característica procesa los paquetes en varios hilos para aprovechar todos los núcleos de los procesadores actuales.
- **Estadísticas de rendimiento:** se encarga de contar varios elementos de rendimiento (tramas/sec, duración), se almacena esta información para posteriormente mostrar al administrador como estadísticas.
- **Detección de protocolos automático:** Suricata posee palabras claves para algunos protocolos como: IP, UDP, TCP, ICMPP, TLS, FTP, HTTP y SMB. Esta característica ayuda en el control y detección de intrusos.
- **Descompresión Gzip:** descomprime archivos Gzip para examinarlos y detectar patrones de ataques.

- **Unified2 Output:** soporte para herramientas y métodos Unified2, el cual busca reducir la carga al sistema, dejando el trabajo para las soluciones externas como Barnyard
- **IP Reputation:** comparte las direcciones IP maliciosas con las demás organizaciones de seguridad.
- **Windows Binaries:** Suricata puede ejecutar sobre versiones de Windows, pero no es recomendable porque algunas funcionalidades pueden fallar.

El motor IDS/IPS Suricata permite realizar detección de intrusos en tiempo real (IDS), un monitoreo de red (NSM), prevención de intrusos en línea (IPS) y procesamiento pcap sin conexión. Suricata realiza una inspección exhaustiva de tráfico, para ello utiliza reglas extensas, potentes y un soporte para secuencias de comandos lua para la detección de ataques complejos. (Suricata, 2015)

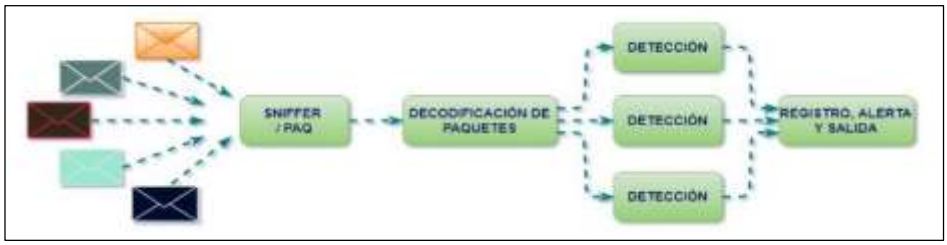


Figura 4-1: Proceso de multi-hilos en Suricata

Realizado por: MORALES, Edwin, 2018

En la figura 4-1, se detalla el proceso que realiza el IDS/IPS Suricata desde que ingresan los paquetes, luego se realiza una decodificación de los mismos, para su posterior análisis por cada uno de los hilos usando la característica Multi-Threading, y finalmente arroja un resultado si es o no un posible paquete malicioso.

1.5 Switch SDN

Actualmente en el mercado existen diversos productos hardware que facilitan la realización de escenarios SDN mediante las tarjetas para experimentación, a continuación, se detallan algunas de estas:

1.5.1 Raspberry Pi 3

La Raspberry Pi 3, como se puede ver en la figura 5-1, es una placa con dimensiones y características aceptables, en donde se le puede dar un sinnúmero de usos. Este tipo de tarjetas pueden realizar funciones de un mini pc, hasta ser un servidor de datos, el principal atractivo que

tiene es su bajo precio con respecto a las especificaciones que brinda. Algunas de las características son: (PcComponentes, 2016)

- Procesador Chipset Bradcom BCM2387 de 4 núcleos a una velocidad de 1.2 GHz.
- Gráficos GPU Broadcam VideoCore IV.
- RAM DDR2 de 1 Giga.
- 4 puertos USB, 1 conector RJ45.
- 802.11b/g/n LAN inalámbrica.
- HDMI rev 1.3 y 1.4.
- Conector GPIO.

Figura 5-1: Raspberry Pi 3 Modelo B

Fuente: https://www.pccomponentes.com/raspberry-pi-3-modelo-b

Una tarjeta Raspberry es la opción más económica para poder realizar el proyecto, pero no es la más óptima, ya que para que realice la función de un switch SDN se le debe instalar el protocolo OpenFlow, el otro impedimento es en la cantidad de puertos Ethernet que posee no puede satisfacer a los requerimientos del escenario o topología planteada.

1.5.2 Tarjeta OnetSwitch 30

Mediante el proyecto OnetSwitch se puede configurar una plataforma Open Source, para la creación de nuevas aplicaciones de red, OnetSwitch es un puerto SBC Quad Gigabit Ethernet del tamaño de una notebook basado en Xilinx Zynq-7000 SoC, que combina la programabilidad del software de los procesadores ARM con la programabilidad de los FPGA, como se observa en la figura 6-1. La tarjeta OnetSwitch consta de: (LinkSprite, 2018)

- Acelerador programable
- FPGA (Sistema eficiente Bitcoin Miner)
- Cinco puertos Gigabit Ethernet

- 3 GB de memoria DRAM DDR3
- Conector SATA e interfaz Mini PCIe para tarjeta WLAN (OpenWrt en Zynq).
- Extensión mini PCI para Wireless.

Figura 6-1: Tarjeta OnetSwitch

Fuente: https://www.kickstarter.com/projects/onetswitch/

La tarjeta OnetSwitch tiene un soporte estable y continuo para la conexión con el controlador Ryu, además como aplicaciones extras se puede conectar con las versiones anteriores de las plataformas OpenDayLight, Floodlight y con NOX, esta tecnología está orientada a los estudiantes, fabricantes e ingenieros, los cuales pueden escribir aplicaciones de software Linux para lograr cualquier función en la red, como NAS, VPN y Firewall.

1.5.3 Tarjeta Zodiac FX

Zodiac FX, figura 7-1, es el conmutador para redes SDN más pequeño del mundo, y es lo suficientemente potente para desarrollar aplicaciones en entornos SDN, puede controlar la prioridad a aplicaciones sensibles, el ancho de banda, transmisiones online, incluso puede dar prioridad a los juegos en línea, esta tarjeta se la puede usar en redes domésticas y en laboratorios experimentales pequeños. Zodiac FX incluye características como: (Zanna, 2015)

- Puertos Fast Ethernet 4 x 10/100 con magnetismo integrado
- Procesador Amtel ATSAM4E Cortex M4
- Soporte para OpenFlow 1.0, 1.3, y 1.4
- Memoria intermedia de 64 KB con almacenamiento no bloqueado y reenvio.

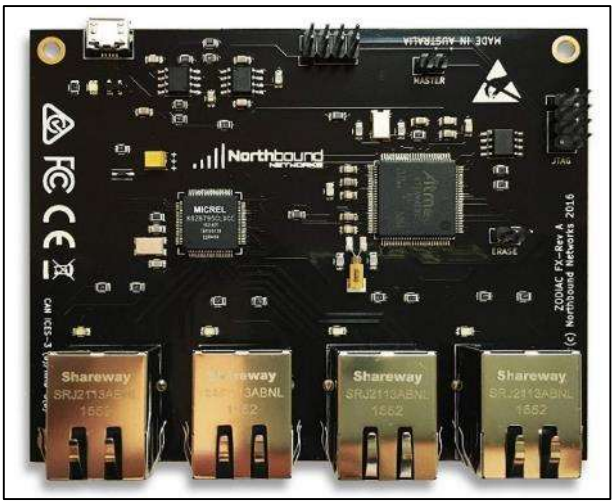


Figura 7-1: Tarjeta Zodiac FX

Fuente: https://www.amazon.com/gp/product/B079MCB16G

1.6 Trabajos previos

En la actualidad las Redes Definidas por Software (SDN) están revolucionando el ámbito de las redes, debido a las mejoras tanto en programabilidad, escalabilidad y bajos costos. Por ello, para el desarrollo del proyecto se basó en información publicada en artículos, documentos científicos, tesis, libros, trabajos realizados, etc…. En este apartado se detallarán algunos trabajos acordes al problema planteado.

- SDN-IPS: Una herramienta para la contención de ataques cibernéticos basada en SDN.

INSERT (Infraestructura de Sistemas para Redes y Telecomunicaciones) creada en 2014 en Brazil, es un grupo de investigación que se centra en las SDN, han realizado un estudio de los Sistemas de Detección de Intrusos con el propósito de crear soluciones de protección para afrontar los ataques cibernéticos. (Insert, 2018)

Consta de, un administrador conectado al controlador, el cual permite: la gestión de topología, flujo, monitorización de tráfico, IPS Suricata, a través del protocolo BGP y el estándar e-Line, y en la parte inferior de la topología se encuentran los conmutadores y usuarios de red. Este proyecto era muy ambicioso pero la limitación en cuanto a recursos monetarios para la adquisición de equipos nunca se pudo implementar físicamente, es decir, solamente fue virtualizado en GNS3, Mininet y Ryu. (Insert, 2018)

- Detección, prevención y mitigación de intrusiones para redes definidas por software: mediante el controlador Floodlight

Fue desarrollado por Pratik Lotia y publicado en la plataforma GitHub. La topología constaba de: 1 switch SDN, 5 hosts, y el controlador Floodligh, al igual que el anterior solo fue virtualizado con las herramientas Mininet y Snort, una vez que se detectaban intentos de intrusión, inmediatamente las alertas eran registradas en un archivo temporal para ser analizadas de acuerdo a reglas establecidas y en caso de no encontrar ninguna coincidencia ser redirigidas al servidor Honeypot. (Lotia, 2017)

- An SDN-based IPS Development Framework in Cloud Networking Environment

Zhengyang Xiong de la Universidad Estatal de Arizona en 2014, se diseñó e implementó un sistema IPS como método de defensa para un entorno de computación en la nube, con el uso de SDN, a fin de tener mejorar en cuanto a seguridad y rendimiento. El prototipo usó el controlador POX, Open Vswitch y Snort. (Xiong, 2014, p. 3)

CAPÍTULO II

2 MARCO METODOLÓGICO

En este capítulo se detalla a trasfondo el diseño metodológico que se usa para la integración de un IDS/IPS a una red SDN, el respectivo escenario físico de la red, el software y hardware necesarios para cada uno de los dispositivos, los cuales deben brindar un buen rendimiento y asegurar la relación de beneficio-costo, de igual manera poder visualizar el resultado de la integración del sistema de detección y prevención de intrusos.

2.1 Metodología de la investigación.

Para el desarrollo del presente proyecto de investigación se emplea un conjunto de técnicas y métodos, que facilitan la recolección, ordenamiento y análisis de los datos obtenidos, estas técnicas y métodos serán detallados a continuación.

2.1.1 Tipo de investigación

El presente trabajo de investigación se estableció como una propuesta tecnológica, la cual tiene sus fundamentos en un tipo de investigación aplicada, se elige este tipo de investigación, ya que el proyecto busca acoplar dos ideas que se encuentran en el mercado, como son las redes SDN y los sistemas IDS/IPS, todo esto con el objetivo principal de mejorar la seguridad en este tipo de redes.

2.1.2 Métodos de investigación.

El presente proyecto es una investigación científica, debido a que está orientado a dar una solución a un problema, con el cumplimiento a ciertos objetivos; se opta en usar los siguientes métodos:

2.1.2.1 Método teórico.

Analítico-sintético: se opta en usar este método, el cual permite realizar un análisis y observación de los dos elementos más importantes dentro de la topología de red: el IDS/IPS y el controlador SDN. Este análisis se lo realizará ante un ataque de denegación servicio sin la presencia de un sistema de detección y prevención de intrusos, y posteriormente se acoplará el IDS/IPS, y se observará su rendimiento.

Inductivo: al aplicar este método al final de la investigación, se podrán exponer conclusiones universales a partir de un análisis de datos particulares, es decir se evaluarán los datos obtenidos del método analítico-sintético.

2.1.2.2 Método empírico.

Observación científica: consiste en la observación y registro de los acontecimientos que se presenten en la implementación del sistema, para su posterior análisis y formular las respectivas conclusiones y recomendaciones.

2.1.3 Técnicas de investigación.

Con el fin de llevar a cabo la presente investigación se aplica técnicas que ayudan a implementar los métodos de investigación descritos en el apartado 2.1.2, para obtener información necesaria previo al desarrollo del proyecto. Algunas de las técnicas a usar son:

- ***Documental:*** mediante esta técnica se recolecta información primaria de libros, revistas, publicaciones, tesis, papers, entre otras; para definir parámetros, elegir dispositivos y softwares aptos para la implementación del sistema.

- **Observación**: esta técnica se la usará luego de realizar la implementación del sistema, pues sirve para analizar el controlador SDN cuando se encuentre bajo un ataque de denegación de servicio, además para poder observar el funcionamiento del sistema IDS/IPS.

2.2 Concepción de la arquitectura de la red

Para el presente proyecto se desarrolla una topología como punto de partida para realizar la integración de un sistema de detección y prevención de intrusos. En la figura 1-2 se detalla el escenario a implementar, los principales elementos de la topología para el correcto funcionamiento.

El escenario consta de seis partes importantes en donde cada una de estas cumplen funciones diferentes, toda esta topología se encuentra conectado mediante el cableado estructurado UTP, y se realiza la comunicación mediante el protocolo OpenFlow.

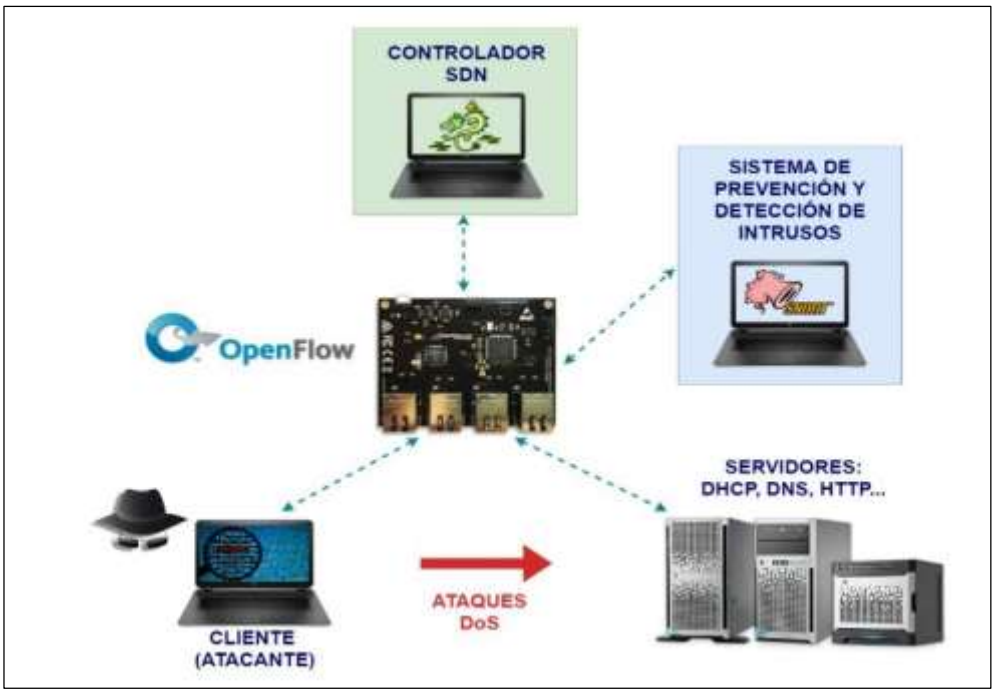


Figura 1-2: Topología para la integración de un IDS/IPS.

Realizado por: MORALES, Edwin, 2018.

Controlador: este es el cerebro de la red, encargado de establecer la conexión con todos los nodos, para este proceso se hace el uso de una plataforma, la cual envía instrucciones o reglas mediante las REST API, y a su vez estas reglas son asumidas y procesadas por el switch SDN.

Switch SDN: cumple la función de conmutación o switcheo, luego de recibir las reglas o instrucciones del controlador, este dispositivo sabrá qué decisión tomar con los paquetes entrantes por cada uno de sus puertos, esta decisión se basa específicamente en las reglas almacenadas en sus tablas de flujo.

Sistema de detección y prevención de intrusos: en esta sección se realiza el control, monitoreo y mitigación de cualquier ataque de seguridad que se detecte en la topología. Esta detección se la hace mediante las reglas de un IDS/IPS.

OpenFlow: es el protocolo de comunicación para las redes SDN, permite él envió de REST API en sentido sur, es decir desde el controlador hacia el switch, y en sentido norte desde el controlador hacia la plataforma SDN o más aplicaciones adicionales.

Servidores web: en un sistema operativo Linux se alojan servidores DHCP, DNS y HTTP, para realizar ataques de denegación de servicio desde los clientes.

Cliente: en esta sección se realiza los ataques de denegación de servicio, mediante aplicaciones que permitan realizar intrusiones DoS.

2.2.1 Funcionamiento de la topología.

Para poder describir el funcionamiento de la topología, debe existir conectividad entre todos los dispositivos o elementos que la conforman, para ello se crean REST API o flows, desde la plataforma del controlador SDN para que generar la conexión. Luego de esto se realiza ataques de denegación de servicio desde el cliente a los servidores, mediante el uso de código o software.

El siguiente paso es configurar mediante reglas de control, el IDS/IPS para que detecte los intrusos, y posteriormente informe de todas estas actividades sospechosas al controlador SDN, y estas alertas sean observadas y tratadas por un administrador de red.

2.3 Recursos que conforman el proyecto.

A continuación, se detalla los recursos necesarios y óptimos para lograr la integración de un sistema de detección y prevención de intrusos (IDS/IPS) a una red SDN.

2.3.1 Análisis de conmutador

Para elegir el conmutador que mejor se ajuste a los requerimientos mínimos, que permita la elaboración del presente proyecto, se toma en cuenta los criterios más relevantes de un conmutador SDN, los cuales se detallan a continuación:

- Compatibilidad con todas las versiones de OpenFlow
- Precio accesible entre 150 y 300
- Procesamiento minino de 1000 paquetes por segundo.
- Numero de puertos necesarios: mínimo cuatro

En base a los parámetros descritos, se realiza un estudio de mercado teniendo como resultado tres dispositivos, los que más se acercan a las especificaciones, los resultados se pueden observar en la tabla 1-2.

Tabla 1-2: Tabla comparativa entre las tarjetas de desarrollo.

Características	Tarjeta Raspberry 3 B+	Tarjeta OnetSwitch 30	Tarjeta Zodiac FX
Precio	$ 50.00	$ 1 284.00	$ 195.00
CPU/ FPGA	Broadcom BCM2387 1.2GHz de 4 núcleos	Dual ARM Cortex-A9 800Mhz	Procesador Amtel ATSAM4E Cortex M4
DRAM	1 GB DDR2	2 GB DDR3	512 para tabla de flujo
Puerto Ethernet	1 x 100Mbps	5 puertos Ethernet Giga	4 x 100 Mbps
Acceso Inalámbrico	Wifi	Mini Pcle WLAN Card	No
Almacenamiento	Tarjeta micro SD	8 GB micro SD/SD /SATA Hard Drive	Soportado
Interfaz SATA	No	SATA / mSATA	No
Controlador compatible	No especificado	Ryu, ODL, Floodlight	Cualquier controlador compatible con OpenFlow.

Fuente: http://revistatelematica.cujae.edu.cu/index.php/tele

Realizado por: MORALES, Edwin, 2018

El siguiente paso es evaluar, cuál es el más apropiado para realizar la elaboración del proyecto. Para ello se usa un método de calificación cuantitativa, con escala del 1 al 5, indicando que 1 es pésimo, 2 es regular, 3 es bueno, 4 es aceptable y 5 es excelente.

Tabla 2-2: Evaluación cuantitativa de la tarjeta de desarrollo.

Características	Tarjeta Raspberry 3 B+	Tarjeta OnetSwitch 30	Tarjeta ZODIAC FX
Precio	5	1	4
CPU/ FPGA	4	5	3
DRAM	4	5	3
Puerto Ethernet	1	5	5
Almacenamiento	4	5	4
Controlador compatible	1	2	5
TOTAL	19	23	**24**

Realizado por: MORALES, Edwin, 2018

La tarjeta de desarrollo más óptima para realizar el proyecto, es la que obtuvo mejor puntaje de la evaluación, como se evidencia en la tabla 2-2, es el dispositivo Zodiac FX, una placa para desarrollar escenarios de red SDN, orientado especialmente para aficionados, investigadores,

desarrolladores, estudiantes, o cualquier tipo de individuo que desee una plataforma de desarrollo de bajo costo. (Networks, 2017)

A pesar de que fue diseñado para ser un simple switch que entienda el protocolo OpenFlow, debido al código abierto del firmware, puede ser utilizado para un sinnúmero de aplicaciones, es decir los usuarios son libres de crear un firmware propio para que el dispositivo pueda realizar diferentes funciones. (Networks, 2017)

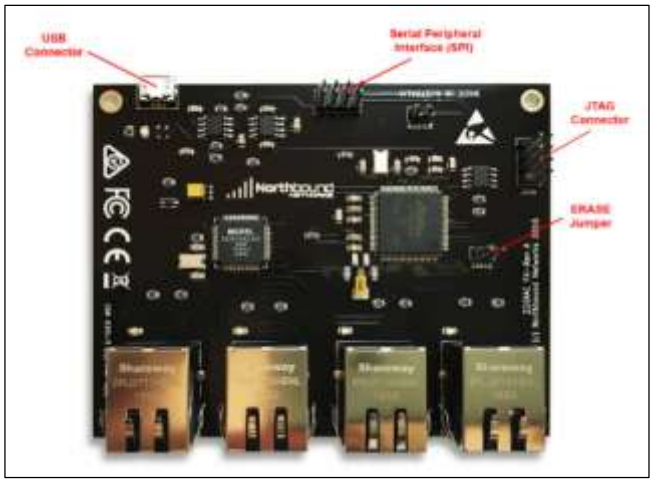


Figura 2-2: Componentes de una tarjeta Zodiac FX

Fuente: www.northboundnetworks.com

En la figura 2-2, se detalla las partes más importantes de la tarjeta Zodiac: el conector USB, el cual sirve para la alimentación de corriente a la tarjeta; el bus SPI, el cual permite la comunicación con otros dispositivos que usen el estándar SPI; el conector JTAG usado para proveer una puerta trasera al sistema para la corrección de errores de código; también se encuentra el jumper de ERASE, el cual permite borrar toda la configuración realizada en la tarjeta, en la tabla 2-2, se especifica con más detalle las características del dispositivo.

Tabla 3-2: Características de la tarjeta Zodiac FX-Rev A

Parámetros	Descripción.
Precio	$ 195.00
CPU/ FPGA	Procesador Amtel ATSAM4E Cortex M4
Velocidad de procesamiento	120 Mhz
Memoria programable	512 para tabla de flujo
Protocolos de comunicación	SPI y I2C
Voltaje de operación (V)	1.62 a 3.6
Dimensiones físicas	100 x 80 mm
Firmware abierto	Código disponible en GitHub

Puerto Ethernet	4 x 100 Mbps
Controlador compatible	Cualquier controlador compatible con OpenFlow.

Fuente: http://revistatelematica.cujae.edu.cu/index.php/tele

Realizado por: MORALES, Edwin, 2018

2.3.2 Análisis del controlador SDN.

Para elegir la plataforma del controlador más óptima de acuerdo a los requerimientos necesarios para la implementación del sistema, se toma en cuenta las características más relevantes las mismas que se detallan a continuación:

- Compatibilidad con las versiones actuales de OpenFlow.
- Manejo de todo tipo de REST API:
- Interfaz grafica
- Código abierto
- Documentación

En base a los aspectos mencionados, se realiza una búsqueda de la plataforma con las características requeridas, esto se puede apreciar en la tabla 4-2.

Tabla 4-2: Características de las plataformas de los controladores.

Características	Floodlight	OpenDayLight	Ryu
Versión de OpenFlow	OpenFlow v1.0	OpenFlow v 1.0	OpenFlow v1.0, v1.2, v1.3 y extensiones Nicra
Virtualización	Mininet y OpenvSwitch	Mininet y OpenvSwitch	Mininet y OpenvSwitch
Lenguaje de desarrollo	Java	Java	Python
Provee REST API	Si	Si	Si (Básica)
Interfaz Grafica	Web	Web	Web adaptable
Soporte de Plataformas	Linux, Mac OS, Windows	Linux, Mac OS, Windows.	Linux
Soporte OpenStack	Si	Si	Si
Multiprocesos	Si	Si	No
Código abierto	Si	Si	Si
Tiempo en el mercado	Desde el 2012	Desde el 2014	Desde el 2013
Documentación	Buena	Media	Excelente

Fuente: http://revistatelematica.cujae.edu.cu/index.php/tele

Realizado por: MORALES, Edwin, 2018

Para conocer cuál es la mejor opción se realiza una evaluación mediante un método cuantitativo, el mismo que se usa en la sección 2.3.1.

