UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

TEMA:

"DISEÑO DE UN SISTEMA DE DETECCIÓN DE INTRUSOS (IDS), BASADO EN REDES NEURONALES PARA UNA RED DEFINIDA POR SOFTWARE (SDN) EN LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA) DE LA UNIVERSIDAD TÉCNICA DEL NORTE."

AUTOR: NICOLALDE QUILCA WILLAMS ANDRES

DIRECTOR: MSC. MAYA OLALLA EDGAR ALBERTO

IBARRA-ECUADOR

2021

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
Cédula de identidad	172184131-8
Apellidos y nombres	Nicolalde Quilca Willams Andrés
Dirección	San Antonio de Pichincha - Reino de Quito y Transito Amaguaña
E-mail	wanicolaldeq@utn.edu.ec
Teléfono móvil	0997012022
DATOS DE LA OBRA	
Título	DISEÑO DE UN SISTEMA DE DETECCIÓN DE INTRUSOS (IDS), BASADO EN REDES NEURONALES PARA UNA RED DEFINIDA POR SOFTWARE (SDN) EN LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA) DE LA UNIVERSIDAD TÉCNICA DEL NORTE.
Autor	Nicolalde Quilca Willams Andrés
Fecha	09/02/2021
Programa	Pregrado
Título	Ingeniero en Electrónica y Redes de Comunicación
Director	Ing. Edgar Maya Olalla, Msc

Firma: .....

Nombre: Willams Nicolalde

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

En la ciudad de Ibarra, 9 de Febrero de 2021

El Autor:

Nicolalde Quilca Willams Andrés

CI: 172184131-8

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

MSC. EDGAR MAYA OLALLA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que, el presente trabajo de titulación "DISEÑO DE UN SISTEMA DE DETECCIÓN DE INTRUSOS (IDS), BASADO EN REDES NEURONALES PARA UNA RED DEFINIDA POR SOFTWARE (SDN) EN LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA) DE LA UNIVERSIDAD TÉCNICA DEL NORTE.", fue realizado en su totalidad por el Sr. Nicolalde Quilca Willams Andrés, portador de la cédula de identidad: 172184131-8, bajo mi supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

Ing. Edgar Maya Olalla, Msc
CI: 100270219-7
DIRECTOR DEL PROYECTO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

Dedico esta tesis,

A Dios por guiar cada paso que doy, también por darme la sabiduría y fortaleza necesaria para continuar siempre con firmeza y perseverancia ante las adversidades.

A mis padres; Marcelo Nicolalde y Mercedes Quilca, quienes son un pilar fundamental en mi vida ya que con su ejemplo me inspiran a nunca darme por vencido y confiar en que cada problema que se me presenta tiene su solución. Además, son las personas que nunca dejaran de velar por mi bienestar, razón por la cual todos mis logros son dedicados a ellos.

A mi hermano Bryan Nicolalde, quien es la persona que me alienta a superarme y a dar lo mejor de mí cada día, incluso a querer ser una persona que tomé como ejemplo para su crecimiento académico.

A mi abuelita Rosa Cualchi; y tíos, Johanna y Zami Quilca, por abrirme las puertas y dejarme ser parte de su hogar desde el momento que fui aceptado en la universidad; lo hicieron sin pensarlo dos veces, además de ofrecerme su ayuda y cariño cada vez que los necesité.

Willams Nicolalde

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTOS

Agradezco,

A mis padres y hermano, por estar presentes en cada etapa de mi vida brindándome su amor, esfuerzo y apoyo incondicional, es gracias a ellos que he logrado culminar mis metas propuestas, siendo una, el finalizar esta etapa de formación académica. También quiero que sepan que todo lo que he logrado hasta ahora, es reflejo de la educación y valores que han inculcado en mí, convirtiéndome en la persona que soy.

A los amigos que conocí a lo largo de la vida universitaria, ya que contribuyeron en mi crecimiento tanto emocional como mental, a través de las experiencias y conocimientos que compartieron conmigo, así como los momentos vividos.

A mi tutor y opositores, quienes me brindaron un apoyo académico y profesional a lo largo de la carrera; y el resultado es un crecimiento en estas áreas que son parte esencial en un estudiante, además de facilitar el desarrollo y culminación de este trabajo de titulación.

Willams Nicolalde

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

RESUMEN

El presente trabajo de titulación pretende la implementación de un Sistema de Detección de Intrusos (Siglas en inglés; IDS) integrado a una red neuronal artificial, como opción para mitigar los riesgos de ataques informáticos que provoquen incremento inusual en el tráfico (ataques de tipo activos) hacia una Red Definida por Software (SDN). La cual se apalanca sobre la infraestructura hiperconvergente en el centro de datos de la Faculta de Ingeniería en Ciencias Aplicadas (FICA) en la Universidad Técnica del Norte, de esta manera se presenta un nuevo modelo de administración hacia los dispositivos de red, lo que acelera los procesos de escalabilidad a un menor costo de infraestructura física.

Para comenzar con esta propuesta se realiza un estudio bibliográfico del funcionamiento de la SDN, para determinar el elemento esencial de esta red; de igual manera con el IDS, para comprender su funcionamiento e indicar cómo la red neuronal artificial puede ser integrada. A continuación se diseñan; un escenario de pruebas SDN dentro de la infraestructura hiperconvergente del data center de la FICA y un Snort+RNA; denominado así por ser un IDS (Snort) y tener incorporado un módulo preprocesador que incluye la red artificial perceptrón múltiple capa (MLP). Esta herramienta se caracteriza por permitir al IDS operar sin la necesidad de configurar reglas para la detección de anomalías, a parte genera alertas ante ataques de tipo activos y presentar un registro del tráfico capturado al finalizar cada análisis, en él se encuentran las variables bajo las que el sistema detecta las anomalías.

El Snort+RNA se puso a prueba con la ayuda del modelo PDCA (Siglas en inglés; Plan, Do, Check, Act) que ofrece el estándar ISO/IEC 27001 y de los procesos

proporcionados por el circulo hacker. Los resultados muestran que Snort+RNA detecta las anomalías que provocan los ataques de tipo activo en contra de la SDN, esto es visible tanto en las alertas generadas como en el registro del tráfico capturado, no obstante, no le es posible analizar todos los paquetes que recibe por parte de ataques de DoS puesto que cierto paquetes se quedan en espera o son rechazados. Demostrando así, que aunque el sistema no analiza todos los paquetes que circulan por la red, sí se encarga de proteger a la SDN al brindar sus alertas cuando terceros la intentaron vulnerar con ataques que ocasionaron incremento en el tráfico de la red.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ABSTRACT

This degree work aims to implement an Intruder Detection System (Acronym in English; IDS) integrated into an artificial neural network, an option to mitigate the risks of computer attacks that cause an unusual increase in traffic (active type attacks) to a Software Defined Network (SDN). Which is leveraged on the hyperconverged infrastructure in the data center of the Facultad de Ingeniería en Ciencias Aplicadas (FICA) at the Universidad Técnica del Norte, in this way, a new management model is presented towards network devices, which accelerates scalability processes at a lower cost of physical infrastructure.

To start with this proposal a bibliographic study of the operation the SDN is carried out, to determine the essential element of this network; in the same way with the IDS, to understand its operation and indicate how the neural network can be integrated. They are then designed; an SDN test scenario within the hyperconverged infrastructure of the FICA data center and a Snort+RNA; so called because it is an IDS (Snort) and has a built-in preprocessor module that includes the artificial network Multiple Layer Perceptron (MLP). This tool is characterized by allowing the IDS to operate without the need to configure rules for the detection of anomalies, besides generating alerts against active attacks and presenting a record of the traffic captured at the end of each analysis, it contains the variables under which the system detects anomalies.

The Snort+RNA was tested with the help of the PDCA (In English; Plan, Do, Check, Act) model offered by the standard ISO/IEC 27001 and the processes provided by the hacker circle. The results show that Snort+RNA detects the anomalies that cause

active attacks against the SDN, It is visible both in the alerts generated and in the record of the captured traffic, however, it is not possible to analyze all the packets that it receives from DoS attacks since certain packets are put on hold or are rejected. Demonstrating that, although the system does not analyze all the packets that circulate through the network, it is in charge of protecting the SDN by providing its alerts when third parties tried to violate it with attacks that caused an increase in network traffic.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I.

ANTECEDENTES

Tema

DISEÑO DE UN SISTEMA DE DETECCIÓN DE INTRUSOS (IDS), BASADO EN REDES NEURONALES PARA UNA RED DEFINIDA POR SOFTWARE (SDN) EN LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA) DE LA UNIVERSIDAD TÉCNICA DEL NORTE.

Problema

En los últimos años el uso de las redes definidas por software (SDN) ha permitido a las organizaciones acelerar la implementación y la distribución de aplicaciones reduciendo drásticamente los costos de TI (Tecnología de la Información) mediante la automatización del flujo de trabajo (Cisco, 2019b). El Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte ya emplea este tipo de red, sin embargo se ha dejado de lado la parte de seguridad de información, misma que podría llegar a presentar problemas en el futuro, ya que como cualquier tipo de red, esta propensa a ataques de terceros, (Ashraf & Latif, 2016) mencionan a dos tipos de ataques los activos, que se caracterizan por presentan anomalías en la red al generar gran flujo de datos en esta; y los pasivos, que a diferencia del anterior pasan desapercibidos, puesto que estos ataques son usados para robar los privilegios de usuarios en la red.

El Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas envía y recibe un gran número de paquetes a través de la red diariamente y es por eso, que se optó por el diseño de la SDN (Red Definida por Software) para una mejor administración, mientras que para el área de seguridad de información que es algo que aún no se ha tomado en cuenta, existen los Sistemas de Detección de Intrusos (IDS).

Las IDS son considerados como una herramienta para reforzar un sistema de seguridad, para esto existen dos metodologías, una es llamado Abusos y la otra es conocida como Anomalía. La primera se encarga de comparar patrones de ataques o firmas, mismos que son indicadores de cuando pueden presentarse alguna penetración a la red. Sin embargo, con el pasar del tiempo aparecen nuevos tipos ataques, haciendo vulnerable esta metodología ya que, si el ataque no se encuentra entre los patrones, no será detectado por el IDS. Mientras que el segundo podría reemplazar el problema de los patrones de ataques ya que esta trabaja con perfiles de comportamiento de la red, es decir, si la red presenta cambios de los cuales no se encuentran registrados en los perfiles, el sistema lo detectara como anomalía, haciéndolo efectivo contra los nuevos tipos de ataques que puedan existir, estos perfiles son creados de forma manual conforme la red tenga cambios, convirtiendo esto en una falencia para esta metodología. (Luna, 2015)

Para disminuir la falencia de esta última metodología se propone un Sistema de Detección de Intrusos (IDS) basado en redes neuronales la cual se encargará de crear los perfiles de comportamiento de la red de manera autónoma y en caso de detectar algún cambio indeseado, este sistema pueda enviar las alertas de las anomalías que ocurran en la red, cubriendo así también el área de la seguridad en las Redes Definidas por Software para el Centro de Datos de la FICA.

Según (Lau Fernández & Iren, 2017) las redes neuronales son una de las técnicas que ha presentado amplias ventajas en la aplicación dirigida a los Sistemas de Detección de Intrusos. Al trabajar prediciendo resultados en función a los datos que ingresan en el clasificador neuronal, es así, que para este tipo de sistemas bastará con ingresar los datos correctos y se encargará de predecir el perfil más adecuado al comportamiento de la red, evitando así futuros ataques mediante las alertas que el IDS envié.

Objetivos

Objetivo General.

Diseñar un Sistema de Detección de Intrusos (IDS), basado en redes neuronales en una Red Definida por Software (SDN), para evitar futuros ataques de tipo activos con la guía del estándar de seguridad ISO/IEC-27001.

Objetivos Específicos.

Estudiar el estado del arte de las redes neuronales, las influencias en los Sistemas de Detección de Intrusos (IDS) y los distintitos modos de ataque de tipo activo que presentan anomalías en una red.

Establecer la situación actual, además de elaborar un escenario basado en una Red Definida por Software en el Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas (FICA) con sus pruebas de funcionamiento.

Realizar los perfiles de comparación del Sistema de Detección de Intrusos (IDS) usando las redes neuronales basados en las anomalías que puedan presentarse en la Red Definida por Software (SDN).

Realizar las pruebas de funcionamiento del Sistema de Detección de Intrusos (IDS) basado en redes neuronales, haciendo ataques de tipo activos con la guía del estándar de seguridad ISO/IEC-27001.

Alcance

El presente proyecto propone cubrir el campo de la seguridad de la información, con un Sistema de Detección de Intrusos basado en redes neuronales, para las Redes Definidas por Software de la Universidad Técnica del Norte dirigidos al Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas.

Lo primero que se realizará en el proyecto es la fundamentación teórica de las redes neuronales, los Sistemas de Detección de Intrusos y las Redes Definidas por Software, además de un estado del arte que existen de las redes neuronales y su relación con los IDS para la prevención de ataques a la red, indicando el cómo estas disminuyen las falencias que presentan los Sistemas de Detección de Intrusos enfocando principalmente en los ataques de tipo activos, mismo que son identificados por presentar anomalías en las redes, y como estos sistemas se han implementado en las SDN, para que de esta manera se pueda elegir el tipo de red neuronal, al igual que su respectivo software, siendo estos los que más se adecuen al proyecto propuesto.

La implementación del IDS será en una Red Definida por Software (SDN), haciendo uso de los módulos que esta contiene para la virtualización de dispositivos, misma que servirá para el diseño de un escenario en el Centro de Datos de la Facultad de Ingeniería en Ciencias Aplicadas en la Universidad Técnica del Norte, especificando los elementos de los cuales constará la red tanto en software como en hardware.

Los IDS trabajan con perfiles de comparación, para el diseño de estos perfiles de comportamiento se hará uso de las redes neuronales, mismas que estas estarán formadas con los datos necesarios que se obtendrán de la captura del tráfico de la SDN tanto en su estado normal como cuando presente anomalías.

Finalmente, con IDS basado en redes neuronales funcionando se hará una verificación de su estado, con ataques de tipo activos, para que posteriormente el sistema alerte acerca de las anomalías en la SDN que presenten estos, haciendo uso de las políticas de seguridad que ofrece el Estándar ISO/IEC-27001 y de ser necesario, reconfigurando el sistema para así mejorar este estado de verificación.

Justificación

Con el pasar del tiempo la cantidad de flujo de datos que viaja a través de la red ha aumentado exponencialmente, haciendo más difícil su administración al requerir nueva tecnología que lograr su propósito, estos incremento de flujo de datos también ha provocado que la información tenga un valor primordial para los usuarios ya que esta puede contener datos que son esenciales para el correcto funcionamiento de una entidad de cualquier índole, llegando a ser tan importantes que terceros quieran hacerse con estos.

La Red Definida por Software (SDN) ha surgido en los últimos años con el fin de solucionar los problemas de administración de red ya que este término se refiere a que esta arquitectura permite separar el plano de control, del plano de datos, para conseguir redes más programables, automatizables y flexibles (Sandoval, 2018).

Todo tipo de red por más actual que esta sea tiene vulnerabilidades, de las cuales se pueden aprovechar los hackers para obtener información, para evitar esto existen varias soluciones (Luna, 2015) y una de ellas son los Sistemas de Detección de Intrusos (IDS), su funcionamiento consiste en detectar o monitorizar eventos dentro de un equipo o a una red en busca de accesos no autorizados (Carrera, Castillo, & Quizhpi, 2011). La desventaja que presentan estos sistemas es que para que puedan detectar alguna anomalía necesitan de un perfil de comportamiento de la red para su comparación el cual debe ser programado manualmente y debe cambiar cada vez que la red cambie.

Es así como, el presente proyecto propone el diseño de un Sistema de Detección de Intrusos (IDS) basado en redes neuronales, las redes neuronales se encargarán de realizar los perfiles de comparación prediciendo los cambios que tendrá la red, haciendo más fácil la adaptabilidad del IDS a cualquier escenario, la verificación de este sistema se lo hará con ataques de tipo activos, además de ser guiados por el Estándar de Seguridad ISO/IEC-27001.

CAPÍTULO 2.

FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza un estudio bibliográfico acerca de la arquitectura que compone una Red Definida por Software (SDN), además de los protocolos de comunicación que se utilizan, con el fin de implementar un Sistema de Detección de Intrusos (IDS) el cual se describirá en esta sección, desde su clasificación, métodos de respuesta, además de las posibles ubicaciones dentro de una red. Este IDS estará compuesto por perfiles realizados con redes neuronales explicando así de forma básica su funcionamiento para finalmente detallar el estándar ISO/IEC 27001, el cual brinda una guía en la seguridad de la información.

2.1. Red Definida por Software (SDN)

De acuerdo con Open Networking Foundation (2020) una SDN se define como una arquitectura que "desacopla el control de red y las funciones de reenvío. Permitiendo que el control de la red se vuelva directamente programable y que la infraestructura subyacente se abstraiga para las aplicaciones y los servicios de red". Sin embargo, Cisco (2019b) dice que una SDN es diseñada con la finalidad de que las redes de datos sean más flexibles y ágiles, es decir, una SDN puede: diseñar, construir y administrar redes, todo esto manteniendo separados los planos de control y de datos.

2.1.1. Arquitectura SDN.

La arquitectura de una SDN está compuesta por tres capas conocidas como: Aplicación, Control y Capa de Infraestructura; las cuales se detallan en la Figura 1 ***Figura 1.*** *Arquitectura básica de red basada en SDN.*

(Sahoo, Mohanty, Tiwary, Mishra, & Sahoo, 2016).

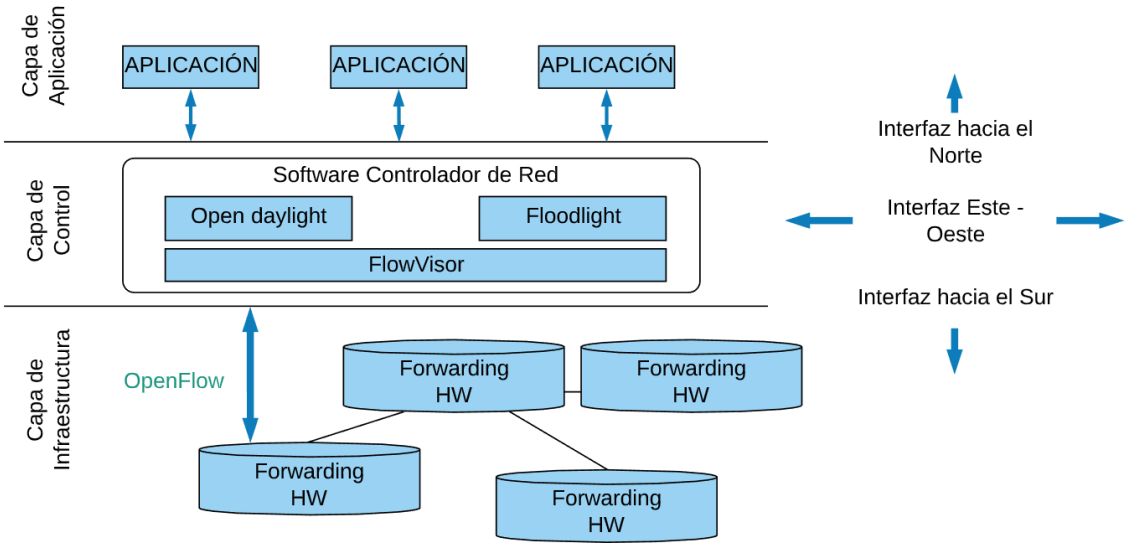


Figura 1. *Arquitectura básica de red basada en SDN.*

Fuente: *(Lakshmanan, 2018). Recuperado de https://bit.ly/2CyFJvj.*

a. Capa de Aplicación

La Capa de Aplicación se encuentra dedicada a los usuarios finales al estar compuesta de aplicaciones y servicios encargados de definir el comportamiento de la SDN. Otra cualidad que la identifica es la comunicación que posee con la capa de Control a través de Interfaces hacia el Norte (NourthBound), un ejemplo de esta es la API Rest (Caicedo B & López S, 2017; Sahoo et al., 2016).

Las aplicaciones de esta capa pueden cubrir ciertas zonas como lo menciona Krishnan, Duttagupta, & Achuthan (2020), las cuales son: Ingeniería de tráfico, Medición y monitoreo, y Seguridad; cada una las zonas mencionadas se describen en el siguiente apartado.

La *Ingeniería de Tráfico* tiene como objetivo: analizar, regular y predecir el comportamiento de los datos en la red. Además, optimiza el rendimiento de la red conforme a los Acuerdos de Niveles de Servicio (SLA) que se encuentren establecidos. Mientras que la *Medición y monitoreo* se encargan de preparar al sistema para algún cambio de flujo en la red; a través de la medición de tráfico realizado con técnicas de

muestreo y estimación, disminuyendo de esta manera la carga en el plano de control. Finalmente, la zona de *Seguridad* cuida las tres capas de la SDN usando su flexibilidad para establecer políticas que salvaguarden la información de la red.

En el caso de la zona de seguridad se toma como ejemplo a la aplicación Defense4All, tal como se presenta en la Figura 2; donde da a conocer la interacción que tiene esta aplicación con el controlador OpenDaylight y como ayuda en la mitigación de ataques DoS/DDoS según menciona Radware (2020).

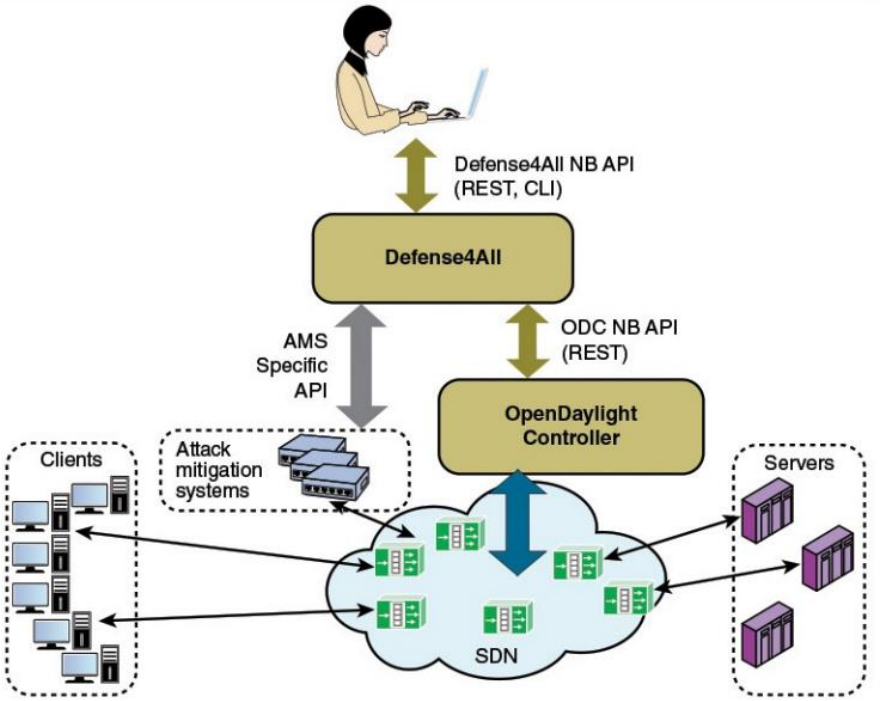


Figura 2. *Aplicación DDoS OpenDaylight*

Fuente: *(Stalling, 2016)*

b. Capa de Control

La Capa de Control, es considerada la más importante de la arquitectura SDN puesto que gestiona a la capa de aplicación y la capa de infraestructura, en esta se proporcionan el control del comportamiento y los recursos del sistema mediante indicaciones que se envían a la capa aplicación. Otra peculiaridad de esta capa es el

manejo de la interfaz NorthBound así como también la interfaz hacia el Sur (SouthBound) (Albowarab, Zakaria, & Abidin, 2018; Oladunjoye, 2017).

La Figura 3 ejemplifica las funciones que se realizan en esta capa y que cualquier controlador debe proporcionar, teniendo en cuenta que cada una de las funciones tiene relación directa con las diferentes interfaces, tal como se señala en la Tabla 1 (Stalling, 2016).

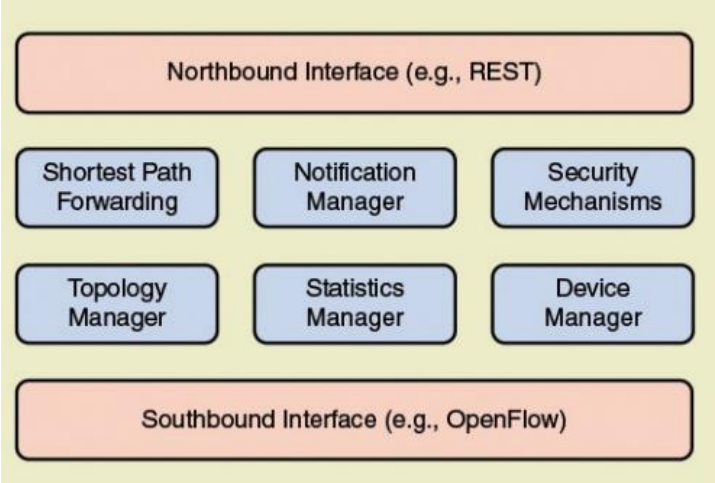


Figura 3. *Funciones e Interfaces del Plano de Control*

Fuente: (*Stalling, 2016)*

Tabla 1

Funciones del Controlador SDN

Función	Descripción	Interfaz
Shortest path forwarding	Utiliza la información de los switches para establecer las rutas preferidas.	
Notification Manager	Recibe, procesa y reenvía eventos a las aplicaciones, tales como cambios de estado y alarmas de seguridad.	Northbound Interface
Security Mechanisms	Proporcionan y seguridad entre aplicaciones y servicios.	
Topology Manager	Construye y mantiene la topología con la información que mantiene la interconexión de switches.	Southbound Interface

Statsitics Manager	Recoge datos sobre el tráfico mediante la información de los switches.
Device Manager	Configura los switches, sus parámetros y atributos, además de gestionar las tablas de flujos.

Fuente: *(Stalling, 2016)*

Al igual que toda red de comunicación, una SDN también necesita una función de *Enrutamiento* (Routing), quien se encarga de aprender rutas para encaminar los datos e información a través de la red. Oladunjoye (2017) señala dos tipos de protocolos de enrutamiento: Los Protocolos de Enrutamiento Interno (con sus siglas en inglés IRP) y los Protocolos de Enrutamiento Externo (con sus siglas en inglés ERP).

El primer tipo, los IRP, operan dentro de un Sistema Autónomo (conocido como AS); mientras que el segundo tipo, es decir, los ERP; trabajan entre Sistemas Autónomos. Es así que una Red Definida por Software centra la función de enrutamiento dentro de la capa de Control, desarrollando una visión del estado de la red para el cálculo de las rutas más cortas adicionalmente la aplicación de políticas de enrutamiento basados en aplicaciones, mejorando el rendimiento de los switch ya que estos no tendrán la carga de procesamiento y almacenamiento que tiene que ver con el enrutamiento (Stalling, 2016).

Dentro del enrutamiento existen a su vez dos funciones más que se encargan que este cumpla con su cometido, dichas funciones son: el descubrimiento de enlace y el gestor de topología. El *Descubrimiento de Enlace* es provocado por el tráfico desconocido que entra en la red del Controlador, ya sea desde un host conectado o desde algún router de redes vecinas. Mientras que en el *Gestor de Topología* involucra a la actualización constante del estado de la red para el cálculo de rutas, estableciendo así el camino más corto entre dos nodos de la capa de Infraestructura o un nodo y un host (Oladunjoye, 2017; Stalling, 2016).

c. Capa de Infraestructura

Por último, está la Capa de Infraestructura, en aquella se centran los elementos físicos de la red (hardware), los que se encargarán del transporte y procesamiento de la información (enrutamiento) según como se haya indicado en la capa de control; esta trabaja con interfaces southbound (Bhushan & Gupta, 2019; Blial, Ben Mamoun, & Benaini, 2016).

Las funciones en las que se centran los dispositivos de red de la capa de infraestructura son: función de apoyo de control y función de reenvío de datos. La primera, la *Función de Apoyo de Control* consiste en la interacción con la capa de control a través de interfaces. Aquí, el switch se comunica con el controlador y lo gestiona mediante el protocolo OpenFlow. Mientras que la segunda función, la *Función de Reenvío de Datos* acepta el flujo de datos que proviene de otros dispositivos de red y sistemas de extremo, a lo largo de los caminos de reenvío que han sido calculados y establecidos de acuerdo con las reglas que definieron las aplicaciones SDN (Blial et al., 2016; Krishnan et al., 2020).

Otra cualidad que cabe mencionar en esta capa, son los protocolos bajo los que operan sus dispositivos de red, ya que consisten en flujos de paquetes IP. Al examinar el encabezado IP y posiblemente otras cabeceras de cada paquete; se puede mencionar protocolos tal como: a TCP, UDP u otro protocolo que sea de transporte o aplicación; basado en esto, se decide la ruta de reenvío (Stalling, 2016); lo ya mencionado en el presente acápite se grafica en la Figura 4.

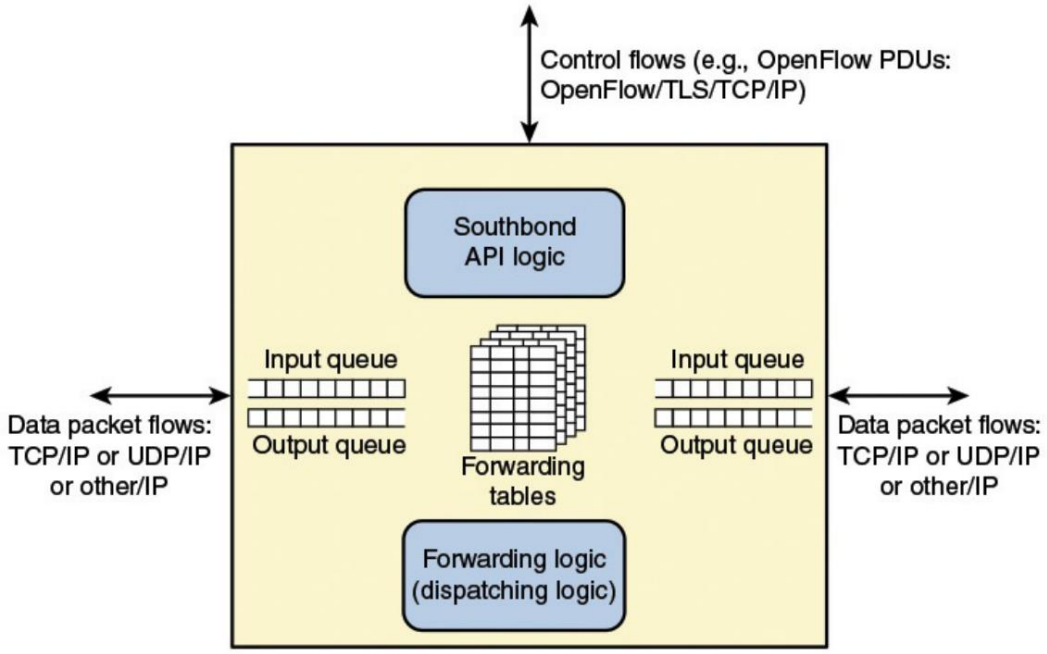


Figura 4. *Dispositivos en Capa de Infraestructura*

Fuente: *(Stalling, 2016)*

2.1.2. Interfaces del controlador SDN.

Como ya se mencionó en el apartado b de la sección 2.1.1, el controlador permite la interacción entre las capas de una SDN, dicha interacción es ocasionada por ciertas interfaces que se usan dependiendo de la dirección en la que viaje la información. Sahoo et al. (2016) menciona a dos clases de interfaces ya citadas que llevan a cabo la comunicación entre capas, llamadas; interfaz Northbound e interfaz Southbound. Sin embargo, E. Pérez (2018) nombra una tercera interfaz conocida como interfaz de protocolos East/Westbound; cada una de las interfaces mencionadas se ilustran en la Figura 5 y son puntualizadas de mejor manera después de este apartado.

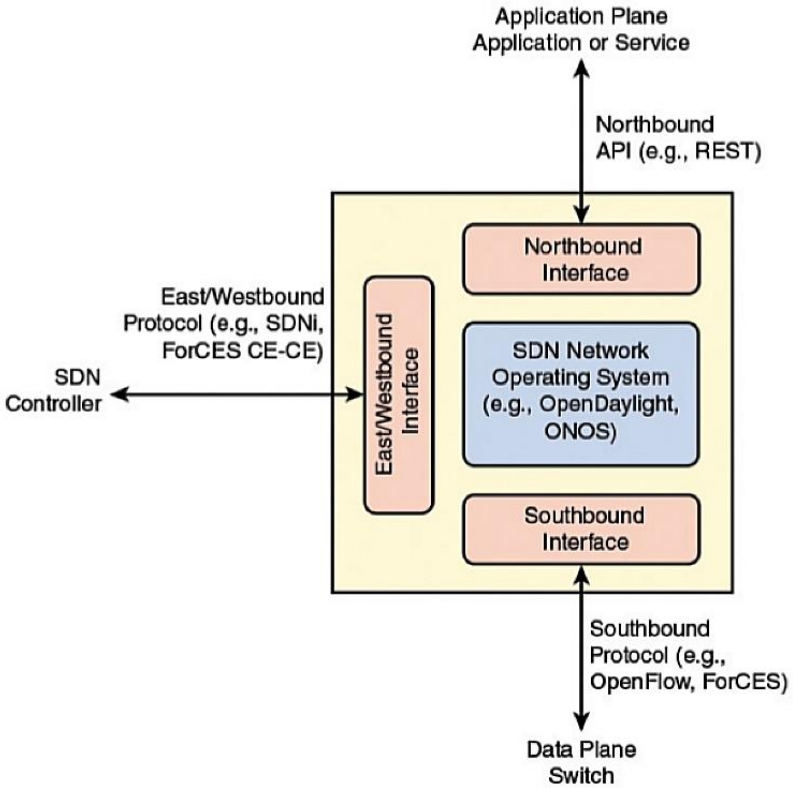


Figura 5. *Interfaces de Controlador SDN*

Fuente: *(Stalling, 2016)*

La *Northbound* es dedicada únicamente a la comunicación entre las capas de aplicación y control, con el propósito de que los desarrolladores de aplicaciones puedan administrar la red a través de software y de esta forma se programen las funciones que debe cumplir cada dispositivo. En contraste con la interfaz anterior, la *Southbound* crea un canal que permite interactuar a las capas de control e infraestructura. Éste canal es creado mediante el protocolo OpenFlow (detallado en la sección 2.1.3), ofreciendo una comunicación segura entre el controlador y los dispositivos de la infraestructura (Sahoo et al., 2016).

En lo que respecta a la *interfaz de protocolos East/Westbound*, es usada en casos especiales, es decir, cuando una arquitectura SDN presenta múltiples controladores; dicha interfaz facilita la comunicación entre varias arquitecturas SDN (E. Pérez, 2018).

2.1.3. OpenFlow.

Open Networking Foundation (2020) menciona que el protocolo OpenFlow se estandarizó por la Open Networking Foundation (ONF) en el año 2012 y Cisco (2019a) lo define como una infraestructura de reenvío que se basa en el flujo y en una Interfaz de Programación de Aplicaciones (API), además, crea un canal seguro entre el controlador y los dispositivos para el reenvío de funciones desde la capa de aplicación hacia la capa de infraestructura.

El funcionamiento del protocolo está basado en que un controlador lógico puede administrar a un switch OpenFlow y cada switch compatible con éste; con la ayuda de una o más tablas de flujo. Las tablas de flujo son las encargadas de la búsqueda de los paquetes, considerando ciertas medidas o reglas con respecto a su reenvío (Denazis, Hadi Salim, Meyer, & Koufopavlou, 2015).

2.1.3.1. Arquitectura del OpenFlow.

La arquitectura de OpenFlow consta de múltiples conmutadores, aquellos están compuestos por un canal seguro y tablas de flujo, además de ser administrados por controladores (E. Pérez, 2018); y ambos elementos, tanto conmutador como controlador son OpenFlow y se comunican a través del protocolo que lleva su mismo nombre; toda esta arquitectura es representada en la Figura 6. Cabe destacar que cada uno de los componentes referidos en este apartado se detallan en los siguientes acápites:

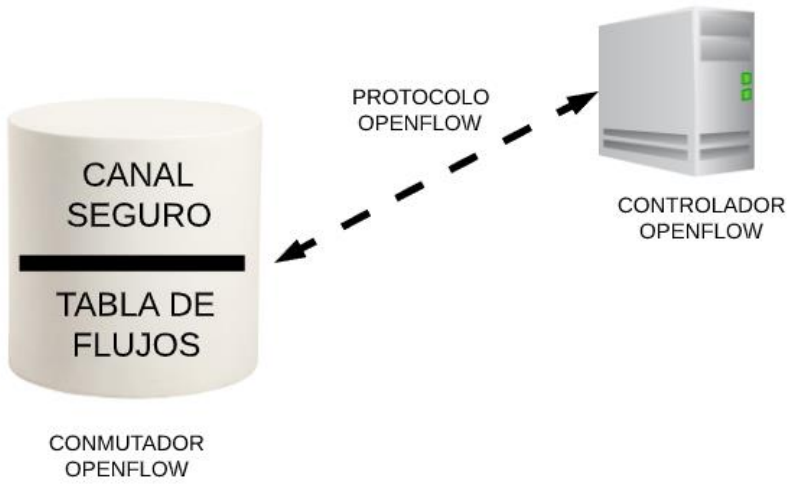


Figura 6. *Arquitectura básica de OpenFlow.*

Fuente: *(Yagües, 2015)*

a) Definición de un flujo

Dentro del tráfico de red, un flujo se entiende como el paso de información ya sea; a través del Protocolo de Control de Transmisión (TCP), mediante paquetes con direcciones MAC o IP, paquetes con etiquetas de Red de Área Local Virtual (VLAN), entre otros (E. Pérez, 2018).

b) Conmutador OpenFlow

Un conmutador OpenFlow está compuesto por una o varias tablas de flujo y una tabla de grupo, con el propósito de crear un canal seguro, al utilizar el protocolo Openflow, el controlador gestiona a los conmutadores OpenFlow que se encuentren habilitados. Dentro existen tablas conformadas por entradas de flujo y estas a su vez se dividen en tres campos: encabezado, contadores e instrucciones; siendo las últimas usadas para asignar coincidencias a los paquetes (Pereira & Gamess, 2017; E. Pérez, 2018).

c) Canal OpenFlow o Canal Seguro

En conformidad a lo explicado en el literal b de este apartado, el canal Openflow permite una comunicación protegida entre los conmutadores OpenFlow y el

controlador, a esta se la conoce como canal seguro ya que utiliza una encriptación de Seguridad de Capa Transporte (TLS) aparte de operar directamente sobre el Protocolo de Control de Transmisión (TCP); dentro del protocolo se admiten mensajes categorizados como: de controlador a conmutador, asincrónico y sincrónico (Yazdinejad, Bohlooli, & Jamshidi, 2018). Las categorías ya nombradas se especifican en el siguiente inciso.

Los *mensajes de controlador a conmutador* se caracterizan porque inician en el controlador y son usados para administrar información acerca del estado del conmutador. Por otro lado, los *mensajes asincrónicos*, son iniciados por el conmutador con el objetivo de actualizar al controlador con eventos de la red o en caso de que haya algún cambio en el conmutador. Mientras que los *mensajes sincrónicos* pueden ser iniciados tanto por el conmutador como por el controlador sin necesidad de solicitud previa (E. Pérez, 2018).

d) Controlador OpenFlow

La función primordial de este componente es gestionar las tablas de flujos de los conmutadores, ya sea agregando o eliminando entradas de flujo. El tener varios controladores genera redundancia e incrementa la confiabilidad de la infraestructura, dado que si algún controlador deja de funcionar otro lo reemplazará y así esta continuará operando con normalidad (Stalling, 2016).

2.1.3.2. Evolución del protocolo OpenFlow.

La Tabla 2 sintetiza las versiones del protocolo OpenFlow que han ido evolucionando con el pasar de los años. Yazdinejad et al. (2018) expone una tabla que contiene las características que cada versión, desde la 1.0 hasta la 1.4, incluido su fecha de lanzamiento, el número de campos que contiene, además de la cantidad de tabla de flujos y si tiene o no la capacidad de brindar una comunicación con varios controladores.

Tabla 2

OpenFlow (OF) campos de comparación.

Versiones de OF	Fecha de lanzamiento	Número de campos en cabecera	Tabla de flujos	Comunicación simultánea con múltiples controladores	Características
OF v1.0	Diciembre 2009	12	Única tabla de flujos	No	Soporta una sola tabla de flujo, comparación fija, campos de comparación L2 y L3 (IPv4)
OF v1.1	Febrero 2011	15	Múltiple tabla de flujos	No	Múltiples tablas, soporta MPLS, VLAN, unicast – multicast, Grupo de tablas, puerto virtual, habilita la aplicación de acciones
OF v1.2	Diciembre 2011	36	Múltiple tabla de flujos	Si	Múltiples controladores, campos de comparación IPv6, cambio de roles, más flexible en comparación y reescritura
OF v1.3	Abril 2012	40	Múltiple tabla de flujos	Si, habilitado conexiones auxiliares	Múltiples canales paralelos entre el conmutador y el controlador, soporta 802.1ah PBB, QoS, capacidad y negociación de versiones, filtro de eventos, entrada de tabla omitida
OF v1.4	Agosto 2013	41	Múltiple tabla de flujos	Si, habilitado conexiones auxiliares	Soporta puertos ópticos, desarrollo y evolución gradual, flujo de tabla sincronizado, monitoreo, desalojo, tabla de salida, paquete programado

Fuente: *Adaptado de (Yazdinejad et al., 2018)*

2.1.4. Software para Controladores.

El controlador dentro de la SDN es la capa que mantiene en armonía a toda su arquitectura, es decir, brinda una comunicación entre capas, Bholebawa & Dalal (2018) define al controlador SDN como el "cerebro" de la red, porque a través de sus distintas interfaces, envía y recibe información de las demás capas. Siendo el controlador un software que se adecua fácilmente con las necesidades del administrador de la red; se lo puede encontrar tanto de forma comercial como gratuita (Código abierto). En la Tabla 3 se describen algunos de los controladores realizados con código abierto, señalando también el lenguaje de programación en el que fueron diseñados, todas las características aquí mencionadas se mantienen en la actualidad, sin presentar ningún cambio desde su creación (Quincozes et al., 2019).

Tabla 3

Lista de controladores OpenFlow.

Nombre del controlador	Lenguaje de programación	Proveedor de licencias
OpenDaylight	Java	EPL-1.0. (Licencia pública Eclipse)
ONOS	Java	Apache
Ryu	Python	Apache
Floodlight	Java	Apache
Faucet	Python	Apache

Fuente: *Adaptado de (Bholebawa & Dalal, 2018; Pereira & Gamess, 2017)*

2.1.5. OpenDaylight.

El proyecto OpenDaylight es de código abierto, organizado por la Fundación Linux en el año 2013 (Lovato, 2019). Además, incorpora la participación de las principales organizaciones de redes; incluidos, los usuarios de la tecnología SDN y vendedores de productos SDN; el enfoque de este es permitir a los participantes de la

industria a unirse para el desarrollo de módulos básicos añadiendo así un valor único (OpenDaylight, 2019).

La Figura 7 plantea una vista de la arquitectura de OpenDaylight, mostrando las cinco capas lógicas de las que está compuesta según (Stalling, 2016).

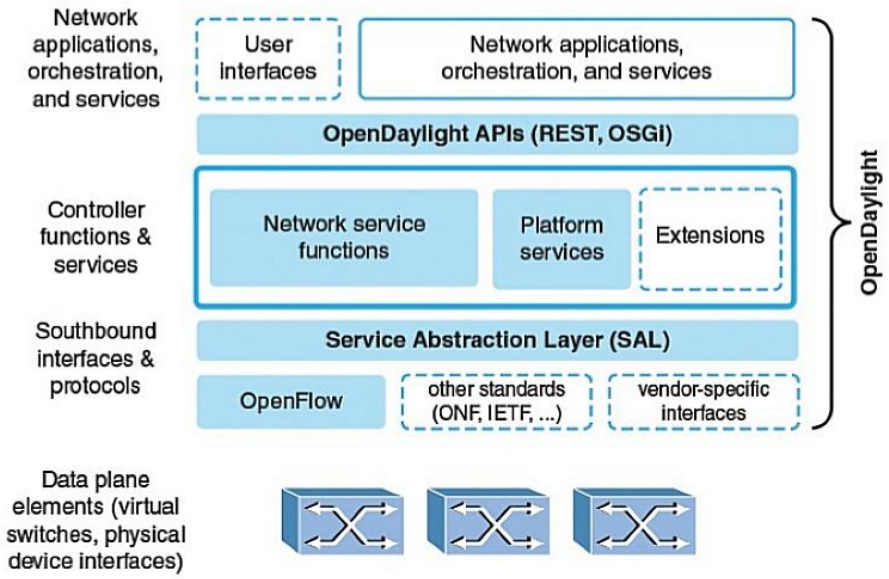


Figura 7. *Arquitectura OpenDaylight*

Fuente: *(Stalling, 2016)*

A través de la Tabla 4, se definen las cinco capas lógicas planteadas en la figura anterior, con el fin de entender la función que cumplen cada una de ellas dentro de OpenDaylight.

Tabla 4

Capas Lógica en OpenDaylight

Capa Lógica	Definición
Aplicaciones de red, orquestación y servicios	Son aplicaciones que usa el controlador para reunir algoritmos, analizar la red y luego orquestar una nueva normativa, a través de la red
Interfaz de Programación de Aplicaciones (APIs)	Es un conjunto de interfaces similares a las funciones del controlador OpenDaylight, brindando una comunicación entre el controlador y las aplicaciones.

Funciones y servicios del Controlador	Funciones y servicios de la capa de Control de una SDN.
Capa de Servicio de Abstracción (SAL)	Proporciona una vista uniforme de los recursos en la capa de infraestructura, de modo que las funciones de la capa de control se implementen independiente de las interfaces y el protocolo.
Interfaces y protocolos Hacia el Sur	Soportes OpenFlow y otros protocolos estándar hacia el Sur.

Fuente: *(Stalling, 2016)*

2.2. Sistema de Detección de Intrusos (IDS)

Un IDS es una herramienta que puede utilizarse en conjunto con otras similares, como es el caso del firewall, brindando así seguridad a los activos de información que posee una organización ya sea privada o pública, además permite una implementación tanto en software como hardware; las ventajas que presenta el primero es su flexibilidad, además de brindar un mayor control y facilitar su actualización, en contraste con el de hardware que ya tiene sus características predefinidas. Otro rasgo que caracteriza a los IDS es: monitorear y detectar cuando una red o host está siendo atacado por alguna persona mal intencionada y dependiendo de cómo esté administrado; envía una alerta de ataque al personal encargado de la red o realiza acciones para contrarrestar el ataque (Iren et al., 2017; Tayyebi & Bhilare, 2018).

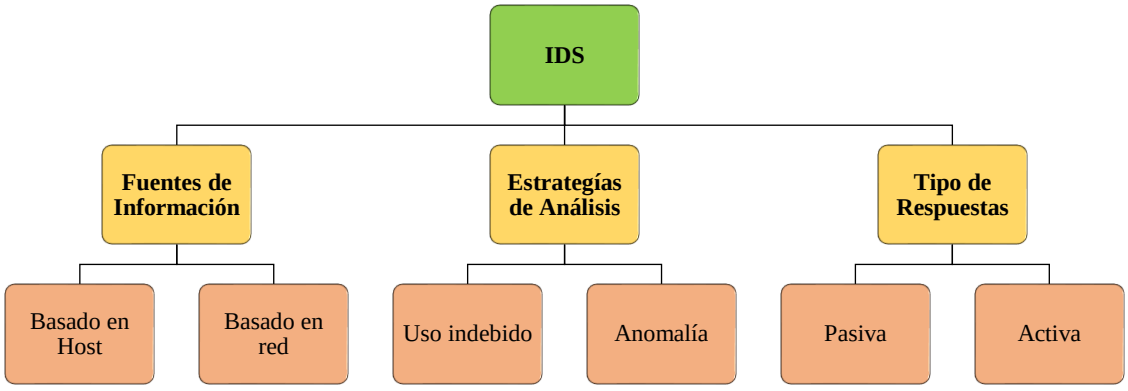


Figura 8. *Diagrama de la estructura de un IDS*

Fuente: *(Iren et al., 2017)*

2.2.1. Clasificación de las IDS.

Según Carvajal (2015) & Iren et al. (2017) un IDS se clasifica en tres grupos; por su fuente de información, dependiendo de las estrategias de análisis y mediante los tipos de respuesta, tal y como se grafica en la Figura 8, en donde se exhibe el diagrama de estructura de un IDS.

La *fuente de información* hace referencia al lugar en el que se enfocará el IDS, puede ser a un host en específico, tomando así el nombre de Host Intruder Detection System (HIDS) o a una red en general, en cuyo caso se denomina Network Intruder Detection System (NIDS) (Iren et al., 2017; Vallejo, 2018).

En tanto que, si se lo clasifica por su *estrategia de análisis,* el método que usa el IDS para determinar si se presentó o no alguna intrusión, requiere que el sistema haya obtenido información previa. Kairi (2017) propone dos estrategias; la primera es la detección del mal uso o uso indebido y la segunda es la detección de anomalías.

Para la detección del uso indebido es necesario contar con una base de datos con los tipos de ataques de los que la red puede ser víctima, y de ataques que ya han sido realizados hacia la red, debido a que esta estrategia se basa en la comparación de la información recopilada y los patrones que estén almacenados en su base de datos, esta es la estrategia más usada comercialmente. Salunkhe & Mali (2017) menciona que la desventaja de este tipo de estrategias es que la red queda propensa a nuevos ataques o a aquellos que no se encuentran registrados en su base de datos. Mientras que la estrategia por detección de anomalías es todo lo contrario, esta no depende de una base de datos, sino de perfiles; según Mardini (2016) la información que estos contienen se basa en el número de intentos fallidos de inicio de sesión, el uso del procesador, recuento de los correos electrónicos, entre otros. Toda la información que se encuentre en los perfiles se

considera como normal y es comparada con eventos que estén fuera de los perfiles, siendo así considerados como anomalías en la red y el IDS enviará una alerta de intrusión.

Finalmente, se tiene un tercer grupo que se basa en el *tipo de respuesta* y esto se presenta una vez que el IDS haya detectado alguna intrusión, existen dos formas de alertar en este grupo, una pasiva y otra activa. En la alerta pasiva, se envía una notificación ya sea mediante un sonido o registro de las acciones ocurridas, mientras que la alerta activa pone en ejecución algún tipo de acción en específico, como puede ser el cierre de una cuenta o la reconfiguración del firewall (Carvajal, 2015).

2.2.2. Parámetros y Métricas de desempeño de un IDS.

En la actualidad, ningún IDS es 100% efectivo tal y como lo afirma Mardini (2016), razón por la cual se establecen tres métricas que contribuyen a mejorar su desempeño: Sensibilidad, Especificidad y Exactitud; estas a su vez se encuentran en función de cuatro parámetros, que son: Falso positivo (FP), Verdadero positivo (VP), Verdadero Negativo (VN), Falso Negativo (FN); dependiendo de dos aspectos determinantes, Detección y Naturaleza del evento, tal como señala la Figura 9.

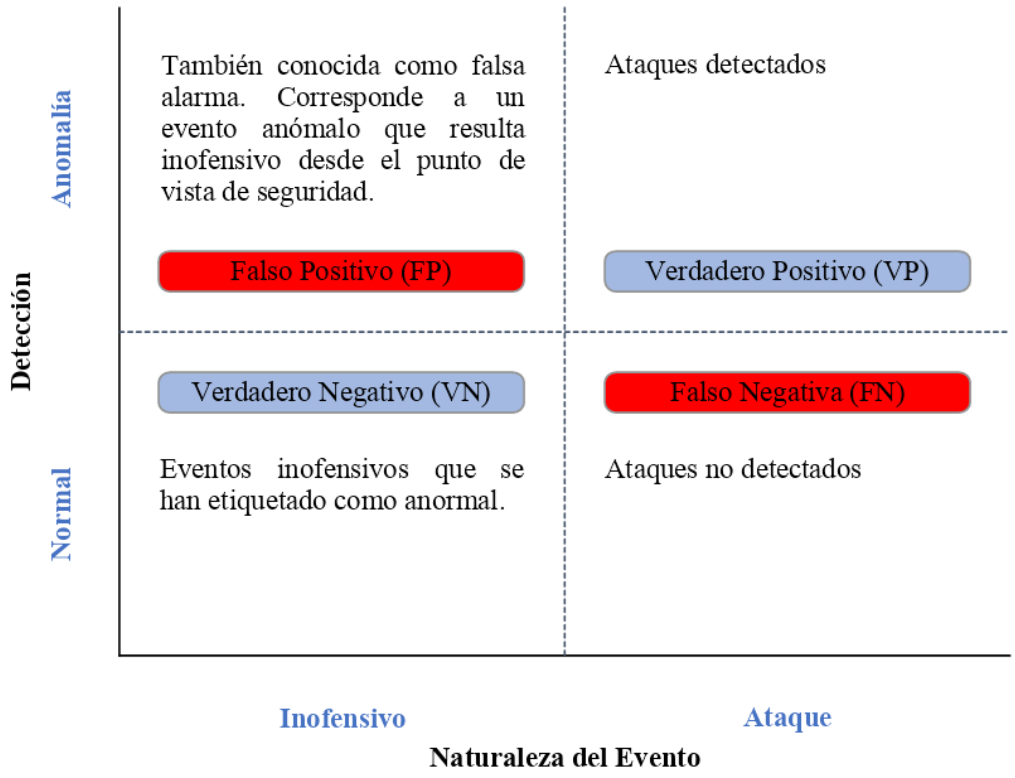


Figura 9. *Parámetros para métrica de desempeño*

Fuente: *Adaptado de (Mardini, 2016)*

Parámetros en un IDS

Los *VP* se caracterizan por indicar que una intrusión o ataque dentro del sistema es verdadero, es decir, que sí presenta una amenaza hacia la seguridad de la información, contrario a los *FP* que presentan alertas que se dan cuando un valor en el sistema sobrepasa el umbral, estos valores pueden ser: consumo de ancho de banda, número de intentos al inicio de sesión, número de peticiones a un servidor, entre otros; aunque no siempre serían causadas por intrusos buscando atacar al sistema sino por posibles descuidos de los propios usuarios, además, cabe recalcar que este parámetro es muy delicado en un IDS, puesto que, si existen demasiados FP el IDS pierde confiabilidad y credibilidad (Camacho, 2017; Mardini, 2016).

Por otro lado, también existe los *VN* y los *FN*, los primeros son alertas correctas que indican que el sistema no está siendo vulnerado, es decir, que la seguridad de la

información no tiene relación con alguna intrusión o ataque en la red de datos. En cuanto a los FN, se caracterizan por presentar un peligro dentro del IDS, ya que en este parámetro se encuentran las intrusiones o ataques que no han logrado ser detectados en el sistema, la razón principal para que se den estos casos, es que los datos intrusivos no sobrepasen los umbrales que se configuraron en los perfiles (Camacho, 2017).

Métricas de desempeño en un IDS

Habiendo comprendido los parámetros que intervienen dentro de un IDS, ahora se describen las métricas que trabajan en función a estos y como es que cada una de ellas varía dependiendo del resultado de los parámetros (Almseidin, Alzubi, Kovacs, & Alkasassbeh, 2017; Kumar, 2015).

La *Sensibilidad* de un IDS consta de un valor numérico, el cual indica los casos que pueden ser catalogados como Verdaderos Positivos (Mardini, 2016), de tal manera que esta métrica depende tanto de los valores de los VP como de los FN, como en la ecuación 1 se indica.

$$Sensibilidad = \frac{VP}{VP + FN} \qquad \textbf{\textit{Ec. 1}}$$

La *Especificidad*, se centra en aumentar los aciertos en función a los Verdaderos Negativos (Kumar, 2015) y se encuentra representado por una expresión matemática la cual se encuentra en la ecuación 2, misma que depende de los VN y los FP.

$$Especificidad = \frac{VN}{VN + FP} \qquad \textbf{\textit{Ec. 2}}$$

La *Exactitud*, que según Mardini (2016), es el grado de cercanía que tienen una cantidad de valores x al de la magnitud real, en un IDS una exactitud del 100% quiere decir que los valores medidos, ya sean: el consumo de ancho de banda, cantidad de inicios

de sesión, número de peticiones a un servidor, entre otros; son idénticos a los valores que se dan, esta métrica viene dada por la ecuación 3 que depende de ciertos parámetros, como: VP, VN, FP y FN.

$$Exactitud = \frac{VP * VN}{VP + FP * FN * VN} \qquad \textbf{\textit{Ec. 3}}$$

2.2.3. Ubicación de un IDS.

La ubicación de un IDS depende de la topología e infraestructura de la red de datos que tenga la organización ya que este sistema puede complementarse con el firewall (Camacho, 2017; Herrera, 2015), es por esta razón que el personal encargado de la seguridad de la información lo coloca delante o detrás del firewall:

Las ventajas que se obtienen al colocar el *IDS delante del Firewall*, es que se logra observar todo el tráfico que se dirige a la red interna desde redes externas, como por ejemplo la Internet; garantizado la detección de ataques que se encuentren dirigidos al firewall principal, sin embargo, al tener demasiados datos que procesar el IDS, este genera demasiados falsos positivos, lo cual no es conveniente en este tipo de seguridad ya que lo vuelve desconfiable (Herrera, 2015; Ocampo, Viviana, Bermúdez, & Solarte Martínez, 2017). La Figura 10 presenta la posición del IDS delante del Firewall.

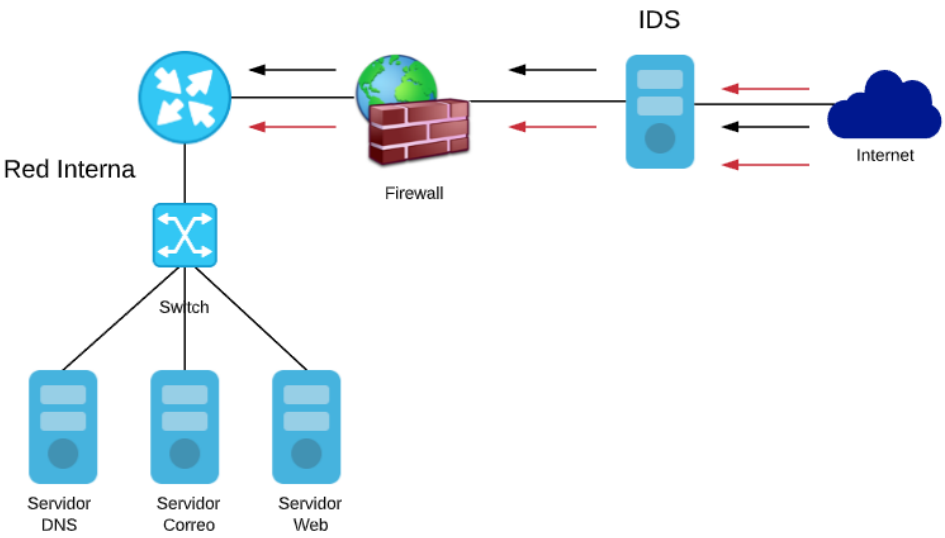


Figura 10. *IDS ubicado delante del firewall.*

Fuente: *(Herrera, 2015)*

En tanto que si se coloca el *IDS después del Firewall* este puede tener un mejor control del tráfico de información, ya que el firewall filtra todo el tráfico no deseado que proviene de redes externas evitando que el Sistema de Detección de Intrusos genere gran cantidad de falsos positivos, dando más confiabilidad al sistema (Cárdenas, 2016; Herrera, 2015). No obstante, al realizar esta configuración existe la posibilidad de que se realicen ataques directamente desde la red interna sin que el IDS los detecte. En otras palabras, el IDS presenta desventajas sin importar la ubicación, es por esta razón que el personal encargado de la seguridad de la información es quien toma la decisión de donde se lo debe ubicar. La Figura 11 exhibe la configuración referida en el párrafo actual.

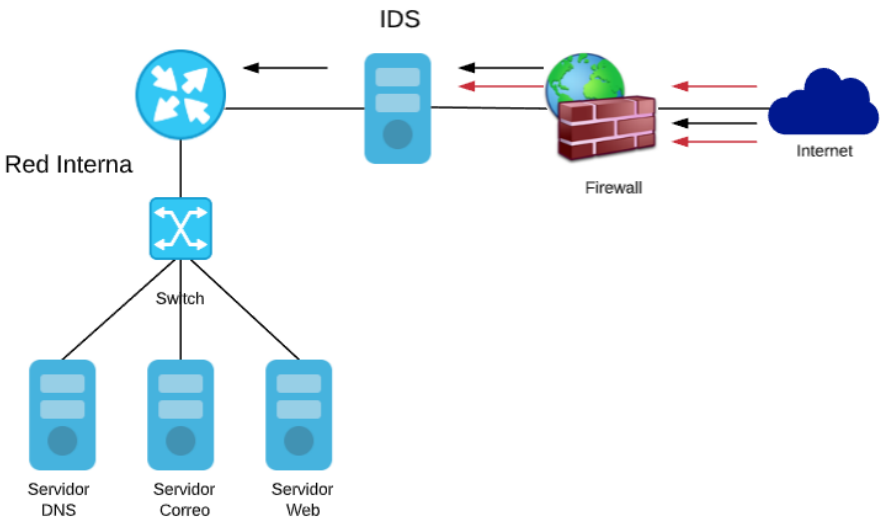


Figura 11. *IDS ubicado detrás del firewall*

Fuente: *(Herrera, 2015)*

2.2.4. Funciones del IDS.

El artículo de Kairi (2017) nombra cuatro funciones que un Sistema de Detección de Intrusos realiza, estas funciones son: Recolección de Datos, Selección de Características, Análisis y Acción; mismas que se describen en acápites posteriores.

La *Recopilación de datos* es la función encargada de que los datos reunidos en la red se transformen en entrada para el IDS, todos estos datos se registran en un solo archivo para posteriormente estudiarlos. Los NIDS al ser enfocados a red, realizan una recopilación de los paquetes de datos que circulan por ella, mientras que los HIDS recopilan detalles más enfocados al host anfitrión del IDS, tales como; el uso del disco y los procesos del sistema (Kairi, 2017; Tayyebi & Bhilare, 2018).

La *Selección de características*, radica en la selección de características particulares de los archivos obtenidos en la función anterior, por ejemplo: direcciones IP, tipos de protocolos, longitud de cabecera, entre otros (Kairi, 2017).

La función de *Análisis* tiene como objetivo determinar la gravedad del ataque, como se mencionó en la sección 2.2.1; existen dos tipos y dependiendo del tipo de

estrategia de análisis que se usa el IDS concluye que acciones ejecutar (Carvajal, 2015; Kairi, 2017).

Finalmente, la función de *Acción* depende directamente del tipo de ataque y de cómo se esté administrando el IDS, ya que en los tipos de repuesta, existen una pasiva y otra activa; la respuesta pasiva envía mensajes de alerta mediante logs, correos electrónicos al administrador, entre otros; por el contrario a la respuesta activa, produce cambios en la red, es decir, rechaza los paquetes o cierra puertos de los servicios afectados; es así que, de acuerdo al tipo de acción configurado en la IDS, éste responde al ataque (Kairi, 2017; Mardini, 2016).

2.2.5. Software IDS.

En el mercado existe gran variedad de software para IDS tanto comerciales como de código abierto (libres); en el acápite posterior se puntualizan cuatro: Snort, Security Onion, Open Source Tripwire y Zeek, de estos se seleccionará el más apto para ser usado en el presente trabajo de titulación.

Snort es un IDS de código abierto, el cual permite analizar el tráfico de la red en tiempo real, brindando posteriormente un registro de las alertas que se presenten, estas alertas pueden ser causadas por coincidencias que se presenten en las reglas definidas, desbordamiento de búfer, ataques de Interfaz de Entrada Común (CGI) y por muchos otros (Snort, 2020). Por otro lado, *Security onion* es una distribución de Linux usado en la exploración de amenazas, además es usado a nivel empresarial ya que está constituida por varias herramientas encargadas de la seguridad de la información; por ejemplo: Elsticsearch, Kibana, Logstacsh, Suricata, entre otras que cumplen propósitos similares (Security Onion, 2020). *Open source Tripwire* es un software destinado a brindar seguridad e integridad a los datos, esto lo logra mediante monitoreo de la red que posteriormente alerta en caso de presentarse cambios en los archivos y/o directorios (Cox,

2018). Finalmente, Zeek es un software open source cuya función es examinar la información que circula por la red y así detectar anomalías que puedan suscitarse en el comportamiento de la misma, esta herramienta es usada en ciberseguridad (Bricata, 2020).

La Tabla 5 ayuda a identificar si los IDS ya citados cumplen con las características que menciona Sáenz & Martínez (2018) para elegir la herramienta que mejor se adapte a esta tesina, entre estas se puede nombrar a: Desarrollo bajo Open Source, análisis de tráfico en tiempo real, fácil administración, establecimiento de reglas, entre otras cualidades que favorezcan su uso en el presente trabajo.

Tabla 5

Características de Software IDS

Criterios	Snort	Security Onion	Open Source Tripwire	Zeek
Desarrollo bajo Open Source	SI	SI	SI	SI
Análisis de tráfico en tiempo real	SI	NO	SI	SI
Fácil de administrar	SI	NO	NO	NO
Compatibilidad únicamente con Linux	NO	NO	SI	NO
Licencia Gratuita	SI	SI	NO	SI
Alta disponibilidad	SI	NO	NO	NO
Establecimiento de reglas	SI	NO	NO	NO

Fuente: *Adaptado de (Sáenz & Martínez, 2018)*

2.3. Ataques Informáticos

Junto a los temas tratados en secciones anteriores, también es necesario considerar en las redes de comunicación los ataques informáticos; siendo esencial empezar por su

definición para una mejor comprensión, de forma que se establece un ataque informático como:

> Aprovechar alguna debilidad o falla (vulnerabilidad) en el software, en el hardware, e incluso, en las personas que forman parte de un ambiente informático; a fin de obtener un beneficio, por lo general de índole económico, causando un efecto negativo en la seguridad del sistema, que luego repercute directamente en los activos de la organización (Vallejo, 2018, p. 13).