Tabla 5-2: Evaluación cuantitativa del controlador SDN

Características	Floodlight	OpenDayLight	Ryu
Soporte OpenFlow	1	1	5
Virtualización	4	4	4
Provee REST API	5	5	4
Interfaz Grafica	5	5	4
Soporte de Plataformas	5	5	4
Código abierto	5	5	5
Documentación	3	3	5
TOTAL	28	28	**31**

Realizado por: MORALES, Edwin, 2018

La plataforma que obtuvo el mejor puntaje de la tabla 5-2, Ryu, y es la más óptima a usar para interactuar con el controlador, figura 3-2, este es un framework de código abierto que brinda las herramientas y bibliotecas que se requiere para crear aplicaciones SDN, el código fuente se encuentra en GitHub, es mantenido y administrado por una comunidad abierta, Ryu está escrito completamente en el lenguaje de programación Python y todo el código está disponible bajo la licencia de apache 2.0. (SDxCentral, 2016)

Figura 3-2: Plataforma Ryu

Fuente: https://osrg.github.io/ryu/

Ryu admite varios protocolos para la administración de red, como NETCONF, OF-config, OpenStack, y el protocolo más antiguo y más usado OpenFlow, este último es capaz de interactuar con el plano de reenvió, es decir conmutadores y enrutadores, para modificar la manera en que maneja la red los diferentes flujos de tráfico. (SDxCentral, 2016)

La plataforma Ryu, puede ser implementada en el sistema operativo Ubuntu 14.04, para ello este S.O. debe tener instalado algunos requerimientos para poder ejecutar Ryu apropiadamente:

- OF-Config: requiere lxml y ncclient.
- NETCONF: requiere paramiko.
- Altavoz BGP (consola SSH): requiere paramiko.
- El servicio de protocolo Zebra (base de datos): requiere SQLAlchemy.

2.3.3 Sistema SNORT para la detección y prevención de intrusos.

El software Snort es gratuito, bajo la licencia de GPL y puede funcionar tanto en sistemas operativos de Windows como en sistemas de UNIX/Linux, este IDS/IPS está probado y es fiable, ya que cuenta con soporte y actualizaciones acorde a nuevas vulnerabilidades. Snort permite la creación de reglas potentes, sencillas y flexibles, la gran ventaja de Snort es que permite o fomenta la compartición de reglas a través del internet para que demás usuarios se beneficien. (Polvereda, 2017, p. 25)

El sistema Snort es fácil de usar, pero existen muchas opciones y no siempre todas van bien juntas, existen tres modos de operación: modo sniffer, modo registro de paquetes y modo de sistema de detección de intrusos de red (NIDS), este último realiza una detección y análisis de trafico de toda la red, este modo es el más complejo y difícil de configurar. (Polvereda, 2017, p. 25)

Luego de instalar el Snort, se agrega un conjunto de reglas para detectar los ataques a la red, si se añaden muchas reglas no significa que detectará más actividades sospechosas, lo contrario muchas reglas genera una sobrecarga en el procesamiento teniendo así una tasa de perdidas muy elevada de paquetes no analizados. A continuación, se muestra la sintaxis para una regla de Snort:

```
<acción> <protocolo> <IP origen> <direccion> < IP destino> < Puerto destino> [(<opción 1;….. ; opción n;.)]
```

En la estructura de las reglas de Snort se especifica todos los detalles que debe cumplir un paquete, en cada política existen dos partes: cabecera y opciones de regla, en la primera se indica las acciones a ejecutarse y en la segunda se encuentra la información necesaria o el condicional para que sea ejecutada, es así que se pueden crear reglas asociadas a la detección de malwares, troyanos, gusanos, virus, ataques de DoS, entre otros. A continuación, en la tabla 6-2, se detalla los parámetros para estructurar una regla.

Tabla 6-2: Parámetros de una regla Snort.

Parámetros	Opciones
Acción	Alert: genera alerta y registra el paquete. Log: solo registra el paquete Pass: ignora el paquete Actívate: active una alerta y llama una regla dinámica Dynamic: funciona cuando se active una regla. Drop: usado en modo inline, elimina el paquete. Reject: usado en modo inline, rechaza el paquete. Sdrop: usado en modo inline, elimina el paquete y no lo registra.
Protocolo	Tcp, Udp, Icmp, Ip
Direcciones IP	IP origen e IP destino o simplemente "any" para todo origen o destino.
Numero de puertos	Todos los puertos de comunicación predeterminados para los diferentes servicios (80:http, 53:udp, 22:ssh, 67 o 68 para icmp, etc…)
Dirección	>: destino <: origen < >: bidireccional
Opciones:	msg, logto, ttl, tos, id, ipoption, fragbits, dsize, flags, seq, ack, itype, icode, icmp_id, icmp_seq, content, content-list, offset, depth, nocase, session, rpc, resp, react, reference, sid, rev, classtype, priority, uricontent, tag, ip_proto, sameip, stateless, regex

Fuente: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/43100/5/fdeharobTFM0615memoria.pdf
Realizado por: MORALES, Edwin, 2018.

En la siguiente viñeta, se muestra un modelo de política para generar alertas cada vez que haya peticiones a través del puerto 80 ya sea desde una dirección IP proveniente del exterior a la red o desde cualquier IP interna, e intente acceder a las contraseñas guardadas en */etc/passwd*, esta acción será tomada como sospechosa y alertada.

- *alert tcp $ETERNAL_NET any → $HOME_NET 80 (msg:" Alguien ha intentado acceder a /etc/passwd"; content:"/etc/passwd";)*

2.3.4 Visualizador de tráfico.

Para poder realizar una visualización y análisis de funcionamiento de la tarjeta Zodiac FX, se opta por usar dos herramientas que son complementarias al escenario planteado, estas herramientas ayudan en gran medida en el análisis de tráfico y a comprender el funcionamiento de un conmutador para redes definidas por software.

2.3.4.1 Wireshark

La primera herramienta a usar es el Wireshark, esta permite la monitorización del tráfico que circula por la red en tiempo real, mediante una interfaz gráfica, como se observa figura 4-2, mediante wireshark se puede extraer información de los paquetes, útil para un administrador, datos como: origen, destino, protocolo, tamaño, puerto, etc…

Para instalar esta herramienta en un sistema operativo Linux, se lo realiza mediante el centro de descargas, o la terminal de sistema digitando los siguientes tres comandos en modo administrador: *sudo add-apt-repository ppa:pi-rho/security, sudo apt-get update, sudo apt-get install wireshark.*

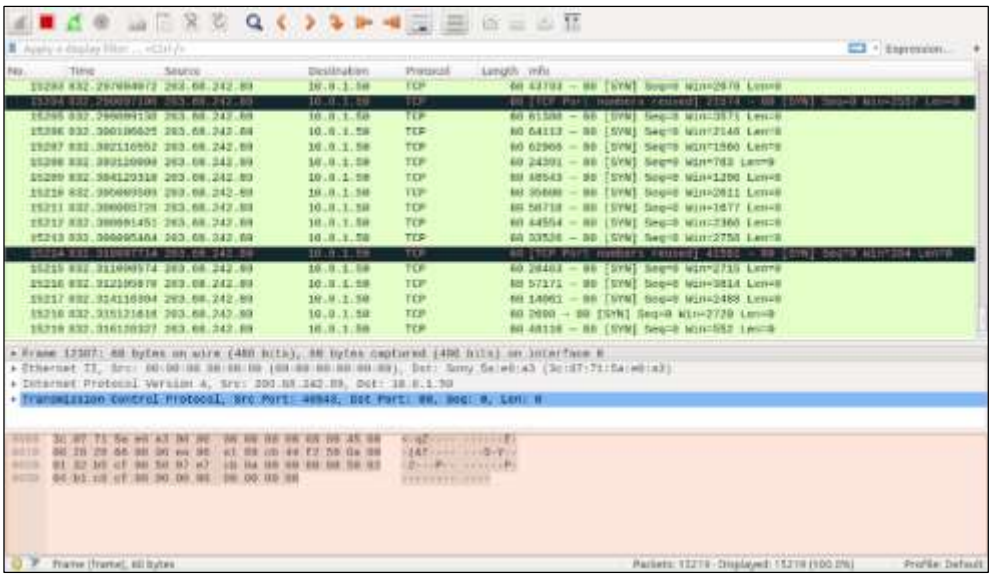


Figura 4-2: Entorno aplicación wireshark.

Realizado por: MORALES, Edwin, 2018.

2.3.4.2 FlowManager.

FlowManager es una aplicación desarrollada específicamente para el controlador RYU, que da al administrador la capacidad tener un control total sobre las tablas de flujo, sin la necesidad de codificar, ya que esta aplicación es con interfaz gráfica de fácil entendimiento para cualquier usuario, como se puede observar en la figura 5-2, algunas características son: (Martimy, 2016)

- Agregar / modificar / eliminar flujos en tablas del conmutador.
- Añadir / modificar / borrar tablas de grupo y medidores.
- Copia de seguridad / restauración de tablas de conmutación a / desde el disco local
- Ver tablas de flujo, tablas de grupos y medidores.

- Ver estadísticas de cambio.
- Ver topología de red.

Para instalar FlowManager en un sistema operativo Linux, la primera opción es descargar de la dirección web https://github.com/martimy/flowmanager, o la segunda mediante la terminal digitar el siguiente comando, *git clone https://github.com/martimy/flowmanager*,

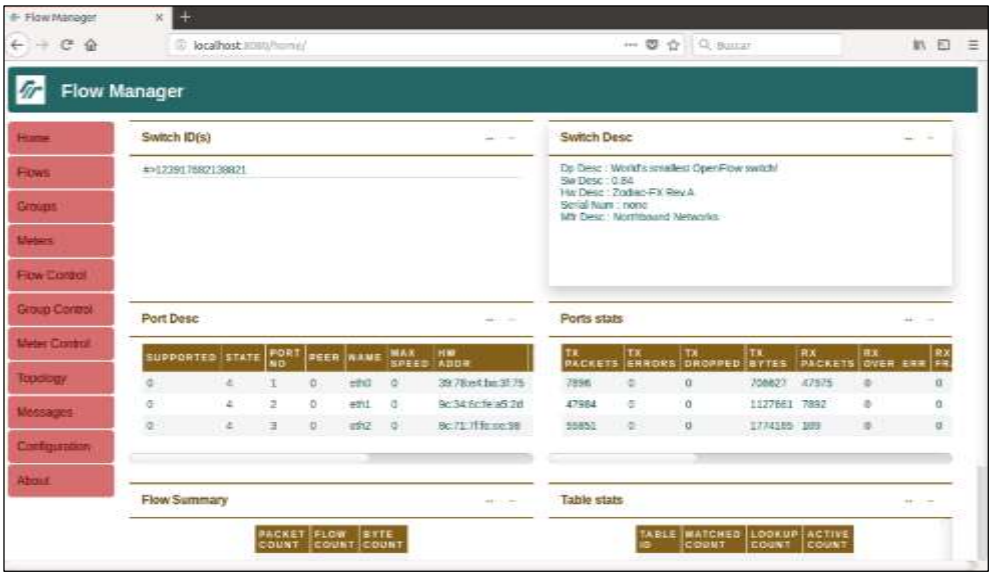


Figura 5-2: Entorno aplicación FlowManager.

Realizado por: MORALES, Edwin, 2018.

2.3.5 Sistema Operativo Ubuntu 14.04 LTS.

La plataforma Ryu solo está disponible para sistemas operativos Linux, por ello se opta usar Ubuntu 14.04 LTS, el cual está basado en la arquitectura Debían, Ubuntu en un S.O de acceso libre para computadoras, lo puede utilizar un usuario sin tener grandes conocimientos gracias a la simplicidad y de fácil interactividad, además existe mucha documentación gratuita para comprender el funcionamiento de todas herramientas disponibles.

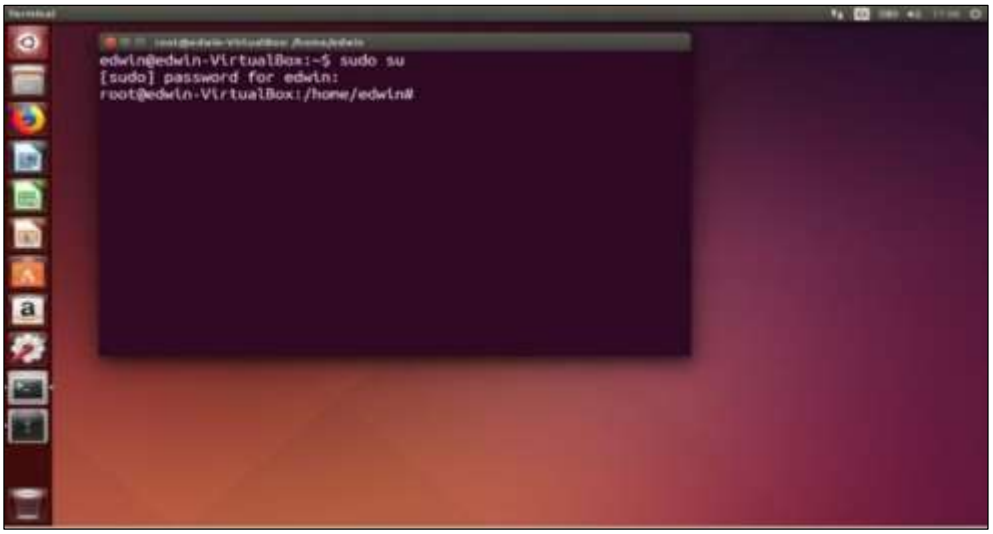


Figura 6-2: Entorno de trabajo de Ubuntu

Realizado por: MORALES, Edwin, 2018.

El patrocinador del sistema operativo Ubuntu, es la empresa "Canonical Ltd", que brinda soporte constante para corregir errores en programación, seguridad y diseño, debido a esto se publican nuevas versiones corrigiendo errores cada seis meses, en la figura 6-2 se observa el entorno de trabajo de un sistema operativo Ubuntu.

2.4 Desarrollo

Después de definir la metodología de investigación, y la selección de software y hardware necesario, se empieza con la implementación, todo lo que concierne a este tema se lo detalla minuciosamente en este apartado, empezando desde el escenario, luego las configuraciones y por ultimo las pruebas pertinentes para evaluar el escenario.

2.4.1 Escenario y direccionamiento de la red.

Como primer punto para el desarrollo del trabajo de titulación, es crear el escenario, en la figura 7-2, se detalla cuáles son los componentes que conforman la red, las direcciones IP y los puertos de conexión,

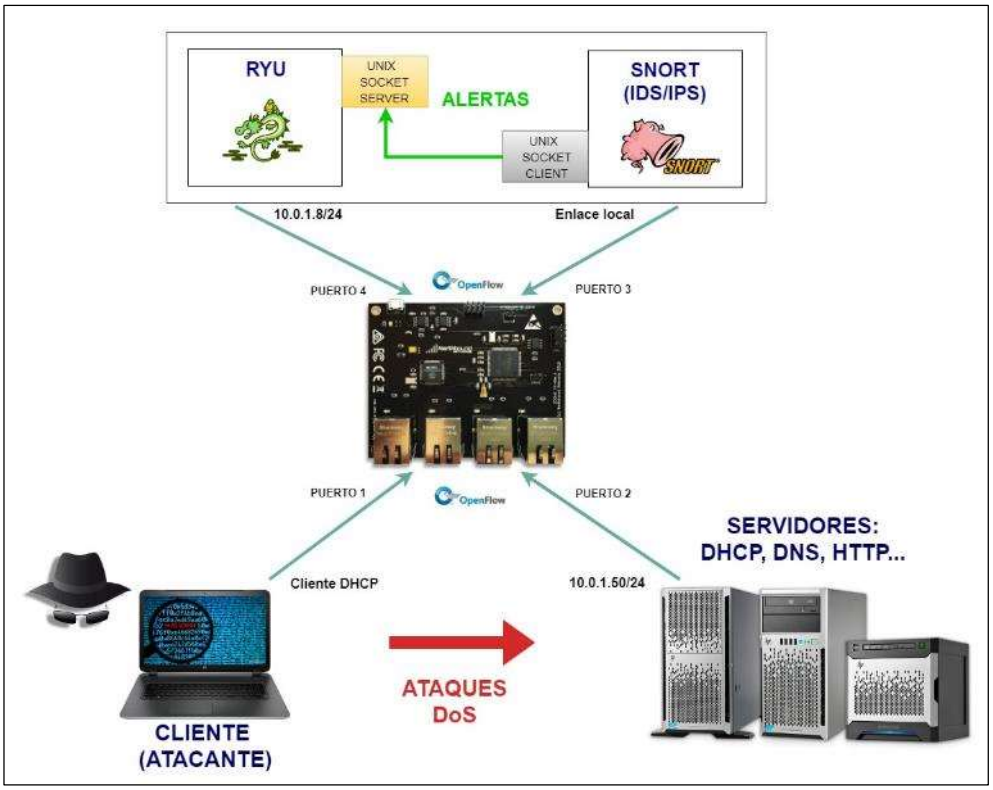


Figura 7-2: Escenario a desarrollar

Realizado por: MORALES, Edwin, 2018.

El escenario de la figura 7-2, es el más idóneo para poder trabajar con la tarjera Zodiac FX adquirida, en ella se puede conectar los dispositivos necesarios, las características de estos elementos se detallan en la tabla 7-2.

Tabla 7-2: Direccionamiento de la red.

Elementos de Red	Sistema Operativo	Dirección IP	MAC tarjeta de red
Enrutador SDN (s1)	Tarjeta Zodiac FX-Rev A	10.0.1.99/24	70-B3-D5-6C-DE-C5
Controlador (c0)	Sistema Operativo Ubuntu 14.04 LTS	10.0.1.8: puerto 6633	00-27-0E-07-43-10
IDS/IPS (h3)	Sistema Operativo Ubuntu 14.04 LTS	Enlace local	00-E0-4C-37-06-0C
Servidores	Sistema Operativo Centos7	10.0.1.50	30-65-EC-2A-1C-0D
Atacante (h1 y h2)	Sistema Operativo Kali Linux	Cliente DHCP	3C-07-71-5A-E0-A3

Realizado por: MORALES, Edwin, 2018

2.4.2 Configuración de la tarjeta Zodiac FX.

La tarjeta para realizar el presente trabajo de titulación se la adquiere desde la compañía de comercio electrónico Amazon, en la fotografía 1-2: se puede observar el contenido de la caja, sin la funda de protección estática, el switch es pequeño así que hay que tener mucho cuidado con él, para la alimentación se lo realiza mediante un puerto micro USB.

Fotografía 1-2: Contenido del paquete comprado en amazon

Realizado por: MORALES, Edwin, 2018

La tarjeta Zodiac Fx, para que funcione como switch, viene acoplada con cuatro puertos Ethernet, donde los tres primeros, fotografía 2-2, contando de izquierda-derecha, son para los dispositivos terminales (PCs), el ultimo o cuarto puerto sirve únicamente para el controlador SDN.

Fotografía 2-2: Tarjeta Zodiac FX

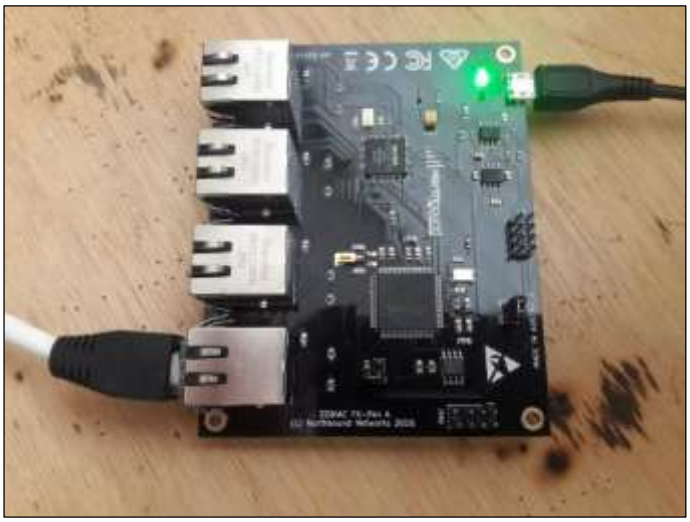
Realizado por: MORALES, Edwin, 2018

El mejor lugar para encontrar información útil sobre la tarjeta Zodiac FX, es en la web oficial del desarrollador Northbound (https://northboundnetworks.com), en la sección Soporte/Community Forum, contiene información actualizada sobre los controladores, guía de usuario, firmware, etc...

En el apéndice A de la guía del usuario que se encuentra en la dirección web mencionada, se halla la información de la configuración de fábrica de la tarjeta. (Networks, 2017)

- Dirección IP: 10.0.1.99
- Máscara de red: 255.255.255.0
- Puerta de enlace predeterminada: 10.0.1.1
- Controlador OpenFlow: 10.0.1.8
- Puerto OpenFlow: 6633
- VLAN: 100 para los dispositivos terminales en los puertos 1,2,3 y 200 para el controlador en el puerto 4.

Para empezar la configuración, la tarjeta Zodiac FX se debe conectar mediante el puerto micro USB a una computadora y mediante un cable ethernet, desde el puerto número cuatro de la tarjeta al puerto ethernet de la máquina, como se observa en la fotografía 3-2.



Fotografía 3-2: Configurando tarjeta Zodiac FX

Fuente: MORALES, Edwin, 2018

En la computadora se debe configurar la dirección IP 10.0.1.8 con la máscara de subred 255.255.255.0 y dirección por defecto 10.0.1.1, ahora mediante el uso de un navegador web se ingresa a la dirección: http://10.0.1.99, como se muestra en la figura 8-2.

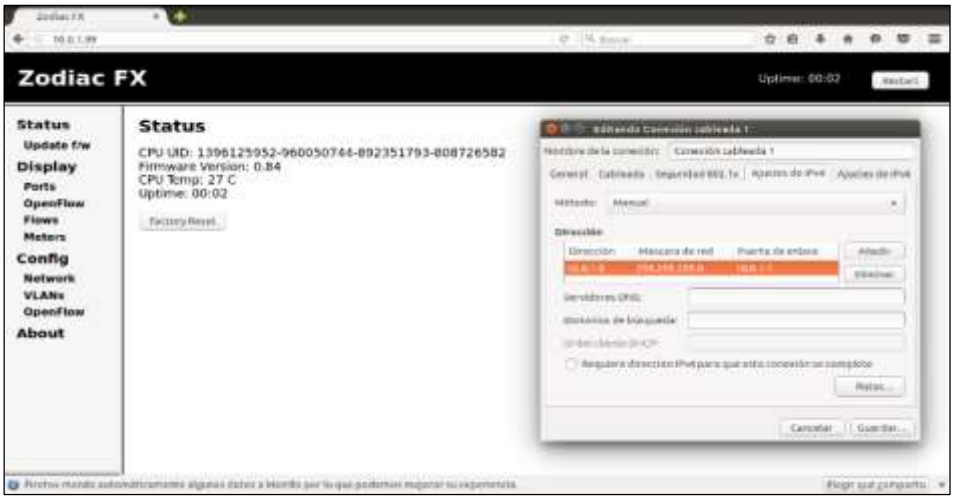


Figura 8-2: Configuración de la tarjeta de red para la Zodiac FX

Realizado por: MORALES, Edwin, 2018.

2.4.2.1 Actualización del firmware

Antes de empezar a usar la tarjeta como un switch SDN, se debe instalar el firmware, es decir el sistema operativo para la Zodiac FX, hasta la fecha, año 2018 la versión actual es la 0.84, en caso de que la posea una versión anterior se debe realizar el cambio respectivo a la versión actual.

Para realizar una actualización del firmware, se debe descargar desde la web oficial en la sección Soporte/Community Forum/Zodiac FX General, figura 9-2, la primera publicación, hay dos maneras de instalar el firmware, a través de SAM-BA y la opción más recomendada y que se usara, es mediante la dirección web, dependiendo de la elección anterior se escoge el archivo *.bin*. para la actualización.

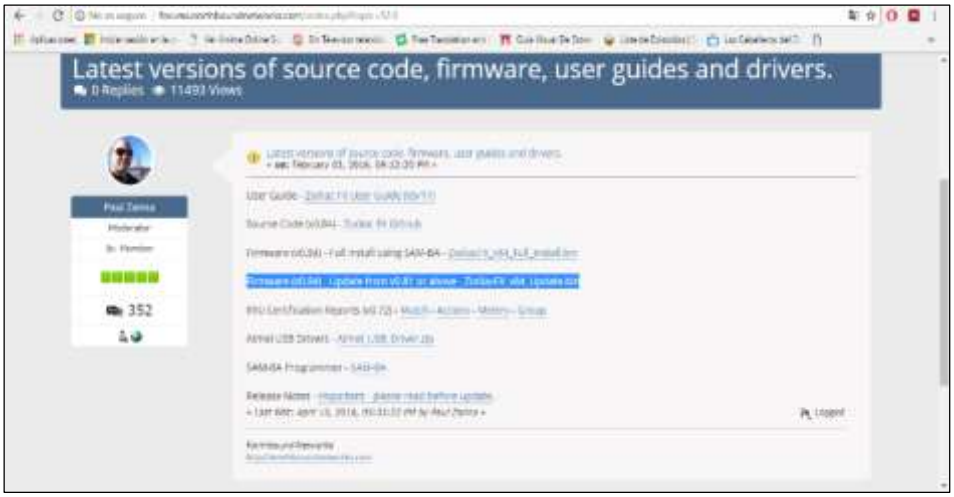


Figura 9-2: Descarga del firmware para la Zodiac FX.

Realizado por: MORALES, Edwin, 2018.

Luego en la página del dispositivo, sección Update f/w, se selecciona el archivo *.bin* descargado y se elige la opción Upload File, se acepta el mensaje de alerta, figura 10-2, y por ultimo aparece un cuadro de reiniciar la tarjeta Zodiac, y listo la tarjeta esta lista para ser usada como un switch SDN, con la versión más actual hasta la fecha.

Figura 10-2: Instalación del firmware para la Zodiac FX.

Realizado por: MORALES, Edwin, 2018.

2.4.3 Instalación de la plataforma Ryu.

El siguiente paso, es tener una PC con características aceptables, tabla 8-2, para que realice la función de controlador SDN, en está pc se debe instalar el sistema operativo Ubuntu 14.04 LTS.

Tabla 8-2: Características de la PC para el controlador.

Parámetros	Valores
Procesador	Intel Core 2 Duo de 2.93Ghz
RAM	2 Gb
Gráficos	Intel G41 x86/MMX/SE2
Almacenamiento	500 Gb

Realizado por: MORALES, Edwin, 2018

Con el sistema operativo Ubuntu 14.04 LTS, se debe instalar la plataforma Ryu para que realice la función de controlador SDN, y para ello en la figura 11-2, se detalla los comandos necesarios para lograr la instalación del Ryu y todos sus complementos.

```
#Autor : Edwin Morales Davila
#Correo : edwin_morales94@hotmail.com
#Programa : instalacion del controlador RYU

#-----------instalar paquetes necesarios para el RYU
sudo apt-get update
sudo apt-get -y install git python-pip python-dev
sudo apt-get -y install python-eventlet python-routes python-webob python-paramiko
sudo apt-get -y install openvswitch-switch

#-----------Descargar RYU desde GitHub
git clone https://github.com/osrg/ryu.git
cd ryu/

#-----------Instalar RYU y sus componentes
sudo pip install six --upgrade
sudo pip install eventlet --upgrade
sudo pip install oslo.config msgpack-python
sudo pip install -r tools/pip-requires
sudo python ./setup.py install
```

Figura 11-2: Pasos para la instalación de la plataforma Ryu.

Realizado por: MORALES, Edwin, 2018.

Para ejecutar el Ryu, se crea una aplicación en el lenguaje de programación Python, esta aplicación puede ser creada desde cero, para ello se debe tener conocimientos avanzados de programación, o a su vez usar o modificar las aplicaciones creadas por defecto que se encuentran en *carpeta personal/ryu/ryu/app*, como se observa en la figura 12-2,

Figura 12-2: Aplicaciones para implementar en la plataforma RYU.

Realizado por: MORALES, Edwin, 2018.

Para el desarrollo de este proyecto, titulado integración de un IDS/IPS, específicamente se usa el sistema Snort, tal como se indica en el apartado 2-3-3. Para su posterior comunicación con el controlador Ryu, se desarrolla una aplicación, que se llama *simple_switch_snort.py*, el funcionamiento se lo describe en la tabla 9-2.

Tabla 9-2: Descripción de las líneas de código de simple_switch_snort.py.

Descripción	Código
>Estas dos líneas de comandos importan las librerías para la codificación mediante Python.	from __future__ import print_function import array
>Se importa las librerías necesarias para ejecutar el controlador Ryu	from ryu.base import app_manager from ryu.controller import ofp_event from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
>Se importa librerías para: usar el protocolo openflow, los paquetes ipv4, tcp, udp, icmp y para el snort. >Estas librerias se encuentran en la dirección */ryu/ryu/lib*. >Todas estas librerías son documentos con código escritos en Python.	from ryu.controller.handler import set_ev_cls from ryu.ofproto import ofproto_v1_3 from ryu.lib.packet import packet from ryu.lib.packet import ethernet from ryu.lib.packet import ipv4 from ryu.lib. packet import icmp from ryu.lib. packet import tcp from ryu.lib. packet import udp from ryu.lib import snortlib
Indica la versión del openflow que se usa para la comunicación, y la librería para enlazar con el archivo de detecciones temporal creado por el snort.	class SimpleSwitchSnort(app_manager.RyuApp): OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION] _CONTEXTS = {'snortlib': snortlib.SnortLib}
En esta sección se debe indicar el puerto del switch al cual está conectado el IDS	def __init__(self, *args, **kwargs): super(SimpleSwitchSnort, self).__init__(*args, **kwargs) self.snort = kwargs['snortlib'] self.snort_port = 3 self.mac_to_port = {}
Esta aplicación puede ser usada en modo unixsock (True) cuando el snort está junto al ryu, y network (False) cuando están en PCs diferentes.	socket_config = {'unixsock': True} self.snort.set_config(socket_config) self.snort.start_socket_server()

>Luego de que esta aplicación ha detectado el archivo temporal del snort, empieza a extraer la información y a ordenarla en los tipos de paquetes ICMP, TCP, IP.
 	
>Si se desea agregar más paquetes hay que añadir las librerías, y codificar en esta sección para extraer la información del archivo temporal del snort.
	def packet_print(self, pkt):
	pkt = packet.Packet(array.array('B', pkt))
	eth = pkt.get_protocol(ethernet.ethernet)
	_ipv4 = pkt.get_protocol(ipv4.ipv4)
	_icmp = pkt.get_protocol(icmp.icmp)
	_tcp = pkt.get_protocol(tcp.tcp)
	_udp = pkt.get_protocol(udp.udp)
	if _icmp:
	self.logger.info("%r", _icmp)
	if _tcp:
	self.logger.info("%r", _tcp)
	if _udp:
	self.logger.info("%r", _udp)

	if _ipv4:
	self.logger.info("%r", _ipv4)
	if eth:
	self.logger.info("%r", eth)

Con este código se envía la información ordenada, para visualizarla en la consola del controlador Ryu.	@set_ev_cls(snortlib.EventAlert, MAIN_DISPATCHER) def _dump_alert(self, ev): msg = ev.msg print('alertmsg: %s' % ''.join(msg.alertmsg)) self.packet_print(msg.pkt)

>estos comandos sirven para que el controlador Ryu se pueda conectar con el conmutador, e intercambien información acerca de las tablas flows, y las características físicas de switch.
 	
>Además esta sección detecta si ha sucedido algún cambio en el switch, estos cambios pueden ser tales como: desconexión de un puerto, Id de switch, flows que envia el Ryu, etc…
	@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
	def switch_features_handler(self, ev):
	datapath = ev.msg.datapath
	ofproto = datapath.ofproto
	parser = datapath.ofproto_parser
	match = parser.OFPMatch()
	actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER, ofproto.OFPCML_NO_BUFFER)]
	self.add_flow(datapath, 0, match, actions)
	def add_flow(self, datapath, priority, match, actions):
	ofproto = datapath.ofproto
	parser = datapath.ofproto_parser
	inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
	mod = parser.OFPFlowMod(datapath=datapath, priority=priority, match=match, instructions=inst)
	datapath.send_msg(mod)

>Esta sección empieza a funcionar en caso de que no se haya podido conectar a un archivo temporal de snort, inmeditamente el ryu procesa todo el tráfico pero no da información acerca del tipo, tamaño, ip y más de origen o destino.	@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER) def _packet_in_handler(self, ev): msg = ev.msg datapath = msg.datapath ofproto = datapath.ofproto parser = datapath.ofproto_parser in_port = msg.match['in_port'] pkt = packet.Packet(msg.data) eth = pkt.get_protocols(ethernet.ethernet)[0] dst = eth.dst src = eth.src dpid = datapath.id self.mac_to_port.setdefault(dpid, {})

>En esta sección el Ryu encuentra las MAC de los host conectados en cada uno de los puertos para luego insertarlos en la tabla de flujo.	self.mac_to_port[dpid][src] = in_port
	if dst in self.mac_to_port[dpid]:
	out_port = self.mac_to_port[dpid][dst]
	else:
	out_port = ofproto.OFPP_FLOOD
	actions = [parser.OFPActionOutput(out_port),
	parser.OFPActionOutput(self.snort_port)]

Realizado por: MORALES, Edwin, 2018

El código de *simple_switch_snort.py*, en su totalidad se lo aprecia en el anexo A. El siguiente paso, es ejecutar esta aplicación con el controlador Ryu mediante la terminal del Ubuntu. Pero para lograr este cometido, se debe usar la herramienta *ryu-manager*, la cual debe ejecutarse exitosamente tal como se muestra en la figura 13-2, y en caso de inconvenientes, verificar la correcta instalación del Ryu.

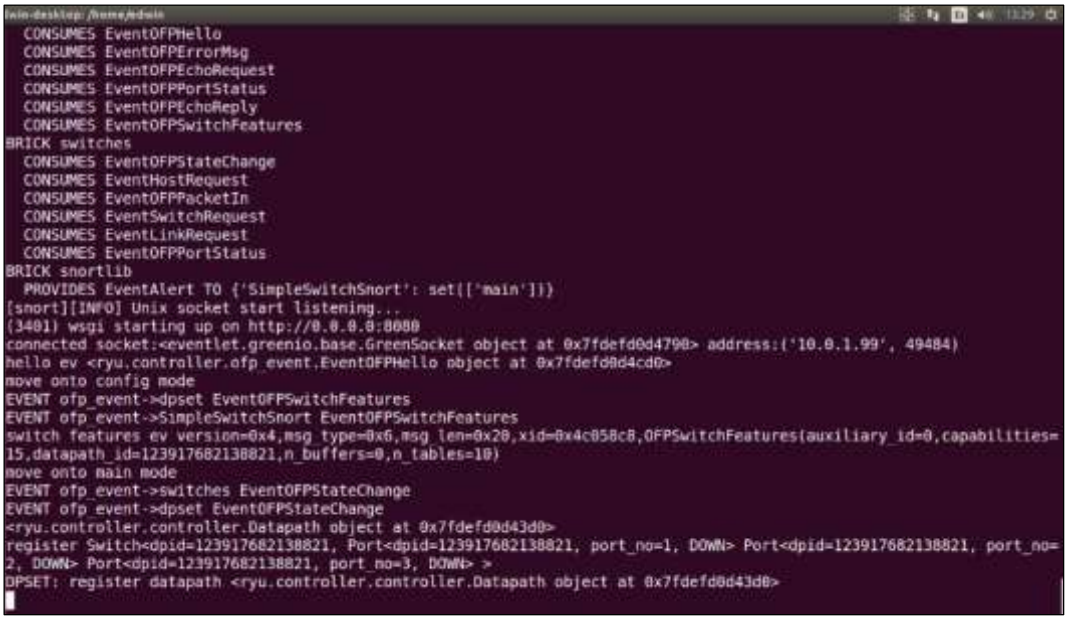


Figura 13-2: Ejecutando plataforma RYU.

Realizado por: MORALES, Edwin, 2018.

Para comprobar la conectividad entre el RYU a la tarjeta Zodiac, se ejecuta el siguiente comando: *ryu-manager --verbose flowmanager/flowmanager.py ryu/ryu/app/simple_switch_snort.py*, luego se hace uso de wireshark, que permite observar el tráfico generado en el enlace conmutador-controlador y se llega a la deducción que se generan tres tipos de paquetes, en donde:

El primero va desde la tarjeta zodiac (10.0.1.99) hasta el controlador (10.0.1.8) mediante el protocolo OpenFlow y es un paquete del tipo OFTP_ECHO_REQUEST, esta trama es una consulta, el segundo va desde el controlador hasta la tarjeta mediante el protocolo OpenFlow, del

tipo OFTP_ECHO_REPLY, este paquete es una respuesta a la consulta, y por último el tercer va desde la tarjeta hacia el controlador mediante el protocolo TCP y puerto 6633, se envía directamente a la interfaz gráfica del RYU. Esta información se observar en la figura 14-2.

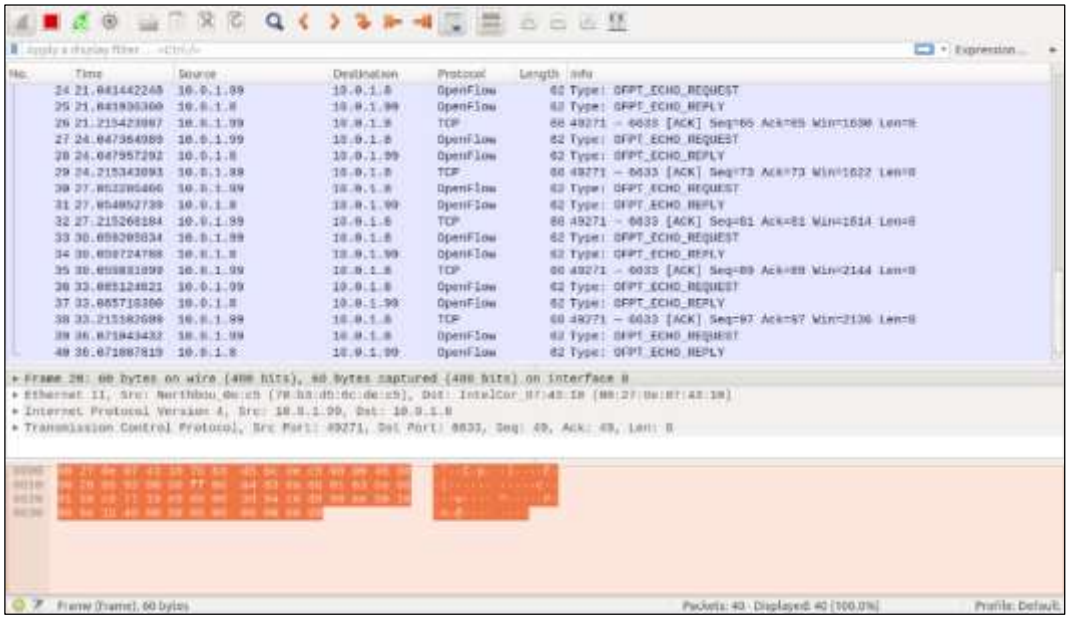


Figura 14-2: Paquetes OpenFlow del controlador observados en wireshark

Realizado por: MORALES, Edwin, 2018.

La comunicación que hay entre la tarjeta y el controlador siempre está representada por las tres tramas, en la primera, la Zodiac pregunta la acción y en el segundo es la respuesta del RYU, luego de que la regla llegue al conmutador, este lo ejecuta y toma una decisión con el paquete que está siendo tratado, no importa el tipo, protocolo, puerto o mensaje de la trama, siempre se va a seguir el mismo procedimiento.

2.4.4 Instalación de IDS/IPS.

Después, en la PC donde se instaló previamente el controlador RYU, se realiza la instalación del sistema de detección y prevención de intrusos SNORT junto a todos sus componentes necesarios para su correcto funcionamiento, para ello se ejecuta los comandos en secuencia como se describe en la figura 15-2.

```
instalarSnort.sh
#Autor : Edwin Morales Davila
#correo : edwin_morales94@hotmail.com
#programa : instalacion del SNORT

#-------------actualizacion de sistema operativo
sudo apt-get update
sudo apt-get dist-upgrade -y
sudo apt-get install -y openssh-server

#modificaciones en la tarjeta de red
sudo nano /etc/network/interfaces
  #ingresar estos cuatro comandos en el archivo y guardar
  auto eth0
  iface eth0 inet dhcp
  post-up ethtool -K eth0 gro off
  post-up ethtool -K eth0 lro off
#######################################
sudo reboot
ethtool -k eth0 | grep receive-offload

#instalacion de librerias para el SNORT
sudo apt-get install build-essential
sudo apt-get install libpcap-dev libpcre3-dev libdumbnet-dev
sudo apt-get install  bison flex

#instalacion del SNORT
sudo apt-get install snort
```

Figura 15-2: Pasos para la instalación del IDS snort.

Realizado por: MORALES, Edwin, 2018.

Una vez que el Snort está instalado, el siguiente paso es crear las reglas o normas para que controlen la red y detecten todas las posibles amenazas o intrusiones, para ello se crea un fichero que contenga todas estas normas, figura 16-2, este fichero de nombre *reglasIDS.rules* debe estar en la dirección */etc/snort/rules*.

```
# Agreement (v 2.0)
#
#----------------
# COMMUNITY RULES
#----------------

# alert tcp $HOME_NET 2589 -> $EXTERNAL_NET any
(msg:"MALWARE-BACKDOOR - Dagger 1.4.0";
flow:to_client,established; content:"2|00 00 00 06 00 00
00|Drives|24 00|"; depth:16; metadata:ruleset community;
classtype:misc-activity; sid:105; rev:14;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET 7597
(msg:"MALWARE-BACKDOOR QAZ Worm Client Login access";
flow:to_server,established; content:"qazwsx.hsq";
metadata:ruleset community; reference:mcafee,98775;
classtype:misc-activity; sid:108; rev:11;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET 12345:12346
(msg:"MALWARE-BACKDOOR netbus getinfo";
flow:to_server,established; content:"GetInfo|0D|";
metadata:ruleset community; classtype:trojan-activity;
sid:110; rev:10;)
# alert tcp $HOME_NET 20034 -> $EXTERNAL_NET any
(msg:"MALWARE-BACKDOOR NetBus Pro 2.0 connection
established"; flow:to_client,established;
flowbits:isset,backdoor.netbus_2.connect; content:"8N|10
00 02 00|"; depth:6; content:"|05 00|"; depth:2;
offset:8; metadata:ruleset community; classtype:trojan-
```

```
alert icmp any any -> any any (msg:"ICMP test";
sid:10000001;)
alert tcp any any -> any any (msg:"acceso port 80";
sid:10000002;)
```

Figura 16-2: Reglas de la comunidad Snort vs reglas propias.

Realizado por: MORALES, Edwin, 2018.

Estas reglas pueden ser de autoría propia, para ello se realiza un previo análisis de la red e incluir normas adecuadas de acuerdo a las necesidades de la topología, la otra opción y más recomendada, es usar las reglas que brindan la comunidad Snort, figura 16-2, estas normas son actualizadas continuamente y se las puede encontrar en la página oficial del sistema de detección.

Luego para que el SNORT ejecute las reglas, se las incluye en el archivo de configuración de Snort, para ello se abre el archivo *snort.conf* en la dirección */etc/snort*, al final de este archivo se debe colocar una línea de comando *include $RULE_PATH/reglasIDS.rules o community.rules*, como se muestra en la figura 17-2.

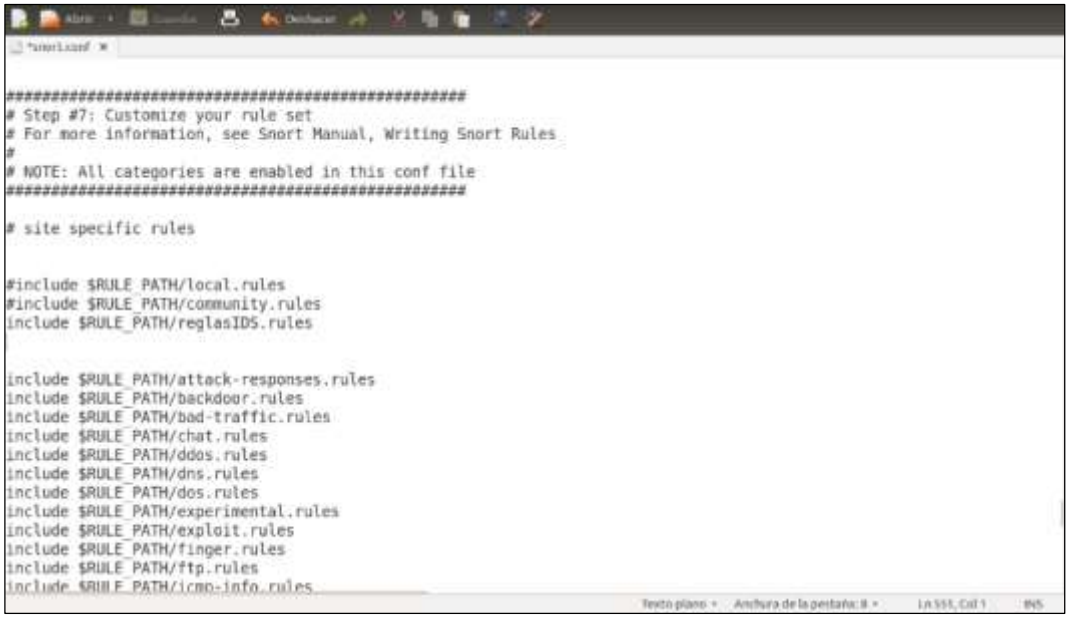


Figura 17-2: Adición de reglas al archivo snort.conf.

Realizado por: MORALES, Edwin, 2018.

Ahora ya se puede iniciar el Snort para que empiece a analizar todo el tráfico mediante el siguiente comando *snort -i eth1 -A unsock -l /tmp -c /etc/snort/snort.conf*, de acuerdo a la figura 18-2, en esta línea de comando ejecutado se debe detallar; que puerto es analizado "*eth1"*, *unsock* indica que se archive las detecciones y */tmp* la dirección del archivo temporal.

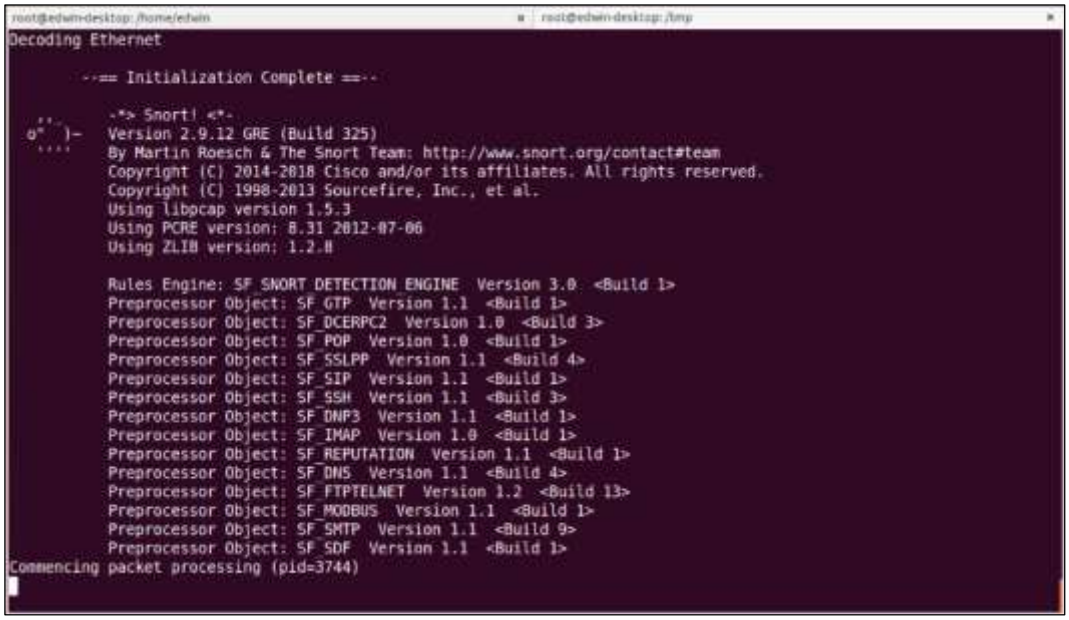


Figura 18-2: Iniciando el snort para la detección de tráfico.

Realizado por: MORALES, Edwin, 2018.

2.4.4.1 Integrar IDS/IPS

Una vez que los elementos de la red (controlador, IDS), estén funcionando correctamente lo siguiente es que se envie información de las detecciones de los ataques al controlador, es decir realizar una integración entre los dos elementos, o que exista comunicación. Para lograr esto, se debe acoplar una aplicación llamada *PigRelay*, el cual se encuentra en el repositorio de GitHub, en la figura 19-2, se muestra los pasos para realizar la instalación.

InstalarSnort.sh ×

```
#autor : Edwin Morales Davila
#correo : edwin_morales94@hotmail.com
#programa : instalar pigrelay

#instalar la plataforma de git
sudo apt-get install git

#descargar pigrelay
git clone https://github.com/John-Lin/pigrelay.git
```

Figura 19-2: Pasos para la instalación del Pigrelay.

Realizado por: MORALES, Edwin, 2018.

La aplicación pigrelay debe conocer la dirección de los archivos temporales que proporciona el SNORT cuándo se ejecuta, estos archivos se encuentran en el directorio */tmp*, como se observa

en la figura 20-2, este archivo temporal es variable, es decir cada vez que se ejecuta el snort se crea un archivo diferente.