2.3.1. Anatomía de un Ataque Informático.

En el transcurso del tiempo se han venido dando diferentes tipos de ataques informáticos, es por esta razón que las personas especializadas en seguridad de la información han establecido fases para un ataque informático, puesto que las personas que hurtan la información siguen este patrón (fases), nombrando a este conjunto de fases como "círculo hacker". El círculo hacker está formado por: Reconocimiento, Exploración, Obtener Acceso, Mantener Acceso y Borrar Huellas, en ese orden; sin embargo, existe un paso adicional, el cual se lo realiza en ocasiones especiales únicamente por auditores informáticos que se encuentren llevando a cabo un proceso de Ethical Hacking, este paso consiste en la presentación de un reporte acerca de las acciones efectuadas y es conocido como Informe de Penetración (Rodriguez & Santibáñez, 2018; Vallejo, 2018).

A continuación, se trata con mayor especificidad cada una de las fases que se encuentran dentro del circulo hacker:

Fase 1: El *Reconocimiento*, esta fase también es conocida como *footprinting* y consiste en la recolección de la mayor cantidad de información de la víctima; esta puede ser desde una persona en específico hasta una organización, es así que para esta fase, las

herramientas más usadas son la Internet (navegadores, redes sociales, entre otros) y la Ingeniería Social (Domínguez L., Maya O., Peluffo O., & Crisanto Ñ., 2017; Romero et al., 2018).

Fase 2: *Exploración*, esta usa la información que se obtuvo en la etapa de Reconocimiento del objetivo o víctima, donde se puede obtener información sobre: direcciones IP, credenciales (usuarios y contraseñas), entre otros. Una de las herramientas que se suele usar durante la exploración es Nmap, ya que permite determinar los equipos disponibles en una red y la información de estos (servicios que ofrecen, sistema operativo, entre otros); sin embargo, la peculiaridad de esta herramienta es su lista de puertos, ya que en ella muestra la información de sus estados (abierto o cerrado), es decir, muestra los puertos de cada servicio y si estos se encuentran abiertos (sin protección) para posibles ataques (NMAP.ORG, 2020; Rodriguez & Santibáñez, 2018).

Fase 3: La tercera fase se denomina *Obtener Acceso*; aquí ya es posible poner en marcha el ataque mediante la explotación de las vulnerabilidades que fueron identificadas en fases anteriores. Esta fase se separa en dos grupos, ataques pasivos y ataques activos (Chávez, 2016), mismos que se aclaran en la sección 2.3.3.

Fase 4: *Mantener Acceso*, es la cuarta fase y consiste en que la vulnerabilidad que el atacante haya encontrado se mantenga así, en otras palabras, el atacante puede acceder a la información desde cualquier sitio siempre que tenga una conexión a internet, es así como se crean pequeños programas que permiten esto, como: backdoors, rootkit, troyano, entre otros; con el requisito de que estos se encuentren instalados en el sistema de la víctima para permitir el acceso a la información (Domínguez L. et al., 2017).

Fase 5: La fase final del círculo hacker es la de *Borrar Huellas*, aquí el atacante intenta borrar todo rastro del acto que cometió, con el fin de que los encargados de la seguridad de la red no lo detecten, por ese motivo es que se centran en eliminar archivos

de registro (logs) o las alarmas que estos hayan generado en el IDS (Rodriguez & Santibáñez, 2018; Vallejo, 2018).

2.3.2. Elementos por proteger.

En una organización existen varios elementos que pueden resultar vulnerables ante posibles ataques, comúnmente el objetivo del ataque es obtener información sin el consentimiento de los administradores, por esta razón es que dichos elementos reciben la seguridad necesaria. Según Romero et al. (2018) & Vallejo (2018) los principales elementos son: Hardware, Software, Datos y Recursos Humanos, mismos que se encuentran descritos en la Tabla 6.

Tabla 6

Elementos de una Organización

Elemento	Descripción
Hardware	En este grupo se engloban los componentes físicos que se encuentran en la organización, entre los principales se encuentran: componentes de comunicación, cableado de la red, CPU, entre otros.
Software	Se encuentra conformado por la parte lógica, por ejemplo: Sistemas Operativos (siglas en inglés S.O.), aplicaciones de la organización, entre otros.
Datos	Este elemento puede ser considerado el más importante en una organización, puesto que aquí se encuentra la información lógica que la conforma, por ejemplo: la base de datos, documentos, archivos, entre otros tipos de información.
Recursos Humanos	Es considerado el eslabón más débil, ya que un error de ellos puede causar la perdida parcial o completa del trabajo en la organización.

Fuente: *Adaptado de (Romero et al., 2018; Vallejo, 2018).*

2.3.3. Tipos de Ataques.

Rodriguez & Santibáñez (2018) menciona dos tipos de ataques que pueden afectar en forma negativa a los principales elementos (Hardware, Software, Datos y Recursos Humanos) de una organización; estos ataques son conocidos como: ataques pasivos y activos.

2.3.3.1. Ataques pasivos.

Un ataque pasivo posee la cualidad de no modificar la información de la víctima en ningún momento, es decir, en este tipo de ataques el objetivo es únicamente escuchar y observar la información que está siendo transmitida a través de la red, para así poder obtener lo que sea más importante para el atacante, como contraseñas y nombres de usuarios (Torruella, 2017).

Un claro ejemplo de este tipo de ataque es el uso de un Sniffer, el cual es un software especializado en el análisis del tráfico de una red para posteriormente detectar y conocer que vulnerabilidades se tiene en ella, con el fin de recabar la información considerada de mayor importancia (Añas, 2018), en la Figura 12 se representa un ataque de hombre en el medio, mismo que suele usar un Sniffer.

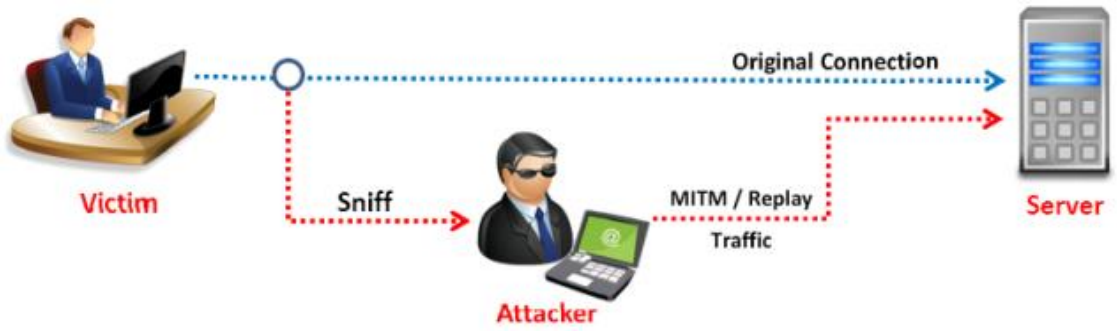


Figura 12. *Ataque de Hombre en el Medio*

Fuente: *(System Hacking, 2017). Recuperado dehttps://bit.ly/2KngpMV.*

2.3.3.2. Ataques activos.

Un ataque activo es identificado por interferir en la transmisión de información, crear un falso flujo de datos o incluso la interrupción de servicios, en otras palabras, esta forma de ataque es más abrupta, ya que aparte de que el atacante quiere hacerse con la información de la víctima, éste trata de penetrar la infraestructura de la red para tomar control de esta.

Los ataques activos “esencialmente se pueden subdividir en: suplantación de identidad, reactuación o repetición, modificación de mensajes y denegación de servicios”, tal como afirma Torruella (2017).

La *Suplantación de identidad* es un método también conocido como Spoofing, y consiste en que el atacante acceda a recursos con privilegios no autorizados dentro de la red, todo esto lo logra haciéndose pasar por un usuario que pertenece a esa red. Unos ejemplos de este tipo de ataque son: IP Spoofing, DNS Spoofing, SMTP Spoofing, entre otros; siempre y cuando cumplan funciones similares (Rodriguez & Santibáñez, 2018). En la Figura 13 se muestra el proceso del ataque de DNS Spoofing, el cual consiste en el falseamiento de la dirección IP cuando esta realiza una consulta de resolución de nombre, dicho en otra forma, resuelve con una dirección IP falsa un cierto nombre DNS o viceversa.

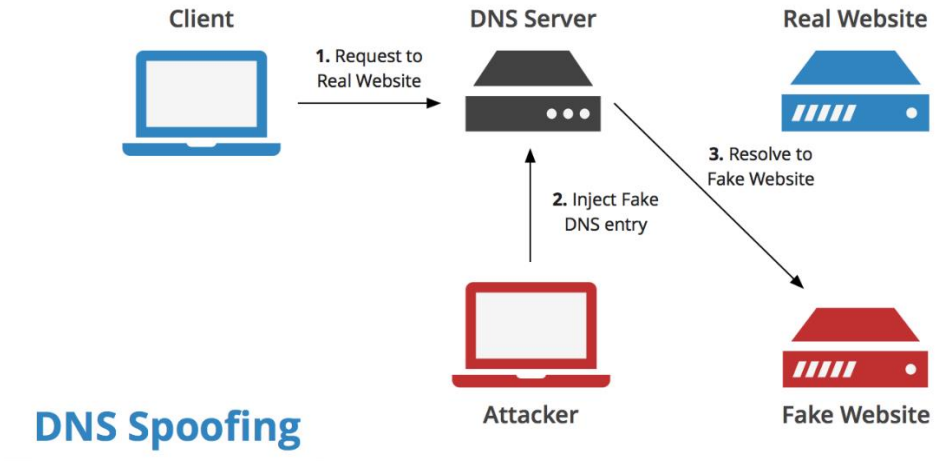


Figura 13. *Ataque DNS Spoofing*

Fuente: *(Keycdn, 2018). Recuperado de https://bit.ly/2qRrHSL.*

La *Repetición* consiste en capturar mensajes que sean legítimos y repetirlos con el fin de que el sistema los detecte como no deseados debido a su retransmisión continua.

El ataque de *Modificación de mensajes* consiste en tomar una porción del mensaje y alterarlo o tomar un mensaje completo y reordenarlo, teniendo en cuenta que estos mensajes sean legítimos, el objetivo de este ataque es provocar un efecto no autorizado. Tomando como ejemplo a Añas (2018) “la falsificación de cuentas para que un cliente deposite cantidades de dinero de manera repetida y en cuentas diferentes a la que en verdad debe realizarse la transacción”.

Por último, la *Denegación de servicios* tiene como objetivo el quitar gran parte del prestigio a una organización, ya que hace imposible el acceso a los servicios que estos provee, esto es causado cuando el atacante inunda el servicio de mensajes o peticiones falsas. Otro uso que le dan a este tipo de ataque es como distracción ya que mientras el personal este resolviendo este problema, el atacante puede estar realizando otro tipo de intrusiones en el sistema (CCN, 2018). Algunos ejemplos de denegación de servicios usados por los atacantes son: Smur and Fraggle, Buffer Overflow, Ping of death, Teardrop, SYN flood. Siendo este último el más usado ya que el atacante envía constantemente paquetes de sincronización (SYN) al servidor y este a su vez responde con paquetes de sincronización reconocida (SYN-ACK) dejando así conexiones abiertas, debido a que el servidor no recibe los paquetes de reconocimiento (ACK) por parte del atacante, siendo estos necesarios para completar la comunicación entre cliente – servidor, al existir estas conexiones abiertas impiden que nuevos clientes legítimos puedan acceder al servicio (Cloudflare, 2020; Romero et al., 2018), en la Figura 14 se manifiesta el proceso de este ataque.

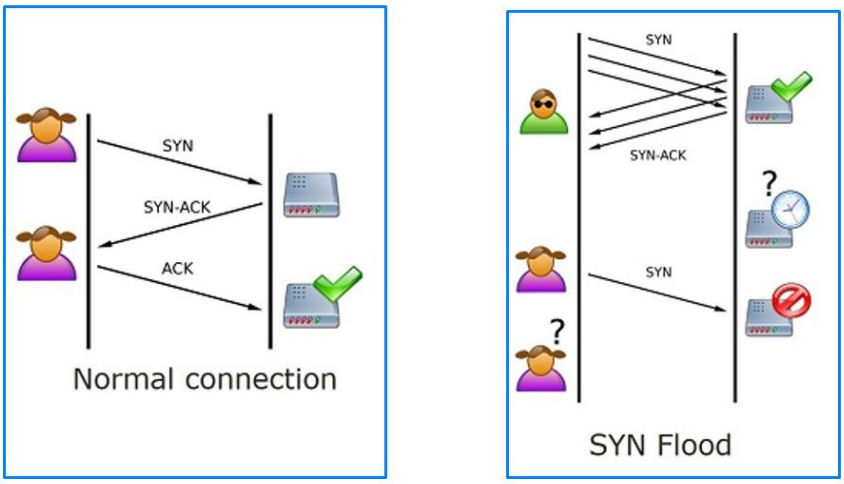


Figura 14. *Conexión Normal y Un Ataque de SYN flood*

Fuente: *(Morón, Ortega, Torres, 2019). Recuperado de https://bit.ly/2O0v81e.*

2.4. Estándar ISO/IEC 27001

En el ISO (2013) indica que la ISO/IEC 27001:2013 especifica cuáles son los requerimientos para la implementación, el mantenimiento y la mejora continua de un sistema de gestión de seguridad, además de incluir requisitos necesarios para evaluar y tratar los riesgos en la seguridad de la información.

El estándar no especifica el software ni hardware que deba usarse, dando a entender que existe la posibilidad de emplear el más apto para la red en la organización; dependiendo del tamaño y las funciones a las que se dedique, siempre y cuando se cumplan con los principios en los que se basa el estándar ISO (2013), como son los de mejorar la disponibilidad, integridad y confidencialidad de la información.

Si las recomendaciones presentadas en el estándar son aplicadas a cabalidad se pueden obtener algunos beneficios al implementar un Sistema de Gestión de Seguridad de la Información (SGSI) dentro de organizaciones, ya sean estas públicas o privadas (ISOTools, 2016), los principales beneficios se enlistan a continuación:

- Tener un mejor control en cuanto a la seguridad de los activos que posee la organización.
- Detectar los riesgos de la red en menor tiempo, facilitando así su control y eliminación.
- Flexibilidad en cuanto a las políticas de seguridad de la red.
- Aumentar la confianza por parte de los usuarios que usan la red.

2.4.2. Objetivos de control y controles.

Los objetivos de control y controles representan las medidas que se ofrecen en una primera fase de seguridad para la información, es así que está permitido la selección de los controles más apropiados al tipo de actividad en las diferentes organizaciones (ISO 27001, 2020).

Los objetivos que se anexan en el estándar mencionan las acciones que la organización está obligada a cumplir para que la seguridad de su información sea la idónea, además de una adecuada gestión para sus activos.

> El anexo A contiene una lista bastante completa de objetivos de control y controles comúnmente relevantes para las organizaciones. Se dirige a los usuarios de este Estándar Internacional como un punto de inicio para la selección de controles a fin de asegurar que no se pase por alto ninguna opción de control importante (ISO, 2013).

2.4.3. Políticas de Seguridad.

Las Políticas de seguridad se encargan de mantener un nivel de protección adecuado para la información que es enviada dentro y fuera de una organización, evitando así, que esta sea interceptada por personas mal intencionadas, en estas políticas se

encuentran procesos que sirven a largo plazo para posibles futuros administradores que serán responsables de esta área de la red (Pública, Manuel, & Viñas, n.d.).

El establecimiento de buenas políticas de seguridad facilita el trabajo de las personas que estén a cargo de la seguridad de la información en la organización, puesto que existen casos en el que los administrativos de esta área no tienen conocimientos previos y con apoyo de las políticas que ofrece el estándar, brindan los procesos que se deben seguir para eliminar estos riesgos en caso de presentarse alguna intrusión o intento de ataque a la red (PMG SSI, 2016).

2.5. Seguridad de la Información

En el ISO (2013) se define a la seguridad de la información como la "preservación de la confidencialidad, integridad y disponibilidad de la información; además, también pueden estar involucradas otras propiedades como la autenticidad, responsabilidad, no-repudio y confiabilidad".

Teniendo en cuenta esta definición se puede observar que la información se ha convertido en un activo con mucho valor dentro de una organización, ya que puede ayudar a que una organización crezca o se destruya, además de que sus trabajadores o usuarios puedan comunicarse con confianza dentro de ella.

Este tema es conformado por cuatro pilares que son fundamentales en los que se basa la seguridad de la información según afirman OBS Business School (2020) & Tecon (2020), dichos pilares se encuentran definidos en la Tabla 7.

Tabla 7

Pilares de la Seguridad de la Información

Pilares	Definición
Disponibilidad	El acceso a la información esta cuando se requiera.

Confidencialidad	La información es accesible solo para personal autorizado.
Integridad	La información correcta sin modificaciones no autorizadas ni errores.
Autenticación	La información procedente de un usuario que es quien dice ser.

Fuente: *(Tecon, 2020)*

2.5.1. Modelo PDCA.

El estándar ISO/IEC 27001 emplea el modelo PDCA (Plan, Do, Check, Act) cada una de sus siglas significa en español: Planificar, Hacer, Verificar y Actuar. Este modelo ofrece una mejora continua (ciclo constante) en la gestión de la seguridad para los activos que presente la organización tal como se exhibe en la Figura 15. En este sentido ESAN (2016) afirma que una organización debe tener claro cuáles son los elementos que se toman en cuenta al momento de establecer las políticas de seguridad para la información. En el siguiente acápite, se profundizan las acciones que llevan a cabo cada una de las etapas ya mencionadas en esta sección.

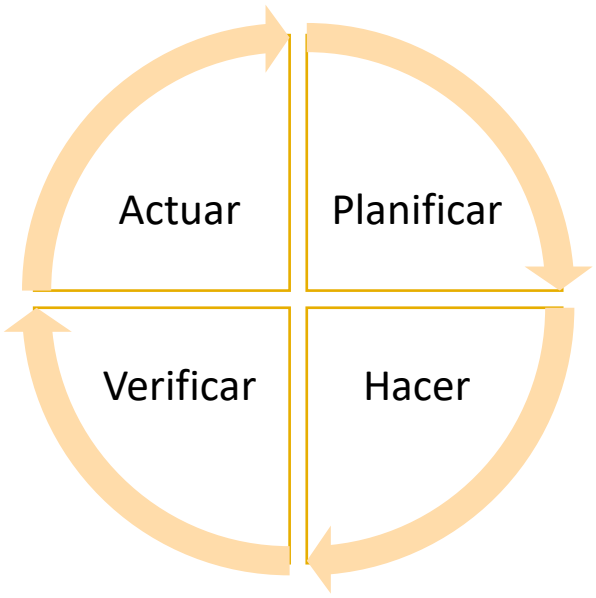


Figura 15. *Modelo PDCA*

Fuente: *Autoría propia*

La etapa de *Planificar* se encarga de definir políticas de seguridad relevantes en la organización, además de realizar un análisis de posibles riesgos posteriores que esta pudiese llegar a tener, estableciendo así las respectivas autoridades y responsables de estas políticas. *Hacer* es la segunda etapa y consiste en ejecutar las políticas que se hayan definido en la fase anterior, esto se logra con la asignación de acciones para cada política de seguridad. La tercera etapa es conocida como *Verificar* y mantiene una revisión continua del desempeño que posee el sistema de seguridad, obteniendo así reportes de la situación en la que se encuentre. La última etapa es *Actuar* y permite efectuar acciones correctivas o preventivas, todo dependiendo de los reportes que se presenten, estas acciones se encargarán de riesgos futuros (ISO, 2013).

2.6. Neuronas Biológicas y Artificiales

Para entender la arquitectura de una neurona artificial es necesario empezar por definir como se constituye una neurona biológica. La Figura 16 indica las partes que conforman una neurona biológica, en donde las Dendritas representan las entradas de información de las neuronas, misma que es procesada por el Núcleo para posteriormente transmitirla a través del Axón, el cual posee ramificaciones que conectan a entradas (dendritas) de otras neuronas; el proceso producido por la conexión entre neuronas es conocido como sinapsis o neurotransmisores. Se dice que hay alrededor de 10^{11} neuronas en el cerebro humano y cada neurona puede recibir información de 5,000 a 15,000 entradas de axones de otras neuronas (Cuevas-Tello, 2018).

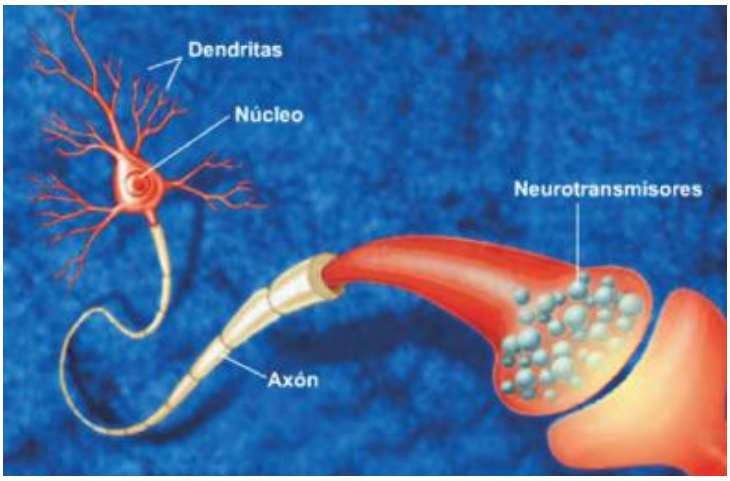


Figura 16. *Neurona biológica*

Fuente: *(Caicedo B & López S, 2017)*

Por otra parte, las neuronas artificiales están compuestas según Novas Otero (2018) & Palacios (2019) por los siguientes elementos: entradas, pesos sinápticos, función de agregación y función de activación; estos elementos son puntualizados más adelante.

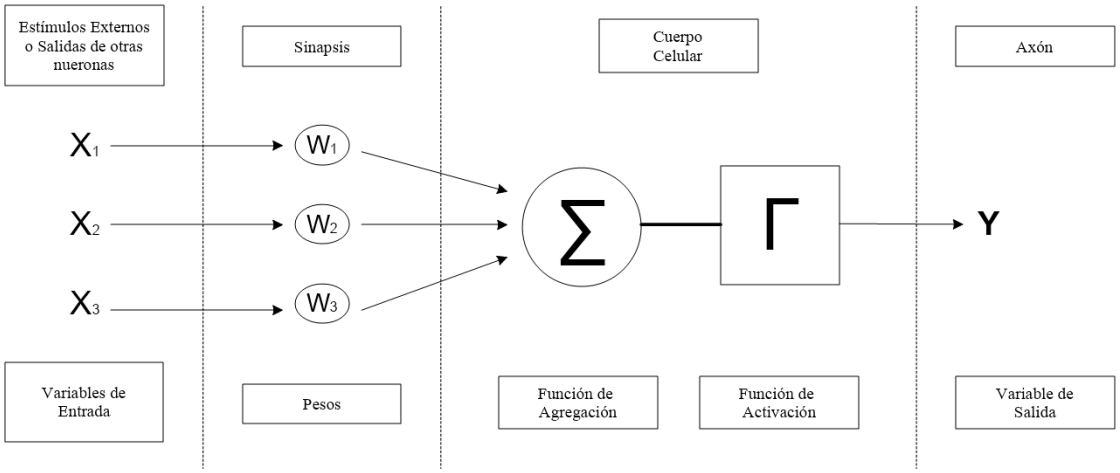


Figura 17. *Modelo de una Neurona Artificial*

Fuente: (*Fernández, 2009). Recuperado de http://www.scielo.org.co/pdf/abc/v14n3/v14n3a6.pdf.*

La Figura 17 describe la arquitectura de una Neurona Artificial, en donde, (X_1, X_2, …, X_n) denotan las variables (estímulos) de entradas o de salidas en otras neuronas; mientras que, (W_1, W_2, …, W_n) indican los pesos sinápticos. El resultado de la multiplicación entre entradas y pesos sinápticos es acumulado en una sumatoria ($\sum$), a

este proceso se lo denomina función de agregación; finalmente los valores resultantes de dicha sumatoria son valorados por una función de activación (Γ) para así obtener una señal de salida Y (Calvo, 2017).

En este sentido, la función de las *Entradas*, es receptar los datos que envían otras neuronas, en comparación a una neurona biológica esta función vendría a ser las dendritas (Palacios, 2019). Por otra parte, los *Pesos Sinápticos* toman valores numéricos que varían durante el entrenamiento de la red neuronal, la variación siempre depende del propósito que tenga la red; en una neurona artificial se referencia un factor de importancia que es el peso asignado a los datos que vienen de otras neuronas (Acevedo, Serna, & Serna, 2017).

La *Función de Agregación* se obtiene luego de haber determinado las entradas y los pesos sinápticos; basado en estos valores se obtiene el valor del potencial postsináptico. La operación empleada en estos casos, es la sumatoria de todas las entradas, sin olvidar el peso sináptico que está relacionado (multiplicado) a cada una de ellas, ya que es la importancia que cada entrada tiene (Grediaga, Ibarra, Ledesma, & Brotons, 2016). A esta operación se la conoce como suma ponderada y está representada por la Ec.4.

$$Net_j = \sum_{i=0}^{N} X_n W_n \qquad \textbf{\textit{Ec. 4}}$$

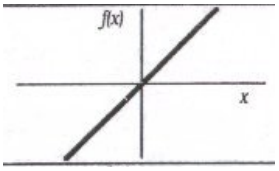
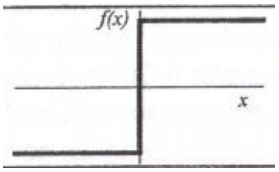
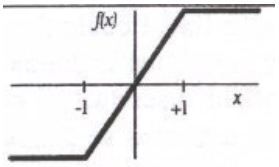
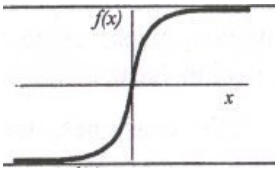
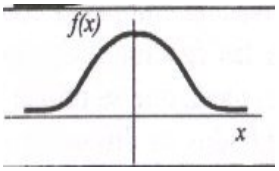
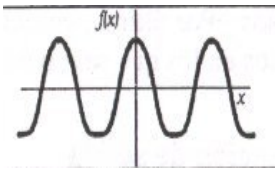
Donde X_n representa las entradas, W_n son los pesos sinápticos y N representa la cantidad de veces que se realiza la operación.

Una vez que se tenga el resultado de la función de agregación, esta se procesa por una función conocida como *Función de Activación* y su resultado viene a ser la salida de la neurona. Existen varias funciones de activación y cada una es usada dependiendo del

propósito que tenga la red neuronal; Novas Otero (2018) nombra las más usuales, las cuales se muestran en la Tabla 8.

Tabla 8

Funciones de activación más usuales

Nombre	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Línea a tramos	$y = \begin{cases} -1, si\ x < -l \\ x, si\ +l \leq x \leq -l \\ +1, si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoide	$y = \frac{1}{1 + e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = Asen(\omega x + \varphi)$	$[-1, +1]$	

Fuente: *(Palacios, 2019)*

2.7. Redes Neuronales Artificiales (RNA)

Las RNA se crean con la necesidad que tiene el ser humano de imitar “lo que por muchos es conocido como la máquina perfecta: el cerebro humano” (Caicedo B & López

S, 2017). Estas RNA surgen con el fin de modelar tanto como; la inteligencia humana, su estructura y los procesos de razonamiento que pasan en el cerebro. En el artículo de Jirón (2015) se define a la RNA como “un procesador masivamente distribuido y paralelo, es decir, es un computador capaz de desarrollar muchas tareas en forma simultánea”. Las neuronas artificiales que conforman las RNA son basadas en las neuronas biológicas.

2.7.1. Arquitecturas de una Red Neuronal Artificial (RNA).

En Jirón (2015) se define a la arquitectura de una RNA como la organización e interacción que tiene cada una de las neuronas que conforman la red, además menciona que hay tres clases diferentes: Redes neuronales monocapa, Redes neuronales multicapa y Redes neuronales recurrentes.

La primera clase está compuesta por las *Redes neuronales monocapa* y se caracterizan por no poseer capas ocultas, es decir, solo cuentan con una capa de entrada y otra de salida, tal como se observa en la Figura 18. Este tipo de redes neuronales son comúnmente usadas en reconstrucción de señales o memorias asociativas (Novas Otero, 2018).

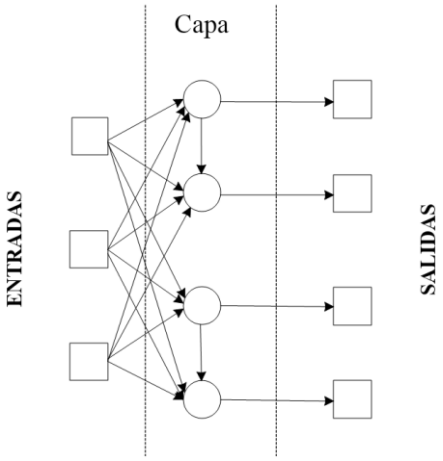


Figura 18. *Red neuronal monocapa*

Fuente: *(J. Pérez, 2017)*

Las *Redes neuronales multicapas* son la segunda clase y se diferencian con la anterior ya que además de componerse por una capa de entrada y otra de salida, estas poseen capas intermedias que llevan el nombre de capas ocultas, son llamadas así, puesto que no interviene con los datos que se presentan al final (Stephen, 2015). Un modelo de esta clase está manifestado en la Figura 19.

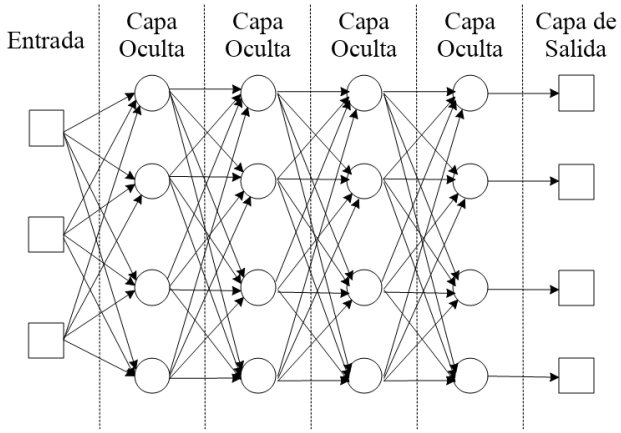


Figura 19. *Red neuronal multicapa*

Fuente: *(J. Pérez, 2017)*

Por último están las *Redes neuronales recurrentes (Feedback)*; algo que tienen en común con las anteriores arquitecturas, es que ambas van en una sola dirección, dicho de otra manera, la información va desde la capa de entrada hacia la capa de salida, sin embargo, la red neuronal recurrente se conecta a capas de niveles anteriores con el fin de generar un proceso de retroalimentación, lo que hace posible que la información de estas redes fluya tanto hacia adelante como hacia atrás (Jirón, 2015). Un modelo de esta red se observa en la Figura 20.

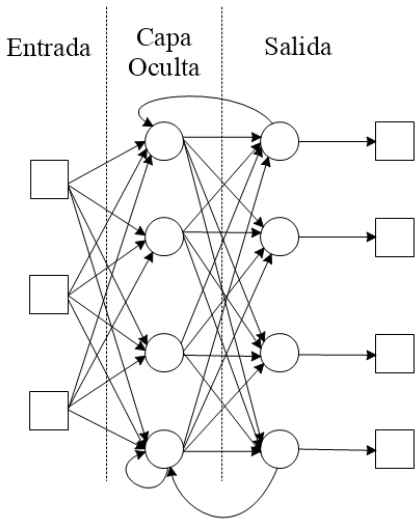


Figura 20. *Red neuronal recurrente*

Fuente: *(J. Pérez, 2017)*

2.7.2. Aprendizaje.

El termino aprendizaje es definido por Caicedo B & López S (2017) como el proceso que permite apropiarse del conocimiento, todo esto mediante previo estudio o experiencias vividas. En las redes neuronales artificiales dicho proceso es similar, ya que estas se van a comportar de forma diferente dependiendo del entorno en que se encuentran y de los pesos sinápticos que tengan programados, estos últimos a su vez van a depender del algoritmo o regla de aprendizaje que se utilice. Aquí se detallan dos tipos de aprendizaje: el supervisado y el no supervisado

El *Aprendizaje Supervisado* se caracteriza por tener un supervisor o maestro, quien es un agente externo encargado de guiar el aprendizaje de la red. Para que esto sea posible el supervisor tiene la obligación de conocer cuáles son las entradas y las salidas correspondientes a cada una de ellas (Caicedo B & López S, 2017).

En tanto que el *Aprendizaje No supervisado* no posee un supervisor que ayude con la guía del aprendizaje, dicho de otra manera, los datos entran directamente a la red sin ninguna etiqueta que identifique su salida. Sin embargo, para el aprendizaje no

supervisado, se necesitan una cantidad menor de datos de entrada para que este brinde el resultado deseado (Stephen, 2015).

2.8. Trabajos Relacionados

Esta última sección menciona trabajos que se relacionan con lo que propone la presente tesina, con el propósito de ser comparados entre sí, de esta manera se puntualiza cuatro de ellos en los siguientes acápites.