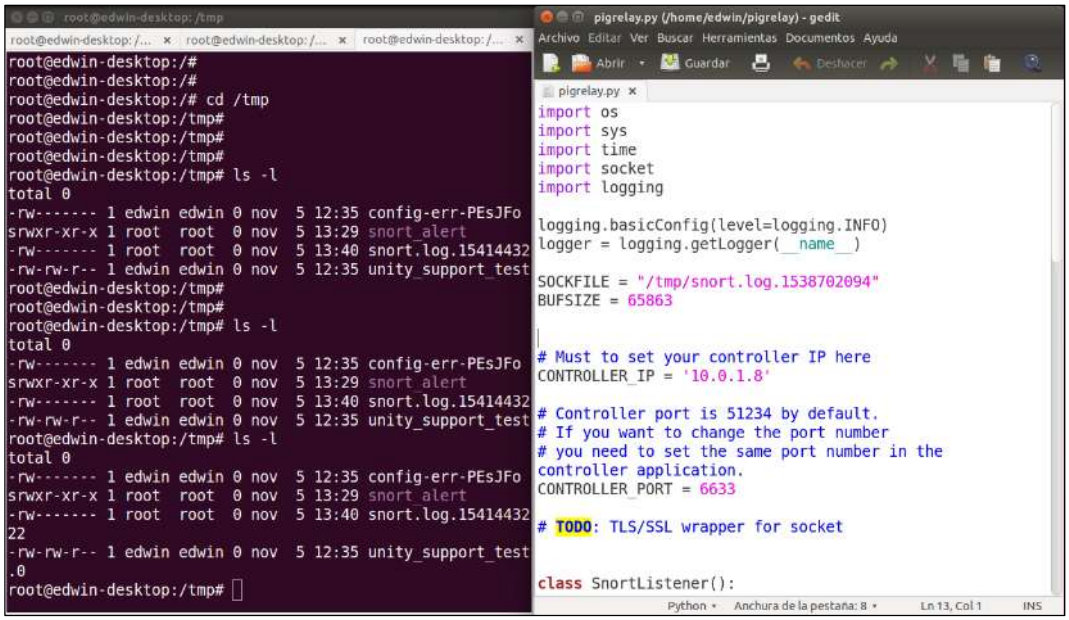


Figura 20-2: Archivos temporales del snort y configuración del pigrelay.

Realizado por: MORALES, Edwin, 2018.

En el archivo *pigrelay.py*, ubicado en el directorio */home/usuario/pigrelay*, se debe colocar la dirección del archivo temporal del snort en el parámetro sockfile, el tamaño del buffer en bufsize, el cual indica la velocidad del envió de la información del snort al controlador. Lo siguiente es indicar la dirección IP del controlador y el puerto por el cual se está comunicando, estos parámetros modificados se lo indican en el anexo B.

Luego de realizar las modificaciones, se guarda y ejecuta esta aplicación mediante el comando, *sudo python pigrelay.py*, esta aplicación toma los registros encriptados del snort que se encuentran en un archivo temporal, como se muestra en la figura 21-2, y posteriormente realiza el envío al controlador.

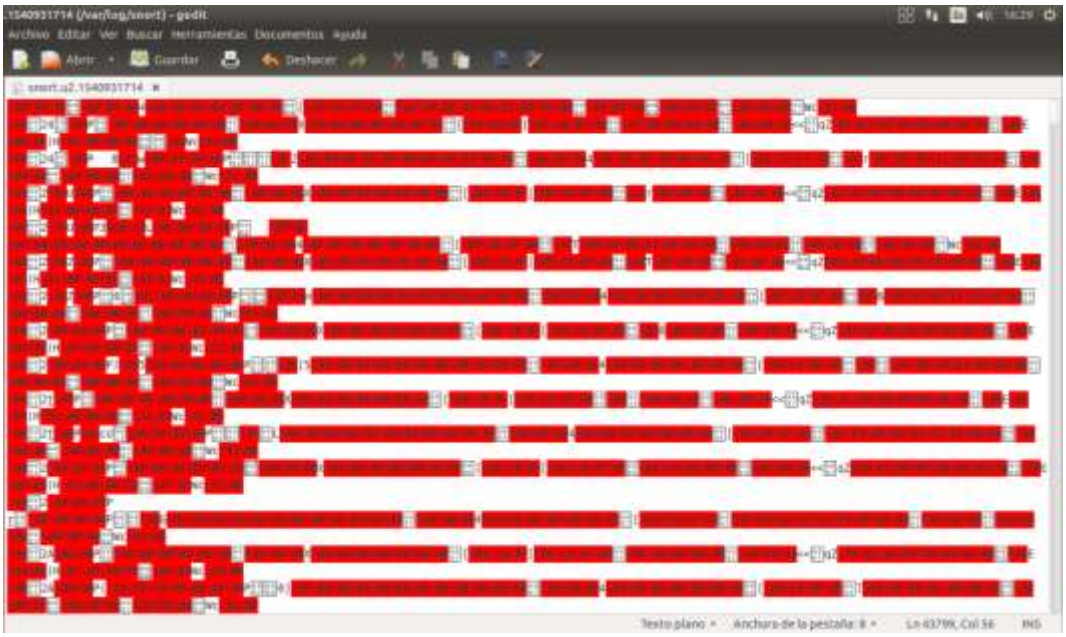


Figura 21-2: datos del archivo temporal del Snort.

Realizado por: MORALES, Edwin, 2018.

Luego de que se envía la información al controlador RYU, este la descifra, diferencia el tipo de paquete, con sus respectivos parámetros y por ultimo organiza esta información para mostrarla en la consola del administrador de red, como se evidencia en la figura 22-2, todo este proceso se lo realiza mediante codificación en la aplicación *simple_switch_snort.py*. que se detalla en el anexo A.

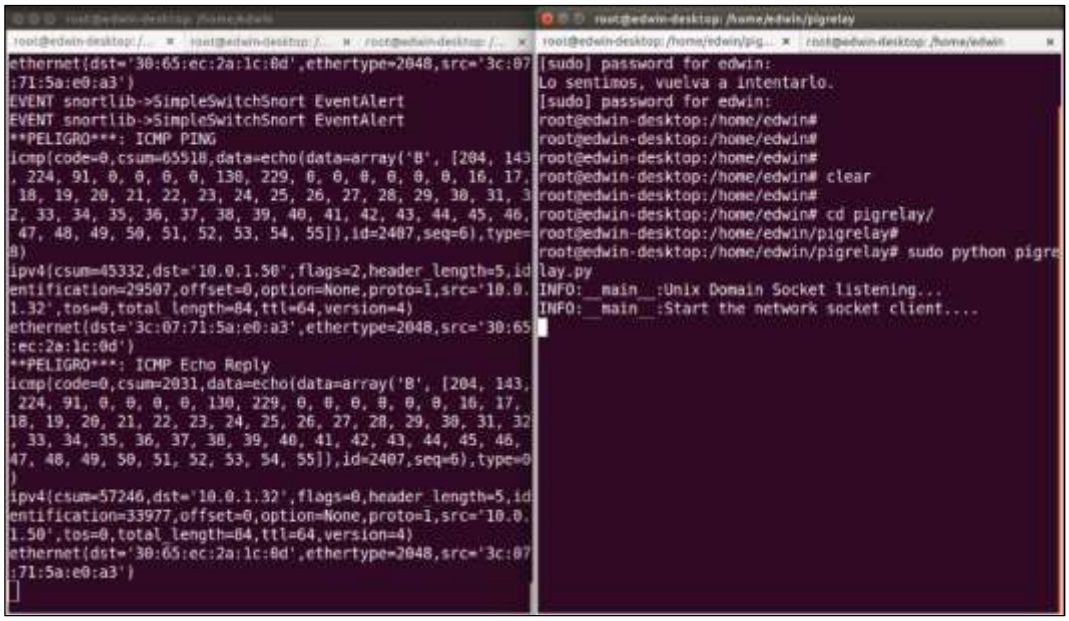


Figura 22-2: Ryu recibiendo alertas del snort mediante el pigrelay.

Realizado por: MORALES, Edwin, 2018.

2.4.5 Ataques

En esta sección se enlista los ataques de denegación de servicio más comunes que afectan a las redes definidas por software.

Tabla 10-2: Ataques DoS más comunes

Nombre	Capa OSI a atacar	Tipo	Descripción
ICMP echo request flood	L3 - capa de red	Recursos	Es él envió de bastantes paquetes, los cuales involucran una respuesta de la víctima, se debe devolver un paquete con el contenido idéntico al del origen.
IP packet fragment attack	L3 - capa de red	Recursos	Es el envío de paquetes IP que llaman automáticamente a otros paquetes, de esta manera se genera un bucle infinito saturando la memoria de la víctima.
SMURF	L3 - capa de red	Ancho de Banda	Es un ataque mediante la saturación ICMP, que roba la dirección de origen, para redirigir múltiples respuestas maliciosas hacia la víctima.
IGMP flood	L3 - capa de red	Recursos	Se trata del envío de grandes cantidades de paquetes IGMP, es cuál es el protocolo para la administración de grupos de internet.
Ping of Death	L3 - capa de red	Explotación	Es el envío de paquetes modificados de ICMP, los cuales explotan vulnerabilidades del sistema operativo,
TCP SYN Flood	L4 - capa de transporte	Recursos	Es el envío de grandes cantidades de solicitudes de conexión mediante el protocolo TCP.
TCP Spoofed SYN Flood	L4 - capa de transporte	Recursos	Es el envío de solicitudes de conexión del protocolo TCP, robando la dirección de origen.
TCP SYN ACK Reflection Flood	L4 - capa de transporte	Ancho de Banda	Son envíos masivos de solicitudes de conexión TCP, varios destinos, cambiando la dirección de origen por la de la víctima, el ancho de banda queda saturado por las respuestas, a dichas peticiones.
TCP Fragmented	L4 - capa de transporte	Recursos	Es el envío de paquetes TCP que se remiten automáticamente a otros paquetes, llegando a saturar la memoria de la víctima.
UDP Flood	L4 - capa de transporte	Ancho de Banda	Envió de grandes cantidades de paquetes UDP, sin haber establecido una conexión previa.
UDP Fragment Flood	L4 - capa de transporte	Recursos	Es el envío de datagramas que invocan automáticamente a otros datagramas, llegando a saturar la memoria de víctima.
Distributed DNS Amplification Attack	L7 - capa de aplicación	Ancho de Banda	Es el envío masivo de solicitudes DNS, desde una dirección de origen usurpada, hacia varios servidores DNS legítimos. Es un ataque de amplificación.

DNS Flood	L7 - capa de aplicación	Recursos	Es el ataque hacia un servidor DNS, a través del envío masivo de peticiones.
HTTP(S) GET/POST Flood	L7 - capa de aplicación	Recursos	Es el ataque a un servidor Web, a través del envío masivo de peticiones.
DDoS DNS	L7 - capa de aplicación	Recursos	Se realiza un ataque desde un gran número de máquinas hacia un servidor DNS mediante el envío masivo de peticiones.

Fuente: https://openwebinars.net/blog/hacking-tutorial-como-hacer-ataque-ddos/

Realizado por: MORALES, Edwin, 2018

Como se puede observar en la tabla 10-2, estas amenazas tienen como objetivo comprometer a diferentes capas del modelo OSI, mediante ataques a diferentes protocolos que funcionan en cada una de estas capas. Para probar la topología creada, se opta por usar ataques enfocados en la capa de red y transporte, es decir ataques a los protocolos ICPM, TCP y UDP.

Para realizar estos ataques de denegación de servicio (DoS) se usó la herramienta Metasploit V4.17.3, figura 23-2, que viene incluida en el sistema operativo Kali Linux 2018, esta herramienta contiene 1795 exploits de todo tipo y aproximadamente unos 100 solo para ataques DoS.

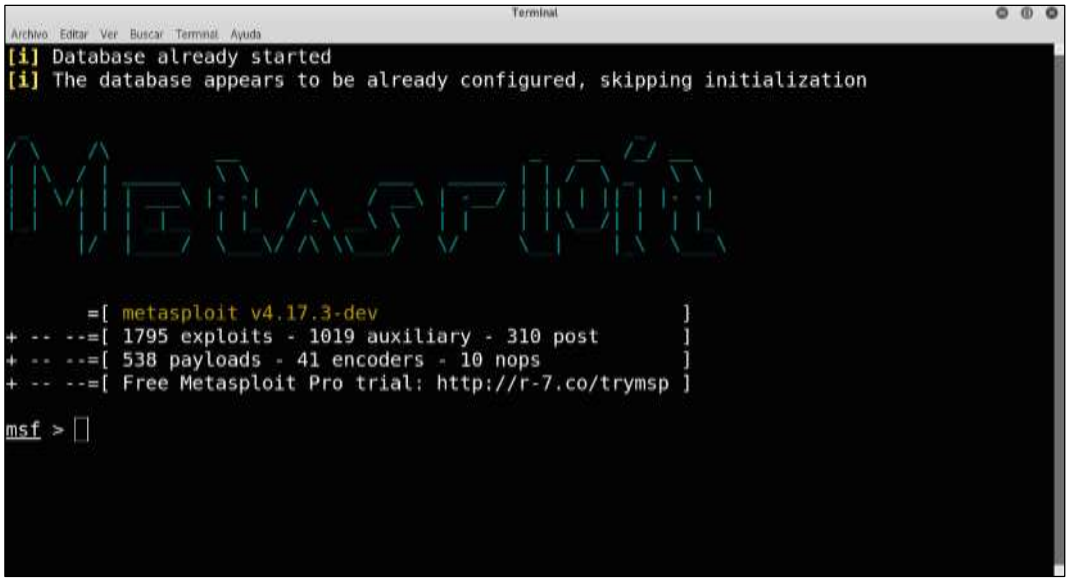


Figura 23-2: Herramienta Metasploit.

Realizado por: MORALES, Edwin, 2018.

2.5 Pruebas de interoperabilidad

Para determinar el correcto funcionamiento de la topología SDN desarrollada, se opta por levantar servicios en un host y acceder desde otro. Se han elegido tres servicios, ya que son los más

comunes que se encuentra en cualquier red sea tradicional o definida por software, además en esta sección mediante wireshark se realiza pruebas para entender la comunicación que realiza la tarjeta con el controlador.

2.5.1 Creación de servidores DHCP, DNS y HTTP.

Para el levantamiento de estos servicios se lo hace en la distribución CentOS de Linux, ya que es uno de los más estables y confiables, además la documentación es una de las más extensas.

2.5.1.1 Servidor DHCP.

El servicio de DHCP (protocolo de configuración dinámica de host), es del tipo cliente/servidor, en donde el servidor establece dinámicamente parámetros de direccionamiento IP (dirección IP, mascara de subred, puerta de enlace, servidores DNS, etc.), este protocolo se encuentra documentado en la norma RFC-1531, los mensajes DHCP usan el puerto 67 en UDP para servidores y 68 en UDP para clientes.

Luego de tener la topología o escenario en funcionamiento, se comienza a instalar el servidor DHCP, para ello una guía detallada se encuentra en el anexo C, el siguiente paso es realizar una verificación de conexión del cliente con el servicio, esto se puede evidenciar en la figura 24-2.

Figura 24-2: Verificación del servicio de DCHP

Realizado por: MORALES, Edwin, 2018.

2.5.1.2 Servidor DNS.

El sistema de nombre de dominio o DNS, consiste en la traducción de nombres de dominio a direcciones IP o viceversa, este proceso lo realiza mediante una base de datos en donde se almacenan todas las direcciones IP, con sus respectivos nombres de dominio. Los servidores DNS usa el protocolo UDP en el puerto 53 para responder todas las consultas unilaterales desde los clientes o servidores, el protocolo TCP se lo usa cuando el tamaño de los paquetes excede los 512 bytes,

En el anexo C se puede encontrar los pasos a seguir para crear un servidor DNS. Para determinar su correcto funcionamiento se realizó la traducción de la dirección IP 10.0.1.50 a un dominio de página web, que se llama *www.tesis-ids-ips.com,* luego mediante el comando *nslookup* mas la direccion Ip del servidor, se puede generar peticiones por el puerto 53 con el objetivo de corroborar el su correcto funcionamiento, los resultados se observan en la figura 25-2

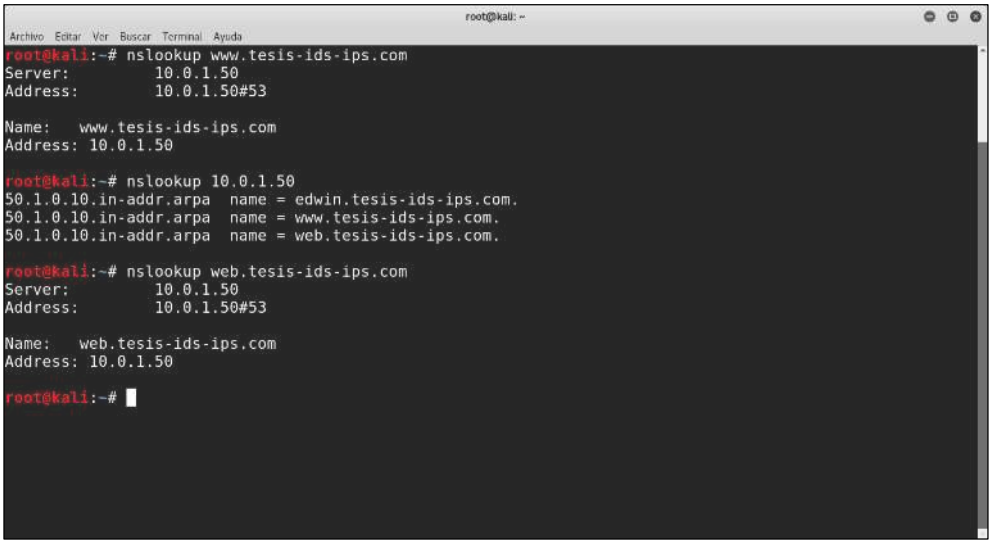


Figura 25-2: Verificación del servicio DNS.

Realizado por: MORALES, Edwin, 2018.

2.5.1.3 Servidor HTTP.

Un servidor web o HTTP, es un servicio de cliente-servidor, que brinda datos en forma de páginas web o html, los cuales están compuestos de textos complejos, figuras, enlaces, botones, objetos animados, etc.., este servidor se complementa con servidor DNS, mediante el protocolo HTTP, la comunicación desde el cliente al servidor se lo realiza mediante el protocolo TCP por el puerto 80 y 8080,

De la mima forma que los anteriores, el procedimiento para la creación del servidor HTTP se encuentra en el anexo C. en la figura 26-2, se puede observar el diseño de la página web, usando una plantilla escrita en código HTML, en este servidor se crea solo la dirección web, y el servidor DNS hace el trabajo de traducción a dirección IP.

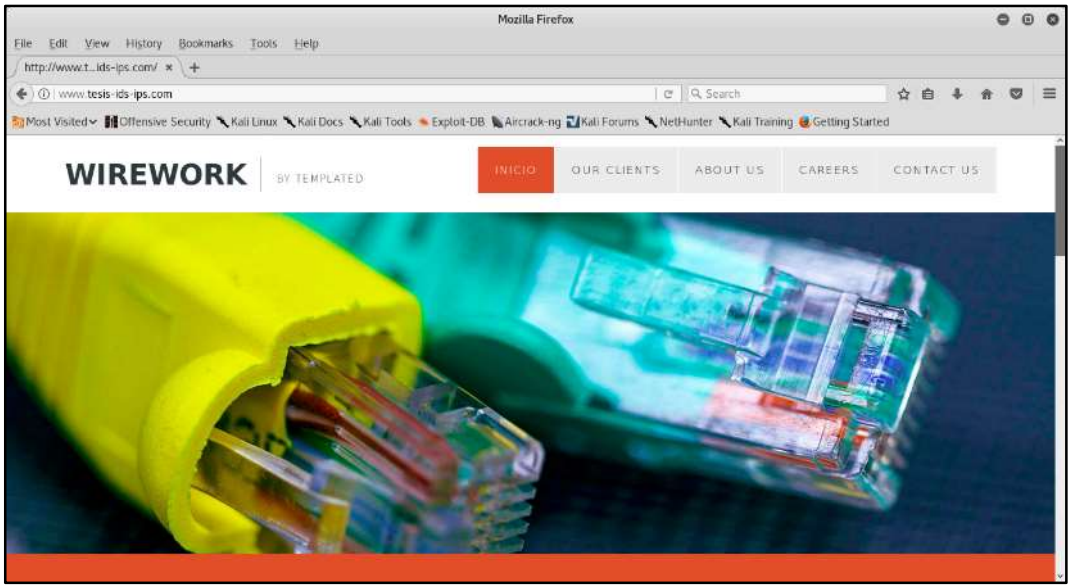


Figura 26-2: Verificación del servicio HTTP

Realizado por: MORALES, Edwin, 2018.

2.5.2 Detección de protocolos ICMP, TCP, UDP en IPv4.

Luego de implementar y configurar los servidores, se comienza con las pruebas de identificación de los protocolos y puertos de comunicación, para esto se crea reglas específicas para poder ser implementadas en el SNORT, y mediante estas realizar un match entre reglas y tipos de paquetes, en la figura 27-2, se muestra las reglas.

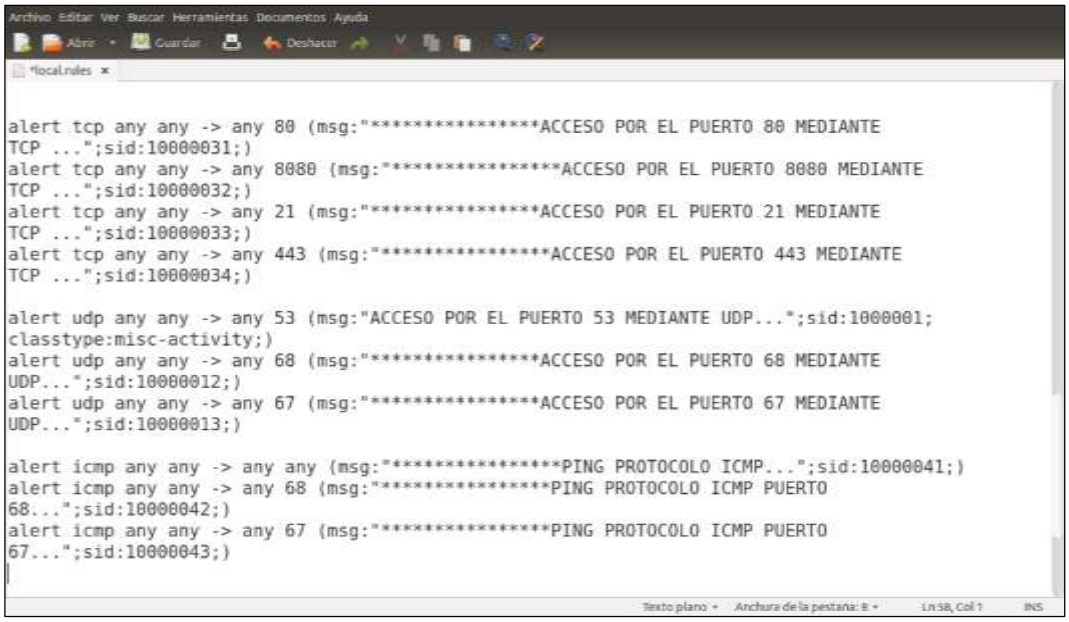


Figura 27-2: Herramienta Metasploit.

Realizado por: MORALES, Edwin, 2018.

2.6 Pruebas de rendimiento

Para determinar un óptimo rendimiento, en la integración del SNORT al RYU, al escenario se le pone a prueba mediante la ejecución de ataques de denegación de servicio, estos ataques están enfocados a los servicios levantados anteriormente, es decir al DNS y HTTP.

2.6.1 Detección de Ataques DNS.

Para el primer ataque DNS, figura 28-2, se usó el exploit *bind_tkey*, el cual envía consultas al servidor DNS con paquetes que tienen un formato incorrecto, este intenta explotar las vulnerabilidades de las consultas tkey del servidor, el ataque bind_tkey es una vulnerabilidad encontrada en las versiones bind de 9.1.0 hasta 9.10.2.

Figura 28-2: Ataque Bind_tkey desde metasploit

Realizado por: MORALES, Edwin, 2018.

Para el segundo ataque DNS, figura 29-2, se usó el exploit *bind_tsig*, el cual explota una vulnerabilidad en el formato los paquetes de reconocimiento DNS, cuando se analiza el buffer.c, en base a esto el exploit bind_tsig construye una respuesta que cumple con los requisitos para ser enviada de regreso al servidor.

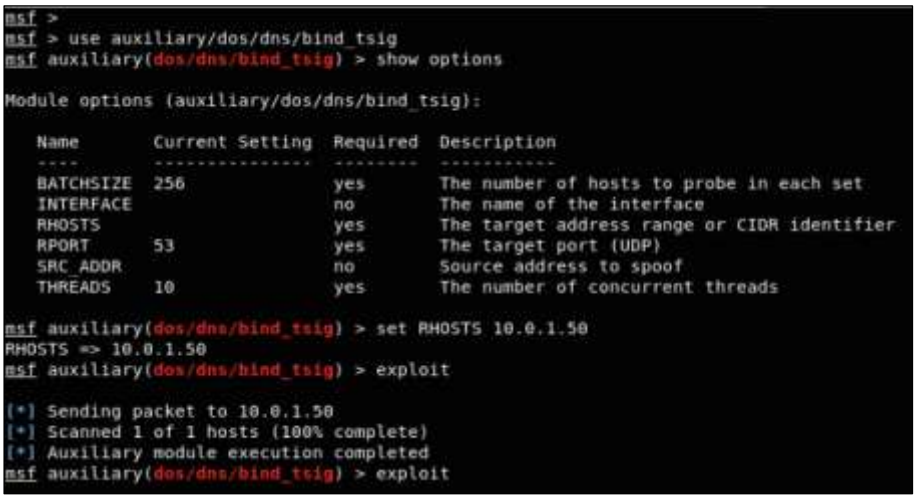


Figura 29-2: Ataque Bind_tsig desde metasploit

Realizado por: MORALES, Edwin, 2018.