El trabajo de Manso, Moura, & Serrão (2019) propone un IDS integrado a la SDN funcionando de esta manera; en el momento que el IDS detecta ataques DDoS, este envía una notificación al controlador de la SDN del ataque suscitado, logrando mitigar los impactos negativos que se presenten en el rendimiento de la red. Con esta propuesta los autores manifiestan que descargan algunas decisiones de reenvío de información al controlado de la SDN. La Figura 21 describe la arquitectura propuesta en este trabajo, ubicando al IDS entre el controlador de la SDN y las redes externas (Internet).

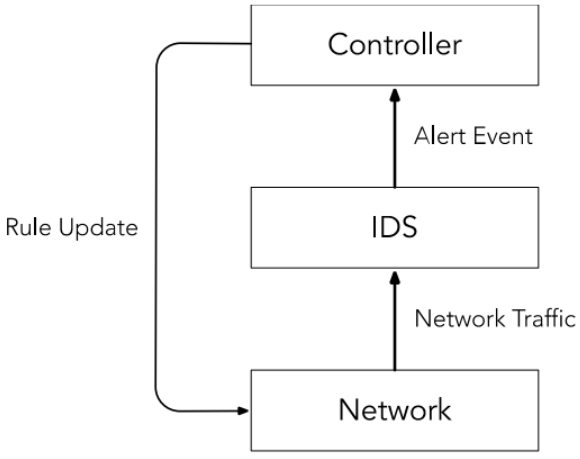


Figura 21. *Arquitectura del IDS aplicado a controlador de SDN.*

Fuente: *(Manso et al., 2019)*

El segundo trabajo pertenece a Reddy & Annapurani Panaiyappan (2018) y propone la integración de un IDS/IPS con el propósito de proteger al controlador de la SDN ya que debido al diseño centralizado que posee, convierte al controlador en el

elemento más importante de la red, exponiéndolo a ataques de personas mal intencionadas. Al igual que el anterior trabajo, este también fue probados con ataques de tipos DDoS. La Figura 22 presenta la arquitectura diseñada para el trabajo mencionado en este acápite.

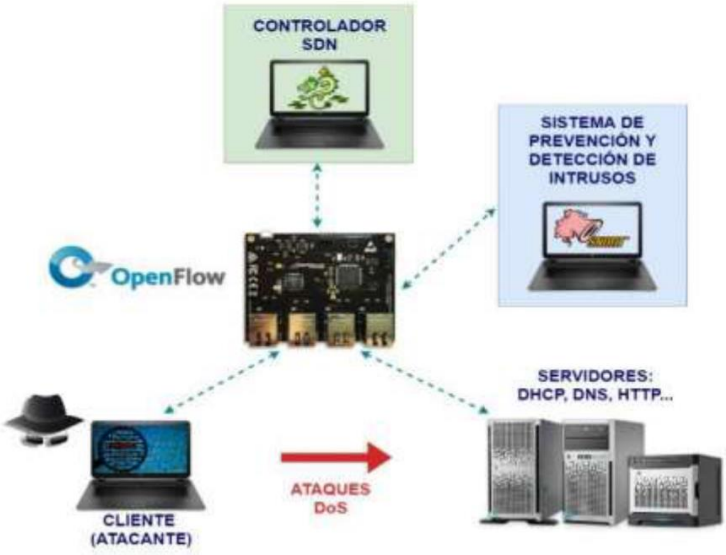


Figura 22. *Arquitectura de IDS/IPS aplicado a controlador SDN.*

Fuente: *(Reddy & Annapurani Panaiyappan, 2018)*

El trabajo que se menciona a continuación es de Varanasi & Razia (2019) y a diferencia de los anteriores, aquí proponen un análisis de varios trabajos que relacionan al IDS con el aprendizaje automático, para este aprendizaje automático se emplean las técnicas de Machine Learning y Deep Learning; logrando demostrar que el IDS puede detectar un ataque basado en los datasets que cada técnica use para su aprendizaje, evitando que usuarios externos deban configurar el sistema cada vez que se integre un nuevo elemento a la red. CIDDS-001 es uno de los datasets usados en este análisis y su esquema se lo puede apreciar en la Figura 23 junto con las categorías que lo conforman.

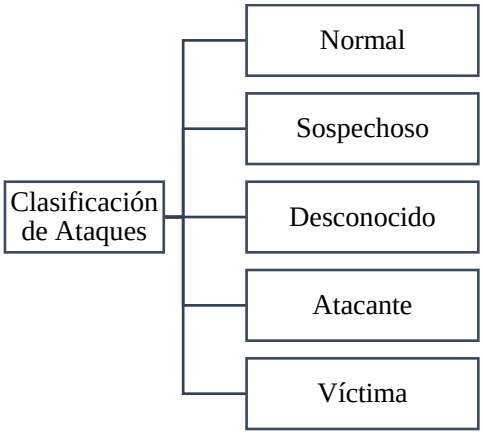


Figura 23. *Esquema de dataset CIDDS-001.*

Fuente: *Adaptado de (Varanasi & Razia, 2019)*

Finalmente, el trabajo de Sultana, Chilamkurti, Peng, & Alhadad (2019) presenta un análisis del diseño de una SDN con un enfoque a la seguridad de la misma, en esta área evalúan técnicas de Deep Learning para el desarrollo de una IDS, concluyendo que el diseño de este tipo de IDS no ha sido probado en infraestructuras con alta velocidad puesto que eso es el futuro que se encuentra tomando estos sistemas con aprendizaje automático. El modelo citado en este trabajo se encuentra en la Figura 24.

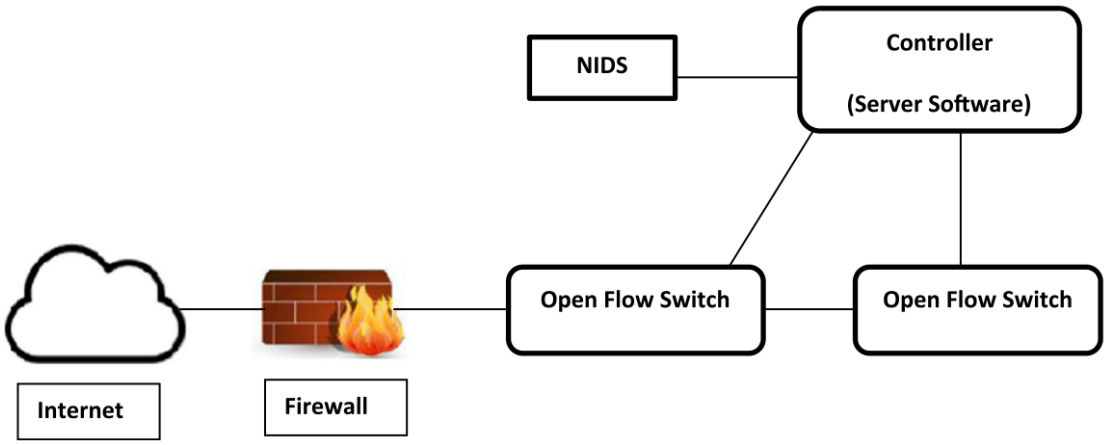


Figura 24. *Modelo de IDS con deep learning aplicado a SDN.*

Fuente: *(Sultana et al., 2019)*

Basado en los trabajos citados se determina que, dentro de los elementos de la SDN el más importante y vulnerable es el controlador, para ellos se presenta al IDS como una alternativa de protección ante ataques informáticos. Mencionando también que este

sistema puede ser optimizado con la ayuda de una red neuronal artificial ya que permite un aprendizaje automáticamente evitando que usuarios configuren constantemente las reglas de control bajo las cuales se administra al IDS. Por estos motivos es que esta tesina propone el diseño de un IDS con red neuronal artificial para la protección del controlador en una SDN, mismo que genera alertas en cuanto se detecte anomalías en dicha red.

CAPÍTULO 3.

SITUACIÓN ACTUAL Y DISEÑO DE SDN

Este apartado se enfoca en la situación actual que se encuentra el Data Center de la Facultad de Ingeniería en Ciencias Aplicadas, incluyendo los servicios y topologías que lo conforman, además se presenta el escenario de la Red Definida por Software (SDN) que será diseñado con el fin de permitir las pruebas para Sistema de Detección de Intrusos (IDS).

3.1. Análisis situación actual Data Center

La Facultad de Ingeniería en Ciencias Aplicadas (FICA) dispone de un data center totalmente equipado, mismo que tiene como objetivo brindar una comunicación adecuada en toda la facultad, tanto hacia internet como a los servicios con los que cuenta, este lugar se encuentra debidamente ambientado y con sus respectivos niveles de seguridad.

3.1.1. Ubicación del Data Center.

El Data Center se encuentra ubicado dentro de la oficina de la Carrera de Ingeniería en Telecomunicaciones (CITEL), está en la planta baja de la FICA; la Figura 25 presenta un diagrama de la infraestructura actual del data center, además indica sus dimensiones físicas, que en este caso son 2,85m de longitud por 3m de ancho, dando un área total de 8,55m2 (Narváez, 2016).

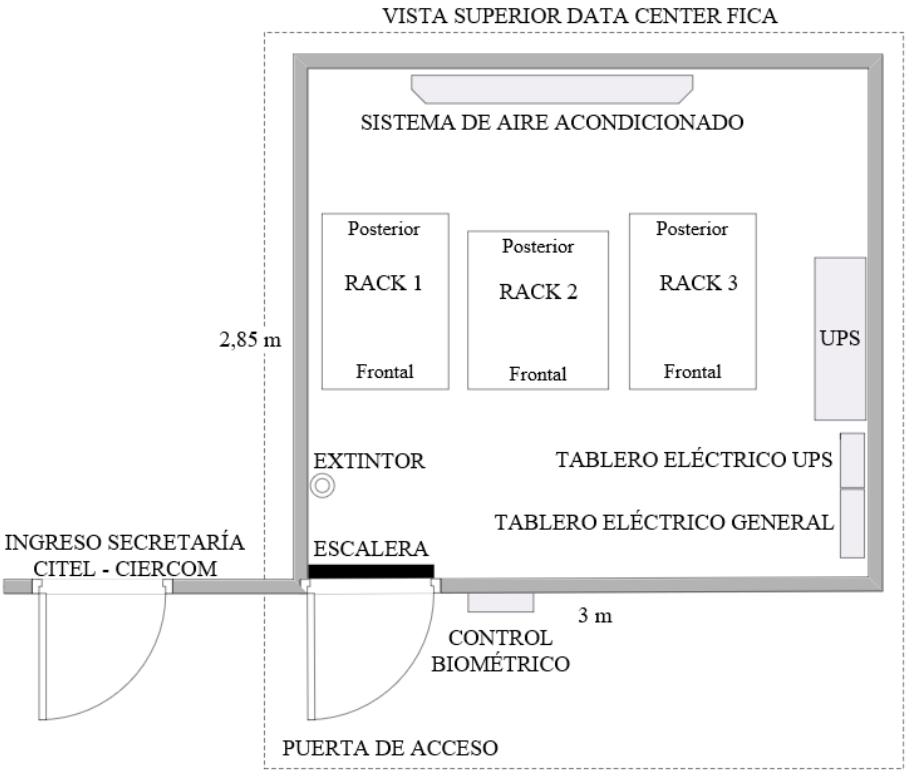


Figura 25. *Infraestructura del Data Center FICA*

Fuente: *Adaptado de (Guerrero, 2019)*

Los sistemas que compone el data center de la FICA se mencionan en los siguientes apartados, siendo estos: sistema eléctrico, sistema de control de acceso, sistema de aire acondicionado, sistema de seguridad y sistema de telecomunicaciones.

3.1.2. Sistema eléctrico.

El sistema eléctrico tiene como función brindar un suministro eléctrico al data center, el recorrido de la acometida eléctrica inicia desde el transformador que se encuentra frente al gimnasio de la universidad, hasta la caja de revisión ubicada en la parte externa de la FICA, realizando un recorrido subterráneo, para posteriormente conectarse al tablero eléctrico general, mismo que está ubicado en la parte interna del data center. Este tablero eléctrico se muestra en la Figura 26, siendo así la parte fundamental de la instalación eléctrica, ya que es aquí donde llega la acometida externa. Además, dicho tablero se encuentra conectado a un Sistema de Alimentación Ininterrumpida (UPS).

Figura 26. *Fotografía del tablero eléctrico*

Figura 27. *Fotografía del UPS*

El UPS que se muestra en la Figura 27, además de estar conectado al tablero eléctrico, también se conecta con el sistema de control de acceso y aire acondicionado, con el fin de que en el caso de haber algún fallo, este pueda suministrar electricidad por un tiempo determinado; otra función que cumple el UPS es proteger a los circuitos eléctricos de cambios bruscos de energía.

3.1.3. Sistema de control de acceso.

El acceso hacia el Data Center debe ser restringido, por este motivo es que se ha implementado un sistema de control de acceso, mismo que es manejado a través de un mecanismo de huella (biométrico) como se muestra en la Figura 28.

Este sistema también está compuesto por una puerta de seguridad metálica, la cual le permite resistir rayones y golpes, esta puerta tiene un diseño hermético con el fin de evitar fugas de aire refrigerado. A parte de lo ya mencionado, también tiene una barra anti-pánico y cerradura electromagnética.

Figura 28. *Fotografía del biométrico*

3.1.4. Sistema de aire acondicionado.

Tal como afirma Jácome (2019), el sistema de aire acondicionado de la Figura 29 es del tipo “split” o también conocido como “conford”, ya que debido a las dimensiones que posee el data center es el más adecuado para mantener las condiciones ambientales de los equipos de comunicación que se encuentran en su interior.

Figura 29. *Fotografía del sistema de aire acondicionado*

3.1.5. Sistema de telecomunicaciones.

El data center en la Facultad de Ingeniería en Ciencias Aplicadas tiene como objetivo alojar equipos TIC, además de funcionar como red de redundancia para el anillo de fibra que posee la Universidad Técnica del Norte, siendo estos equipos ubicados en el Rack 1. La Figura 30 muestra la distribución de los equipos TIC con los que cuenta el data center, dando un total de 3 Rack; a continuación se puntualizan los equipos con los que están conformados cada uno de estos.

El Rack 1 cuenta con un switch capa tres, mismo que es usado como equipo de borde para la conexión del data center con el edificio central y hacia Internet, a través de dos cables de fibra óptica. Mientras que el Rack 2 es en donde se ubican cuatro servidores denominados Proxmox Virtual (PV), sin embargo, únicamente tres (PV1, PV2 y PV3) brindan la infraestructura física para virtualizar equipos, el software usado para hacer posible este proceso se denomina Proxmox. Además en este mismo rack hay un servidor Radius, tres servidores que pertenecen a la Carrera de Ingeniería en Sistemas Computacionales (CISIC), un switch 3Com 4500G administrable y un router Mikrotik que permite la administración de la red inalámbrica. Finalmente, en el Rack3 se encuentran alojados un switch QPcom y 4 servidores, estos últimos son dedicados para

servicios administrativos y educativos dirigidos a docentes y estudiantes, estos servidores son: Opina, Reactivos, Biométricos y Revista Universitaria.

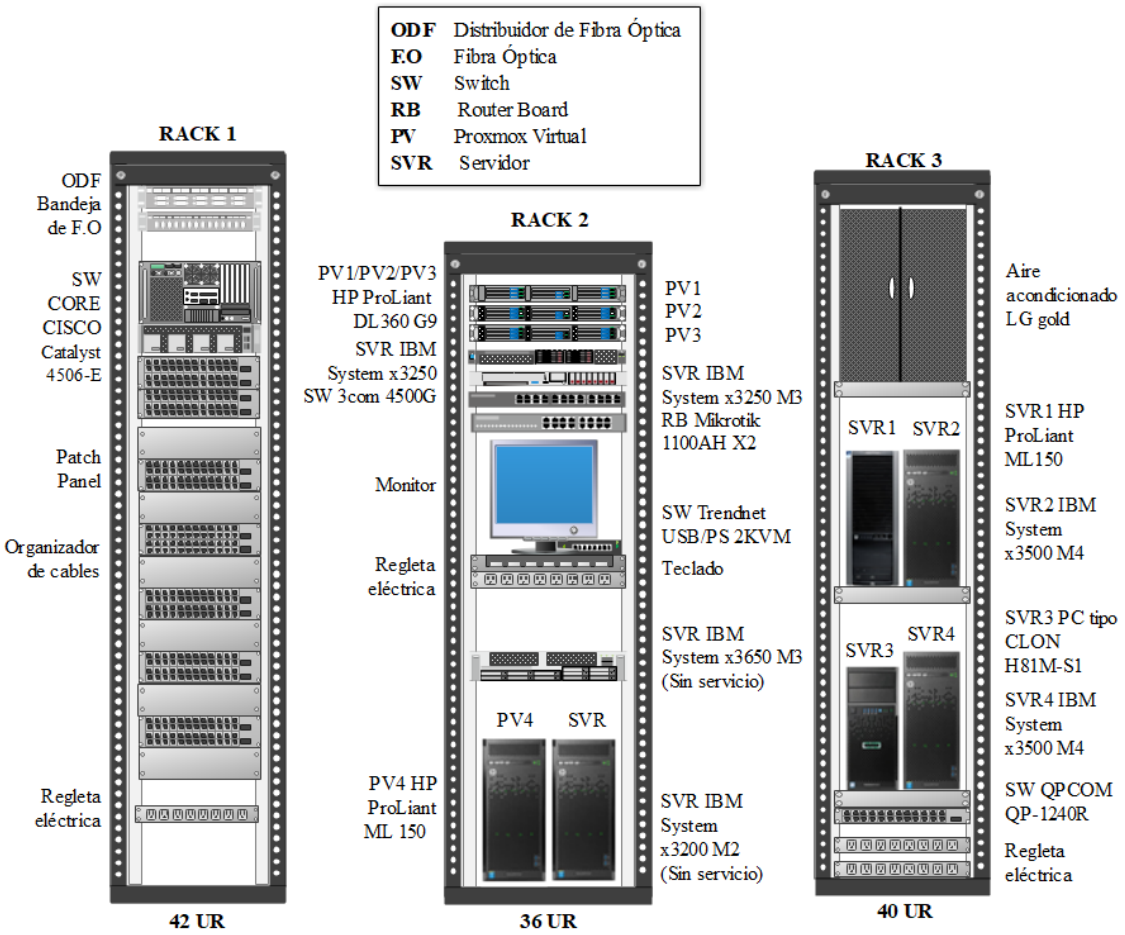


Figura 30. *Sistema de Telecomunicaciones Data Center – FICA*

Fuente: *Adaptado de (Jácome, 2019)*

3.2. Administración del data center

Para que el data center funcione de una manera eficiente debe existir una debida documentación de cómo se encuentra administrada la red, de manera similar a lo que presentan Guerrero (2019) & Jácome (2019) sus trabajos de titulación. Es así como se presentan topologías tanto físicas como lógica, mismas que se especifican en las secciones 3.2.1 y 3.2.2. Estas topologías permiten conocer la distribución de los equipos TIC y los segmentos de red que dispone, esto de manera gráfica.

Además es necesario mencionar que con las topologías se observan posibles puntos de vulnerabilidad que tiene la red en cuanto a su seguridad, permitiendo así establecer normativas de seguridad ya que actualmente no se cuenta con alguna, sin embargo, recientemente se realizó un trabajo de titulación, el cual propone un plan dirigido a la gestión de riesgos de la información, además establece ciertas políticas de seguridad para el data center en la FICA, este plan se basa en la Metodología MAGERIT v3.0 (Jácome, 2019), no obstante este plan aún no se lo ha implementado y sus políticas no han sido socializadas.

3.2.1. Topología física.

En la Figura 31 se exhibe la topología física del data center, la cual posee conexión a Internet y servicios, ambos a través de un enlace de fibra óptica que va desde la Dirección de Desarrollo Tecnológico e Informático (DDTI), para posteriormente terminar en el switch de core Catalyst 4506-E.

La red implementada internamente en la FICA se encuentra segmentada físicamente en tres partes, cada uno de estos segmentos se puntualizan a continuación:

El *primer segmento* va desde el puerto 28 en el switch de core Catalyst 4506-E mencionado anteriormente hasta el switch 3Com 3226. Esta conexión permite el acceso a ciertos servidores, tales como; Opina, Biométrico, Reactivos y Revista Universitaria

En tanto que el *segundo segmento* conecta el puerto 27 del switch de core Catalyst 4506-E con un switch LinkSys SR224G, la función de esta conexión es permitir la comunicación de los tres servidores que administra la Carrera de Ingeniería en Sistemas Computacionales (CISIC).

Por último, el *tercer segmento* brinda una conexión a la red inalámbrica que posee la FICA y a la infraestructura virtualizada Proxmox, éste empieza en el puerto 30 del

switch Catalyst 4506-E y conecta a un equipo Mikrotik RouterBoard 1100, mismo que se encarga del enrutamiento del tráfico, este último a su vez va conectado a un switch 3Com 4500G en el cual se encuentran enlazado: un switch QPCom QP-1240R y a la infraestructura hiperconvergente de Proxmox; el primero tiene la función de mantener conectados a los puntos de acceso (A.P.) que se ubican en toda la FICA, además este switch permite la comunicación con el servidor Radius, por otro lado la infraestructura de Proxmox se administra a través de los servidores PV1, PV2 y PV3.

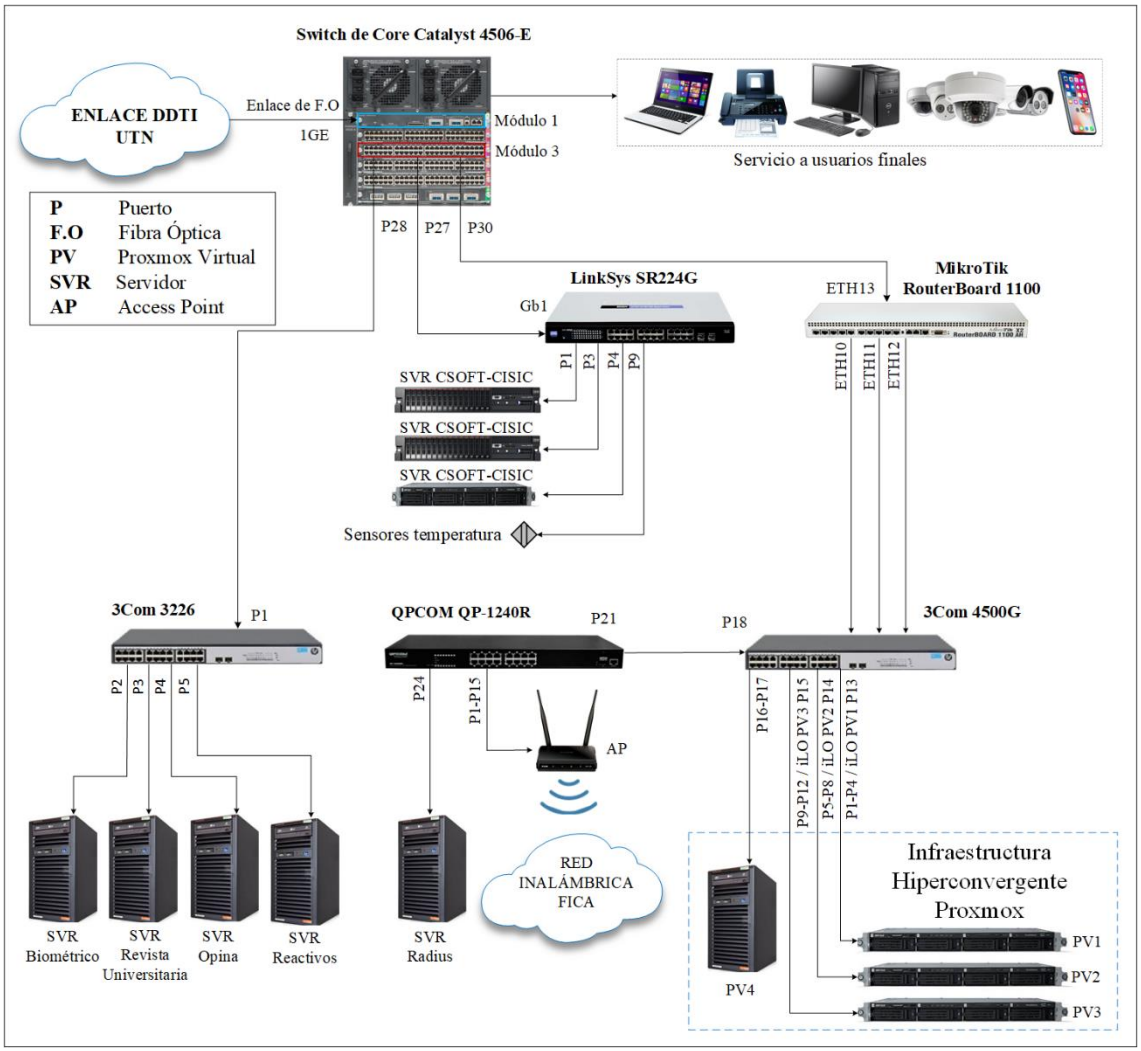


Figura 31. *Topología física del Data Center*

Fuente: *Adaptado de (Jácome, 2019)*

3.2.2. Topología lógica.

La topología lógica, como su nombre lo dice, se centra en la distribución lógica de la red, misma que se conforma por VLANs o segmentos de red con una finalidad especifica. La Figura 32 gráfica la topología en mención incluyendo las diferentes VLANs que se encuentran configuradas dentro de la misma.

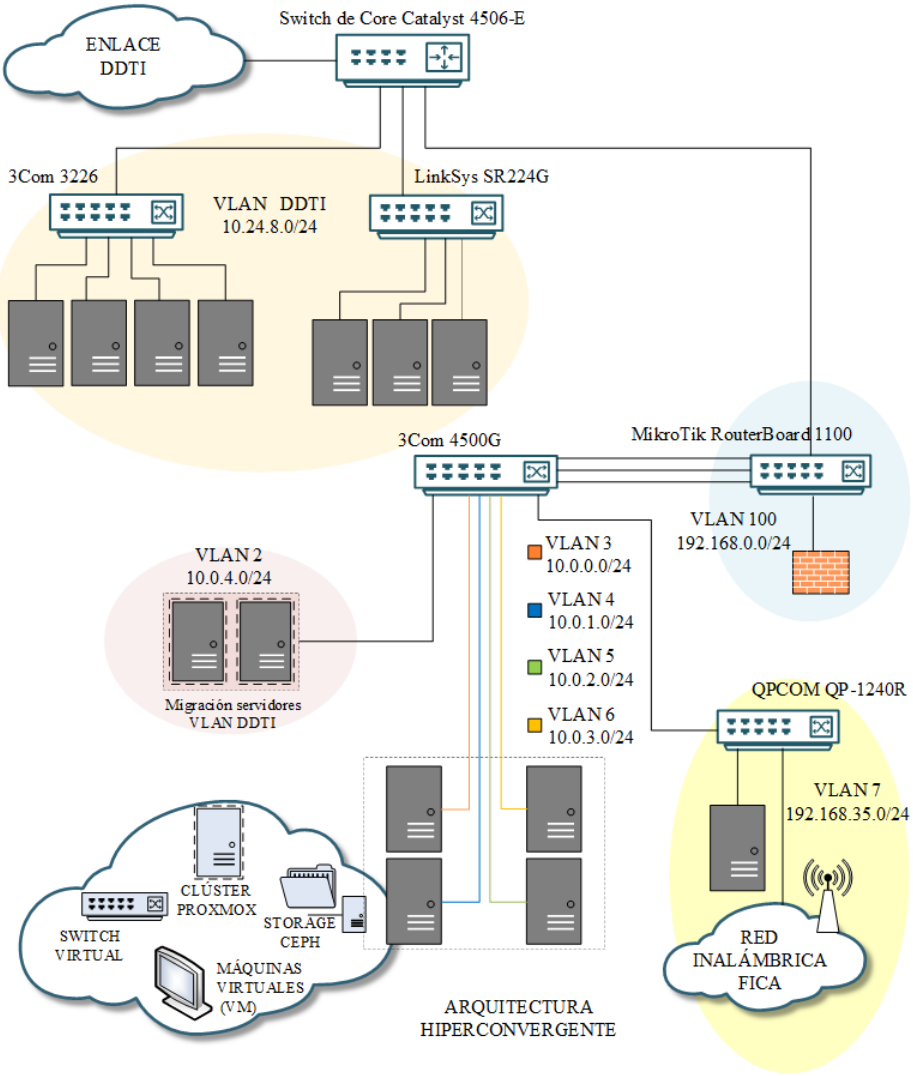


Figura 32. *Topología Lógica del Data Center*

Fuente: *Adaptado de (Guerrero, 2019)*

La Tabla 9 menciona las siete VLANs que se encuentran configuradas dentro del switch 3Com 4500G incluyendo una descripción del uso que tiene cada una de ellas, además del número y nombre con las que son identificadas en la topología lógica.

Tabla 9

VLAN's del Data Center

VLAN	Nombre	Descripción
VLAN 2	ENLACE_RED_ANTIGUA	Migración de servidores DDTI
VLAN 3	VLAN_SERV_PROXMOX	Servidores físicos de Proxmox
VLAN 4	VLAN_ILO_SERV_PROXMOX	Administración remota de servidores Proxmox
VLAN 5	VLAN_SERV_VIRTUAL	Servidores virtualizados en Proxmox
VLAN 6	STORAGE_CEHP	Conecta el almacenamiento compartido
VLAN 7	VLAN_WIFI_ADMIN	Administra la red inalámbrica en la FICA.
VLAN 100	TO_MIKROTIK	Indica red interna de Data Center.

Fuente: *Data Center FICA*

De la misma manera que hay una distribución de las VLAN's, también existe una distribución de direccionamiento IPv4, estas direcciones se ubican en la Tabla 10, especificando su VLAN y el equipo al que corresponde cada una. Además, es importante de claro que el presente direccionamiento IPv4 corresponde únicamente a la red interna que conforma el data Center de la FICA.

Tabla 10

Direccionamiento IPv4 del Data Center

	VLAN	EQUIPO	SUBRED	IP	GATEWAY	MASCARA
3Com 226	VLAN DDTI	OPINA	10.24.8.0	10.24.8.x	10.24.8.254	255.255.255.0
		Revista Universitaria		10.24.8.x		
		Reactivos Biométricos		10.24.8.x		
		CISIC 1		10.24.8.x		
		CISIC 2		10.24.8.x		
		CISIC 3		10.24.8.x		
		Sensores		10.24.8.x		
Linksys	2	Migración Servidores VLAN DDTI	10.0.4.0	10.0.4.x	10.0.4.254	255.255.255.0
3Com 4500G	3	PV1	10.0.0.0	10.0.0.1	10.0.0.254	255.255.255.0
		PV2		10.0.0.2		
		PV3		10.0.0.3		
		PV4		10.0.0.4		
	4	iLO PV1	10.0.1.0	10.0.1.1	10.0.1.254	255.255.255.0
		iLO PV2		10.0.1.2		
		iLO PV3		10.0.1.3		
	5	Máq.Virtuales (VM)	10.0.2.0	10.0.2.x	10.0.2.254	255.255.255.0
	6	CEHP PV1	10.0.3.0	10.0.3.1	10.0.3.254	255.255.255.0
		CEHP PV2		10.0.3.2		
		CEHP PV3		10.0.3.3		
		CEHP PV4		10.0.3.4		
	7	QPcom	192.168.35.0	192.168.35.x	192.168.35.1	255.255.255.0
		Radius		192.168.35.x		
		AP’s		192.168.35.x		
	100	MikroTik	192.168.0.0	192.168.0.x	192.168.0.254	255.255.255.0

Fuente: *(Jácome, 2019)*

3.3. Servidores Data Center

En el Data Center FICA se encuentran implementados varios servidores de aplicación basados en las necesidades de los usuarios, que además de permitir el acceso a la red inalámbrica, los restringe de usuarios no autorizados. Los servidores que se encuentran implementados actualmente son: Proxmox, Radius, Opina, Reactivos, Revista Universitaria y Control de Acceso Biométrico, mismos que son puntualizados en los siguientes acápites.

- **Servidor Proxmox**

  La plataforma de Proxmox es completamente de código abierto en cuanto a virtualización empresarial respecta (Proxmox, 2019); en el Data Center se encuentran cuatro servidores, mismos que permiten tanto el despliegue como la gestión de máquinas virtuales, presentando algunas ventajas como: el migrar servidores físicos, obtener copias de seguridad y brindar alta disponibilidad; todo esto dirigido a la red.

- **Servidor Radius**

  Radius son siglas que en inglés significan Remote Authentication Dial-In User Service y es un servidor que tiene como objetivo el verificar la autenticidad de los usuarios para así permitirles el acceso a la red. Este servidor usa un protocolo basado en un modelo de cliente-servidor, esto quiere decir que los usuarios deben tener las credenciales correctas para conectarse con el servidor.

- **Servidor Opina**

  La función del servidor Opina es la de modelar encuestas, sin importar el sitio, siempre y cuando tenga acceso a internet. En este caso los usuarios que se benefician de estas encuestas son los docentes y estudiantes de la FICA.

- **Servidor de Reactivos**

El servidor de Reactivo se encarga de gestionar los contenidos, mismos que esta orientados al ámbito educativo, a fin de que se pueda; descargar, visualizar e incluso interactuar con las actividades que el administrador proponga.

- **Servidor Revista Universitaria**

La plataforma Open Journal System (OJS) es la que ayuda en el funcionamiento del servidor de la Revista Universitaria, que como su nombre lo dice, es un sistema que permite la Administración y Publicación de documentos especiales como; revistas o periódicos (Seriadas), todo esto en Internet. Aparte de ello el sistema brinda un manejo eficiente y unificado sobre el proceso editorial, con el propósito de acelerar la difusión en los contenidos e investigación elaborados por la Universidad.