Estas vulnerabilidades la *bind_tkey* y *bind_tsig*, fueron descubiertas en el mes de agosto del 2017 con las versiones anteriores a BIND 9.10.5, en servidores que aun usen versiones bind antiguas este tipo de ataques son eficaces.

2.6.2 Detección de ataques TCP SYN Flooding

El siguiente ataque mediante al protocolo TCP, al servidor web HTTP, el SYN Flood es uno de los más sencillos de efectuar, pero también es uno de los más efectivos para lograr saturar servidores, debido a la arquitectura propia del protocolo TCP, en donde los clientes deben usar el three-way handshake para empezar él envió de datos y es esta vulnerabilidad que explota el SYN Flood.

Este ataque consiste en enviar múltiples paquetes SYN al servidor, para lograr que reserve todos los recursos para dar conexión a estas peticiones, pero dichas conexiones no se realizan pues el atacante tiene una dirección IP falsa, o una MAC inexistente, lo cual permite rechazar la conexión, logrando así saturar al servidor solo con peticiones.

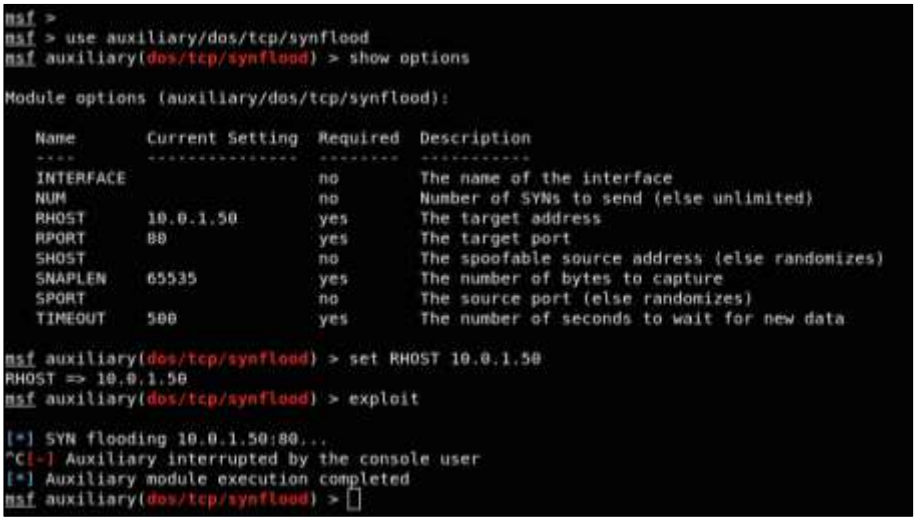


Figura 30-2: Ataque synflood desde metasploit

Realizado por: MORALES, Edwin, 2018.

En la figura 30-2, se observa el ataque SYN realizado al servidor web, como información adicional se tiene parámetros los cuales deben ser cambiados dependiendo de la capacidad de procesamiento del servidor que se esté atacando, si un servidor se encuentra con poca demanda, el ataque SYN con los parámetros por defecto no lograra su objetivo, para ello se debe realizar cambios en los parámetros: RHOST, SNAPLEN y TIMEOUT.

CAPITULO III

3 MARCO DE RESULTADOS

En este capítulo se analiza los resultados obtenidos de las diversas pruebas realizadas con los dispositivos que forman el escenario planteado, estas pruebas se basan en la interoperabilidad, mediante la creación de servidores, y pruebas de rendimiento, a través de ataques de denegación de servicio ejecutados para probar la integración de un sistema de detección y prevención de intrusos a un controlador de redes definidas por software.

3.1 Análisis de resultados: Pruebas de interoperabilidad

En este apartado se realizó la detección de la comunicación que se estableció entre el cliente y los servidores DHCP, DNS y HTTP,

3.1.1 Detección de Protocolos ICMP, TCP, UDP en IPv4.

Una vez habilitados los servicios, se procedió a realizar pruebas para comprobar la conectividad y funcionalidad, mediante el envío de paquetes IP, para que posteriormente estos sean detectados por el sistema SNORT, y luego enviar la información obtenida al controlador RYU.

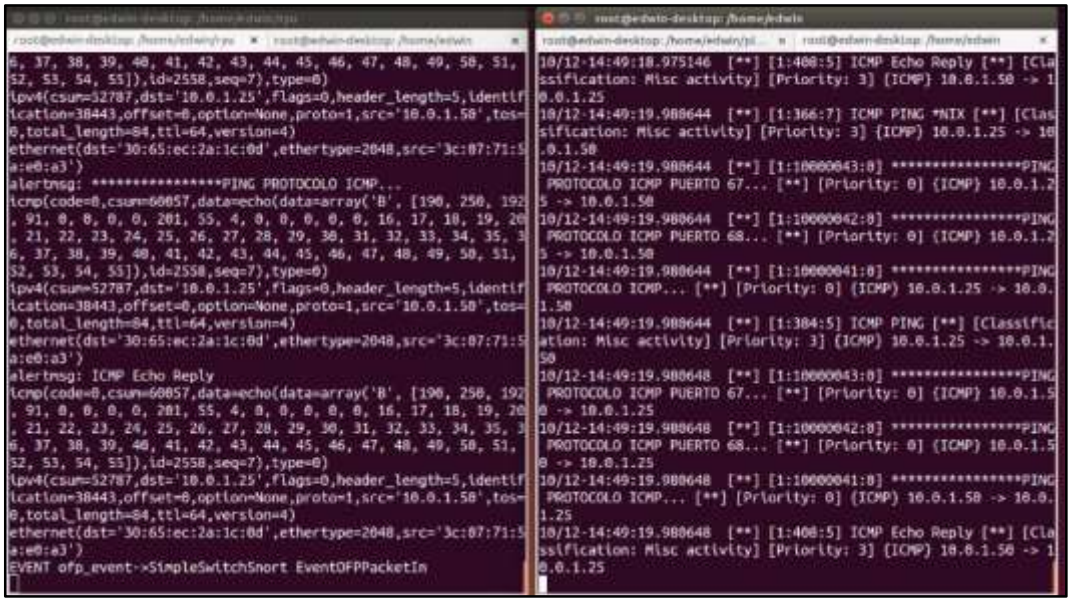


Figura 1-3: Paquetes DHCP detectados por el SNORT.

Realizado por: MORALES, Edwin, 2018.

La primera prueba es con el servidor DHCP, para ello se realizó el envío de paquetes desde el cliente hasta los servidores mediante el protocolo ICMP por el puerto de comunicación 67 y 68, en la figura 1-3, a la derecha se observa los mensajes que envía el SNORT y a la izquierda los mensajes que son recibidos por el RYU

De igual manera la segunda prueba, es con el servidor DNS, para ello se usó el comando NSLOOKUP desde el cliente hacia la dirección IP del servidor, como se puede observar en la figura 2-3, a la izquierda los mensajes que recibe el SNORT, indican el puerto de comunicación y el protocolo, también hay información complementara como la dirección IP de origen y de destino.

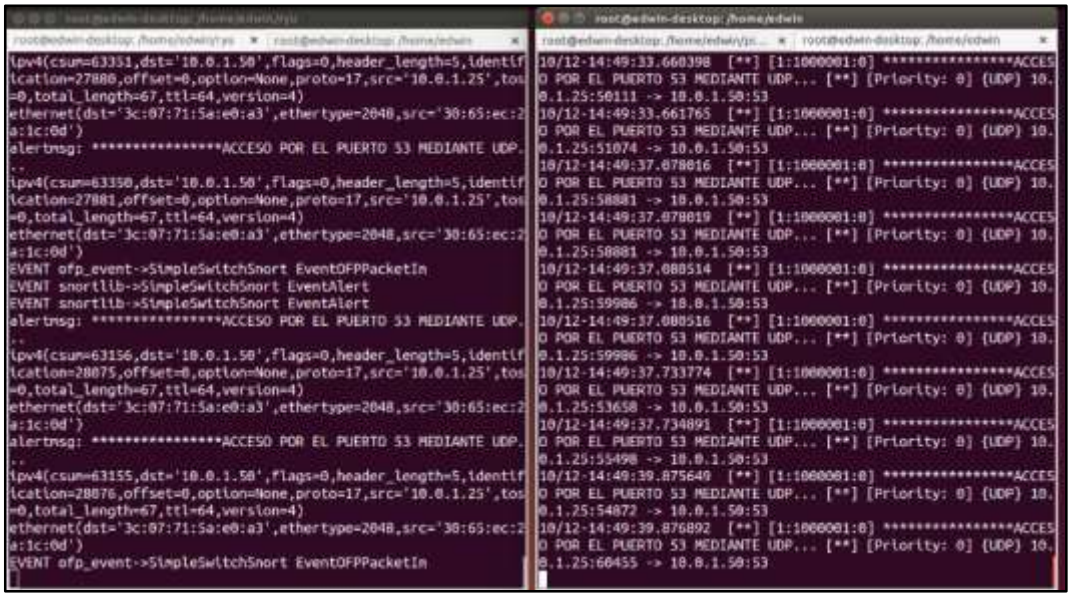


Figura 2-3: Paquetes DNS detectados por el SNORT

Realizado por: MORALES, Edwin, 2018.

La tercera prueba es con el servidor de la página web HTTP, así mismo desde el cliente, se accedió mediante un navegador a la dirección web www.tesis-ids-ips.com, en la figura 3-3, a la izquierda se observa mensajes que son enviados por el SNORT, con parámetros característicos del protocolo TCP como puerto, IP origen, IP destino, flags, identificadores, ttl.

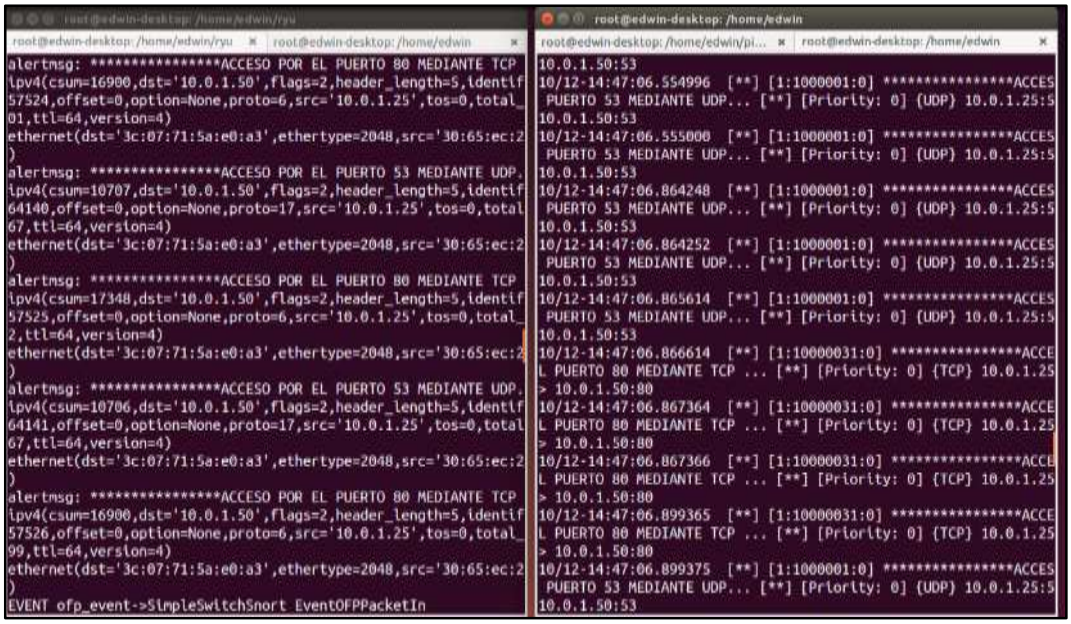


Figura 3-3: Paquetes TCP detectados por el SNORT

Realizado por: MORALES, Edwin, 2018.

Como se observa en las figuras de esta sección, los servidores funcionan correctamente y el SNORT está realizando su trabajo de detectar en informar al RYU de todos los eventos que suceden en la red, estos eventos pueden ser normales es decir paquetes sin malformaciones, o eventos maliciosos cuando los paquetes están adulterados y contengan información para desencadenar un posible ataque DoS.

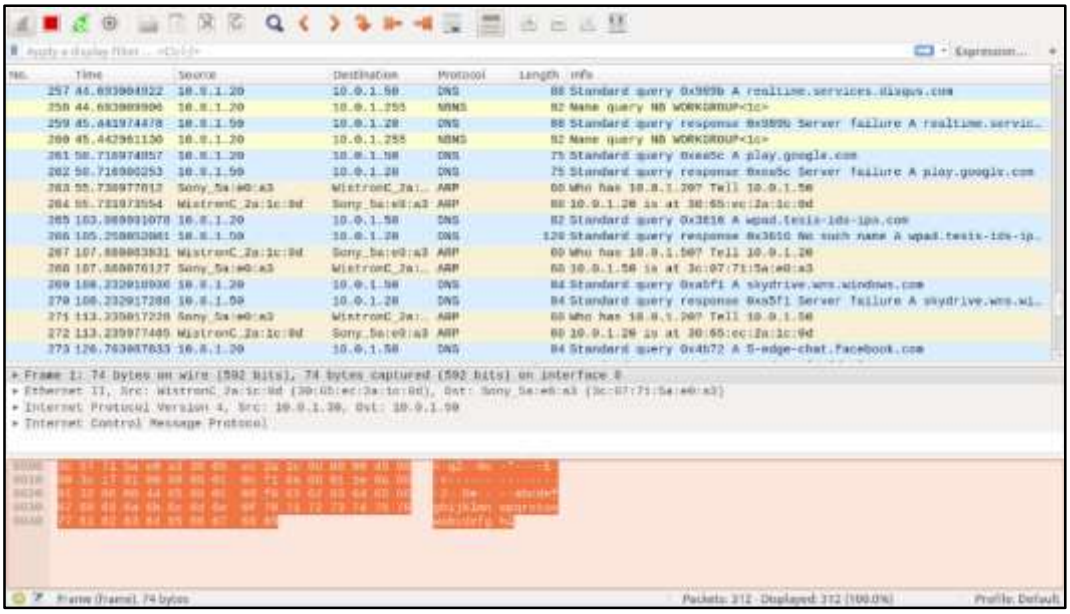


Figura 4-3: Paquetes del controlador detectados en Wireshark

Realizado por: MORALES, Edwin, 2018.

En la figura 4-3, con la ayuda de la herramienta wireshark se observa los paquetes adicionales que automáticamente se generan cuando se inician los servidores, estas tramas adiciones a las de reconocimiento son las siguientes ARP, DNS; en donde con la primera los hosts intercambian las direcciones MAC, y la segunda de reconocimiento de servidores DNS.

3.2 Análisis de resultados: Pruebas de rendimiento.

En esta sección se realiza un análisis de comportamiento de la red SDN bajo ataques de denegación de servicio (DoS), con o sin un sistema de detección de intrusos (IDS/IPS).

3.2.1 Detección de los ataques DNS y SYN Flood sin IDS/IPS.

Luego de ejecutar los ataques de denegación de servicio indicados en la sección 2.6, la información que se obtuvo en el controlador sobre los paquetes entrantes es nula, lo único que muestra es que por cada paquete hay tres mensajes que solo informan que algo ha sido detectado, pero nunca una información más detallada, como se muestra en la figura 5-3.

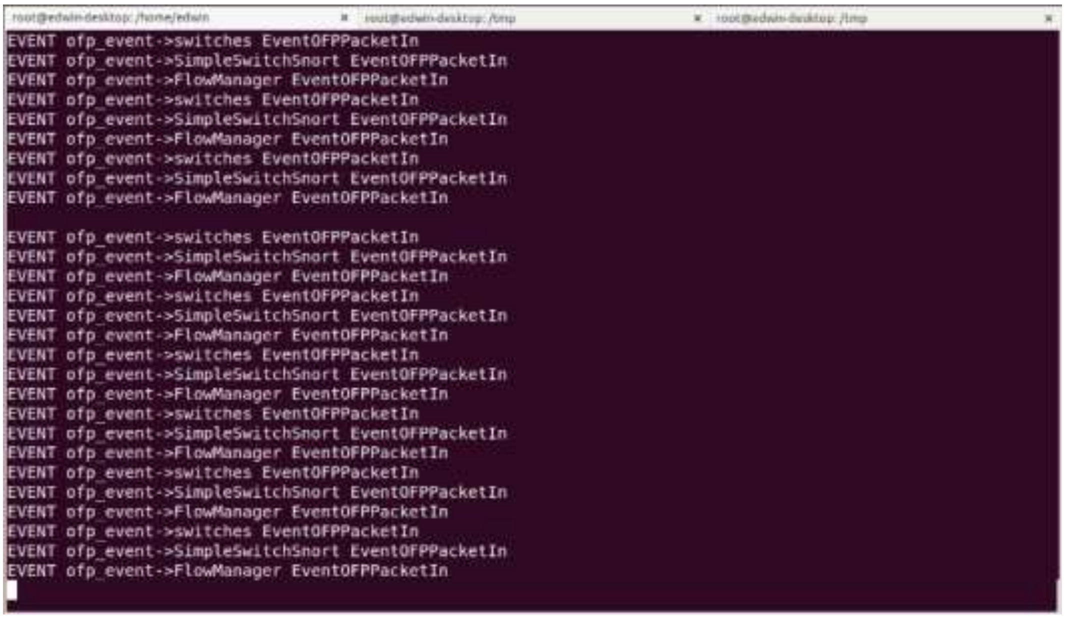


Figura 5-3: Paquetes detectados por el controlador sin IDS

Realizado por: MORALES, Edwin, 2018.

Para un administrador de red la información sobre qué tipos de paquetes, tamaño del mismo, protocolo y puerto que se usa para la comunicación, direcciones IP de origen y destino, identificadores MAC, etc… es muy importante, ya que con esto puede realizar mantenimientos preventivos y correctivos en la infraestructura de red.

3.2.2 Detección de los ataques DNS y SYN con IDS/IPS.

La segunda prueba de estos ataques realizados es bajo la integración de un IDS/IPS en el escenario, con la ayuda del SNORT, se realizó un escaneo completo a tiempo real, conforme a las reglas establecidas, esta información invaluable es enviada al controlador para que un administrador de red tenga noción acerca de lo que está transitando por la infraestructura.

Para poder realizar la detección de las amenazas se configuro reglas de acuerdo a la estructura de la red SDN, estas reglas deben ser precisas para que no exista confusión entre paquetes que contengan información verídica e inofensiva y paquetes que sean corrompidos llegando a ser una amenaza, en la figura 6-3, se observa las reglas que ayudaron a detectar los ataques realizados desde el cliente con la herramienta Metasploit.

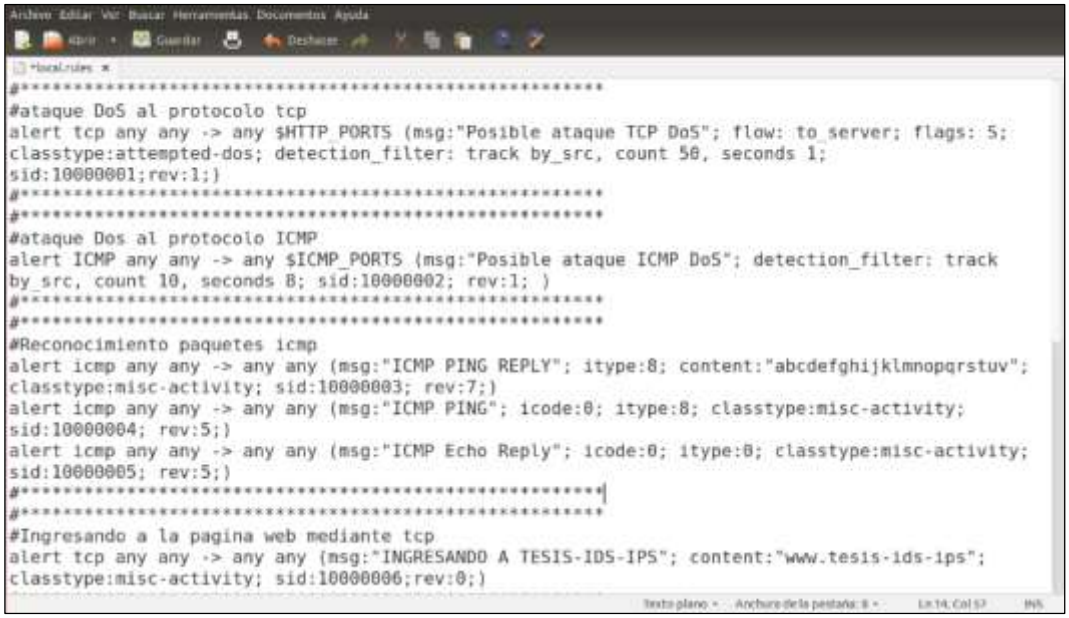


Figura 6-3: Reglas del SNORT para detectar los ataques DoS

Realizado por: MORALES, Edwin, 2018.

Para que se realice la detección de alguna amenaza de manera correcta, las tramas que son escaneadas por el SNORT, deben pasar por una comprobación de match con cada una de las reglas creadas, con la regla que se generado un match, el IDS genera un mensaje de acuerdo a lo especificado en la regla, y posteriormente es enviado al controlador, en caso de que la trama no haya realizado ningún match, simplemente se la ignora.

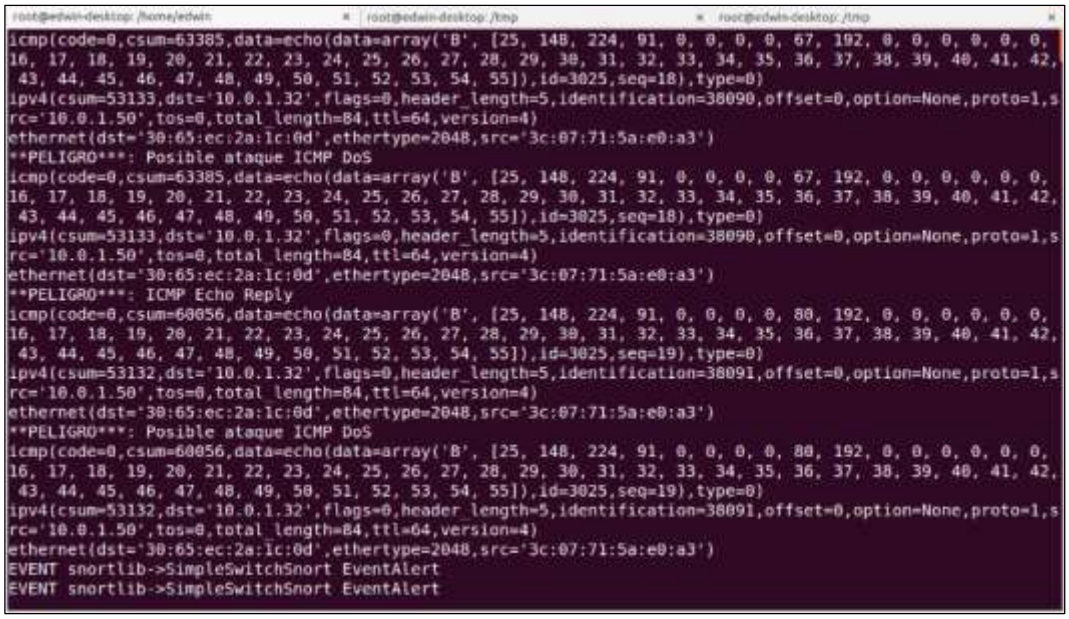


Figura 7-3: Mensajes recibidos en el RYU sobre un posible ataque ICMP DoS

Realizado por: MORALES, Edwin, 2018.

La figura 7-3, muestra unos mensajes recibidos en el controlador, donde indica que posiblemente se esté realizando un ataque DoS mediante el protocolo ICMP. Este tipo de ataque generalmente se lo detecta cuando se recibe grandes cantidades de paquetes del mismo tipo en tiempos muy cortos, lo cual desencadena una posible denegación de servicio, mediante el agotamiento de los recursos.

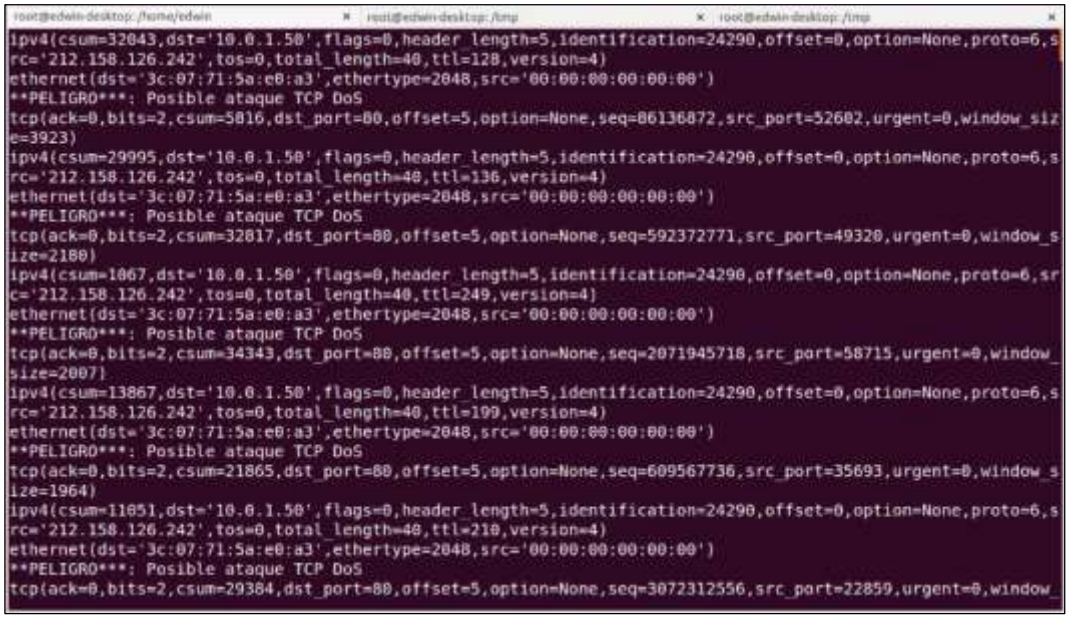


Figura 8-3: Mensajes recibidos en el RYU sobre un posible ataque TCP DoS

Realizado por: MORALES, Edwin, 2018.

El segundo ataque detectado se lo indica en la figura 8-3, esta amenaza es un posible ataque DoS SYN Flood, el cual mediante el uso del protocolo TCP, atenta directamente al servidor web HTTP, el aviso recibido en el controlador, contiene varios parámetros, los cuales ayudan a corroborar si efectivamente se trata de una amenaza del tipo SYN Flood.

Estos parámetros son: ack, tamaño en bits, csum, puerto de comunicación, la dirección IP y MAC del destino final de la trama, las banderas (slot característico de una trama TCP), el identificador de trama, la longitud, el ttl, y la versión, pero el parámetro a enfocarse es la dirección IP de origen 202.127.53.154 y MAC del host 00:00:00:00:00:00.

Estos valores (IP y MAC de origen) efectivamente no son propios de los equipos físicos del escenario, que puedan generar tráfico, es decir estas tramas son generadas artificialmente con el único objetivo malicioso de provocar una denegación de servicio mediante el agotamiento de los recursos lógicos (procesamiento y memoria) del servidor o del conmutador.

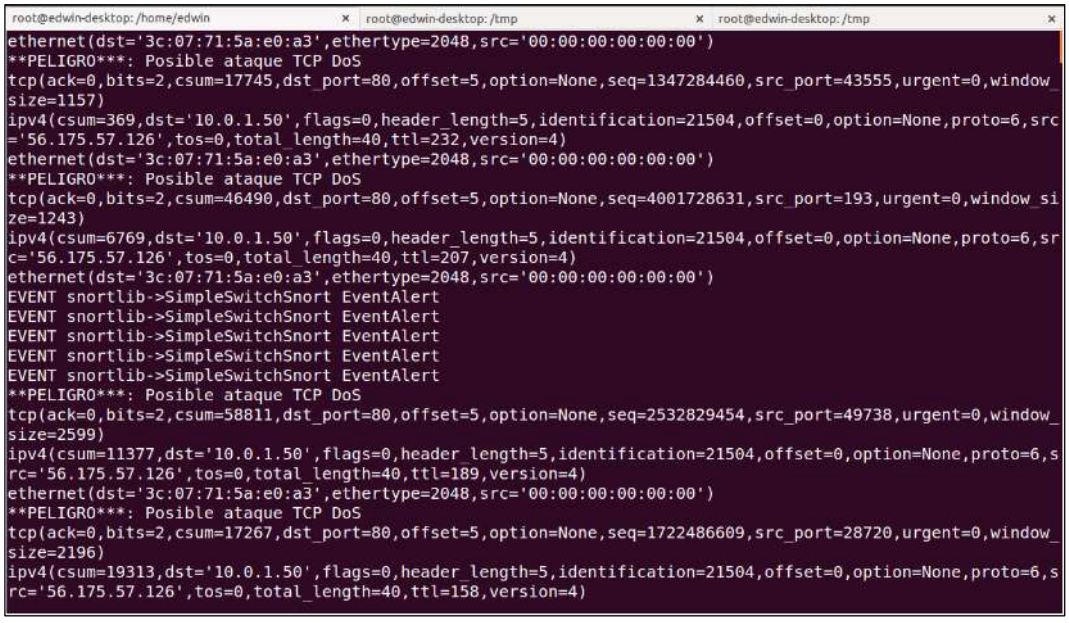


Figura 9-3: Mensajes recibidos en el RYU sobre dos posible ataque DoS

Realizado por: MORALES, Edwin, 2018.

En la tercera prueba, al mismo tiempo se realizaron los dos ataques DoS a los servidores al mismo tiempo, en la figura 9-3, se observa los mensajes de alerta que el SNORT envía al controlador, como resultado se obtuvo que, mientras más ataques simultáneos se realice, los servidores y el conmutador más rápido se verán afectados.

3.2.2.1 Margen de error de la detección.

Para poder determinar el margen de error, se realizó un conteo de paquetes que se generaron en la fuente, y los que se detectaron en el sistema Snort. Para el conteo de paquetes enviados desde el origen se usó la herramienta wireshark por un tiempo de 70 segundo. Como se observa en la figura 10-3. Los resultados se los tabuló en la tabla del anexo D, deduciendo que en el tiempo transcurrido es decir los 70 segundo se han detectado 41800 paquetes.

.

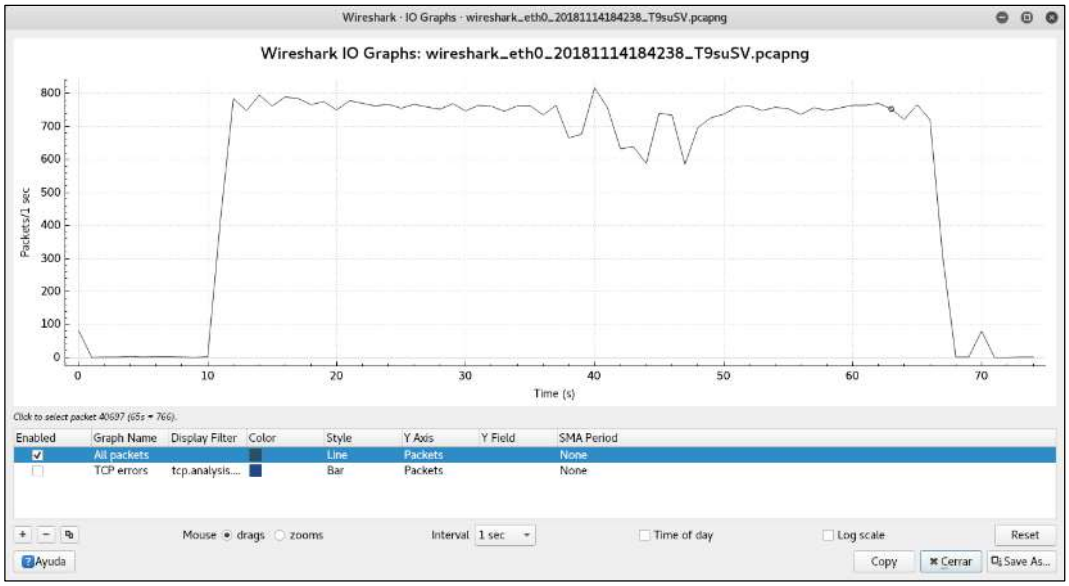


Figura 10-3: Conteo de paquetes por segundo en Wireshark

Realizado por: MORALES, Edwin, 2018.

Para el conteo de paquetes, que el sistema Snort ha detectado, se usa las herramientas barnyard2 combinado con la aplicación BASE, el primero crea una base de datos en donde almacena toda la información registrada por el snort, y el segundo organiza la información y la muestra al usuario, de manera resumida y de forma gráfica. Durante el mismo intervalo de tiempo, 70 segundo, se observa que se han detectado 41728 paquetes, como se lo evidencia en la figura 11-3

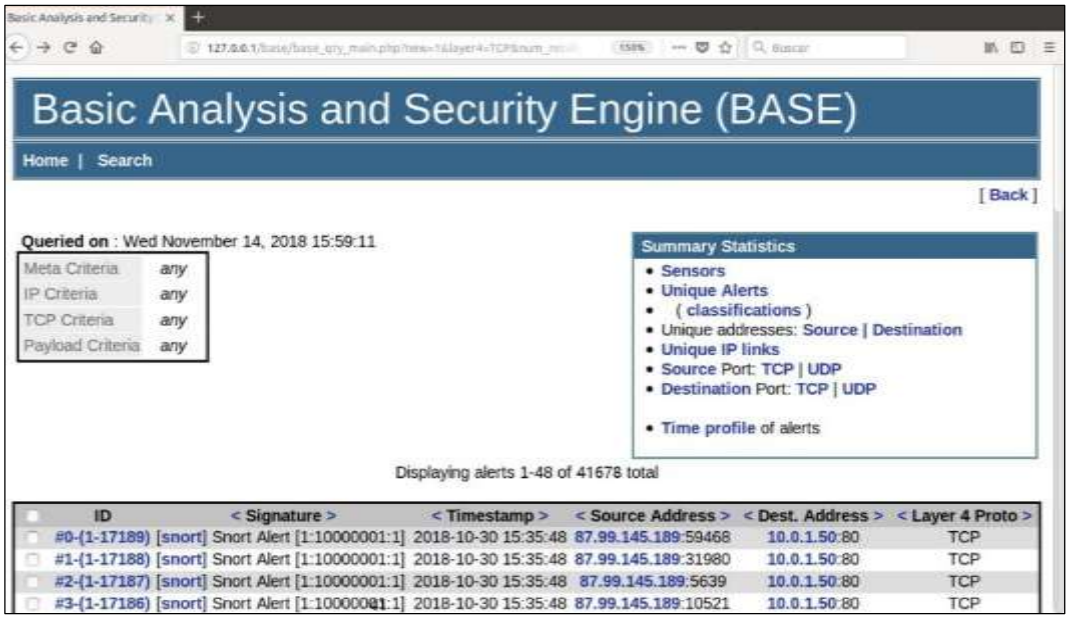


Figura 11-3: Conteo de paquetes por segundo en BASE

Realizado por: MORALES, Edwin, 2018.

Al comparar la cantidad de paquetes que se han generado y que se han detectado, se puede deducir cuantos paquetes no han sido analizados por el sistema, esto indica un margen error de 0.01, que puede ser calculado con la ayuda de la fórmula del porcentaje de error usada comúnmente en el ámbito de estadística, como se muestra en la figura 12-3.

$$\%\ error: \left|\frac{valor\ aprox - valor\ exact}{valor\ exact}\right| * 100$$

$$\%\ error: \left|\frac{41768 - 41800}{41800}\right| * 100 = 0.077 \cong 0.1$$

Figura 12-3: Margen de error de paquetes detectados.

Realizado por: MORALES, Edwin, 2018.

3.2.3 Mitigación de ataques DNS y SYN con IDS/IPS.

En esta sección se realizó la mitigación de los ataques Usando la herramienta DAQ, la cual viene incluida en los paquetes instalados previamente junto al IDS, por defecto esta opción se encuentra deshabilitada, para que empiece a funcionar se debe modificar varios parámetros el archivo *snort.conf*, que se encuentra en la dirección /etc/snort, Los cambio se lo indica a continuación:

- Config daq: afpacket
- Config daq_dir: /usr/local/lib/daq
- Config daq_mode: inline
- Config daq_var:buffer_size_mb=512

Luego de haber realizado las modificaciones, el SNORT tendrá la capacidad de mitigar los ataques previamente detectados, Para ello se debe modificar un parámetro de las reglas, en lugar de *alert* se debe colocar *drop* o *reject*, Como se indica la figura 13-3.

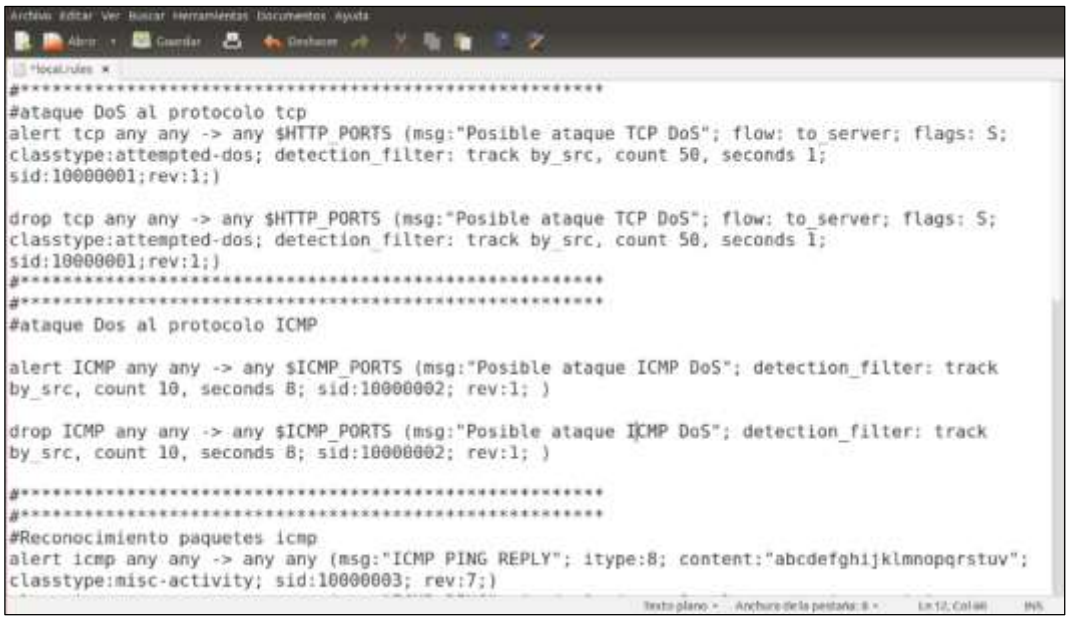


Figura 13-3: Reglas para la mitigación de amenazas.

Realizado por: MORALES, Edwin, 2018.

De este modo, la mitigación se realizó automáticamente, previamente la detección de una amenaza era alertada al controlador, de la misma forma la mitigación debe ser informada, para cumplir este objetivo se opta por colocar una regla con *alert* y otra con *drop*, los mensajes de alerta que se recibe en el RYU se observan en la figura 14-3.

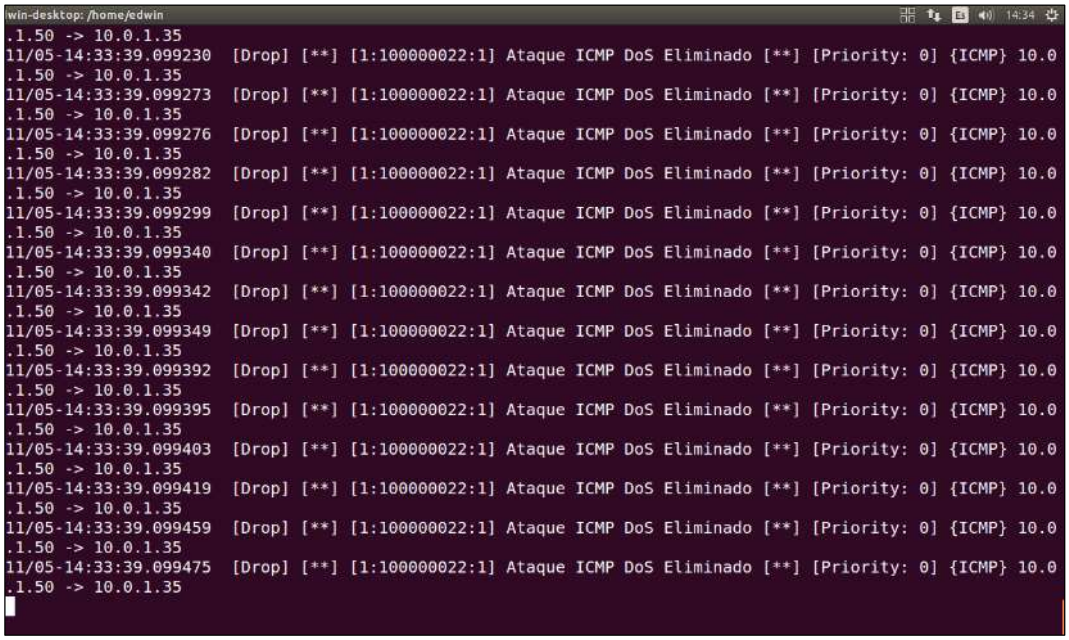


Figura 14-3: Mensajes de la mitigación de amenazas.

Realizado por: MORALES, Edwin, 2018.

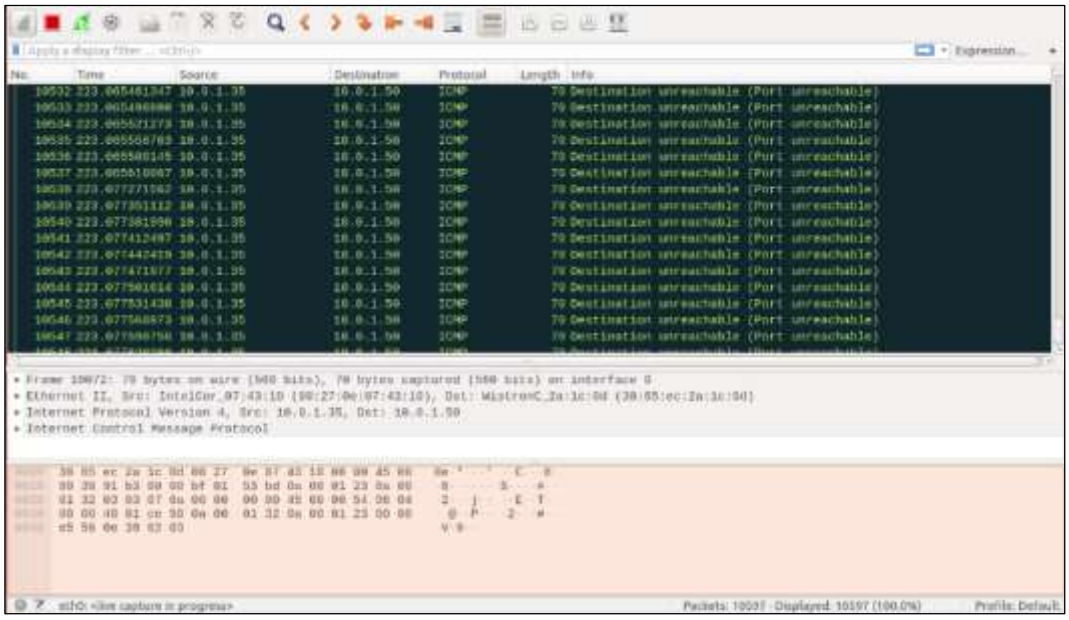


Figura 15-3: Verificación del bloqueo de paquetes con wireshark.

Realizado por: MORALES, Edwin, 2018.

En la figura 15-3, mediante la herramienta wireshark, se observó que los paquetes generados para realizar un ataque DoS son rechazados y no encuentran camino hacia su víctima, debido a que con anterioridad se crearon reglas con DROP, es decir que elimine las tramas infectadas, y no transiten por la red intentando causar cualquier tipo de daño.

3.3 Comparativa de tiempos de respuesta en red tradicional y red SDN.

En esta sección se determinó el perfomance o rendimiento de una red SDN versus una red tradicional, para ello, se realizó pruebas con la finalidad de obtener los tiempos de respuesta cuando la red se encuentra sin o con un flujo abundante de tráfico. Para ello se generaron 10 paquetes ICMP, tanto en el primer como en el segundo escenario.

La parte derecha de la figura 16-3, muestra los tiempos de respuesta cuando la red se encuentra sin tráfico de datos, obteniendo como promedio 0.6236 milisegundos, la parte izquierda muestra los tiempos de respuesta cuando la red se encuentra bajo ataques DoS, teniendo como promedio 1.185 milisegundos.

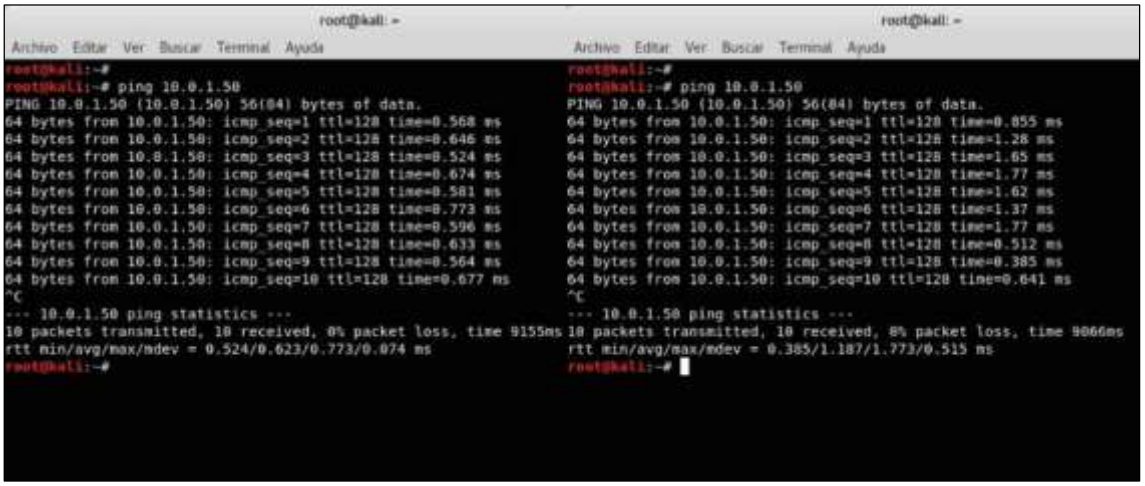


Figura 16-3: Tiempos de respuesta de una red SDN.

Realizado por: MORALES, Edwin, 2018.

El siguiente paso fue realizar la misma prueba, pero en una red tradicional para ello se reemplazó la tarjeta Zodiac FX por un router marca Tp-Link, modelo TL-WR720N, se escogió este dispositivo por poseer características de hardware similares a la tarjeta. Lo cual permite que las velocidades de procesamiento sean parecidas.

Luego de realizar las configuraciones en el router Tp-Link, se realizó las pruebas anteriores, teniendo como respuesta la figura 17-3, en la parte derecha se evidencia un promedio de tiempo de respuesta de 2.439 milisegundos, mientras que en la parte izquierda un valor de 8.390 milisegundos como media del tiempo de respuesta cuando la red se encuentra bajo ataques.

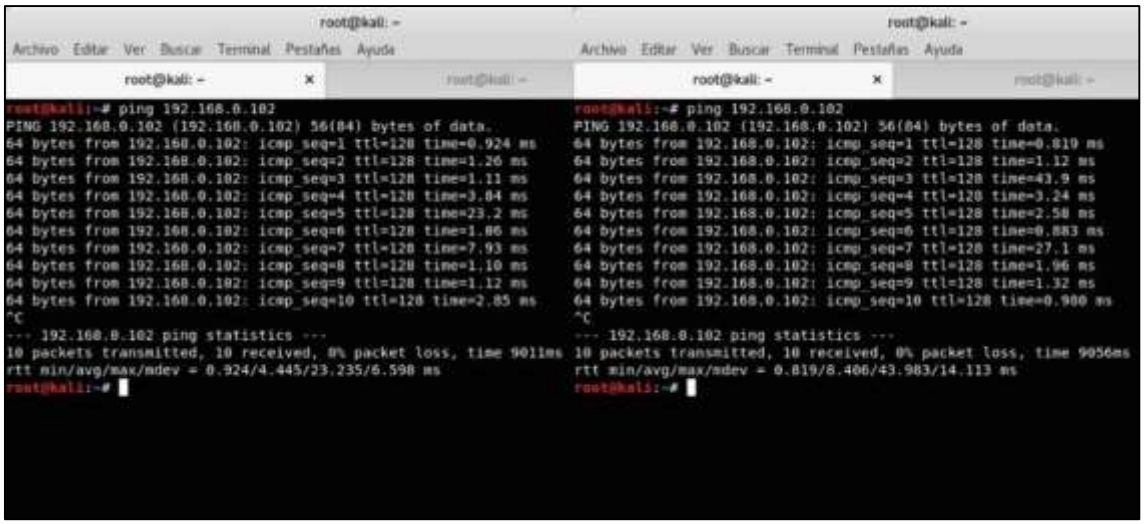


Figura 17-3: Tiempos de respuesta de una red SDN.