- **Servidor de Control de Acceso Biométrico**

Dentro del servidor Biométrico se almacenan cierta información de los docentes como por ejemplo; la huella dactilar, nombres y apellidos, entre otros; para posteriormente almacenar esta información en forma de bases de datos; lo hace con el fin de realizar un registro de los horarios de cada uno, tanto de entrada como salida, además de ser utilizado como medio de autenticación para el ingreso a los diferentes lugares (laboratorios, aulas y oficinas) que se encuentran en la FICA.

3.4. Diseño de la Red Definida por Software

Para la implementación de la Red Definida por Software (SDN) hace falta una plataforma que sirva como infraestructura de hardware base para que apalanque las capas que componen una SDN (Capa de Aplicación, Capa de Control, Capa de Infraestructura), en el presente trabajo se usó la infraestructura hiperconvergente que proponen en la investigación de Domínguez et al. (2020), puesto que en ella se permite la virtualización

de otras infraestructuras como es el caso de la SDN, dicha infraestructura se encuentra implementada en los servidores proxmox (PV1, PV2 y PV3) del data center.

Una vez establecida la infraestructura física ya es posible la implementación de cada una de las capas que componen la SDN, empezando por la más importante, la capa de control, a esta le sigue la capa de Aplicación y finalmente la capa de infraestructura. A continuación, y en este orden se especifica el proceso que se toma para cada una de las capas.

a. Capa de control

Como se mencionó en la sección 2.1.4 existen diversos fabricantes en lo que a controladores SDN se refiere, por este motivo la Tabla 11 presenta una comparación entre los controladores citados en el capítulo dos; a los cuales se les da una calificación con puntos que van entre 0 y 1, en el cual, 0 significa que no cumple con el parámetro mencionado y 1 significa que sí cumple con dicho parámetro. Al final, el controlador que tenga mayor cantidad de puntos será el más idóneo a usar para la presente propuesta.

Los parámetros a tomar en cuenta son: Soporte openflow, soporte de plataformas, código abierto, API REST, interfaz web y documentación. Cada uno de los parámetros mencionados se los describe a continuación:

- *Soporte openflow*, hace referencia a que el controlador debe soportar este protocolo para la comunicación con la capa de infraestructura.
- *Soporte de plataformas,* quiere decir que el controlador permita su implementación tanto en un sistema operativo libre como en uno comercial.
- *Código abierto,* se lo toma en cuenta ya que de este modo el administrador de la red tiene mayor libertad en cuanto a la configuración del controlador.

- *API REST*, permite al controlador generar u obtener datos en otros formatos como: XML o JSON, todo esto mediante sistemas que usen HTTP para su comunicación.
- *Interfaz web*, evita que los encargados de la SDN usen constantemente comandos para su administración, además que la interfaz permite a la API REST ejecutar su función.
- Finalmente, *documentación*, para que un controlador sea más sencillo de operar este debe disponer de una amplia documentación con el propósito de solucionar de una manera más eficiente posibles fallos en la SDN.

Tabla 11

Selección de controlador SDN

Criterios\Controladores	**ODL**	**ONOS**	**Floodlight**	**RYU**	**FAUCET**
Soporte OpenFlow	1	1	1	1	1
Soporte de plataformas	1	0	1	0	0
Código abierto	1	1	1	1	1
API REST	1	1	1	0	0
Interfaz web	1	1	1	1	1
Documentación	1	0	0	1	0
Total de puntos	6	4	5	4	3

Fuente: *Autoría propia*

Con un total de seis puntos, el controlador ODL cumple con todos los parámetros necesarios para el diseño de la SDN, por lo cual será implementado en una máquina virtual dentro la infraestructura hiperconvergente.

b. Capa de aplicación

La capa de aplicación no implica configuración alguna aparte de instalar en la misma máquina virtual del controlador ODL los módulos de la interfaz web y API REST que este dispone, permitiendo su administración de acuerdo con las necesidades que

presenten los usuarios en la SDN, como; la convergencia con nuevas redes, implementación de servidores, bloqueo de interfaces, entre otras necesidades que se puedan suscitar.

c. Capa de Infraestructura

La capa de infraestructura garantiza la interacción entre usuarios finales y servidores implementados en la SDN, para que esto suceda se necesitan dispositivos de red compatibles con el protocolo que usa el controlador ODL; en este caso openflow. De momento los dispositivos de red físicos que conforman la infraestructura del data center en la FICA no soportan el protocolo openflow, por lo tanto se configura el software Open Virtual Switch (OVS) en una máquina virtual creada dentro de la infraestructura hiperconvergente, cumpliendo la misma función de un switch físico, con la diferencia de que OVS gestiona las tablas de flujos utilizadas por openflow.

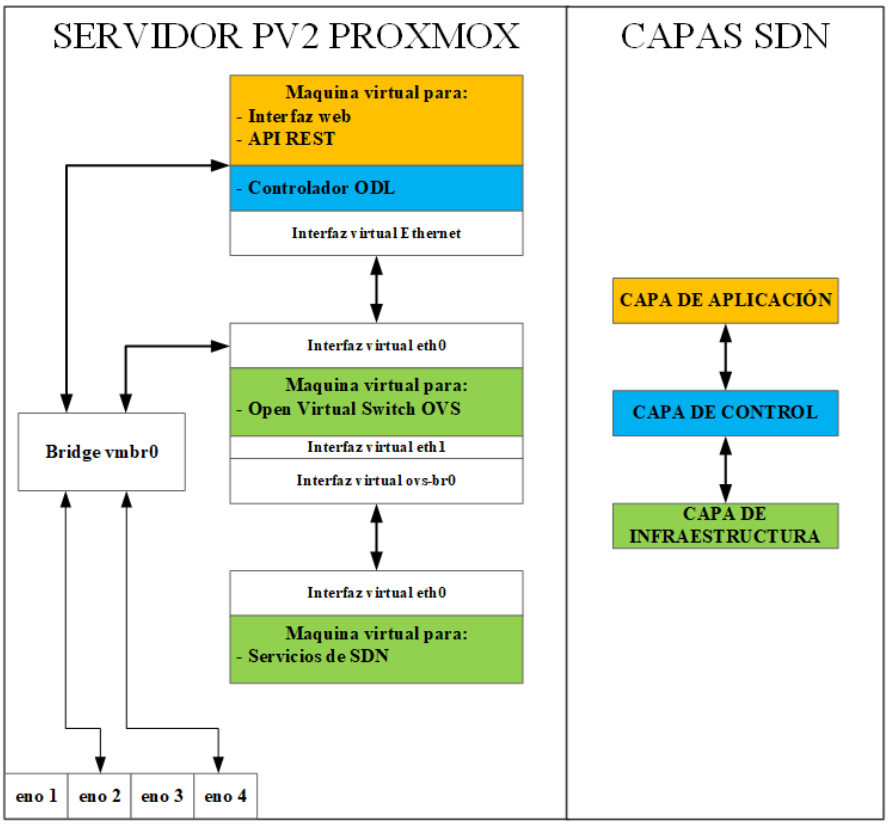


Figura 33. *Diseño de SDN en servidor PV2*

Fuente: *Autoría propia*

La Figura 33 muestra el diseño propuesto, precisando su implementación en el servidor físico PV2 de la infraestructura hiperconvergente citada al inicio de esta sección, además, señala las máquinas virtuales creadas y conexiones entre interfaces virtuales desde la capa de control hasta la capa de infraestructura y todas estas a su vez con el PV2; a continuación se aclara el proceso llevado a cabo en cada capa.

3.5. Implementación de la Red Definida por Software

La implementación de la SDN empieza por la capa de control puesto que es la parte esencial en este tipo de red y basado en el diseño de la sección 3.4, el software escogido es Opendaylight (ODL). ODL es una infraestructura con algunos beneficios para el despliegue de SDN, entre ellos están: alta disponibilidad, escalabilidad, multi-protocolos, entre otros (OpenDaylight, 2019).

ODL se instala dentro de una máquina virtual (VM) con sistema operativo Windows, la cual esta creada en el servidor físico PV2 de la infraestructura hiperconvergente, los requerimientos de esta máquina, tanto en hardware como software se especifican en la Tabla 12, teniendo en cuenta que hubo un incremento en el espacio de almacenamiento en disco duro (HDD) aparte de lo recomendado por OnApp (2020), con el propósito de que el administrador tenga libertad de agregar aplicaciones extras en el servidor. El proceso de instalación se detalla en el Anexo 1.

Tabla 12

Características de Máquina Virtual

Software	Versión	Hardware recomendado	Hardware configurado
Windows	10 Profesional de 64 bits	16 GB HDD 8 GB RAM	100 GB HDD 8 GB RAM
Opendaylight	Beryllium SR4		

Fuente: *Adaptado de (OnApp, 2020)*

A continuación se instala los módulos principales que necesita el controlador para establecer la comunicación con las demás capas de la SDN, dichos módulos se definen en la Tabla 13.

Tabla 13

Módulos de ODL

NOMBRE	DESCRIPCIÓN	COMANDO
DLUX	Proporciona una interfaz gráfica de usuario intuitiva para OpenDaylight.	odl-dlux-all
Switch L2	Proporciona reenvío L2 (Ethernet) a través de conmutadores OpenFlow conectados y soporte para seguimiento de host.	odl-l2switch-switch
Soporte RESTCONF API	Permite el acceso de la API REST a MD-SAL, incluido el almacén de datos.	odl-restconf
Model-Driven SAL (MD-SAL)	Conjunto de servicios de infraestructura destinados a proporcionar soporte común y genérico a los desarrolladores de aplicaciones y complementos.	odl-mdsal-apidocs
OpenFlow Plugin	Interfaz de comunicaciones estándar para permitir la interacción entre las capas de una arquitectura SDN.	odl-openflowplugin-all

Fuente: *Adaptado de (OpenDaylight Project, 2016)*

Una vez finalizada la instalación de los módulos principales, tanto la interfaz web como la API REST que forman parte de la capa de aplicación están disponibles para administrar el controlador, tal como se exhibe en la Figura 34. EL objetivo de la interfaz web es visualizar la topología que ODL detecta automáticamente a partir de la información que los dispositivos de red envían mediante el puerto 6653, mientras que la API REST gestiona la comunicación de la capa de control con la capa de infraestructura.

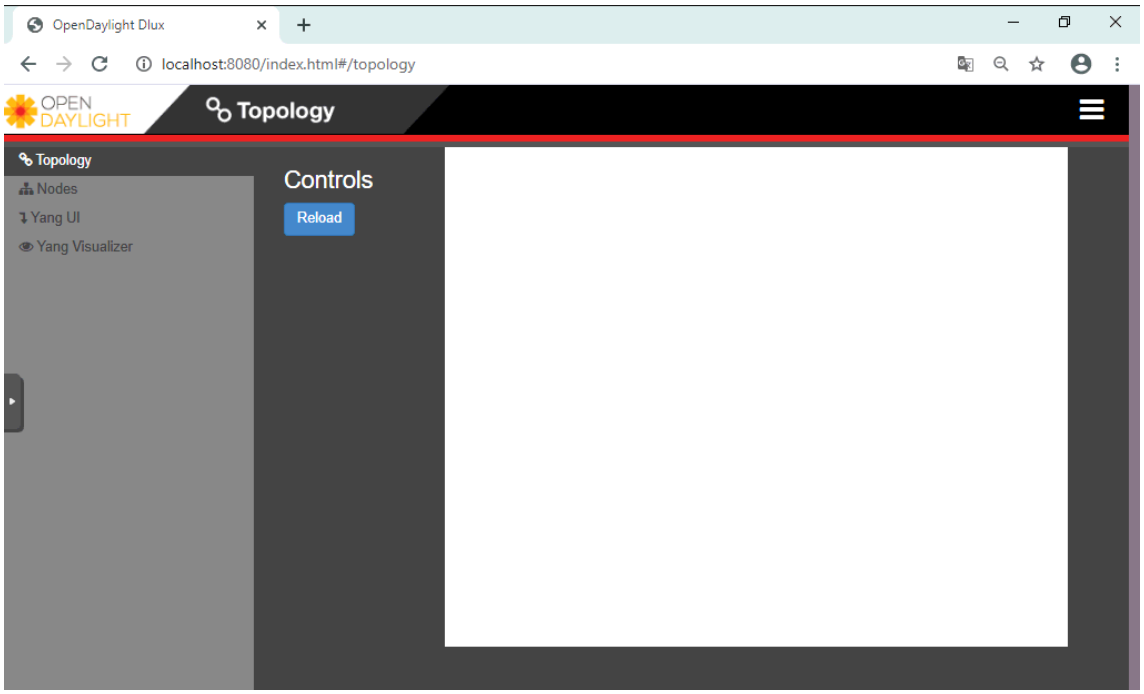


Figura 34. *Interfaz Gráfica de ODL*

Fuente: *Autoría propia*

Finalmente, la capa de infraestructura de la SDN usa a Open vSwtich (OVS) como switch virtual para el propósito ya citado en el apartado c de la sección 3.4, mismo que se implementa en otra VM con sistema operativo Centos creada en el PV2 de la red hiperconvergente, esta máquina debe contar con los requerimientos señalados en la Tabla 14. Cabe aclarar que estos requerimientos son mayores a los sugeridos por CentOS (2020) puesto que se espera que el tráfico de toda la SDN atraviese el OVS sin ningún inconveniente y la documentación de OVS no especifica este tipo requerimientos para su ejecución. El Anexo 2 contiene el procedimiento completo de su instalación.

Tabla 14

Características de Máquina Virtual para OVS

Software	Versión	Hardware sugerido	Hardware configurado
Centos	7 de 64 bits	20 GB HDD 1 GB RAM	100 GB HDD 4 GB RAM

Fuente: *Adaptado de (CentOS, 2020)*

Con el OVS implementado, lo siguiente es conectarlo al controlador para que la SDN funcioné, con la comunicación ya establecida los paquetes openflow se transmiten por la red. La comunicación se verifica de dos formas, la primera es observar los paquetes openflow a través de un sniffer, la Figura 35 muestra este proceso en el sniffer denominado wireshark.

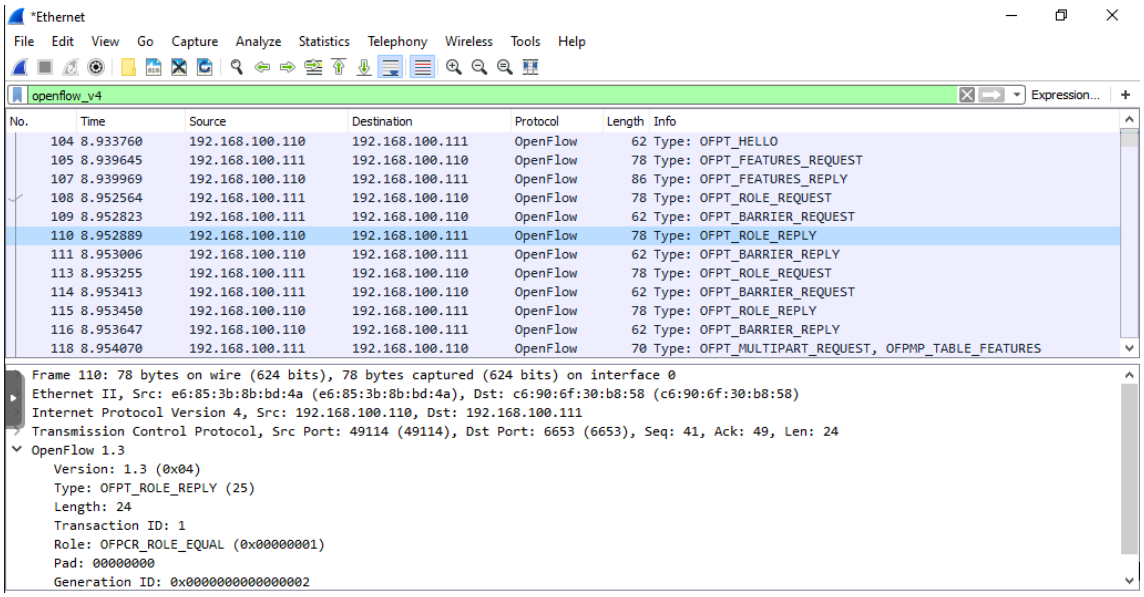


Figura 35. *Paquetes openflow capturados por wireshark*

Fuente: *Autoría propia*

Una segunda forma de verificar que la comunicación se ha establecido de forma correcta es en la interfaz web del controlador puesto que ahí se crea una topología de la red, tal como lo indica la Figura 36.

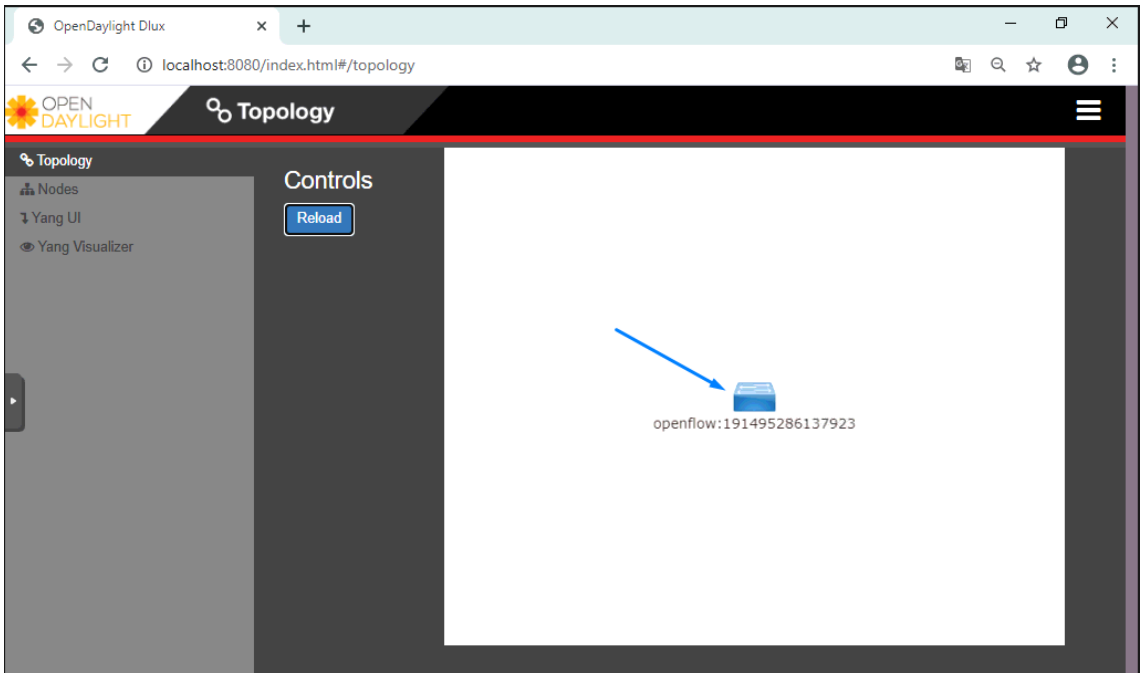


Figura 36. *OVS detectado por controlador SDN*

Fuente: *Autoría propia*

3.5.1. Diseño del escenario de pruebas SDN.

Con la infraestructura SDN implementada, para el escenario de pruebas se necesita instalar servidores: web, transferencia de archivos (FTP) y moodle a la SDN para posteriormente integrar el sistema de detección de intrusos con redes neuronales a esta red. De modo que para existir convergencia en dicho escenario hace falta un direccionamiento IPv4, mismo que se encuentra en la Tabla 15.

Tabla 15

Direccionamiento IPv4 de la SDN

Nombre Dispositivo	IP	Submáscara de red	Puerta de enlace
ODL	192.168.100.111	255.255.255.0	192.168.100.110
OVS	192.168.100.110	255.255.255.0	192.168.100.100
	192.168.150.10	255.255.255.0	
Servidor Web	192.168.150.11	255.255.255.0	192.168.150.10

Servidor FTP	192.168.150.12	255.255.255.0	192.168.150.10
Plataforma Moodle	192.168.150.13	255.255.255.0	192.168.150.10

Fuente: *Autoría propia*

La Figura 37 representa una topología lógica de cómo se conecta cada dispositivo que compone el escenario de pruebas SDN, entre ellos se incluyen al controlador ODL, al OVS y los servidores ya citados.

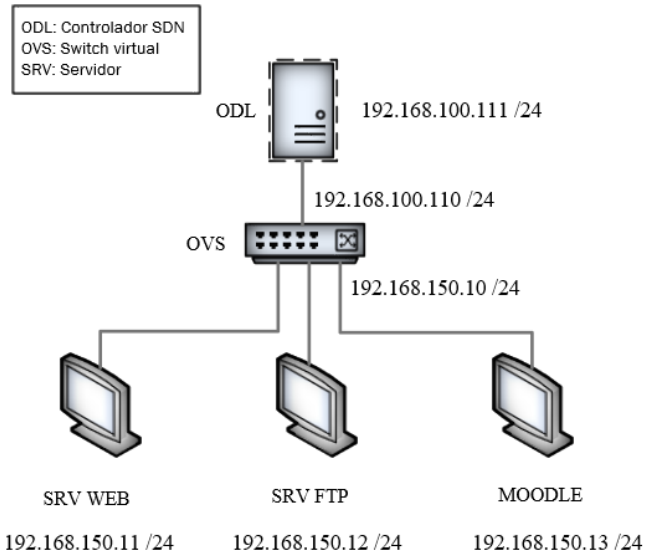


Figura 37. *Topología lógica del escenario SDN*

Fuente: *Autoría propia*

3.5.2. Implementación de servidores.

Para la implementación de los servidores hacen falta máquinas virtuales que alberguen cada servicio, las cuales se alojan en el PV2 de la infraestructura hiperconvergente para posteriormente pasar a pertenecer a la SDN, la Tabla 16 indica los requerimientos en hardware y software con los que cuentan que cada máquina virtual, mismos que se basan en CentOS (2020), quien da a conocer los requerimientos mínimos en hardware que necesita CentOS 7 para la ejecución de cada uno de los servicios mencionados al inicio de la sección 3.5.1. No obstante, estos requerimientos poseen una

variación a la recomendada, con el propósito de que no presenten problemas en caso de haber crecimiento en la SDN.

Tabla 16

Requisitos de máquinas virtuales para servidores

Servicio	Software	Versión	Hardware recomendado	Hardware configurado
Web	Centos	7 de 64 bits	20 GB HDD 1 GB RAM	40 GB HDD 2 GB RAM
FTP	Centos	7 de 64 bits	20 GB HDD 1 GB RAM	40 GB HDD 2 GB RAM
Moodle	Centos	7 de 64 bits	20 GB HDD 1 GB RAM	60 GB HDD 6 GB RAM

Fuente: *Adaptado de (CentOS, 2020)*

Una vez creadas las máquinas virtuales, se implementa cada uno los servicios anteriormente mencionados, los cuales son tratados de manera individual en los siguientes literales.

a. Servidor Web

Un servidor web tiene como objetivo, la gestión de una aplicación entre servidor y cliente, el código que recibe el segundo por lo general es interpretado por un Navegador Web. Esta transmisión puede ser mediante dos protocolos, uno es el Protocolo de Transferencia de Hipertexto (HTTP), mismo que trabaja en el puerto 80 y el segundo es el Protocolo Seguro de Transferencia de Hipertexto (HTTPS) que trabaja en el puerto 443; la diferencia que existe entre ambos es el nivel de seguridad al momento de la comunicación, puesto que HTTPS usa Protocolo de Capa de Conexión Segura (SSL) o Seguridad de la Capa Transporte (TLS) para brindar una mejor confianza en la transferencia de datos, al contrario de HTTP que transmite los datos en texto plano, es decir, sin ningún tipo de seguridad.

En cuanto a la aplicación que se instala en el servidor web, es Apache, puesto que según El computador 2020 (2020) se encuentra entre las más usadas en los sistemas operativos Linux. El proceso de instalación completo de este servidor se explica en el Anexo 3.1.

Tan pronto como la instalación haya finalizado se realiza una prueba de funcionamiento con cualquier navegador de un dispositivo (computador, celular, Tablet, entre otros) que se encuentre en la misma red que el servidor web, basta con escribir la IP del servidor en la barra de búsqueda; la prueba de funcionamiento de dicho servidor se encuentra en la Figura 38.

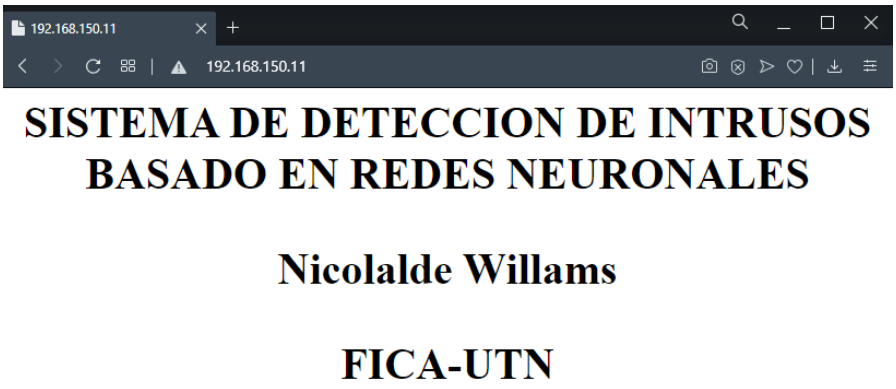


Figura 38. *Test de funcionamiento servidor web*

Fuente: *Autoría propia*

b. Servidor FTP

El servidor FTP tiene como función, la transferencia de archivos entre dispositivos que se encuentren conectados a una red de tipo TCP/IP, además trabaja en un modelo cliente-servidor, este servidor usa dos puertos de red para su funcionamiento, el puerto 20 lo hace para la transferencia de datos y el puerto 21 para el control, el proceso descrito es graficado en la Figura 39.

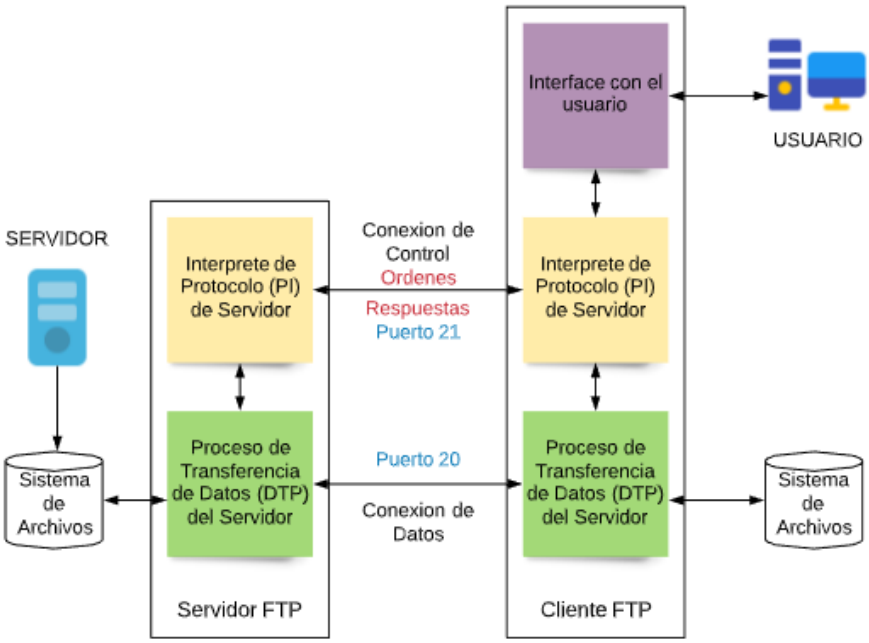


Figura 39. *Proceso del servidor FTP*

Fuente: *Adaptado de https://bit.ly/2LXtsoD*

El proceso de instalación para el servidor FTP se detalla en el Anexo 3.2; en seguida de que el servidor esté listo, se realiza una prueba de funcionamiento desde un ordenador externo, el cual accede al servicio con las credenciales apropiadas, tal como se presenta en la Figura 40.

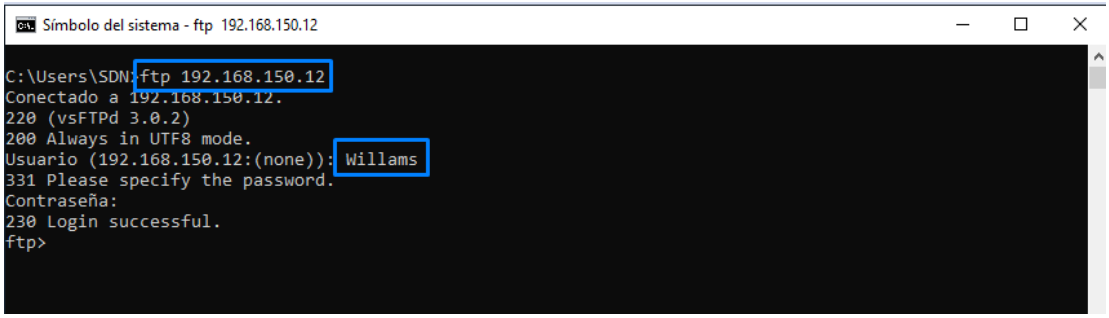


Figura 40. *Test de funcionamiento servidor FTP*

Fuente: *Autoría propia*

c. Moodle

En cuanto a Moodle, Moodle (2020) lo define como “una plataforma de aprendizaje diseñada para proporcionarle a educadores, administradores y estudiantes un

sistema integrado único, robusto y seguro para crear ambientes de aprendizaje personalizados". Al funcionar a través del servidor web, los puertos que usa son los mismo que se mencionó en el literal a de esta sección; puerto 80 para HTTP y puerto 443 para HTTPS.

Al igual que los servicios anteriores, la implementación de Moodle se describe en el Anexo 3.3, sin embargo, la Figura 41 constata el funcionamiento de la plataforma en mención.

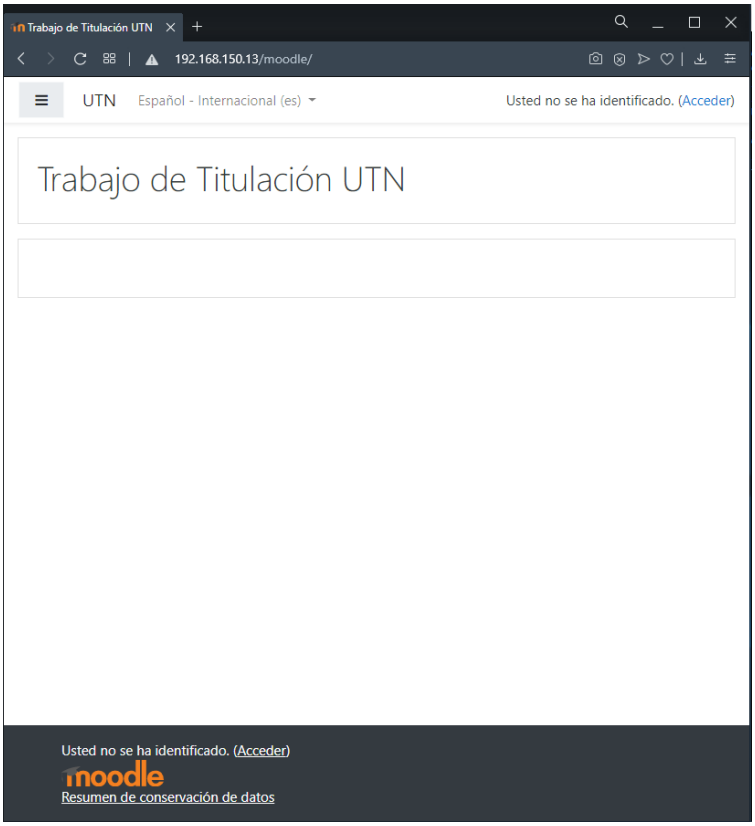


Figura 41. *Test de funcionamiento de Moodle*

Fuente: *Autoría propia*

Para una mejor comprensión de lo que se acaba de implementar se grafica una topología lógica partiendo de la topología original (véase Figura 32) y mostrando como esta queda una vez que el escenario de pruebas SDN se encuentre listo en el data center de la FICA. Dicha topología se encuentra en la Figura 42 y se aprecia cada uno de los

elementos citados a lo largo de esta sección, como son: el controlador ODL, el switch virtual OVS y los servidores.