Realizado por: MORALES, Edwin, 2018.

Los valores encontrados en esta sección se los puede apreciar de mejor manera en la tabla 1-3. Cuando la red tradicional o la SDN, no está bajo ataques DoS, el sistema Snort logra detectar todos los paquetes sin ninguna complicación. Pero al comparar las dos redes cuando están sometidas ataques de denegación de servicio, la red SDN tiene mejores resultados en cuanto al análisis de reporte de paquetes gracias al que el tiempo de respuesta es más corto y existe menos probabilidad de que alguno no sea analizado.

Tabla 1-3: Tiempos de respuesta en red tradicional y SDN

Red	Situación	Promedio de tiempos de respuesta
Tradicional	Sin ataques	2.439
	Con ataques	8.390
SDN	Sin ataques	0.623
	Con ataques	1.185

Realizado por: MORALES, Edwin, 2018.

3.4 Comparación entre red, con IDS/IPS y sin IDS/IPS.

El uso de una herramienta para la detección de intrusos, es sin duda primordial para cualquier infraestructura de red. Como se puede observar en la figura 18-3, la imagen del lado izquierdo representa una red que está bajo ataques DoS, pero no cuenta con un sistema de detección, mientras que la imagen del lado derecho cuenta con la herramienta SNORT integrada a una red definida por software.

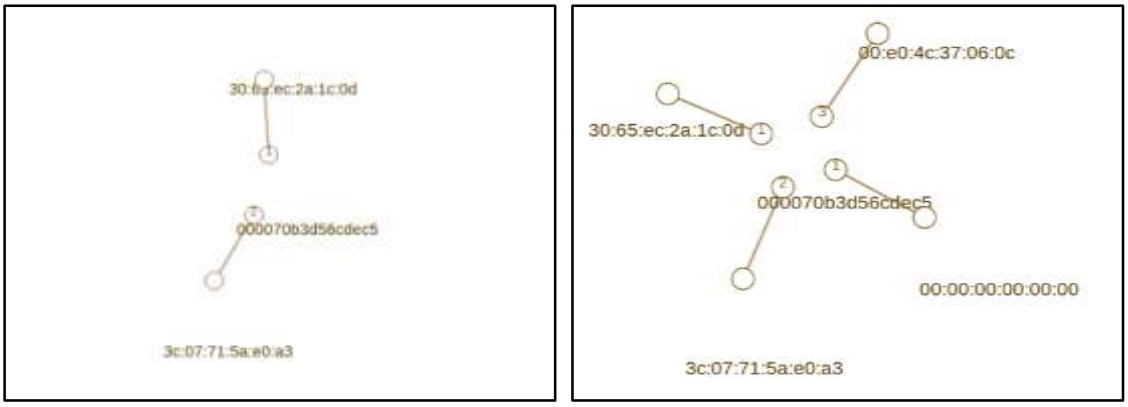


Figura 18-3: Topología de una red sin o con IDS/IPS bajo ataques DoS

Realizado por: MORALES, Edwin, 2018.

En la imagen del lado derecho de la figura 18-3, se dice que está bajo amenazas de intrusos DoS, porque en el puerto uno se encuentra una dirección MAC (00:00:00:00:00:00) errónea, es decir no debería existir debido a que no representa a ningún equipo físico conectado a la red. Esta figura ha sido obtenida de una aplicación denominada FlowManager que se detallará su funcionamiento más adelante.

Otra comparación en una red con IDS y sin IDS, es mediante las alertas, que se reciben en la consola del controlador RYU. En la tabla 2-3, se puede observar características de los principales tipos de tramas (ICMP, UDP, TCP).

En la tabla 2-3, se puede observar que mientras no exista un sistema de detección el controlador RYU, recibirá tres alertas por cada paquete (indiferente del tipo) que se genere de algún host, estas tres alertas solo indican que un paquete se ha sido tratado, mientras que, si un IDS está integrado al controlador, este le informara detalladamente que tipo de trama es, de donde ha sido generado, a donde se dirige, el tamaño, el tiempo de convergencia, el identificador, etc…

Tabla 2-3: Tipos de paquetes y su parámetros enviados cuando se usa un IDS.

Protocolos	Sin IDS/IPS	Con IDS/IPS
TCP	>Switches EventOFPPacketIN >SimpleSwitchSnort EventFPPacketIN >FlowManager EventOFPPacketIN	>Paquete tcp: ack, bits, csum, dst_port, offset, src_port, urgent, size. >Paquete IPv4: csum, IP destino, flags, length, id, offset, options, proto, IP origen, tos, length, ttl, versión. >Ethernet: MAC origen, MAC destino.
UDP	>Switches EventOFPPacketIN >SimpleSwitchSnort EventFPPacketIN	>Paquete IPv4: csum, IP destino, flags, length, id, offset, options, proto, IP origen, tos, length, ttl, versión.

	>FlowManager EventOFPPacketIN	>Ethernet: MAC origen, MAC destino.
ICMP	>Switches EventOFPPacketIN >SimpleSwitchSnort EventFPPacketIN >FlowManager EventOFPPacketIN	>Paquete ICMP: code, csum, data, id, seq, type. >Paquete IPv4: csum, IP destino, flags, length, id, offset, options, proto, IP origen, tos, length, ttl, versión. >Ethernet: MAC origen, MAC destino.

Realizado por: MORALES, Edwin, 2018.

3.5 FlowManager

Como se puede observar en los resultados obtenidos de las pruebas realizadas en el escenario, la integración se la realizo con éxito, toda detección y mitigación de amenazas que realizaba el SNORT, eran enviados al controlador sin ningún tipo de excepción. Pero ahora ¿que se podría realizar luego de recibir las alertas en el RYU?, la opción más recomendable sería crear reglas o métricas en el controlador en contra de los ataques detectados.

Hay dos maneras de crear estas reglas o métricas; la primera, mediante codificación en Python, crear reglas y ejecutarlas junto a la aplicación *simple_switch_snort.py*, pero para ello se debe tener conocimientos avanzados en programación, la segunda opción, es mediante una aplicación que se comunica con el RYU en sentido NorthBound API, esta aplicación se llama FlowManager, en ella se puede crear, editar o eliminar flows de las tablas de conmutador, crear métricas, observar la topología que controla el RYU, etc…

Para que la aplicación funcione junto al controlador se lo debe ejecutar con el siguiente comando: *ryu-manager --verbose flowmanager/flowmanager.py ryu/ryu/app/simple_switch_snort.py*, luego el FlowManager empieza a funcionar en conjunto con el RYU, en la figura 19-3 se puede observar como la herramienta obtiene la información del conmutador automáticamente, y en la figura 20-3, se observa la pantalla principal de la herramienta, para acceder a ella se lo hace mediante la dirección web *https://localhost/8080/home/*,

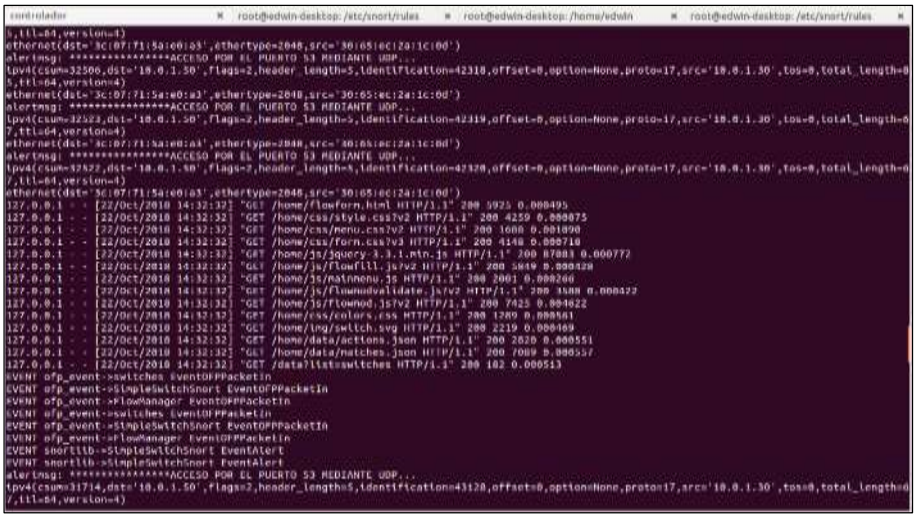


Figura 19-3: Flow Manager tomando información del Ryu

Realizado por: MORALES, Edwin, 2018.

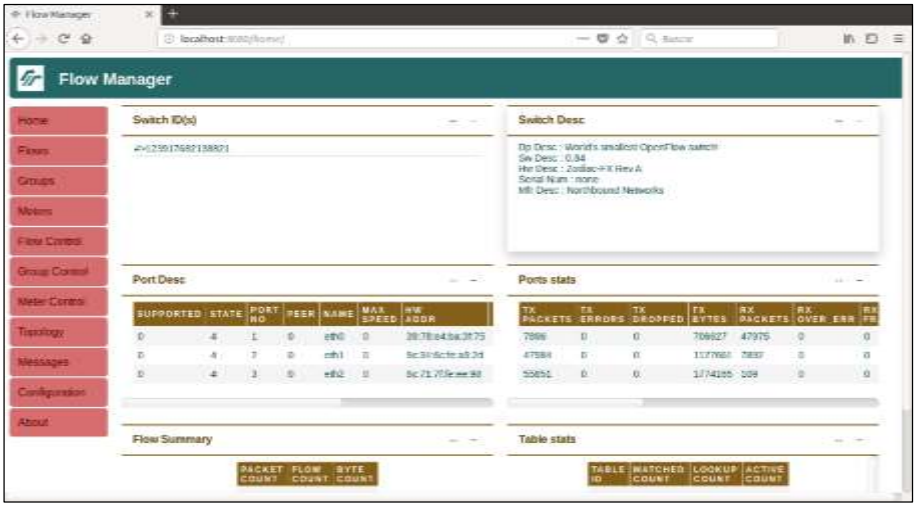


Figura 20-3: Entorno del Flow Manager

Realizado por: MORALES, Edwin, 2018.

En el entorno del FlowManager se tiene varias opciones en el menú, las más importantes para la administración: son Flows control y Meters control, como se observa en las figuras 21-3, 22-3, en estas secciones se debe crear los flows o métricas para ello se debe llenar los campos con las especificaciones que de desea.

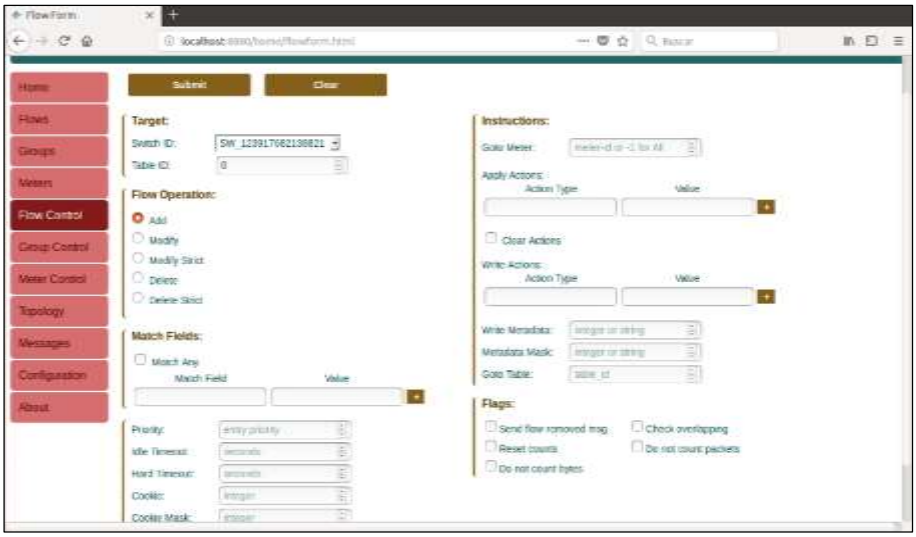


Figura 21-3: Sección Flow Control del FlowManager

Realizado por: MORALES, Edwin, 2018.

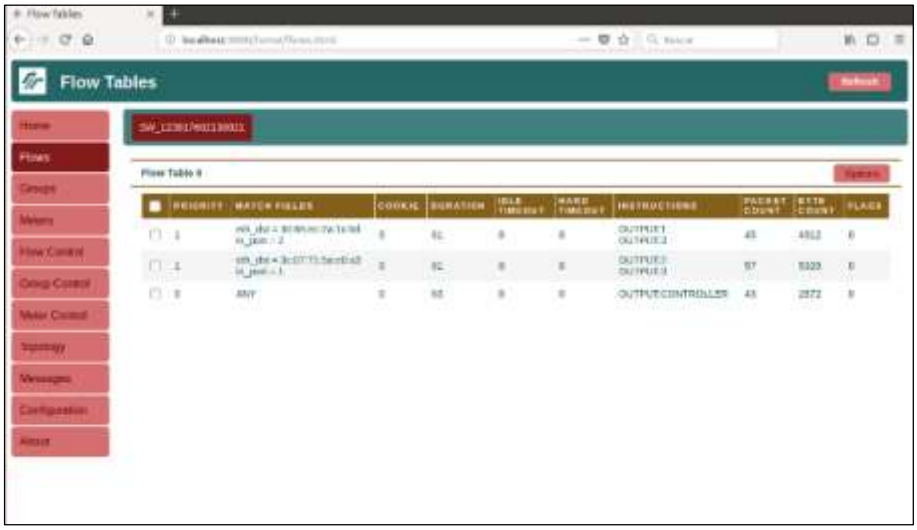


Figura 22-3: Sección Meter Control del FlowManager

Realizado por: MORALES, Edwin, 2018.

En las figuras 23-3 y 24-3, se tiene dos secciones para la visualización de la topología de la red y mensajes, en la primera se puede observar escenario que está por debajo del controlador con sus respectivas direcciones MAC, y la segunda permite la visualización de mensajes de los match realizados entre los Flows y los paquetes que transitan por la red.

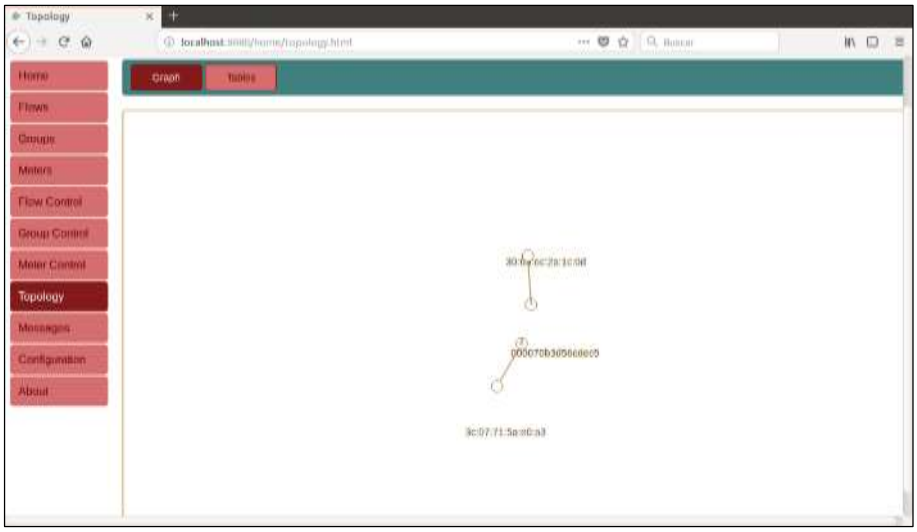


Figura 23-3: Sección Topology del FlowManager

Realizado por: MORALES, Edwin, 2018.

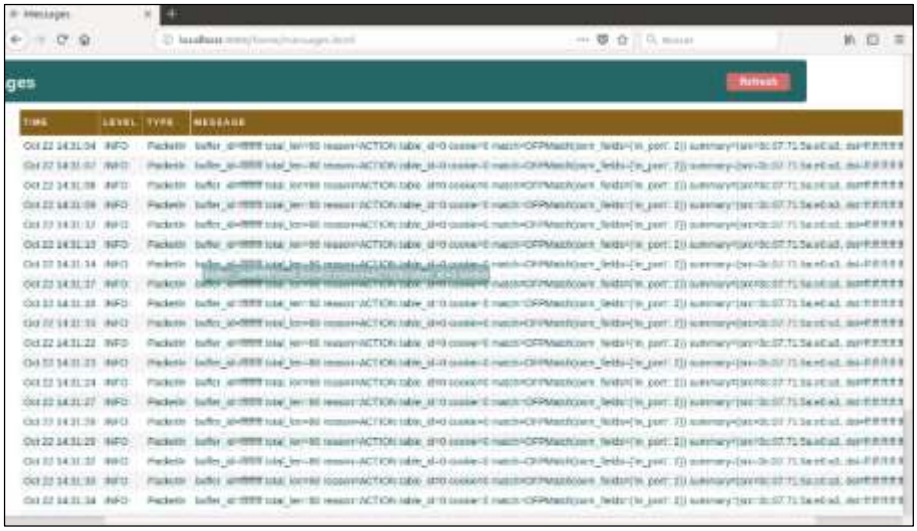


Figura 24-3: Sección messages del FlowManager

Realizado por: MORALES, Edwin, 2018.

CONCLUSIONES

- Se diseñó e implemento un escenario de bajo costo para integrar un IDS/IPS a un controlador SDN para la prevención y detección de ataques de seguridad (DoS), mediante el uso del conmutador de la marca Zodiac, el cual se encargó de realizar el encaminamiento de los paquetes desde su origen hasta el destino con la ayuda del protocolo OpenFlow.

- Se investigó sobre los principales tipos de ataques DoS que afectan la correcta funcionalidad de los recursos (software y hardware) de las redes definidas por software, lo que permitió elaborar un conjunto de reglas precisas para ser incrustadas al sistema de detección de intrusos Snort con el fin de contrarrestarlos. entre los más destacados se encuentran los ataques por inundación mediante los protocolos ICMP, UDP y TCP.

- Con las pruebas realizadas se pudo comprobar que el Snort fue capaz de detectar fácilmente cualquier intento de intrusión en tiempo real, sea del tipo ICMP, TCP o UDP. Posteriormente almacenaba esta información en un archivo temporal para ser analizado y desencriptado por el controlador Ryu.

- La plataforma Ryu que representa al controlador, mediante programación en Python fue capaz de buscar los registros del snort, luego, analizarlos y clasificar según los protocolos usados para la comunicación de los servicios y por ultimo mostrar resultados en una interfaz.

- Para comprobar la integración se realizó pruebas en dos escenarios, el primer escenario, el cual no contaba con un sistema de detección de intrusos, la información que se obtenía sobre los paquetes entrantes era nula, mientras que en el segundo escenario con la integración del sistema Snort se tuvo la capacidad de analizar exhaustivamente el flujo de datos.

- El prototipo implementado fue capaz de reaccionar rápidamente en tiempo real, mitigando o eliminando los paquetes sospechosos previamente detectados. Con error de 0.077, es decir cada mil paquetes 1 no era detectado.

RECOMENDACIONES

- Al momento de obtener el conmutador Zodiac FX se recomienda revisar la guía de usuario del proveedor, para una correcta configuración y manipulación.

- Es recomendable tener amplio conocimiento en uso de sistema operativo Linux, y así mismo sobre programación en Python para comprender el funcionamiento de cada línea de código, pues si se modifica una mínima parte, el algoritmo realizara acciones diferentes a la deseada.

- Se recomienda realizar un escenario virtual mediante la herramienta MiniNet integrada con GNS3, o a su vez usar el simulador y emulador de paga EstiNet. Para el ahorro de recursos Hardware.

- Es recomendable comprender a su totalidad la composición de una regla de Snort, pues si se crea una sin un objetivo específico, puede no solo bloquear trafico malicioso, sino también bloquea a usuarios que generan tráfico de comunicación legal.

BIBLIOGRAFÍA

ASTUDILLO, J., et al, *Adaptación del ids/ips suricata para que se pueda convertir en una solución empresarial* [en línea]. (tesis) (pregrado). ESCUELA SUPERIOR POLITECNICA DEL LITORAL. 2011. [Consulta: 26 julio 2018]. Disponible en: https://www.dspace.espol.edu.ec/bitstream/123456789/19502/2/tesina_seminario0.6.pdf.

CRUZ, N., *Los 5 mejores software de detección de intrusos para PC | 1000 Tips Informáticos. 1000tipsinformaticos* [en línea]. 2017. [Consulta: 25 junio 2018]. Disponible en: https://www.1000tipsinformaticos.com/2017/11/los-5-mejores-software-de-deteccion-de-intrusos-para-pc.html.

FIGUEROLA, N., *SDN – Redes definidas por Software* [en línea]. 2013. Tecnología y Tendencias. [Consulta: 12 marzo 2018]. 2013. Disponible en: https://articulosit.files.wordpress.com/2013/10/sdn.pdf

GASTÓN, B., *Análisis de botnets y ataques de denegación de servicio.* [en línea]. Argentina: 2016. [Consulta: 17 marzo 2018]. Disponible en: http://premios.eset-la.com/universitario/pdf/analisis_de_botnets_y_ataques_DDoS.pdf.

GIMENEZ, M., *Utilización de Sistemas de Detección de Intrusos como Elemento de Seguridad Perimetral* [en línea]. (tesis) (pregrado). UNIVERSIDAD DE ALMERÍA. 2008. [Consulta: 14 junio 2018]. Disponible en: http://www.adminso.es/images/1/1d/PFC_marisa.pdf.

HERVÁS, Ó., *Software Defined Networking* [en línea]. (tesis) (pregrado). UNIVERSIDAD POLITECNICA DE CATALUNYA. 2014. [Consulta: 18 marzo 2018]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2099.1/21633/Memoria.pdf.

IBÁÑEZ, F., et al, *Diseño e implementación de una herramienta de visualización para análisis en tiempo real de redes sdn/openflow* [en línea]. (tesis) (pregrado). UNIVERSIDAD COMPLUTENSE DE MADRID. 2016. [Consulta: 17 marzo 2018]. Disponible en: http://eprints.ucm.es/38709/1/TFG_DisenoeImplementacion_SDN.pdf.

INCERA, J., et al, *Redes Digitales: Presente y Futuro.* [en línea]. México: 2007. [Consulta: 16 marzo 2018]. Disponible en: http://allman.rhon.itam.mx/~jincera/IntroRedesDigitales.pdf.

INSERT, *SDN-IPS – Insert – Ufba. Insert-Ufba* [en línea]. 2018. [Consulta: 14 noviembre 2018]. Disponible en: http://insert.ufba.br/sdn-ips/.

ISA, R., *Implementación de un laboratorio virtual para aprendizaje de SDN (Virtual lab implementation for SDN learning)* [en línea]. (tesis) (pregrado). UNIVERSIDAD DE CANTABRIA. 2016. [Consulta: 17 marzo 2018]. Disponible en: https://repositorio.unican.es/xmlui/bitstream/handle/10902/9377/387880.pdf.

LINKSPRITE, *ONetSwitch30. LinkSprite* [en línea]. 2018. [Consulta: 24 julio 2018]. Disponible en: http://store.linksprite.com/onetswitch30/.

LOTIA, P., *SDN-Intrusion-Prevention-System-Honeypot. GitHub* [en línea]. 2017. Disponible en: https://github.com/pratiklotia/SDN-Intrusion-Prevention-System-Honeypot.

MACÍAS, J., *Evaluación de los ataques denegación de servicios (ddos), formas de protección en la carrera de ingeniería en sistemas computacionales de la universidad estatal del sur de manabí* [en línea]. (tesis) (pregrado). UNIVERSIDAD ESTATAL DEL SUR DE MANABÍ. 2017. [Consulta: 14 marzo 2018]. Disponible en: http://repositorio.unesum.edu.ec/bitstream/53000/841/1/UNESUM-ECU-COMPR-18.pdf.

MARTIMY, M., *FlowManager. GitHub* [en línea]. 2016. Disponible en: https://github.com/martimy/flowmanager.

MARTINEZ, M., *Diseño y simulación de un sistema para detección de intrusos en redes de comunicaciones utilizando OMNET++* [en línea]. (tesis) (pregrado). UNIVERSIDAD DE LAS AMÉRICAS PUEBLA. 2010. [Consulta: 14 junio 2018]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/martinez_l_ma/.

NETWORKS., *Zodiac FX: User Guide* [en línea]. 2017. Northbound Networks. [Consulta: 7 agosto 2018]. 2017. Disponible en: www.northboundnetworks.com.

OCAMPO, C., et al, *sistema de detección de intrusos en redes corporativas. Scientia et technica* [en línea]. Pereira: 2017. 22, 1. Disponible en: http://revistas.utp.edu.co/index.php/revistaciencia/article/view/9105.

OPENDAYLIGHT PROJECT, *Platform Overview*. *OpenDaylight* [en línea]. 2016. [Consulta: 29 mayo 2018]. Disponible en: https://www.opendaylight.org/what-we-do/odl-platform-overview.

PARACUELLOS, J., *Defensa proactiva y reactiva ante ataques DDoS en un entorno simulado de redes definidas por software* [en línea]. (tesis) (pregrado). UNIVERSIDAD ZARAGOZA. 2016. [Consulta: 13 marzo 2018]. Disponible en: http://webdiis.unizar.es/~ricardo/files/PFCs-TFGs/Defensa-Proactiva-Reactiva-DDoS-SDN/Memoria_TFG_DefensaProactivaReactivaDDoSSDN.pdf.

PCCOMPONENTES, *Raspberry Pi 3 Modelo B. PcComponentes* [en línea]. 2016. [Consulta: 24 julio 2018]. Disponible en: https://www.pccomponentes.com/raspberry-pi-3-modelo-b.

PEGADO, M., *Desarrollo de un sistema de monitorización para SDNs (Software Defined Networks)* [en línea]. (tesis) (pregrado). UNIVERSIDAD AUTONOMA DE MADRID. 2015. [Consulta: 17 marzo 2018]. Disponible en: https://repositorio.uam.es/bitstream/handle/10486/669460/Pegado_Boureghida_Maria_Teresa_tfg.pdf?sequence=1.

PÉREZ, G. y MARÍN, M., *Redes definidas por software: Solución para servicios portadores del Ecuador* [en línea]. 2015. Investigatio Research Review. 2015. Disponible en: http://revistas.uees.edu.ec/index.php/IRR/article/view/23.

POLVEREDA, J., *Sistema de monitorización del ids snort* [en línea]. (tesis) (pregrado). UNIVERSIDAD POLITÉCNICA DE VALENCIA. 2017. [Consulta: 9 agosto 2018]. Disponible en: https://riunet.upv.es/bitstream/handle/10251/88474/LLOPIS - Sistema de monitorización del IDS Snort.pdf?sequence=1.

SÁNCHEZ, G., *Aplicación de SDN para el control del tráfico de red en base a usuarios Using SDN for user based network traffic control* [en línea]. (tesis) (pregrado). UNIVERSIDAD DE LA LAGUNA. 2017. [Consulta: 29 mayo 2018]. Disponible en: https://riull.ull.es/xmlui/bitstream/handle/915/4285/Aplicacion de SDN para el control del trafico de red en base a usuarios..pdf?sequence=1.

SÁNCHEZ, M., *Análisis de redes SDN utilizando Mininet e implementación de un Deep Packet Inspector*. [en línea]. (tesis) (pregrado). UNIVERSIDAD DE GRANADA. 2015. [Consulta: 16 marzo 2018]. Disponible en:

http://dtstc.ugr.es/it/pfc/proyectos_realizados/downloads/Memoria2015_ManuelSanchez.pdf.

SDXCENTRAL, *What is Ryu Controller? sdxcentral* [en línea]. 2016. [Consulta: 8 agosto 2018]. Disponible en: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/.

SURICATA, *Suricata. Open Source IDS / IPS / NSM engine* [en línea]. 2015. [Consulta: 26 julio 2018]. Disponible en: https://suricata-ids.org/.

VIÑES, V., *análisis de sistemas de detección de intrusos* [en línea]. (tesis) (pregrado). 2005. UNIVERSIDAD ROVIRA I VIRGILI. [Consulta: 24 junio 2018]. 2005. Disponible en: http://deim.urv.cat/~pfc/docs/pfc375/d1126516530.pdf.

XIONG, Z., *An SDN-based IPS Development Framework in Cloud Networking Environment* [en línea]. (tesis) (postgrado). ARIZONA STATE UNIVERSITY. 2014. [Consulta: 14 noviembre 2018]. Disponible en: https://repository.asu.edu/attachments/137329/content/Xiong_asu_0010N_14223.pdf.

ZANNA, P., *Zodiac FX: The world's smallest OpenFlow SDN switch. kickstarter* [en línea]. 2015. [Consulta: 25 junio 2018]. Disponible en: https://www.kickstarter.com/projects/northboundnetworks/zodiac-fx-the-worlds-smallest-openflow-sdn-switch?ref=nav_search&result=project&term=Zodiac FX.

ANEXOS

Anexo A: Aplicación para el controlador RYU

```
from __future__ import print_function
import array

from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import
CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import
set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ipv4
from ryu.lib.packet import icmp
from ryu.lib.packet import tcp
from ryu.lib.packet import udp
from ryu.lib import snortlib

class
SimpleSwitchSnort(app_manager.RyuApp):
  OFP_VERSIONS =
[ofproto_v1_3.OFP_VERSION]
  _CONTEXTS = {'snortlib': snortlib.SnortLib}

def __init__(self, *args, **kwargs):
    super(SimpleSwitchSnort,
self).__init__(*args, **kwargs)
    self.snort = kwargs['snortlib']
    self.snort_port = 3
    self.mac_to_port = {}
    socket_config = {'unixsock': True}
    self.snort.set_config(socket_config)
    self.snort.start_socket_server()

def packet_print(self, pkt):
    pkt = packet.Packet(array.array('B', pkt))
    eth =
pkt.get_protocol(ethernet.ethernet)
    _ipv4 = pkt.get_protocol(ipv4.ipv4)
    _icmp = pkt.get_protocol(icmp.icmp)
    _tcp = pkt.get_protocol(tcp.tcp)
    _udp = pkt.get_protocol(udp.udp)
    if _icmp:
      self.logger.info("%r", _icmp)
    if _tcp:
      self.logger.info("%r", _tcp)
    if _udp:
      self.logger.info("%r", _udp)
    if _ipv4:
      self.logger.info("%r", _ipv4)
    if eth:
      self.logger.info("%r", eth)

@set_ev_cls(snortlib.EventAlert,
MAIN_DISPATCHER)
  def _dump_alert(self, ev):
    msg = ev.msg
    print('alertmsg: %s' %
''.join(msg.alertmsg))
```

```
        self.packet_print(msg.pkt)


@set_ev_cls(ofp_event.EventOFPSwitchFeat
ures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
 match = parser.OFPMatch()
        actions =
[parser.OFPActionOutput(ofproto.OFPP_CO
NTROLLER,

ofproto.OFPCML_NO_BUFFER)]
        self.add_flow(datapath, 0, match,
actions)
    def add_flow(self, datapath, priority,
match, actions):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        inst =
[parser.OFPInstructionActions(ofproto.OFPIT
_APPLY_ACTIONS,
                                     actions)]
        mod =
parser.OFPFlowMod(datapath=datapath,
priority=priority,
                          match=match,
instructions=inst)
        datapath.send_msg(mod)
```

```
@set_ev_cls(ofp_event.EventOFPPacketIn,
MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        in_port = msg.match['in_port']
        pkt = packet.Packet(msg.data)
        eth =
pkt.get_protocols(ethernet.ethernet)[0]
        dst = eth.dst
        src = eth.src
        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})


self.mac_to_port[dpid][src] = in_port
        if dst in self.mac_to_port[dpid]:
            out_port =
self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD
        actions =
[parser.OFPActionOutput(out_port),

parser.OFPActionOutput(self.snort_port)]



if out_port != ofproto.OFPP_FLOOD:
```

```
            match =
parser.OFPMatch(in_port=in_port,
eth_dst=dst)

            self.add_flow(datapath, 1, match,
actions)

        data = None

        if msg.buffer_id ==
ofproto.OFP_NO_BUFFER:

            data = msg.data

        out =
parser.OFPPacketOut(datapath=datapath,
buffer_id=msg.buffer_id,

                            in_port=in_port,
actions=actions, data=data)

        datapath.send_msg(out)
```

Anexo B: Aplicación para la integración mediante PIGRELAY

```
import os
import sys
import time
import socket
import logging

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

SOCKFILE = "/tmp/snort_alert"
BUFSIZE = 65863

# Must to set your controller IP here
CONTROLLER_IP = '127.0.0.1'

# Controller port is 51234 by default.
# If you want to change the port number
# you need to set the same port number in the controller application.
CONTROLLER_PORT = 51234

# TODO: TLS/SSL wrapper for socket

class SnortListener():

    def __init__(self):
        self.unsock = None
        self.nwsock = None

    def start_send(self):
        '''Open a client on Network Socket'''
        self.nwsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            self.nwsock.connect((CONTROLLER_IP, CONTROLLER_PORT))
        except Exception, e:
            logger.info("Network socket connection error: %s" % e)
            sys.exit(1)

    def start_recv(self):
        '''Open a server on Unix Domain Socket'''
        if os.path.exists(SOCKFILE):
            os.unlink(SOCKFILE)

        self.unsock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.unsock.bind(SOCKFILE)
        logger.info("Unix Domain Socket listening...")
        self.recv_loop()

    def recv_loop(self):
        '''Receive Snort alert on Unix Domain Socket and
        send to Network Socket Server forever'''
```

```
        logger.info("Start the network socket
client....")
        self.start_send()
        while True:
            data = self.unsock.recv(BUFSIZE)
            time.sleep(0.5)
            if data:
                logger.debug("Send {0} bytes of
data.".format
                             (sys.getsizeof(data)))
                # data == 65900 byte
                self.tcp_send(data)
            else:
                pass


    def tcp_send(self, data):
        self.nwsock.sendall(data)
        logger.info("Send the alert messages to
Ryu.")



if __name__ == '__main__':
    server = SnortListener()
    server.start_recv()
```

Anexo C: Servidores

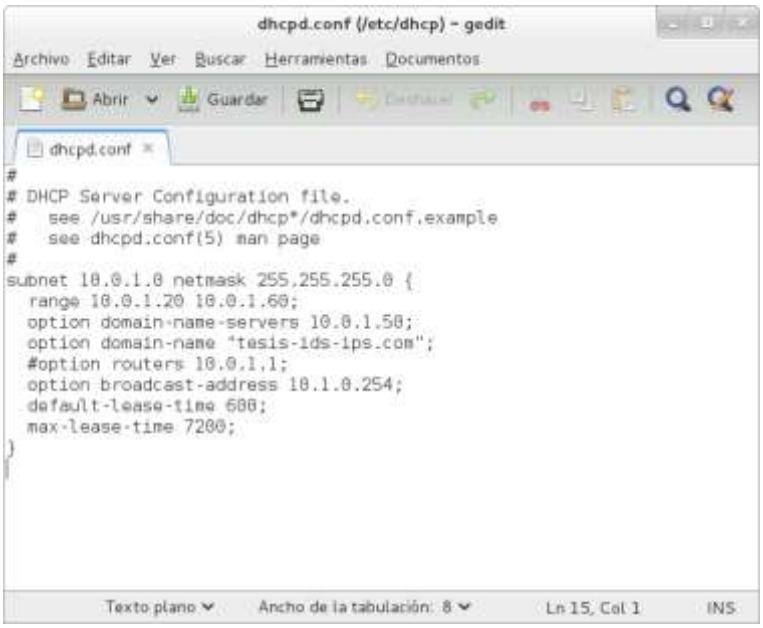
SERVIDOR DHCP

Descarga los ficheros para el servidor DHCP

```
yum install dhcpd
rpm -q dhcpd
```

Modificar el archivo con los parámetros para el servicio DHCP

```
nano /etc/dhcp/dhcpd.conf
```



Iniciar el servicio DHCP

```
systemctl dhcpd start
service dhcpd status
service dhcpd start
```

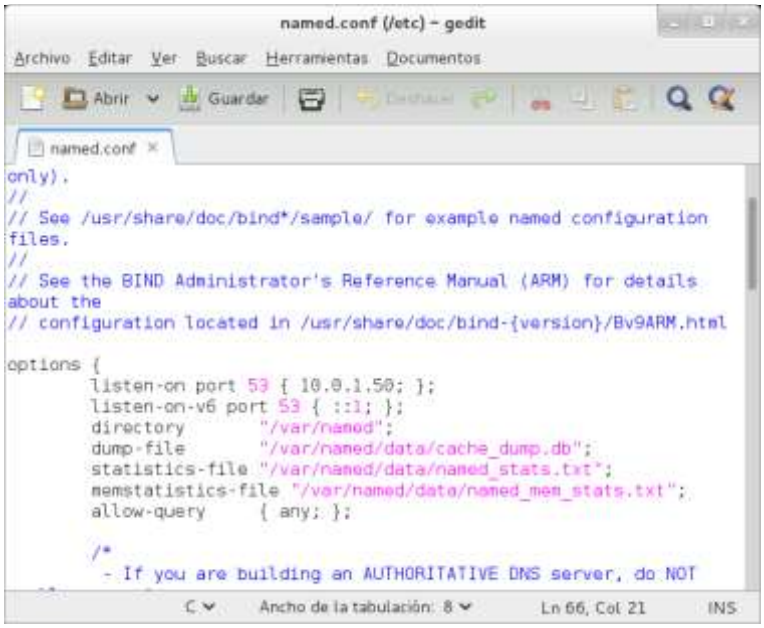
SERVIDOR DNS

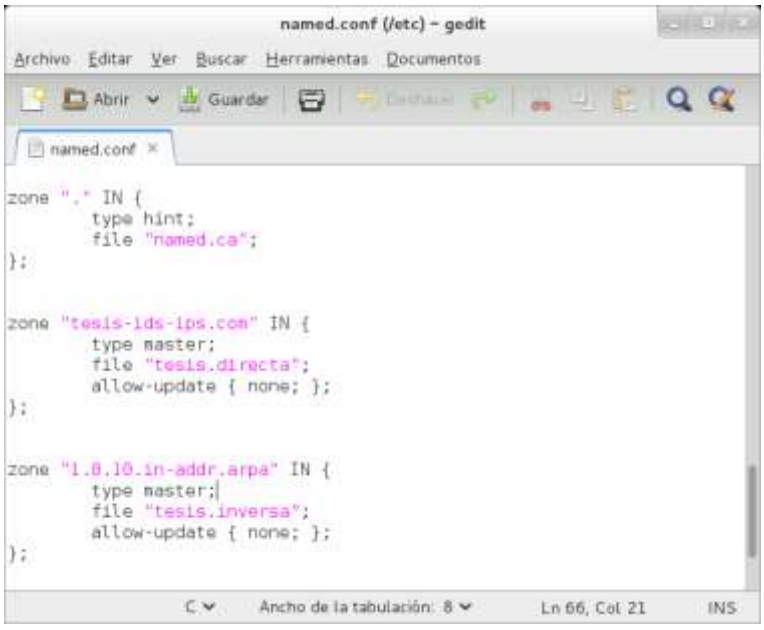
Descargar los ficheros para el servidor DNS

```
yum install bind
rpm -q bind
```

Modificar el archivo con los parámetros para el servicio DNS

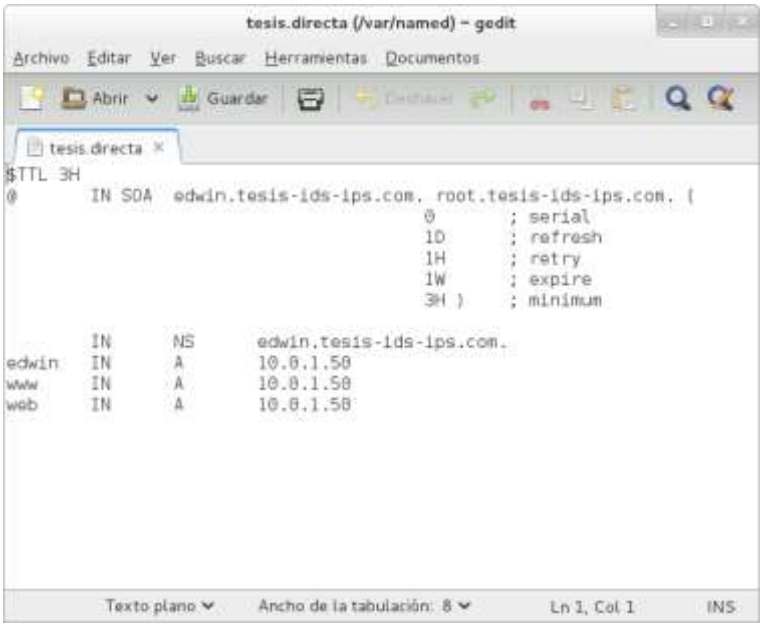
```
cd /etc
nano named.conf
```



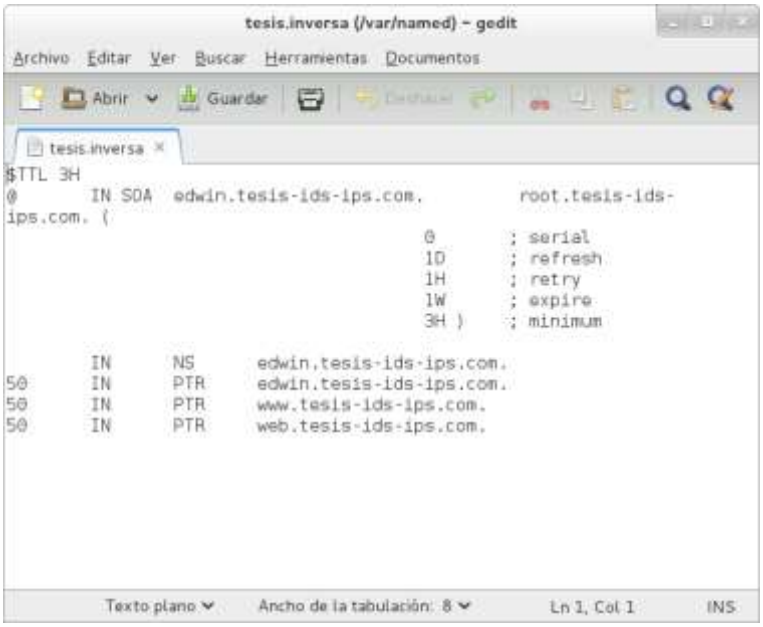


Crear dos archivos con la conexión para el servidor DNS

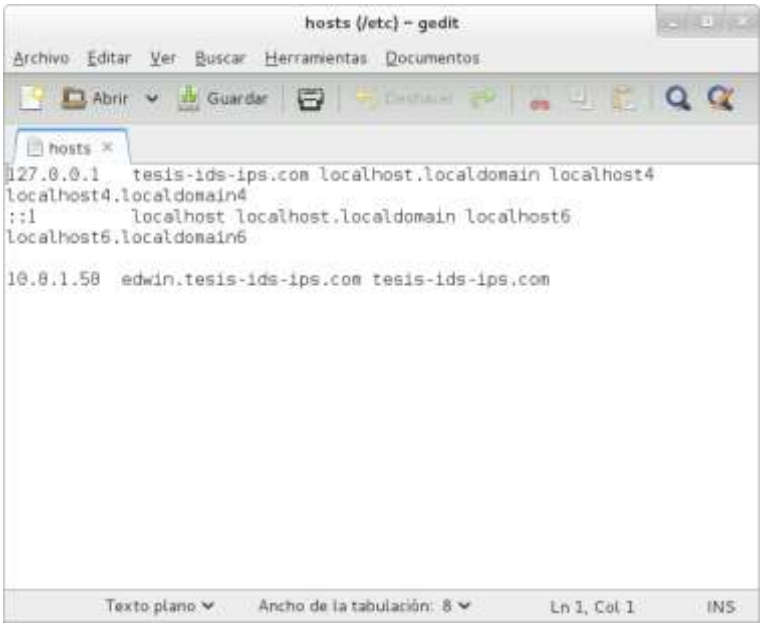
```
cd /var/named
touch tesis.directa
touch tesis.inversa
nano tesis.directa
```



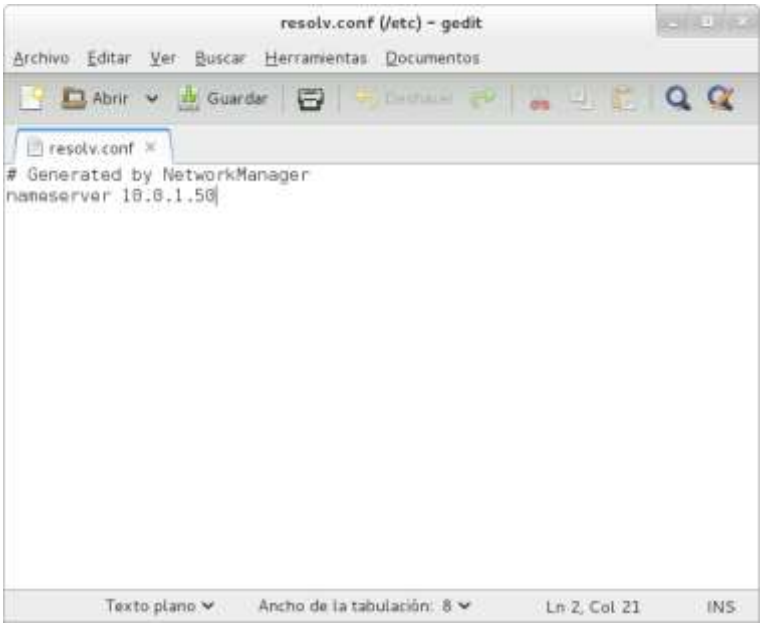

```
nano tesis.inversa
```




```
nano /etc/hosts
```



nano /etc/resolv.conf



Modificar la configuración de los archivos

```
chgrp named prueba.directa
chgrp named prueba.inversa
```

Deshabilitar SELINUX del sistema y ejecutar el servicio

```
cd /ect/selinux
selinux=disabled

systemctl start named
service named start
```

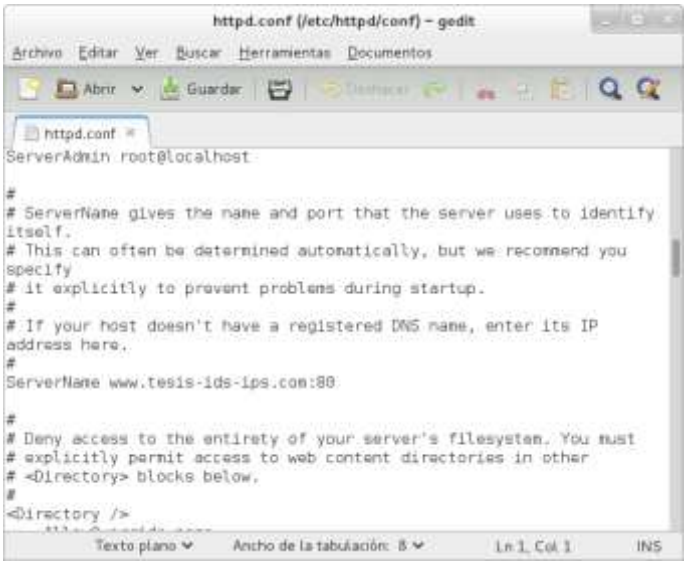
SERVIDOR HTTP

Descargar los ficheros para el servidor HTTP

```
yum install httpd
```

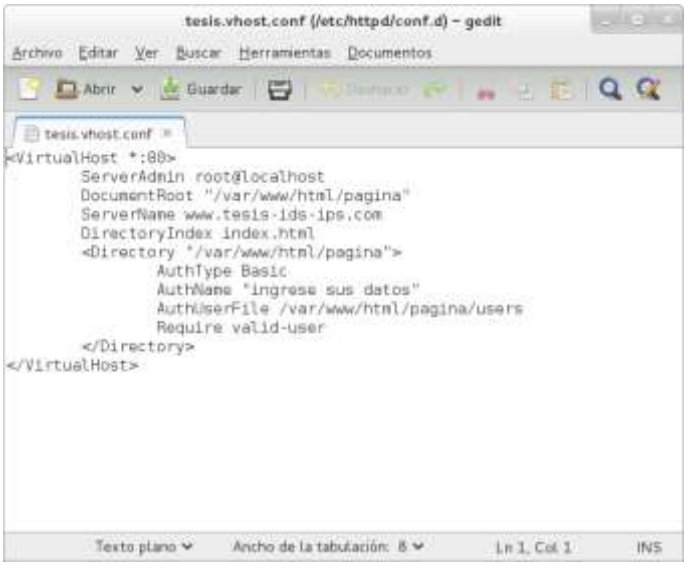
Modificar el archivo con los parámetros para el servicio HTTP

```
cd /etc/httpd/conf
nano httpd.conf
```



Crear el archivo para la configuración de os parámetros del servicio HTTP

```
cd conf.d
touch tesis.vhost.conf
nano tesis.vhost.conf
```



Ubicación de la plantilla de la página web

```
cd /var/www/html
cd pagina
nano index.html
```

Iniciar el servicio

```
systemctl start httpd
service httpd start
```

Anexo D: Tabla de paquetes generados en un intervalo de tiempo

SEGUNDO	CANTIDAD
0	80
1	0
2	1
3	1
4	3
5	1
6	2
7	2
8	1
9	0
10	2
11	421
12	785
13	747
14	795
15	762
16	789
17	785
18	766
19	775
20	750
21	778
22	770
23	762
24	767
25	755
26	767
27	759
28	752
29	769
30	747
31	764
32	761
33	746
34	762
35	763
36	735
37	764
38	665
39	677
40	817
41	756
42	632
43	639
44	588
45	740
46	734
47	585
48	696
49	726
50	737
51	759
52	762
53	748
54	758
55	754
56	736
57	757
58	748
59	756
60	764
61	764
62	770
63	753
64	721
65	766
66	720
67	301
68	1
69	1
70	80
Total	**41800**