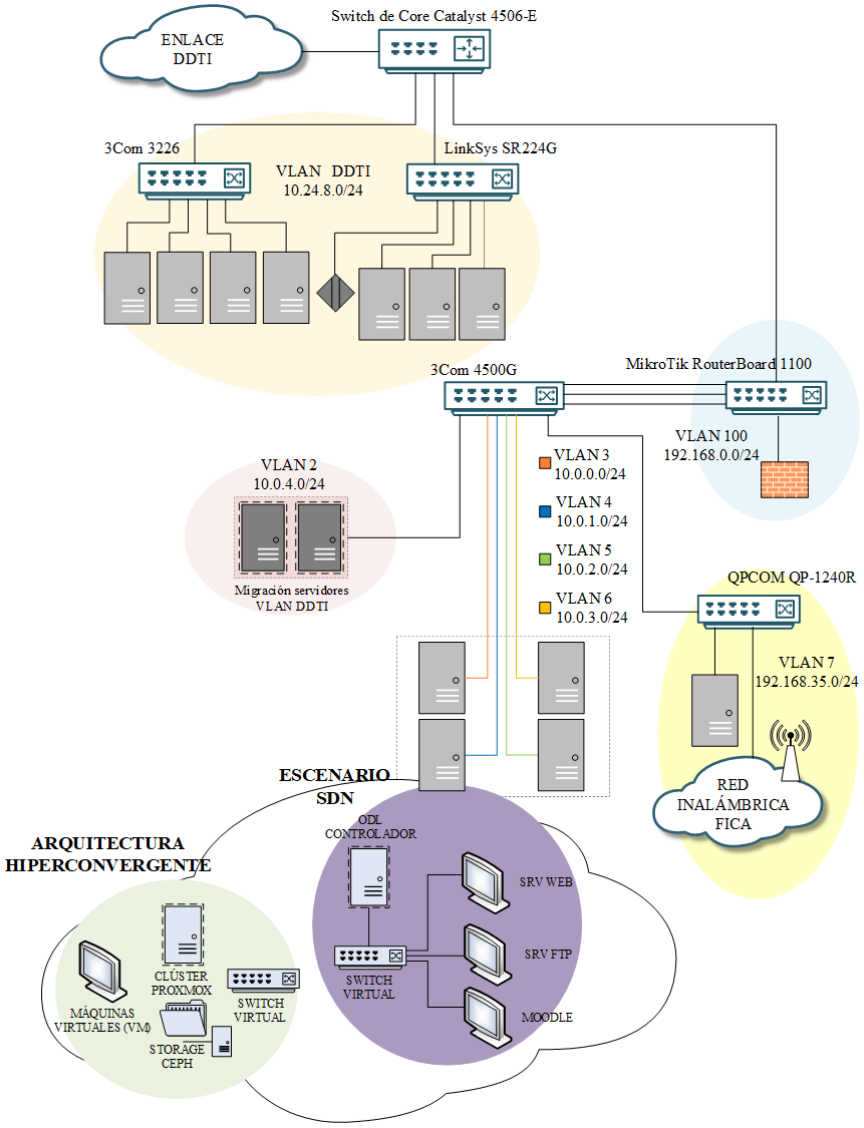


Figura 42. *Topología Lógica Data Center y SDN*

Fuente: *Adaptado de Data Center FICA*

Por otra parte, este mismo escenario se lo observa en la interfaz web que dispone el controlador ODL, si la implementación se ejecuta de forma correcta en la interfaz web debe mostrarse una topología similar a la Figura 43, con alguna variación en cuanto a la posición de cada dispositivo.

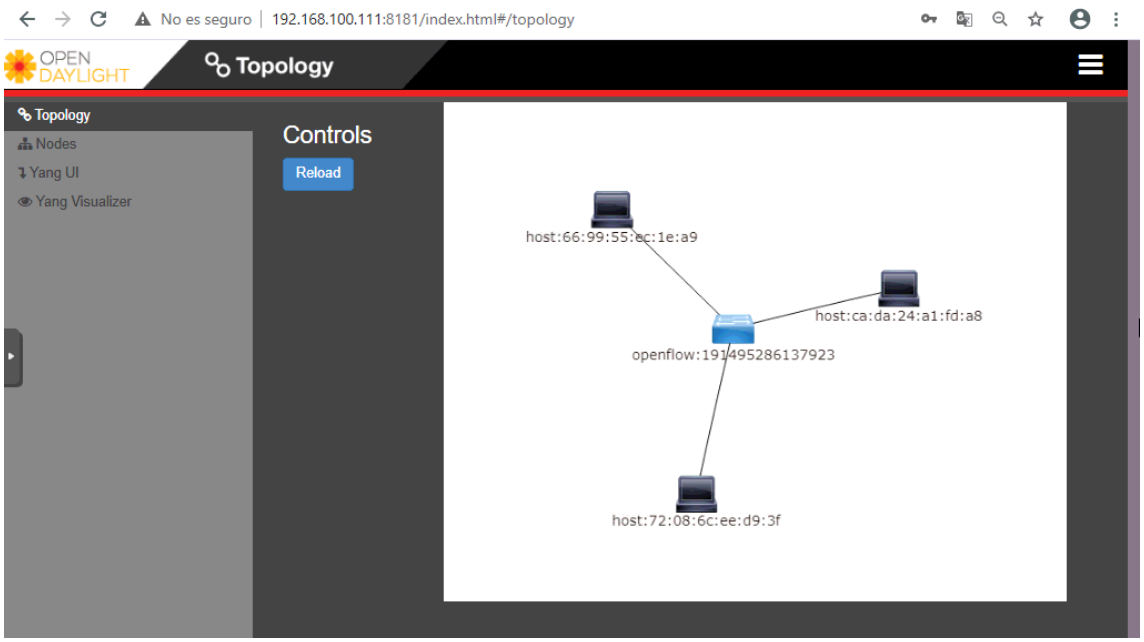


Figura 43. *Topología en controlador ODL*

Fuente: *Autoría propia*

CAPÍTULO 4.

DISEÑO DEL IDS CON RNA

Este capítulo abarca toda la informacion relacionada con el Sistema de Deteccion de Intrusos (IDS), el presente trabajo propone el software denominado Snort, puntualizando sus cualidades, arquitectura y funcionamiento. Además de detallar la integración con la Red Neuronal Artificial (RNA) y como este nuevo IDS forma parte de la SDN que se encuentra en el Data Center de la FICA.

4.1. Selección de IDS

Basado en la Tabla 5, se determinó el software más idóneo para la detección de intrusos en esta propuesta, el cual es Snort, mismo que cumple con todas las características expuestas en dicha tabla. Es así como los siguientes apartados dan a conocer la información necesaria sobre este IDS, desde su definición hasta su arquitectura y como operan cada uno de los módulos que la conforman.

4.2. Snort

Snort es un IDS disponible en código abierto que trabaja bajo las licencias GPL de GNU, además que se desarrolló en lenguaje C (Snort, 2020), esto con el propósito que las personas conozcan su programación y así lo acoplen a las diferentes necesidades que se presenten al momento de brindar seguridad a la red de comunicación. Lo que caracteriza a este IDS es el análisis en tiempo real al tráfico de la red; Snort se lo puede descargar de su página oficial www.snort.org, misma que cuenta con las últimas versiones y documentación correspondientes a cada una de ellas. En otras palabras, Snort es similar a un sniffer ya que analiza los paquetes que se transmiten a través de la red, la diferencia en Snort es que, al detectar una anomalía en la red, este genera alertas. Las alertas que presenta un IDS se encuentran detalladas la sección 2.2.1.

4.2.1. Funcionamiento

Una vez identificado las cualidades del IDS a utilizarse en el presente trabajo, lo siguiente es comprender su esquema de funcionamiento, el cual se menciona en (Sáenz & Martínez, 2018) y la Figura 44 señala que los datos parten desde el *tráfico de red* hasta el *modo de salida* para posteriormente continuar con la trayectoria hacia su destino sea cual fuere dentro de la red. En este sentido, para una mejor comprensión del esquema de funcionamiento se describe cada uno de los módulos que lo conforman.

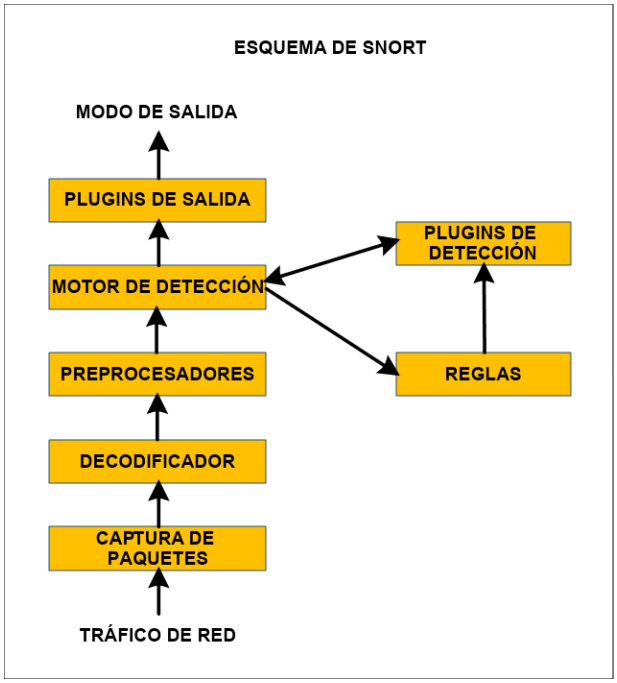


Figura 44. *Esquema original de Snort*

Fuente: *Adaptado de (Sáenz & Martínez, 2018)*

El primer módulo que conforma el esquema de Snort es *Captura de paquetes* y tal como su nombre lo dice, se encarga de capturar los paquetes que circulan por la red para ser analizados por el resto de módulos; este proceso lo realiza gracias a una librería que en el caso de Linux es *libcap* y en Windows es llamada *winpcap*.

Decodificador; el siguiente módulo se encarga de interpretar los paquetes capturados con el propósito de conocer el protocolo que usa cada uno y de esta manera

estructurar sus datos para un posterior análisis. Snort es capaz de decodificar protocolos Ethernet, protocolos de IP de Línea Serial (SLIP) y Protocolos Punto a Punto (PPP).

El tercer módulo se denomina *Preprocesador* y se encuentra conformado por programas codificados en lenguaje C; su principal función es analizar los paquetes decodificados y ordenarlos; es decir, la gran cantidad de paquetes que provienen de la red y han sido decodificados, el preprocesador los ordena con el propósito de que puedan ser identificados de mejor manera por el motor de detección.

Motor de detección; es parte fundamental del esquema de Snort ya que su principal labor es analizar paquetes ya ordenados anteriormente y detectar ataques, todo esto basándose en reglas previamente establecidas; en otras palabras, este módulo se encarga de parear a los paquetes con las reglas existentes y así ejecutar una acción pertinente. Además, que pueden ayudarse de módulos auxiliares (plugins de detección) permitiendo un análisis óptimo.

Plugin de detección; es un módulo auxiliar que forma parte de la compilación ejecutada por Snort y suele usarse para alterar el funcionamiento del motor de detección.

Las *Reglas* rigen el análisis de los paquetes detectados, comúnmente usados por el Motor de detección para asignar acciones a paquetes que se emparejen con ciertas reglas. Adicionalmente tienen su propia sintaxis, misma que se divide en dos: cabecera y opciones. La primera parte; *Cabecera*, se encarga de dar a conocer que acción pertenece a cada regla, que protocolo pertenece a cada paquete, además de sus direcciones IP tanto de origen como destino y de igual manera con los puertos usados en dicho paquete. Toda la estructura ya mencionada con respecto a la cabecera se representa en la Figura 45. Mientras que la Tabla 17 exhibe un ejemplo de una regla ya establecida en Snort.

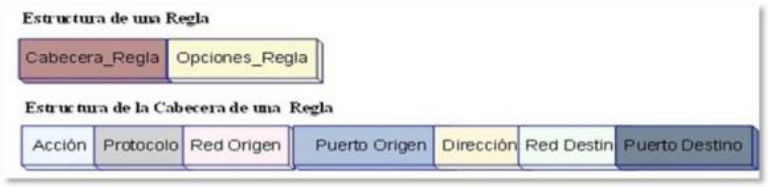


Figura 45. *Sintaxis de reglas en Snort*

Fuente: *(SNORT, 2020) Obtenido de https://snortudenar.wordpress.com/snort-2/*

Tabla 17

Ejemplo de regla en Snort

Ejemplo de regla en Snort						
Acción	**Protocolo**	**Red Origen**	**Puerto Origen**	**Dirección**	**Red Destino**	**Puerto Destino**
alert	tcp	$EXTERNAL_NET	any	➔	$HOME_NET	53

Fuente: *Adaptado de (Snort, 2020)*

En la estructura ya mencionada se identifican cinco acciones posibles que se aplican en las reglas y a cada una se las define en la Tabla 18.

Tabla 18

Acciones de una regla en Snort

Tipo de acción	Definición
Log	Genera un registro correspondiente al paquete.
Alert	Genera una alerta y posterior a esto genera un log del paquete que activa la regla.
Pass	Permite el paso de paquetes que correspondan a la regla, de igual manera generando un log.
Activate	Origina una alerta, posteriormente activa una regla dinámica.
Dynamic	Permanece en forma pasiva hasta que un activate provoca su activación, su propósito es dar una información adicional a la que se obtendría con la regla inicial.

Fuente: *Adaptado de (Sánchez Lorente, 2015)*

Por otra parte, las *Opciones* se encargan de detallar la información que un paquete debe tener para activar su regla correspondiente. Aquí se encuentran cuatro categorías, las cuales son definidas en la Tabla 19.

Tabla 19

Categorías en Opciones de reglas

Categoría	Definición
Generales	Brinda información de una regla sin afectar a la detección.
Detección de contenido	Explora patrones en el interior de la carga útil que posee un paquete
Detección de non-payload	Explora patrones en otros campos del paquete que no sean referentes a la carga útil.
Postdetección	Activa reglas específicas posterior a que otra regla se haya accionado.

Fuente: *Adaptado de (Sánchez Lorente, 2015)*

El último módulo es conocido como *Plugin de salida* y tiene como misión el establecer los métodos de guardado para las alertas generadas, ya sean en bases de datos, servidores de registro, archivos de texto, entre otros.

4.2.2. Modos de operación

Basado en el funcionamiento de Snort es posible aclarar cuatro modos en los que este funciona: modo sniffer, modo packet logger, modo NIDS y modo inline; a continuación, se puntualiza dichos modos:

Modo Sniffer; este modo consiste en que Snort únicamente analiza los paquetes que circulan por la red, mostrando al encargado de la red la información de cada paquete capturado, esto lo efectúa mediante su interfaz gráfica.

Modo Packet Logger; el segundo modo que posee Snort, es similar al anterior modo, sin embargo, aquí se almacenan los paquetes capturados con el propósito de que estén disponibles para un futuro análisis.

El siguiente modo se denomina *Modo NIDS*; este modo permite un análisis de los paquetes que circulan por la red en busca de intrusiones, esto lo hace comparando los paquetes con las reglas que se establezcan en los módulos ya citados en la sección 4.2.1.

Finalmente, el *Modo Inline*; la cualidad de este modo es que interactúa con el Firewall de la red, obteniendo sus paquetes y determinando así el paso o bloque de estos, dicho de otra manera, el Firewall permite el paso del tráfico basándose en las reglas que se encuentren programas en Snort.

4.2.3. Selección de modo de operación

Ahora bien, con los modos de operación que ofrece Snort es necesario determinar cuál de ellos se acopla mejor a las necesidades de esta propuesta, estas necesidades se las representa mediante características en la Tabla 20.

Tabla 20

Características del modo de operación de Snort

Características \ Modos Snort	Sniffer	Packet logger	NIDS	Inline
Capturar paquetes	SI	SI	SI	SI
Analizar paquetes	NO	NO	SI	SI
Alertar anomalías	NO	NO	SI	SI
Independiente de firewall	SI	SI	SI	NO

Fuente: *Autoría propia*

En base con la Tabla 20 se concluye que el modo que mejor se adapta al presente trabajo es el NIDS puesto que permite el análisis del tráfico en la red, además de alertar

a los administradores de esta en caso de existir el intento de alguna intrusión, todo esto independientemente de cómo se encuentre configurado el firewall de la red.

4.3. Selección de la Red Neuronal

Apoyado en las arquitecturas referidas del capítulo dos, aquí se presentan a tres redes que operan bajo estas arquitecturas, dichas redes son: Perceptrón Múltiple Capa (MLP*: Multi Layer Perceptron*), Mapas de Kohonen y Red Elman.

Perceptrón Múltiple Capa (MLP); esta red se sujeta a dos arquitecturas mencionadas en la sección 2.7.1 y son: la red multicapa y red neuronal recurrente. Esto quiere decir que posee capas ocultas entre las capas de entrada y salida; además de realizar una función de retroalimentación tal como se nota en la Figura 46; con el propósito de alterar los pesos existentes en las conexiones que se presenta entre las entradas y salidas de cada neurona que conforma la red. Esto a su vez permita que los resultados de la red MLP se acerque en lo posible a la solución deseada.

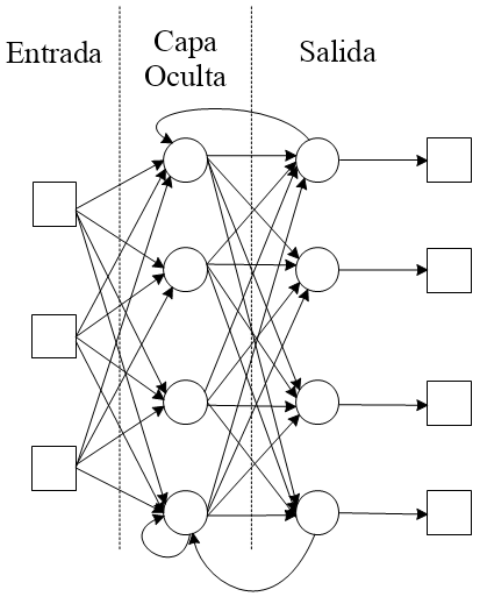


Figura 46. *Red MLP*

Fuente: *Adaptado de (Calvo, 2018)*

Otra cualidad de la red MLP es que opera bajo un aprendizaje supervisado, dicho en otras palabras, necesita que un usuario (supervisor) indique el resultado (la salida) deseado con el fin de que dicha red se aproxime a éste.

En tanto que la red *Mapas de Kohonen*; atribuye este nombre a su creador Teuvo Khonen y esta red opera bajo un aprendizaje no supervisado, es decir, no necesita que un usuario (supervisor) establezca un resultado (salida) deseado. Lo que ocasiona que la red se autoorganice por sí misma. En cuanto a su arquitectura, este tipo de red es monocapa tal como se grafica en la Figura 47, dicho de otra manera, posee dos capas; capa de entrada y capa de salida, lo que ocasiona que los datos recibidos por la primera capa sean transmitidos directamente a la capa de salida.

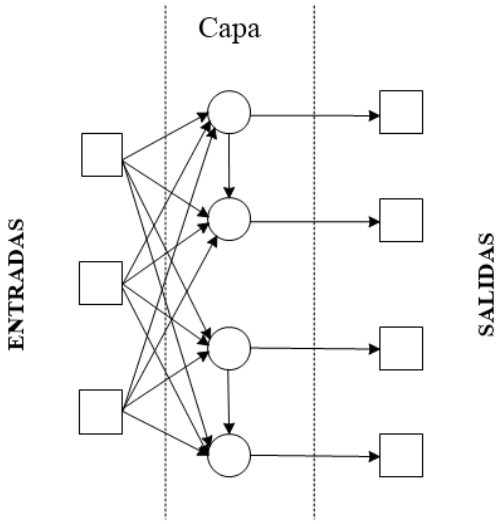


Figura 47. *Mapas de Kohonen*

Fuente: *Autoría propia*

Por último, está la *Red Elman*; este tipo de red es similar a MLP ya que posee una arquitectura multicapa y retroalimentación, la diferencia está en la segunda cualidad puesto que la red Elman se retroalimenta únicamente entre sus capas ocultas justo como se presenta en la Figura 48. Este tipo de retroalimentación limita en cierta manera al aprendizaje puesto que los cambios se basan únicamente en las variaciones que suceden

en sus capas ocultas provocando un aprendizaje más prolongado, a diferencia de la red MLP ya que los cambios en su retroalimentación dependen directamente de los resultados (capa de salida) provocando que el aprendizaje se realice en menor tiempo. Cabe recalcar que la red Elman opera con un aprendizaje supervisado permitiendo que se establezcan los resultados deseado y la red se aproxime a estos con cierta exactitud.

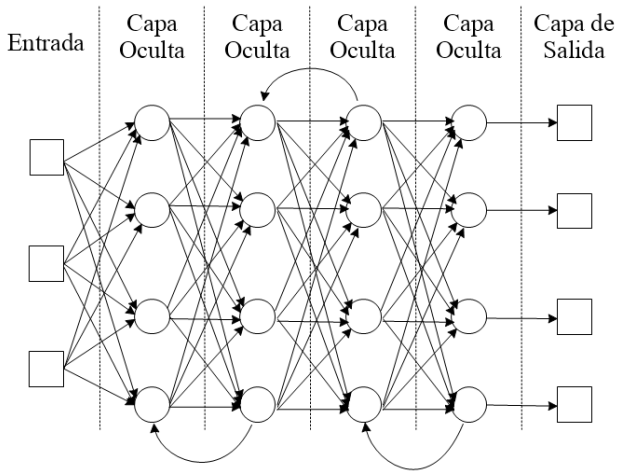


Figura 48. *Red Elman*

Fuente: *Adaptado de (Díaz, 2016)*

En base a las RNA mencionadas, se presenta la Tabla 21, la cual tiene la intención de compararlas en función a las características que mejor se ajusten a este proyecto y que posteriormente garanticen una correcta integración al IDS. Las características consisten en que la RNA permita llegar a una pronta solución mediante modelos sencillos (sencillez del diseño), también que la información que se obtenga de esta se fácil de interpretar (compresión de la información en las salidas), además de que la RNA no consuma muchos recursos del computador tanto en memoria como en procesador (consumo bajo de recursos), finalmente se observa que esta disponga de un software para su diseño (software de diseño existente).

Tabla 21

Comparación entre RNA

Características	MLP	Mapas de Kohonen	Red Elman
Sencillez del diseño	SI	NO	NO
Comprensión de la información en las salidas	SI	NO	SI
Consumo bajo de recursos	SI	NO	SI
Software de diseño existente	SI	SI	SI

Fuente: *Autoría propia*

Observando los resultados de la comparación, se determina que la RNA que mejor se adapta es la Red Perceptrón Múltiple Capa (MLP) puesto que presenta mayor cantidad de resultados a favor, al contrario de la red de Mapas de Kohonen y la red Elman.

4.4. Integración del preprocesador con RNA a Snort

Una vez determinado Snort como software IDS a utilizarse y MLP como la RNA, lo siguiente es integrarlos; para ello se empieza por el diseño de la RNA y posteriormente se profundiza en la parte del módulo preprocesador. Este último se incorpora a Snort con el propósito de que use la red MLP en el análisis del tráfico de una red.

El diseño de la red MLP tiene como fin el brindar la información necesaria para la codificación del preprocesador, por este motivo, este diseño se encuentra elaborado en un software llamado JavaNNS; mismo que se especializa en la simulación de RNAs. Al ser un diseño predefinido, la red MLP se compone por dos etapas: capas y set de entrenamiento; en cuanto a la primera etapa que son las *Capas*, se crea una red MLP tal como indica la Figura 49; se forma por siete capas de entrada, cuatro capas ocultas y dos capas de salida. Cabe recalcar que al ser una red sin entrenar todos sus valores se encuentran seteados en cero, tanto los valores de todas las capas (capas de entrada, capas ocultas y capas de salida), como los correspondientes a los pesos; además, las distintas

capas de entrada representa a una variable que analiza la versión de Snort propuesta por Bedon (2020), esta versión lleva el nombre de Snort+RNA, la variables mencionadas se tratan en la Tabla 22.

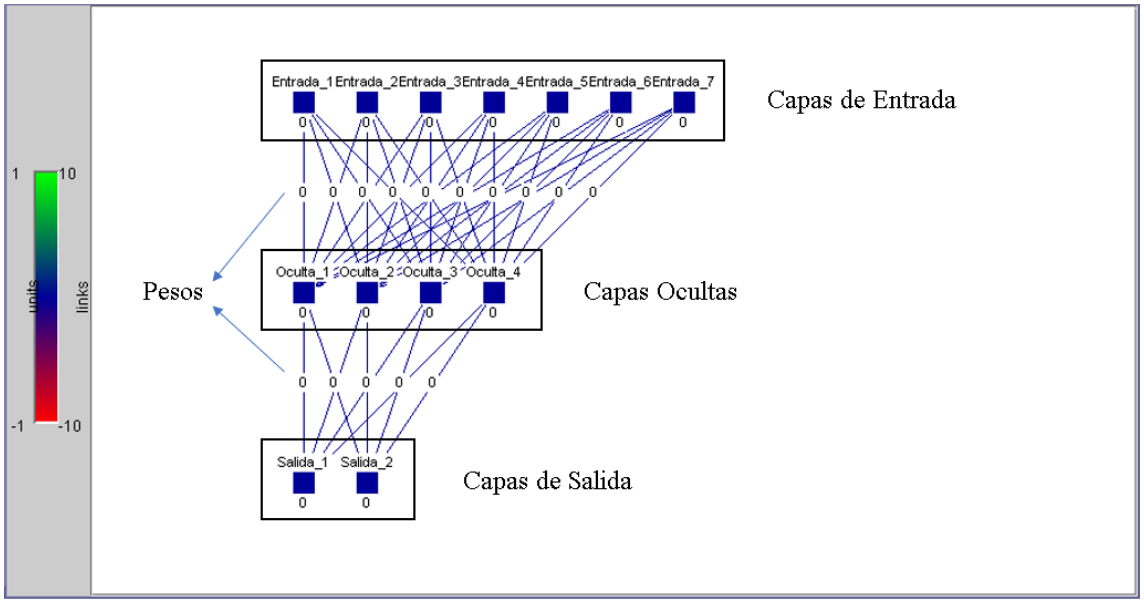


Figura 49. *Red MLP sin entrenar*

Fuente: *Adaptado de Bedon (2020)*

La segunda etapa en el diseño de la red MLP es el *Set de entrenamiento*, mismo que se obtiene de https://sourceforge.net/projects/snort-ai/ y debe ser cargado en JavaNNS para su uso; Bedon (2020) es su autor ya que integró un módulo (módulo de entrenamiento) a Snort, el cual se encarga de generar este set, almacenando las variables necesarias para el entrenamiento de la red MLP y así esta determine si existe una anomalía o no en la red. Las variables que contiene este set de entrenamiento son: *hits_as_dst*, *hits_as_src*, *av_rcv_time*, *av_snd_time*, *ack_rst_resp*, *rh/has* y *rh/had*; y se definen en la Tabla 22.

Tabla 22

Variables del set de entrenamiento.

Variable	Definición
hits_as_dst	(hits as destination – hits como destino): Número de paquetes de inicio de conexión (flows) en los cuales esta dirección IP figuraba como destino. Es decir, número de intentos de conexión que la IP recibió.
hits_as_src	(hits as source – hits como origen): Número de paquetes de inicio de conexión (flows) en los cuales esta dirección IP figuraba como origen. Es decir, número de intentos de conexión enviados por la IP.
av_rcv_time	(average receive time – tiempo promedio de recepción): Tiempo (segundos) promedio entre peticiones de conexión recibidas por esta IP. Esta medida dice con qué frecuencia están llegando peticiones de conexión a un equipo. Si este número es inferior a 10^{-3} quiere decir que existe una anomalía en la red.
av_snd_time	(average send time – tiempo promedio de envío): Tiempo (segundos) promedio entre peticiones de conexión hechas por esta IP. Esta medida dice con qué frecuencia un equipo está enviando peticiones de conexión. Si este número es inferior a 10^{-3} quiere decir que existe una anomalía en la red.
ack_rst_resp	(Respuestas ACK y RST): Número de respuestas negativas que una dirección IP envía. Una respuesta negativa (valor de uno) es el mensaje que un host genera al recibir una petición de conexión en alguno de sus puertos cuando éste está cerrado, las respuestas negativas para el protocolo TCP están dadas generalmente por el envío de un paquete con las banderas RST y ACK fijadas.
rh/has	(Conexiones de relación como origen): Indica el número de inicios de conexión que son enviados por un origen específico y que han llegado al destino.
rh/had	(Conexiones de relación como destino): Indica el número de inicios de conexión que son recibidos por un destino específico y que han venido de un origen destino.

Fuente: *Adaptado de (Novillo & Guaño, 2012)*

Además, el set de entrenamiento se constituye de 2094 patrones entre ellos se encuentran; tráfico normal y tráfico con anomalías, cada patrón representa un análisis realizado por Snort con el módulo de entrenamiento y se compone por cuatro líneas; la primera línea contiene un comentario con los nombres de las variables almacenadas por

este módulo, en la siguiente línea están los valores de las variables mencionadas en el acápite anterior, es necesario aclarar que las cantidades de: *hits_as_dst, hits_as_src, ack_rst_resp, rh/has* y *rh/had* se encuentran normalizadas entre 1 y 0 usando la Ec 5, misma que se calcula con tres valores: máximo (val_max), mínimo (val_min) y de la variable a normalizar (var_norm). Aquí es necesario indicar que los valores, tanto máximo como mínimo no son expuestos por el autor del set de entrenamiento, razón por la cual se decide tomarlos tal como se encuentran en la documentación. En tanto que los valores de las variables *av_rcv_time* y *av_snd_time,* se obtienen por el cálculo de la media en cada una de ellas, dicho en otras palabras, se suma los tiempos de todas las peticiones y se los divide para el número de peticiones capturados en cada patrón, ya sean de recepción o remisión, dependiendo de la variable.

$$var\_norm = \frac{var\_norm - val\_min}{val\_max - val\_min} \qquad \textbf{Ec. 5}$$

En cuanto a la tercera línea, es otro comentario clasificando los tipos de tráficos, es decir, tráfico norma o tráfico con anomalía; y la última línea presenta dígitos que identifican cada uno de estos, siendo 1 0 para el tráfico normal y 0 1 para el tráfico con anomalía. La Figura 50 es ejemplo de tráfico normal puesto que presenta valores normales en los tiempos promedio de transmisión (av_rcv_time, av_snd_time), también tiene un valor de cero en la variable ack_rst_resp.

```
#In: hits_as_dst    hits_as_src     av_rcv_time     av_snd_time     ack_rst_resp    rh/has      rh/had
    1.000000        1.000000        0.046602        0.046602        0.000000        1.000000    1.000000
#Out: Normal o atacante o atacado
    1 0
```

Figura 50. *Tráfico Normal*

Fuente: *Adaptado de (Bedon, 2020)*

Por otro lado, la Figura 51 es un ejemplo de tráfico anormal debido que las variables citadas en el ejemplo anterior se encuentran con valores fuera de los establecidos.

```
#In: hits_as_dst    hits_as_src     av_rcv_time     av_snd_time     ack_rst_resp    rh/has      rh/had
    0.117057       0.464968        0.001858        0.000922        1.000000        0.479452    1.000000
#Out: Normal o atacante o atacado
    0 1
```

Figura 51. *Tráfico con anomalía*

Fuente: *Adaptado de (Bedon, 2020)*

Para finalizar este diseño se entrena la red MLP creada (véase Figura 49) con el uso del set de entrenamiento cargado en JavaNNS; el proceso consiste en configurar los parámetros de acuerdo al anexo 4 y así empezar con el entrenamiento, en cuanto lo haga los números de los pesos y de todas las capas tanto entradas, ocultas como salidas varían y esto se debe a que la red MLP posee retroalimentación, lo que ocasión que los valores en ella se modifiquen conforme a la configuración de sus parámetros iniciales, la variación es aleatoria con respecto a lo siguiente: los valores en los pesos fluctúan entre -12 y 12, este rango provoca un entrenamiento más acelerado, ya que como indica la sección 2.6, en una RNA se suman todas las multiplicaciones entre pesos y entradas para obtener una salida, de forma que los pesos afectan de manera directa al resultado en la RNA provocando una disminución en el lapso de tiempo para su entrenamiento.

Por otro lado, los valores del resto de capas oscilan únicamente en el rango de 0 a 1 por la normalización realizada en el set de entrenamiento; esto continua hasta que la red MLP aprenda cuando el tráfico presenta anomalías y cuando no. Dicho en otras palabras, el entrenamiento finaliza en el momento que la red presenta 0 y 1 o valores próximos a estos en sus capas de salidas, tal como muestran los resultados de la Figura 52, mismos que confirman un 99% de certeza en la red MLP y la razón es porque sus capas de salida son de 0 y 0.999.

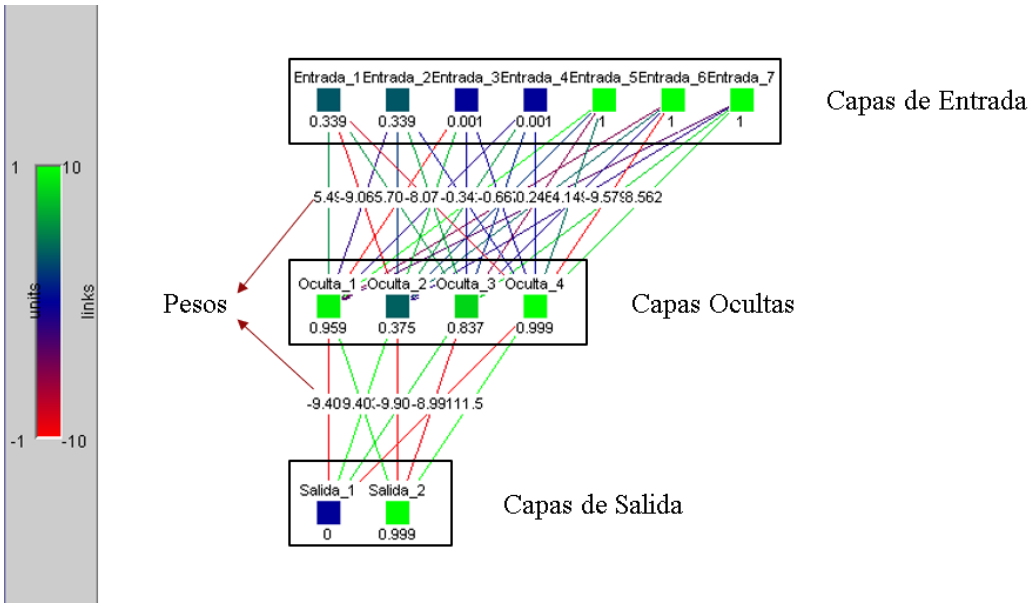


Figura 52. *Red MLP entrenada*

Fuente: *Adaptado de (Bedon, 2020)*

Para obtener los pesos de la red MLP entrenada, en JavaNNS se guarda esta red con formato .net, este archivo se lo abre en Notepad++ con el propósito de observar la información de la red MLP entrenada, la Figura 53 contiene los valores de los pesos, incluyendo las capas que se relacionan a ellos, para mejor comprensión se las dividen entre; capas origen (target_src) y capas destino (target_dst); las target_src se encuentran separadas de los pesos por dos puntos (:), en tanto que las target_dst se ubican en la columna de la izquierda.

Red MLP:

target_dst	target_src : peso
8	1 : 5.49060, 2 : -1.69696, 3 : -10.18235, 4 : -1.20383, 5 : 9.77381, 6 : -4.98751, 7 : -3.27252
9	1 : -9.06317, 2 : 2.00491, 3 : 7.20834, 4 : 5.87302, 5 : 1.99658, 6 : 3.81068, 7 : -1.50080
10	1 : 5.70754, 2 : 6.25378, 3 : -0.30728, 4 : 1.42349, 5 : -5.13755, 6 : -0.88458, 7 : 8.34217
11	1 : -8.07394, 2 : -0.34257, 3 : -0.66740, 4 : 0.24634, 5 : 4.14917, 6 : -9.57955, 7 : 8.56291
12	8 : -9.40204, 9 : 9.90298, 10 : 8.99104, 11 : -11.49687,
13	8 : 9.40369, 9 : -9.90499, 10 : -8.99199, 11 : 11.50027,

Nomenclatura:

target_src	Nombre_Capa
1	Entrada_1
2	Entrada_2
3	Entrada_3
4	Entrada_4
5	Entrada_5
6	Entrada_6
7	Entrada_7
8	Oculta_1
9	Oculta_2
10	Oculta_3
11	Oculta_4

target_dst	Nombre_Capa
8	Oculta_1
9	Oculta_2
10	Oculta_3
11	Oculta_4
12	Salida_1
13	Salida_2

Figura 53. *Pesos obtenidos de la red MLP*

Fuente: *Adaptado de (Bedon, 2020)*

El propósito de los pesos presentados en el párrafo anterior, es usarse en la codificación del módulo preprocesador que se encuentra programado en lenguaje C, la Figura 54 presenta la sección de los pesos codificados en un array de valores tipo flotante; luego de la codificación se consigue un módulo preprocesador con RNA (preprocesador RNA) listo para la integración a Snort, el cual se compone por dos archivos; spp_portscanai.c y spp_portscanai.h, la diferencia en ambos es su función ya que el primero contiene información de lo que realiza el módulo, en tanto que el segundo contiene información del módulo que será compartida con otros archivos usados por Snort.

```
/* Pesos definidos */
static float Weights[] =  {
5.490600, -1.696960, -10.182350, -1.203830, 9.773810, -4.987510, -3.272520,
-9.063170, 2.004910, 7.208340, 5.873020, 1.996580, 3.810680, -1.500800,
5.707540, 6.253780, -0.307280, 1.423490, -5.137550, -0.884580, 8.342170,
-8.073940, -0.342570, -0.667400, 0.246340, 4.149170, -9.579550, 8.562910,
-9.402040, 9.902980, 8.991040, -11.496870,
9.403690, -9.904990, -8.991990, 11.500270,
};
```

Figura 54. *Pesos establecidos en preprocesador*

Fuente: *Adaptado de (Bedon, 2020)*

Ahora es posible la integración del preprocesador RNA a Snort, Bedon (2020) menciona que para ello es necesario configurar cuatro archivos del IDS, estos son: snort.conf, plugbase.c, Makefile.in y flow_callback.c. La Tabla 23 presenta información de cada uno los archivos configurados, incluyendo nombre, ubicación del archivo dentro de la carpeta de instalación y función que cumple.

Tabla 23

Archivos configurados en Snort

Nombre	Ubicación	Función
snort.conf	etc/	Carga parámetros de los módulos al momento de ejecutar un análisis.
plugbase.c	src/	Llama a las funciones de los preprocesador que componen a Snort.
Makefile.in	src/preprocessors/	Carga la información de los módulos durante la instalación de Snort.
flow_callback.c	src/preprocessors/flow	Analiza los flujos (conexiones) capturados durante el análisis del IDS.

Fuente: *Adaptado de (Bedon, 2020)*

Se debe agregar la configuración realizada en cada archivo, con el objetivo de entender cómo se completa la integración del preprocesador RNA a Snort y se obtiene el Snort+RNA presto para instalar; estas configuraciones se explican en los siguientes literales.

a. En *snort.conf* se agregan parámetro bajo los que opera el proprocesador RNA durante un análisis del IDS, estos son tres: el primero es *ignorebc*, mismo que indica si el preprocesador RNA toma en cuenta o no a las IPs de broadcast para identificar las anomalías en el tráfico de red, otro parámetro es net_topology, en él se indica que la RNA a usar es MLP y el último parámetro es log_method, el cual dice si se quiere generar o no, un registro del tráfico capturado al

finalizar el análisis del IDS. La Figura 55 indica de mejor manera estos parámetros.

```
#######################################
#######Snort-AI Preprocessors#########
#######################################
# PortscanAI: Detects portscans using Artificial Intelligence
# ------------------------------------------------------------
#
# The PortsanAI is based on Flow and uses an Artificial Neural Network to take
# its decisions. The flow preprocessor must be enabled.
#
# The following options are available for portscanai
#    ignorebc <1|0> - the preprocessor ignores broadcast traffic and IP address
#                     with its last octet equal to zero (0). 0 for take all IPs,
#                     1 to ignore bcast/network (default).
#    net_topology <0> - Which neural network to use. Now the preprocessor has
#                         Multi-Layer Perceptron (MLP). Put 0 for
#                         MLP (default).
#
# Log generation options:
#
#    log_method <0|1> - PortscanAI can generate useful log information that can be
#                         analyzed with the PHP Console.
#                         0 for disable logging, 1 for logging in files. Default is 0.

preprocessor portscanai: ignorebc 1 \
                         net_topology 0 \
                         log_method 1
```

Figura 55. *Configuración de snort.conf*

Fuente: *Adaptado de (Bedon, 2020)*

b. El segundo archivo configurado es plugbase.c, en él únicamente se incluye el spp_portscanai.h para que pueda ser llamado cuando el Snort+RNA lo necesite, la Figura 56 indica como se realiza este proceso.

```
/* built-in preprocessors */
#include "preprocessors/spp_rpc_decode.h"
#include "preprocessors/spp_bo.h"
#include "preprocessors/spp_stream4.h"
#include "preprocessors/spp_stream5.h"
#include "preprocessors/spp_arpspoof.h"
#include "preprocessors/spp_perfmonitor.h"
#include "preprocessors/spp_httpinspect.h"
#include "preprocessors/spp_flow.h"
#include "preprocessors/spp_sfportscan.h"
#include "preprocessors/spp_frag3.h"

/*Snort-AI preprocessors*/
#include "preprocessors/spp_portscanai.h"
```

Figura 56. *Configuración de plugbase.c*

Fuente: *Adaptado de (Bedon, 2020)*

c. En lo que respecta a Makefile.in, se agrega la información del preprocesador RNA con el propósito de que sea incluido al momento de la instalación de Snort+RNA y así pueda ejecutarse cuando el IDS lo necesite. La Figura 57 detalla esta configuración.

```
CONFIG_HEADER = $(top_builddir)/config.h
CONFIG_CLEAN_FILES =
LIBRARIES = $(noinst_LIBRARIES)
ARFLAGS = cru
libspp_a_AR = $(AR) $(ARFLAGS)
libspp_a_LIBADD =
am_libspp_a_OBJECTS = spp_arpspoof.$(OBJEXT) spp_bo.$(OBJEXT) \
    spp_rpc_decode.$(OBJEXT) spp_stream4.$(OBJEXT) \
    snort_stream4_session.$(OBJEXT) snort_stream4_udp.$(OBJEXT) \
    stream_ignore.$(OBJEXT) spp_perfmonitor.$(OBJEXT) \
    perf.$(OBJEXT) perf-base.$(OBJEXT) perf-flow.$(OBJEXT) \
    perf-event.$(OBJEXT) sfprocpidstats.$(OBJEXT) \
    spp_httpinspect.$(OBJEXT) snort_httpinspect.$(OBJEXT) \
    spp_flow.$(OBJEXT) portscan.$(OBJEXT) spp_sfportscan.$(OBJEXT) \
    spp_frag3.$(OBJEXT) str_search.$(OBJEXT) spp_stream5.$(OBJEXT) \
    stream_api.$(OBJEXT) spp_portscanai.$(OBJEXT) \
(...)
SUBDIRS = flow HttpInspect Stream5
libspp_a_SOURCES = spp_arpspoof.c spp_arpspoof.h spp_bo.c spp_bo.h \
spp_rpc_decode.c spp_rpc_decode.h  \
spp_stream4.c spp_stream4.h stream.h \
snort_stream4_session.c snort_stream4_session.h \
snort_stream4_udp.c snort_stream4_udp.h \
stream_ignore.c stream_ignore.h \
spp_perfmonitor.c spp_perfmonitor.h \
perf.c perf.h \
perf-base.c perf-base.h \
perf-flow.c perf-flow.h \
perf-event.c perf-event.h \
sfprocpidstats.c sfprocpidstats.h \
spp_httpinspect.c spp_httpinspect.h \
snort_httpinspect.c snort_httpinspect.h \
spp_flow.c spp_flow.h \
portscan.c portscan.h \
spp_sfportscan.c spp_sfportscan.h \
spp_frag3.c spp_frag3.h \
str_search.c str_search.h \
spp_stream5.c spp_stream5.h \
stream_api.c stream_api.h \
spp_portscanai.c spp_portscanai.h
```

Figura 57. *Configuración de Makefile.in*

Fuente: *Adaptado de (Bedon, 2020)*

d. Finalmente, en flow_callback se agrega la información del preprocesador RNA con el fin de que este pueda hacer uso de la información de los flujos capturados durante el análisis del IDS, la Figura 58 muestra la configuración de este archivo.

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <stdio.h>
#include <time.h>


#include "packet_time.h"

#include "flow_callback.h"
#include "flow_cache.h"

/* needed for flow stats callback */
#include "flow_stat.h"

/* portscan detector */
#include "portscan/flowps_snort.h"

/*AI Preproc*/
#include "spp_portscanai.h"
```

Figura 58. *Configuración de flow_callback.c*

Fuente: *Adaptado de (Bedon, 2020)*

4.5. Instalación y funcionamiento de Snort+RNA

Antes de empezar con la instalación es necesario aclarar que el Snort+RNA con el que se cuenta ahora posee un esquema de funcionamiento similar al original, la diferencia con el esquema original (véase Figura 44) es que este emplea un preprocesador RNA para la detección de anomalías en los análisis de tráfico en tiempo real, automatizando en cierta forma el proceso de análisis ya que el administrador del IDS no necesita configurar las reglas que se encargan de esto; la Figura 59 muestra el nuevo esquema de funcionamiento que se acaba de describir.

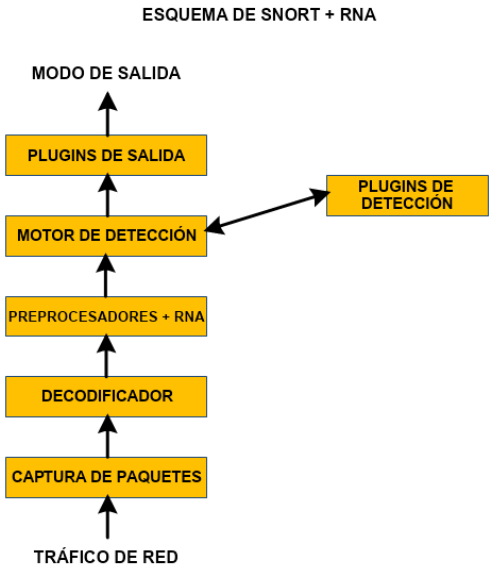


Figura 59. *Esquema de funcionamiento Snort+RNA*

Fuente: *Adaptado de (Sáenz & Martínez, 2018)*

En cuanto a la instalación de Snort+RNA, se especifica en el anexo 5 y es necesario dejar claro que la presente propuesta emplea la versión de Bedon (2020), razón por la cual los archivos usados se obtienen de https://sourceforge.net/projects/snort-ai/, al igual que la documentación necesaria para realizar la instalación. Por otro lado, este proceso se efectúa en una máquina con características específicas de software y hardware; la parte del software necesita el sistema operativo Centos minimal en la versión 7; mientras que el hardware se basa a las recomendaciones de CentOS (2020), de este modo, las características de hardware empleadas son:

- Memoria Ram 2 GB
- Disco Duro 100 GB
- Dos interfaces de red

Acerca del funcionamiento de Snort+RNA, empieza por el análisis, mismo que permite detectar las posibles anomalías que aparezcan en la red; si el IDS detecta una anomalía genera alertas, la Figura 60 es un ejemplo y en ella que se indican algunos datos

importantes como: certeza, protocolo de comunicación, IP y puerto tanto de origen como destino.

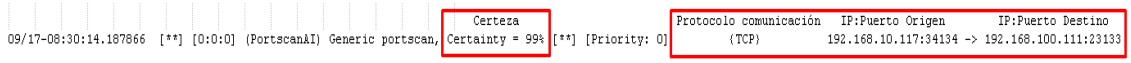


Figura 60. *Ejemplo de alerta en Snort+RNA.*

Fuente: *Autoría propia*

La *certeza* indica un porcentaje que identifica si la alerta generada es real o no, dicho en otras palabras, si la alerta generada presenta un valor inferior al señalado en el diseño de la red MLP (99%) se considerada falso positivo; el *protocolo de comunicación* representa al protocolo que los host usan para la comunicación, estos pueden ser TCP o UDP; en cuanto a las *IPs y puertos* mostradas en la alerta, identifican la IP y puerto donde inicia el ataque (Origen) y la IP y puerto hacia donde se dirige (Destino).

Una última característica en el funcionamiento de Snort+RNA, es su registro de tráfico, el cual se genera al finalizar cada análisis con las teclas Ctrl + C y se encuentra formado por variables que requiere el IDS para identificar las anomalías, estas variables son: *hits_as_src*, *hits_as_dst*, *av_rcv_time*, *av_snd_time*, *win_count* y *rel_hits*; cada una de ellas se definen en la Tabla 24, denotando su utilidad y los valores que debe presentar el tráfico de la red para considerarse anormal, esta cualidad también ayuda en la verificación falsos positivos ya que si la alerta generada se relaciona con valores normales del tráfico, no se puede considerar como anomalía. El registro es presentado en una página web a la que se accede a través de un navegador web con la dirección http://192.168.100.100/tesis/file y los pasos a seguir para la configuración de ella se exponen desde el anexo 6.1 hasta el anexo 6.4.

Tabla 24

Variables analizadas por Snort+RNA

Variable	Definición
hits_as_dst	(hits as destination – hits como destino): Número de paquetes de inicio de conexión (flows) en los cuales esta dirección IP figuraba como destino. Es decir, numero de intentos de conexión que la IP recibió. Un valor superior a 10^3 en este parámetro puede significar que la IP está siendo víctima de un escaneo.
hits_as_src	(hits as source – hits como origen): Número de paquetes de inicio de conexión (flows) en los cuales esta dirección IP figuraba como origen. Es decir, numero de intentos de conexión enviados por la IP. Un valor superior a 10^3 de este parámetro puede significar que el host poseedor de dicha IP está realizando un escaneo.
av_rcv_time	(average receive time – tiempo promedio de recepción): tiempo (segundos) promedio entre peticiones de conexión recibidas por esta IP. Esta medida dice con qué frecuencia están llegando peticiones de conexión a un equipo. Un valor inferior a 10^{-3} de este parámetro puede significar que este host está siendo escaneado.
av_snd_time	(average send time – tiempo promedio de envío): tiempo (segundos) promedio entre peticiones de conexión hechas por esta IP. Esta medida me dice con qué frecuencia un equipo está enviando peticiones de conexión. Un valor inferior a 10^{-3} puede indicar que el host está realizando un escaneo.
win_count	(contador de ventanas): Este valor indica cuantas veces inician las mediciones, cada medición consta de 1599 conexiones en las que una IP es marcada como destino.
rel_hits	(relation hits – hits de relación): Número de peticiones de inicio de conexión que se realizan entre dos host dados. Este parámetro sería superior a 10^3 en el caso de un escaneo uno a uno.

Fuente: *Adaptado de (Bedon, 2020)*

4.6. Prueba de Funcionamiento de Snort+RNA

Esta sección consiste en probar el funcionamiento de Snort+RNA, para ello se elabora un escenario de pruebas en base a la topología de la Figura 61, la cual consiste en tener 3 hosts configurados; el primero contiene a Kali Linux como atacante, el segundo es un servidor web como víctima y el último es Snort+RNA, que se ubica entre el atacante y la víctima. De esta forma, el atacante cumple dos acciones que son la tratar de vulnerar a la víctima y la de realizar simples consultas a la misma, mientras que la función de

Snort+RNA es detectar cuando el atacante realiza las distintas acciones a partir de los resultados obtenidos en cada análisis del IDS; esto quiere decir que se ejecuta un análisis independiente para las dos acciones del atacante. De este modo se presentan los siguiente literales con el proceso a seguir para el cumplimiento de las pruebas de funcionamiento.

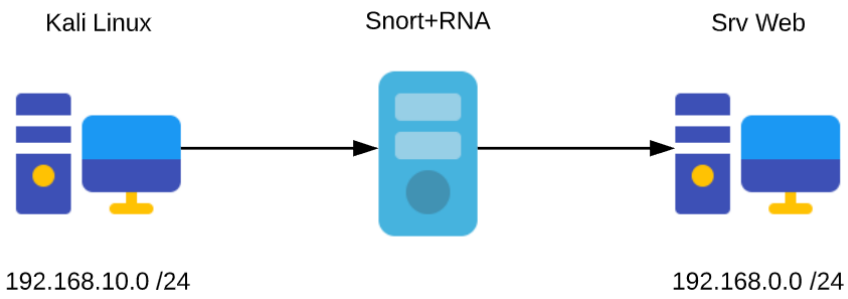


Figura 61. *Topología para prueba de Snort+RNA.*

Fuente: *Autoría propia*

a. Identificar las herramientas del atacante.

La máquina atacante usa dos herramientas: la primera es para tratar de vulnerar a la víctima y la segunda únicamente realiza consultas. Para tratar de vulnerar a la víctima se utiliza la herramienta Hping3, misma que se caracteriza por permitir ocultar la IP del atacante y desde esta inundar con paquetes de diferentes tipos a un host en específico (la víctima); en tanto que para las consultas al servidor web se emplea la herramienta JMeter, su cualidad es realizar pruebas de rendimiento a un servicio específico, en este caso el servicio es web.

b. Usar la herramienta de vulneración.

En este paso, lo primero es iniciar el análisis de Snort+RNA, para lo ello se necesita el comando:

```
snort -i ens32:ens33 -A console -c /etc/snort/snort.conf -l /var/log/Snort
```

En el comando anterior se identifican las interfaces a analizar (*ens32:ens33*), se establece que las alertas serán mostradas mediante la consola de Snort+RNA (*-A console*), además es necesario establecer la dirección en la que se ubica el archivo que contiene las configuraciones de ejecución de Snort (*-c /etc/snort/snort.conf*) y finalmente se señala la dirección en donde se guarda el registro del tráfico capturado una vez que el proceso del IDS termine (*-l /var/log/snort*). La Figura 62 es una captura de la consola de Snort+RNA ya iniciada.

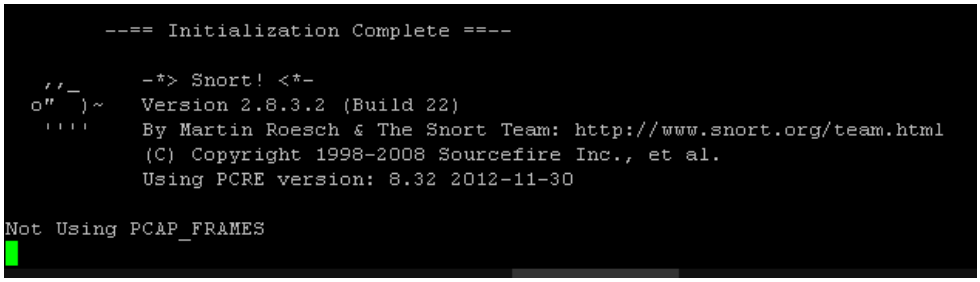


Figura 62. *Inicio de Snort+RNA.*

Fuente: *Autoría propia*

A continuación la Figura 63 muestra la ejecución de Hping3 y explica que la IP 10.10.10.10 inunda de paquetes de sincronización (Sync) al puerto 80 de la IP 192.168.100.112.

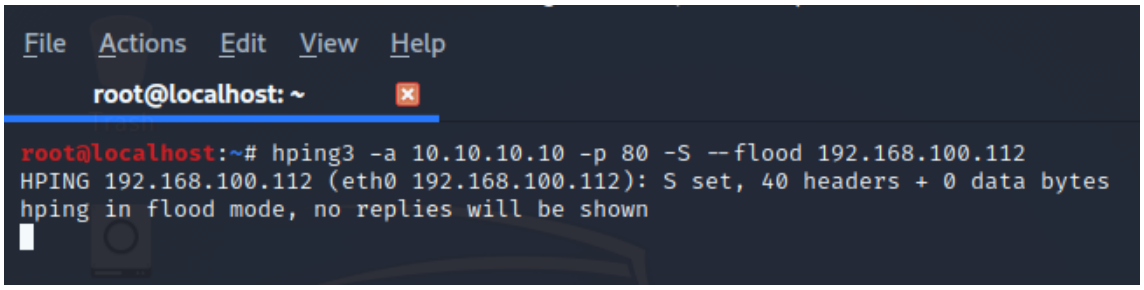


Figura 63. *Ejecución de Hping3*

Fuente: *Autoría propia*

En este momento Snort+RNA empieza a detectar el ataque de Hping3, es así como genera alertas con la información mencionada en la sección 4.5; la certeza del 99%

clasifican estas alertas como reales; en la Figura 64 se encuentran parte de las notificaciones generadas.

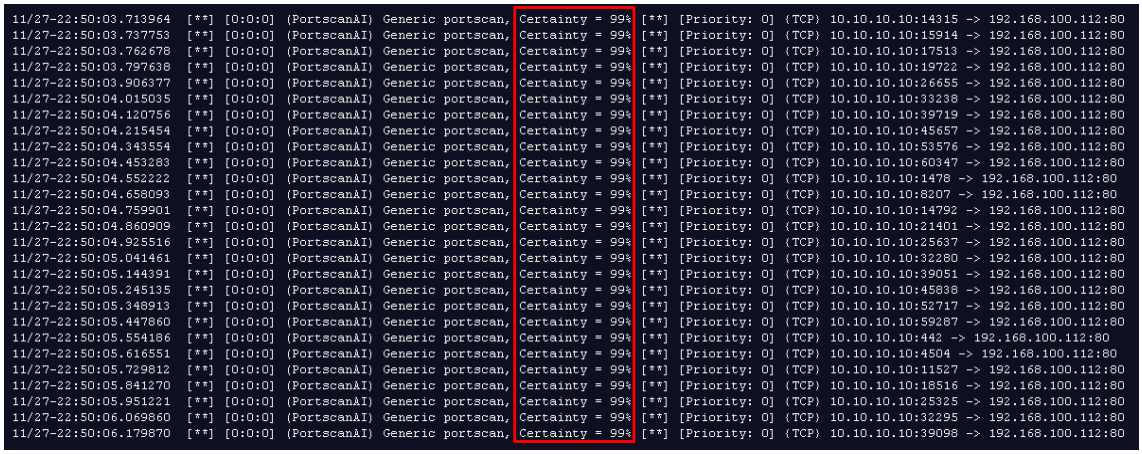


Figura 64. *Alertas en Snort+RNA de Hping3.*

Fuente: *Autoría propia*

c. Analizar el registro de tráfico del intento de vulneración.

Para analizar el registro del tráfico primero se detiene el proceso de Snort+RNA, enseguida se ingresa la dirección que lleva a su página web y muestra un registro igual a la Figura 65, ahí se buscan las IPs 10.10.10.10 y 192.168.100.112 para examinar la información, la cual determina que efectivamente se realizó el ataque debido a que los valores en sus variables son superiores a lo que se indica en la Tabla 24.

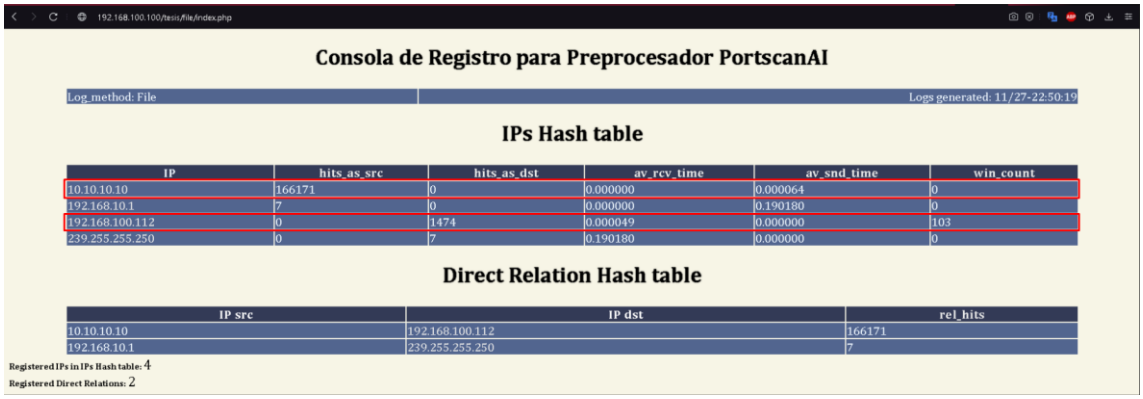


Figura 65.*Registro de tráfico para Hping3.*

Fuente: *Autoría propia*

d. Usar herramienta de consultas al servicio web.

En cuanto a la segunda herramienta, lo primero es ejecutar Snort+RNA con el comando usado en el literal b de esta sección, luego se establecen los parámetros de consulta en JMeter igual a los de la Figura 66, estos señalan que se realiza 60 consultas en un lapso de 10 segundos y esto se repite una sola vez.

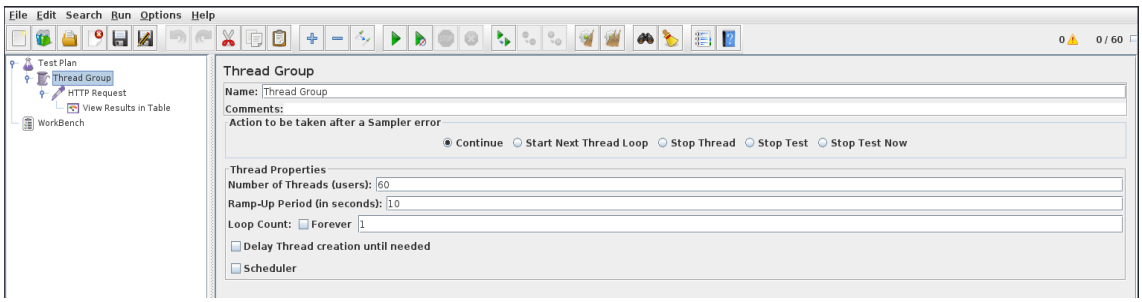


Figura 66. *Configuración de JMeter.*

Fuente: *Autoría propia*

El proceso de JMeter se ejecuta tres veces y al termina se observa que Snort+RNA genera un falso positivo, porque la alerta posee un 97% de certeza tal como presenta la Figura 67, además que marca como origen del ataque la IP del servidor web y como destino la IP del atacante, razón por la cual es necesario acudir al registro del tráfico para verificar esta alerta.

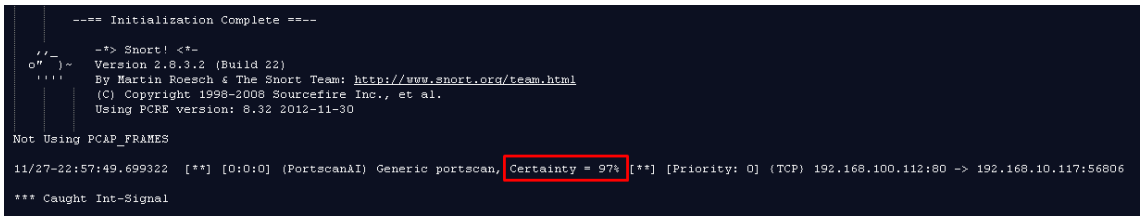


Figura 67. *Alerta en Snort+RNA de JMeter.*

Fuente: *Autoría propia*

e. Analizar el registro de tráfico de las consultas al servicio web.

Este análisis es similar al que se realiza en el literal c en cuanto al acceso al registro del tráfico, sin embargo, sus resultados son diferentes, la Figura 68 muestra el registro

que esta vez genera Snort+RNA, en él se buscan las IPs 192.168.100.112 y 192.168.10.117, las cuales presentan valores normales según señala la Tabla 24. Esto quiere decir que la notificación generada fue un falso positivo, no obstante en las tres veces que se ejecutan las consultas únicamente se genera una notificación lo que indica que Snort+RNA aprende que ese tráfico es normal, caso contrario hubiese una alerta por cada proceso de consulta.

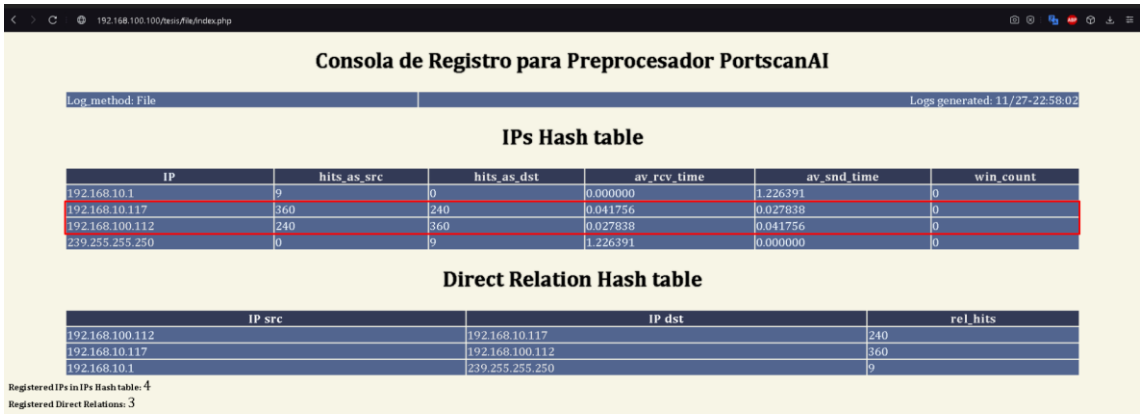


Figura 68. *Registro de tráfico para JMeter.*

Fuente: *Autoría propia*

4.7. Integración de Snort+RNA a SDN

Esta sección se centra en la integración de Snort+RNA a la SDN, para empezar su proceso se plantea el diseño de la Figura 69, el cual consiste en conectar un host físico al servidor físico PV2 por medio de sus interfaces de red ethernet y eno2; el host físico contiene a Kali Linux y Snort+RNA en dos máquinas virtuales distintas, mientras que el servidor físico proxmox (PV2) aloja parte de la infraestructura hiperconvergente que sirve como plataforma para la SDN.

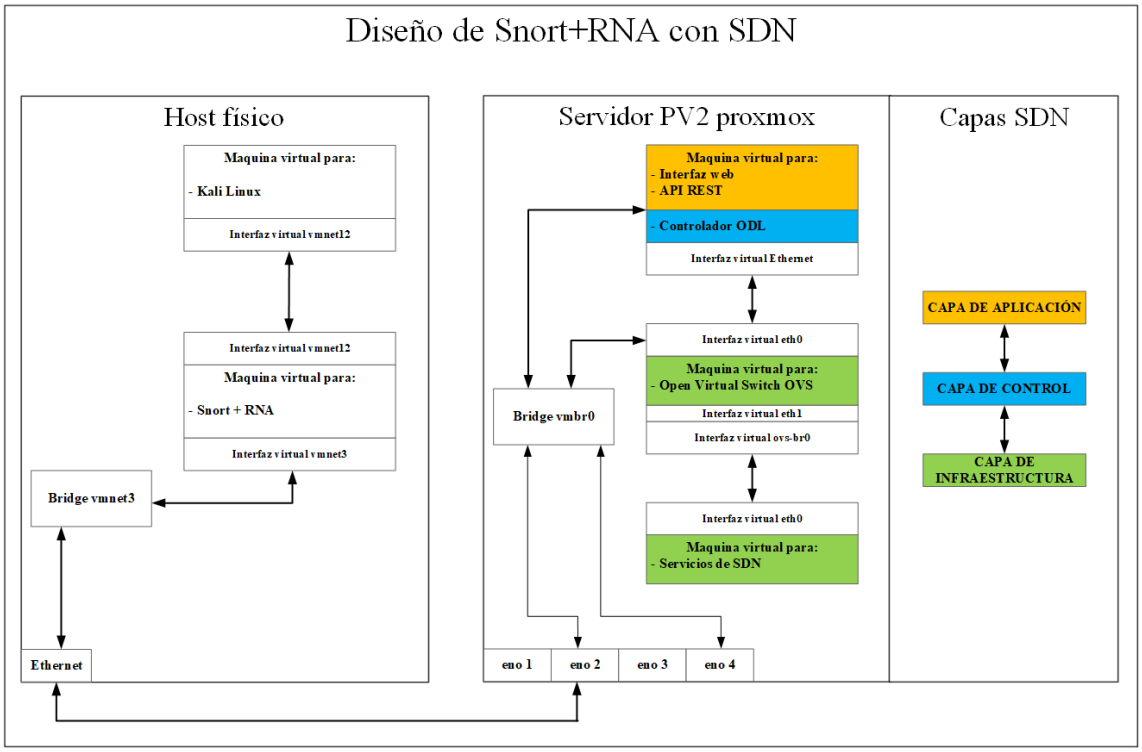


Figura 69. *Diseño de Snort+RNA con SDN.*

Fuente: *Autoría propia*

Se debe agregar a la integración un direccionamiento IPv4 debido a que esto facilita la convergencia en toda la red, de esta manera cada host tiene la obligación de configurar su respectiva IP de acuerdo con la Tabla 25.

Tabla 25

Asignación de IPs en IDS y SDN.

Host	IP	Máscara de red	Puerta de enlace
Kali Linux	192.168.10.117	24	192.168.10.129
Snort+RNA	192.168.10.129	24	
	192.168.100.100	24	192.168.100.110
OVS	192.168.100.110	24	192.168.100.100
	192.168.150.10	24	

ODL	192.168.100.111	24	192.168.100.110
SRV-WEB	192.168.150.11	24	192.168.150.10
SRV-FTP	192.168.150.12	24	192.168.150.10
SRV-MOODLE	192.168.150.13	24	192.168.150.10

Fuente: *Autoría propia*

Lo siguiente es la configuración de las interfaces de red tanto en el servidor físico proxmox PV2 como en el host físico, cada proceso se la clasifica en dos etapas distintas; con esto dicho, la primera etapa consiste en preparar la interfaz de red (eno2) a la cual se conecta el host físico. Esta configuración se ejecuta en la interfaz gráfica del segundo servidor físico de proxmox PV2, la interacción con dicha interfaz es mediante acceso remoto desde otro host que disponga de un navegador web y se encuentre en la misma red del PV2. El proceso detallado se encuentra en el anexo 7.1, sin embargo; en este apartado se menciona el resultado de la configuración, la misma que radica en incluir a la interfaz de red a usar dentro de un puente virtual (vmbr0), la función del vmbr0 en proxmox es que varios clientes virtuales se comuniquen con una sola interfaz de red física que este caso sería lo ideal para la integración. En la Figura 70 se verifica que eno2 se encuentra dentro de vmbr0, estando lista para su uso.

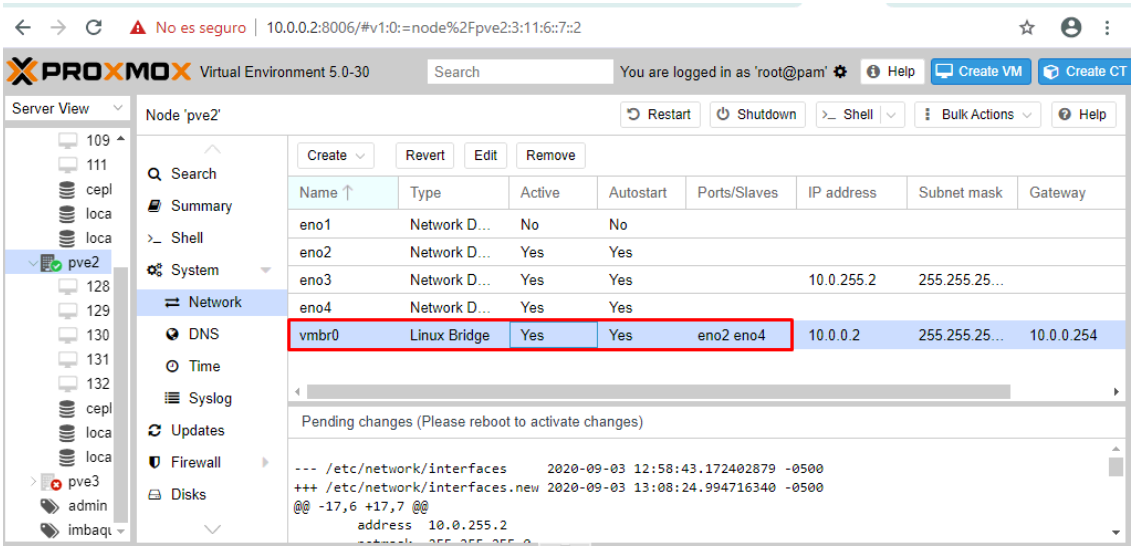


Figura 70. *Interfaz de red dentro del puente en pv2*

Fuente: *Autoría propia*

En tanto que la segunda etapa es similar a la primera con la diferencia que la configuración de la interfaz de red se la realiza en VMware ya que es el software que contiene a la máquina atacante y a Snort+RNA propuesto en este trabajo de titulación, es así que de la misma forma que antes, se coloca a la interfaz que conecta este host físico con el servidor físico PV2 dentro de un puente virtual, la Figura 71 indica como debería quedar la configuración mencionada en este acápite, no obstante, en el anexo 7.2 es posible visualizar este proceso con más detalles.

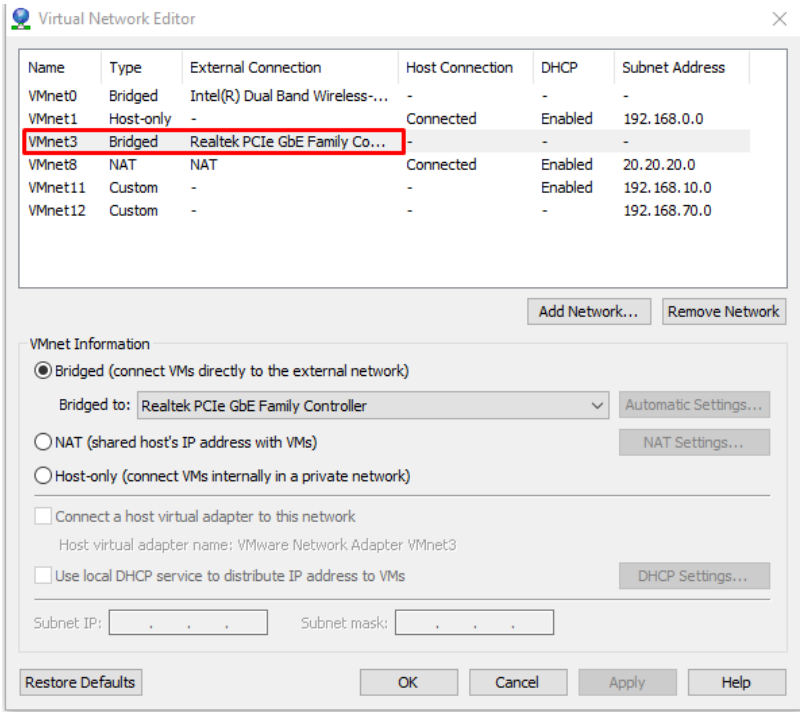


Figura 71. *Interfaz de red dentro del puente en VMware*

Fuente: *Autoría propia*

En cuanto las interfaces de red se encuentren configuradas, es usa el OVS para cumplir dos funciones; la *primera función* es la comunicación entre las máquinas alojadas dentro de Proxmox y el host físico. La *segunda función* que lleva a cabo es la comunicación entre los servidores virtuales y el controlador ODL de la SDN.

El proceso que se efectuó dentro del OVS se lo puntualiza en el anexo 2, en tanto que la Figura 72 describe las interfaces ya configuradas usando el comando `ip add`, ahora solo basta conectar un patch cord al host y el servidor físicos PV2.

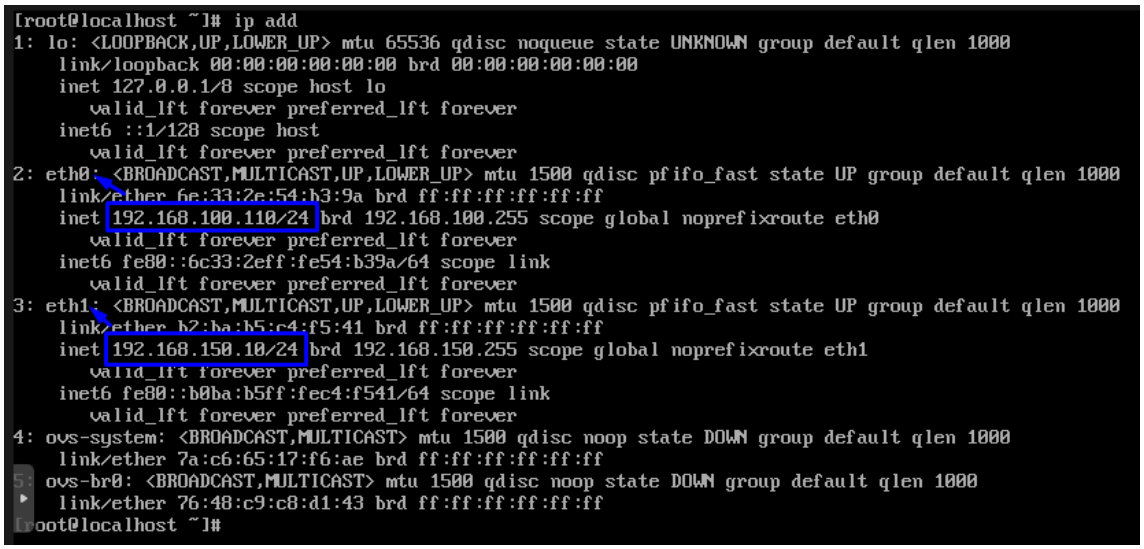


Figura 72. *Interfaces configuradas de OVS*

Fuente: *Autoría propia*

Para determinar que la integración fue exitosa se realiza una prueba de ping entre el host y las máquinas alojadas dentro de PV2, la Figura 73 señala el proceso del ping que en este caso se realizó desde Kali Linux hacia el OVS y al servidor web (srv-web), demostrando así que Snort+RNA se encuentra integrado a la SDN.

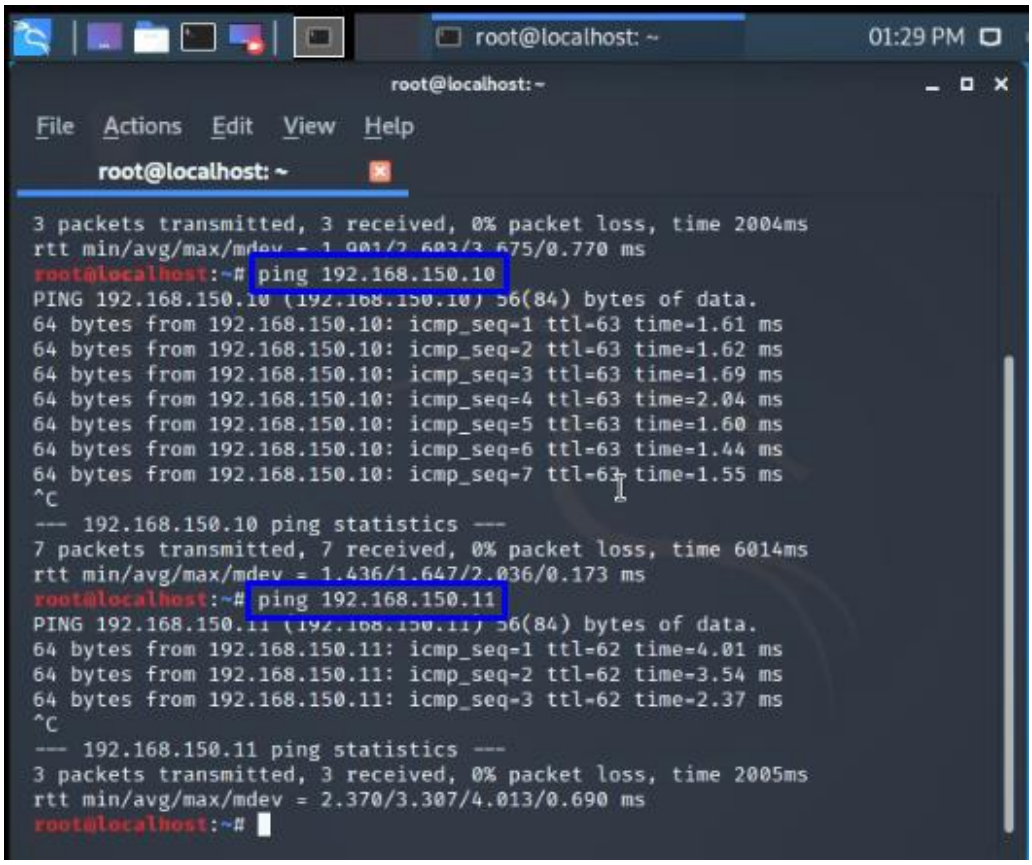


Figura 73. *Prueba ping de Kali Linux hacia OVS y srv-web*

Fuente: *Autoría propia*

El resultado final de la integración es una topología similar a la Figura 74, en la que se presenta la conexión entre el host y el servidor PV2 físicos, incluyendo su direccionamiento de red IPv4 con su respectiva mascara de red.

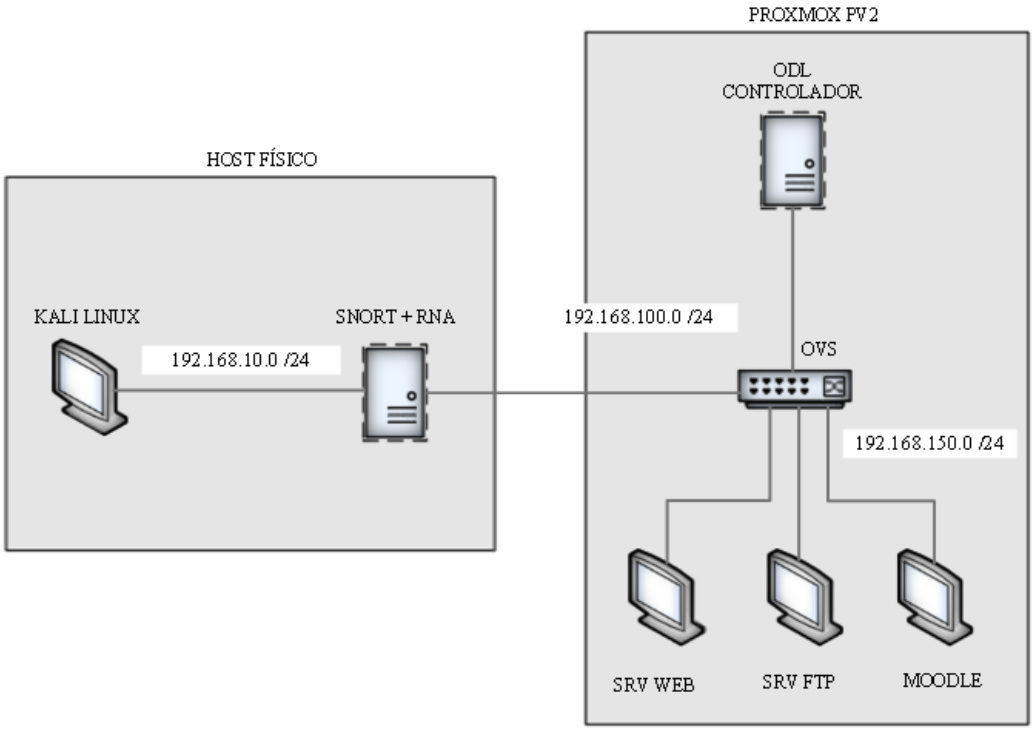


Figura 74. *Topología de Snort+RNA con SDN.*

Fuente: *Autoría propia*

CAPÍTULO 5.

PRUEBAS DE FUNCIONAMIENTO

Este capítulo final consiste en el desarrollo de pruebas de funcionamiento del sistema propuesto en el presente trabajo de titulación, el cual ofrece una herramienta para la seguridad de la información denominada Snort+RNA, misma protege a la SDN contra ataques de tipo activos, dicha protección aprovecha el modelo PDCA que menciona el estándar ISO/IEC-27001, como guía para la seguridad en la información que circula por la misma.

5.1. Modelo PDCA

El modelo PDCA (por sus siglas en inglés Plan-Do-Check-Act) es usado para gestionar la seguridad de la red, en este caso la SDN propuesta. Además, el modelo está compuesto por cuatro etapas: Planificar (Plan), Hacer (Do), Verificar (Check) y Actuar (Act); en donde con cada etapa mencionada se comprueba el funcionamiento del diseño de Snort+RNA planteado en el presente trabajo de titulación. Dicho de otra manera, el modelo PDCA expone el proceso adecuado que permitirá determinar si la herramienta implementada para la seguridad de la información (Snort+RNA) funciona y para ello se puso a prueba con ataques a la red de tipo activos, esto con el fin de generar anomalías que sean alertadas por la herramienta de seguridad.

La etapa de *planificar* consiste en indicar los motivos necesarios para brindar seguridad en una SDN, en este sentido se destacan la selección de los diferentes elementos que lo compone tanto software como hardware, determinando que este tipo de red es centralizada, razón por la cual expone al controlador (Opendaylight) a ataques informáticos debido que administra el comportamiento del resto de su infraestructura. Es así como esta propuesta presenta al Snort+RNA como método de protección, de igual manera que en la SDN aquí se selecciona los elementos que la componen teniendo en

cuenta cuales son los más idóneos para este propósito. Todo lo que se menciona en este párrafo se lo realiza en el capítulo 2 y 3 de este trabajo de titulación.

A continuación la etapa *hacer* pretende implementar todo lo planteado en la primer etapa, en otras palabras, todo lo que se planificó, es así como se centra en la implementación del Snort+RNA en SDN, teniendo en cuenta su funcionamiento y configuraciones necesarias para que cumpla con su objetivo de protección. Es aquí donde se aclara que todo este proceso se encuentra documentado en el capítulo 4.

Por otro lado, la etapa *verificar*, tal como su nombre lo indica, verifica si el Snort+RNA cumple lo que propone, es decir, si el Snort+RNA protege al controlador (Opendaylight) de la SDN. Esta verificación se realiza con ataques de tipo activos y bajo el proceso que proporciona el circulo hacker, en el cual se detallan las herramientas que se usan para cumplir el objetivo de esta etapa. Al finalizar el proceso del circulo hacker se presentan los resultados obtenidos para un posterior análisis.

Finalmente, la etapa de *actuar* consiste en analizar los resultados obtenidos en la etapa anterior y de esta manera sugerir posibles correcciones al IDS o de ser posible recomendar otras herramientas para mejorar la seguridad en esta área (capa control) de la SDN.

5.1.1. Planificar

Como se menciona en el capítulo 2, la SDN permite una administración más eficiente de la red, permitiendo que la comunicación se gestione mediante órdenes que son enviadas al controlador desde APIs y este a su vez se encarga de enviarla a dispositivos de red (router y/o switch) con el fin de que estos últimos realicen la correcta conexión entre dispositivos finales. Así pues, se determina al controlador como punto

débil ante ataques por parte de personas mal intencionadas puesto que si este es vulnerado entonces toda la infraestructura SDN puede colapsar.

Por esta razón el Snort+RNA propone la protección del controlador en la SDN, mas no de los dispositivos finales, en este caso los servidores, puesto que, en el diseño realizado en el capítulo 3, estos al estar dentro de la SDN se comunican con dispositivos de red que trabajan bajo el protocolo openflow, el cual evita que usuarios de redes externas se comuniquen con dichos servidores a menos que se encuentren en la misma red de los servidores.

La Figura 75 detalla el proceso que realiza el sistema propuesto de Snort+RNA en SDN, el cual inicia con un ataque en Kali Linux (host atacante) y finaliza en el registro de tráfico mediante una página web o en la conclusión del ataque. Dicho en otras palabras, significa que si el sistema en mención detecta el ataque originado en Kali Linux, este genera una alerta; es ahí donde el personal encargado de la seguridad de la red tiene la potestad de detener el análisis, para luego dirigirse a observar el registro de tráfico en la interfaz web con el fin de determinar su veracidad o a su vez puede dejar que el sistema continúe analizando en caso de presentarse alertas de otros tipos de ataques. Por el contrario, si el sistema no detecta el ataque, este concluye en la vulneración del controlador ODL de la SDN (host víctima).

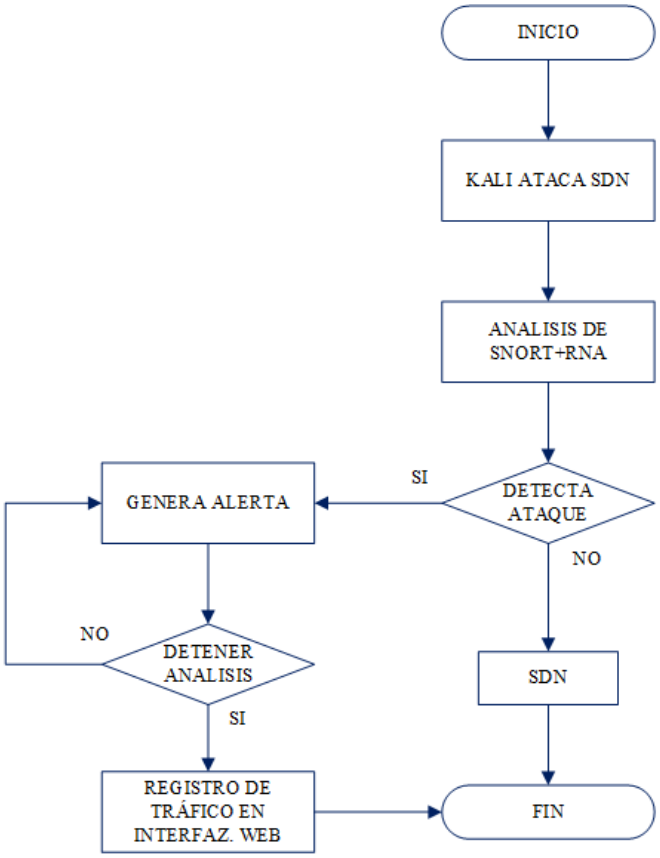


Figura 75. *Diagrama de flujo del proceso de ataque*

Fuente: *Autoría propia*

5.1.2. Hacer

Esta etapa consiste en la ejecución del diseño propuesto (Snort+RNA en SDN), mismo que se encuentra plasmado en el capítulo 4. El capítulo citado abarca la selección del IDS y la red neuronal a usar, posteriormente el diseño y por último la implementación de este, lo que significa que en esta sección únicamente se describe cómo opera el sistema citado; el cual radica en que el Snort+RNA genera alertas en el momento que alguna persona mal intencionada intente vulnerar el servidor que aloja al controlador ODL. Cabe aclarar que el análisis del Snort+RNA es en tiempo real, no obstante, la visualización del registro del tráfico no lo es, por lo que, para visualizar el registro del tráfico analizado durante el ataque es necesario detener el análisis.

Una vez que el sistema haya detenido su análisis, el personal encargado de la seguridad de la red podrá observar mediante una interfaz web si los valores expuestos en dicho registro pertenecen a un tráfico normal o con anomalías tal como lo indica el capítulo 4.

5.1.3. Verificar

La tercera etapa tiene como fin, el verificar el funcionamiento de la herramienta destinada a la seguridad de la SDN y como se menciona al inicio de la sección 5.1, se basa en el circulo hacker, mismo que propone un proceso adecuado para atacar una red o un host en específico, siendo este el controlado ODL. Dicho proceso consta de 5 fases (reconocimiento, exploración, obtener acceso, mantener acceso y borrar huellas) mismas que se aclaran conforme avance el proceso en mención.

Para iniciar las fases del circulo hacker, al host atacante se lo denomina Kali Linux y al host víctima ODL, además se aclara que, durante el ataque informático con las diferentes herramientas, el Snort+RNA ejecuta un análisis por separado y en basado a los resultados arrojados en cada uno de ellos se determina su funcionamiento. Esto se lo hace únicamente por ser un ambiente de pruebas ya que así los resultados de cada ataque informático son más exactos.

5.1.3.1. Reconocimiento

La primer fase busca obtener la mayor cantidad de información posible de la víctima, sin embargo, estas pruebas pretenden obtener únicamente la dirección IP de la víctima y para ello se emplea la herramienta denominada *Nslookup*, misma que tiene la función de traducir direcciones IP de un host en específico, en este caso el ODL, a partir de un dominio, además permite realizar consultas de tipo DNS inversa, es decir, el usuario ingresa una dirección IP y como resultado se obtiene el dominio al que esta pertenece.

Para esta prueba se usa la primera función para conocer la IP del dominio www.controlador.odl.com, para lo cual se digita el comando (`nslookup`) y el dominio (`www.controlador.odl.com`); cabe recalcar que el dominio se encuentra configurado de forma local y no pertenece a algún dominio de la Internet. La Figura 76 indica los resultados arrojados por esta herramienta, en donde, Nslookup muestra que la IP del host perteneciente al dominio www.controlador.odl.com es la **192.168.100.111**, misma que se pretende atacar con diversas herramientas en cada una de las siguientes fases.

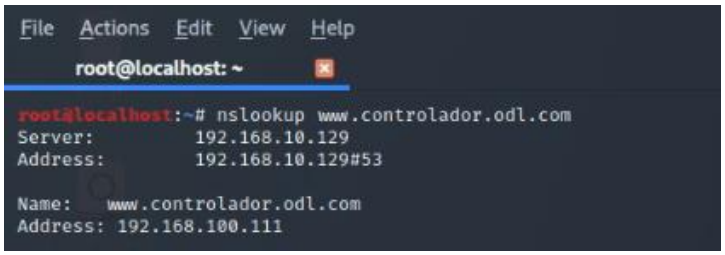


Figura 76. *Resultados de Nslookup*

Fuente: *Autoría propia*

5.1.3.2. Exploración

Aquí se tiene como fin la determinación de las posibles vulnerabilidades que posee la víctima; es así que una vez conocida la IP del ODL se utilizan dos herramientas con funciones similares que son la de escanear los puertos de la víctima, sus nombres son: Sparta y Nmap; la diferencia entre ellas radica en su interfaz de administración, ya que la primera herramienta (*Sparta)* presenta una interfaz gráfica para analizar el ODL, mientras que *Nmap* cumple la misma función, pero mediante líneas de comando.

Una vez que Nmap (Figura 77) terminan su análisis se observa la información detallada acerca del ODL, no obstante; el apartado que se necesita conocer es sobre los puertos que se encuentran sin protección, es decir que se encuentran abiertos, con ella se determina la posibilidad de ejecutar ataques con las herramientas que se presentan en la

siguiente etapa. Es necesario aclarar que en comando de Nmap se indican que todos los puertos serán escaneados (-p-), se trata de conocer la versión del servicio que se encuentra en el puerto identificado (`-sV`) y la IP de la víctima (`192.168.100.111`).

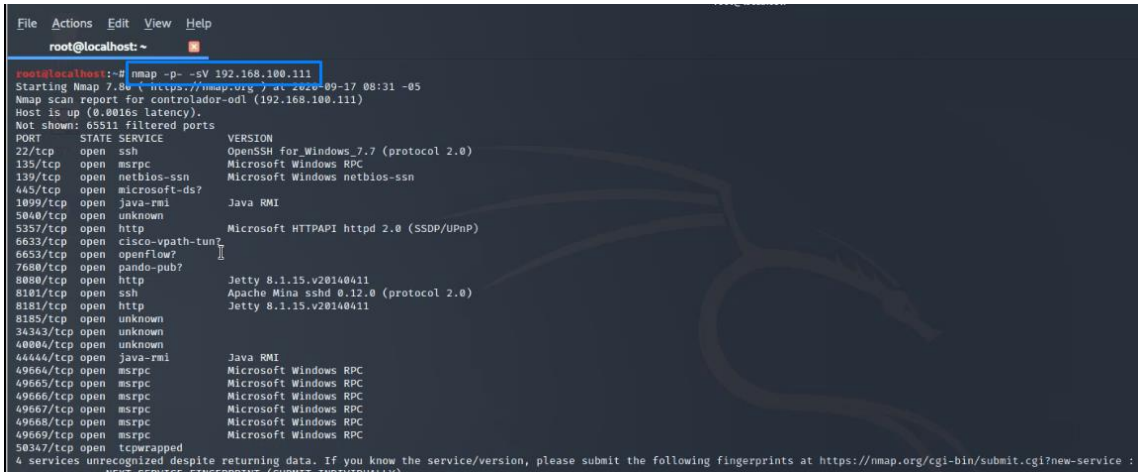


Figura 77. *Resultados de Nmap*

Fuente: *Autoría propia*

Por otro lado, para obtener resultados en Sparta (Figura 78) únicamente se establece la IP de la víctima y se inicia el proceso de escaneo.

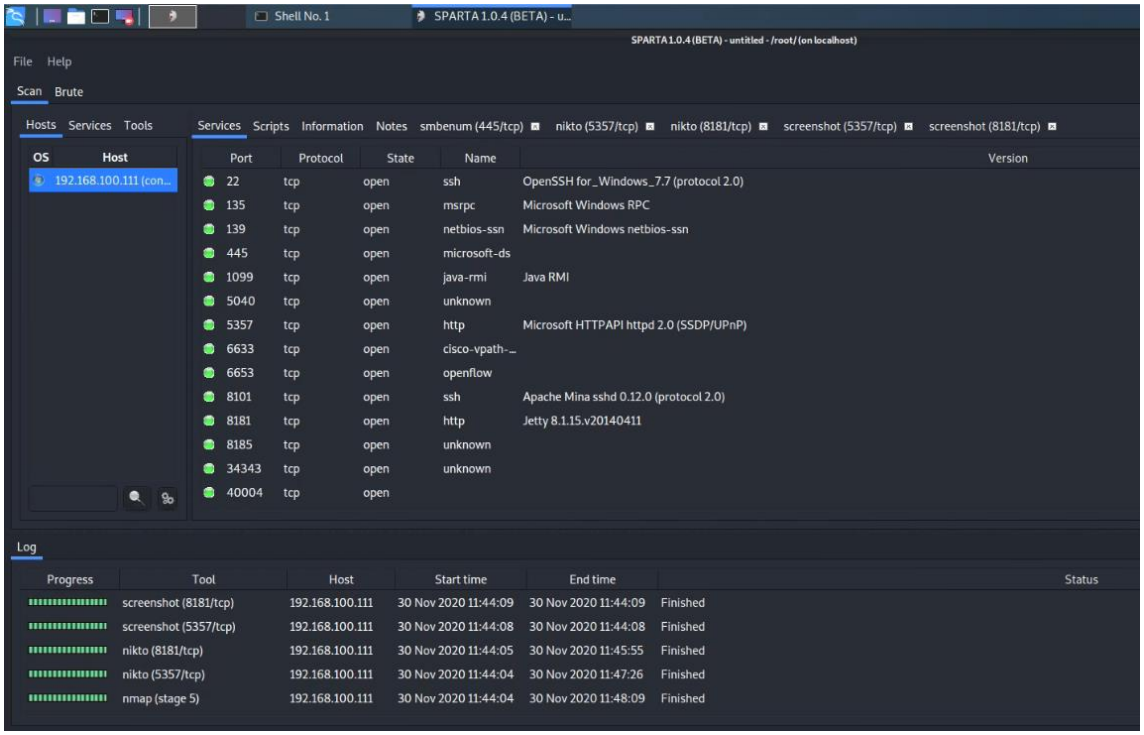


Figura 78. *Resultados de Sparta*

Fuente: *Autoría propia*

Los resultados obtenidos por las dos herramientas muestran que el ODL se encuentra alojado en un sistema operativo Windows, además de poseer varios puertos TCP sin protección, es así como se usan estas vulnerabilidades para obtener acceso a este host.

5.1.3.3. Obtener acceso

En la tercera fase se aprovechan las vulnerabilidades obtenidas de la exploración a la víctima con diferentes propósitos, como por ejemplo; el acceder a la información que se encuentra en ella, modificar su información, tratar de truncar su servicio, entre otros. Por este motivo, tan pronto como se conoce las vulnerabilidades, se empiezan a explotarlas y para ello se usan cuatro herramientas que son: Metasploit, Crunch, Hydra y Hping3. A continuación se detalla cómo se utiliza cada una dentro de las pruebas efectuadas.

Antes de usar la herramienta de Metasploit se verifica si esta es capaz de vulnerar el ODL, para ello es necesario digitar el comando `searchsploit "Windows"` en la terminal del atacante, si el resultado es similar a la Figura 79 quiere decir que la herramienta contiene funciones útiles para el objetivo de esta fase, de esta manera la función que se emplea en esta prueba se denomina reverse_tcp (recuadro rojo).

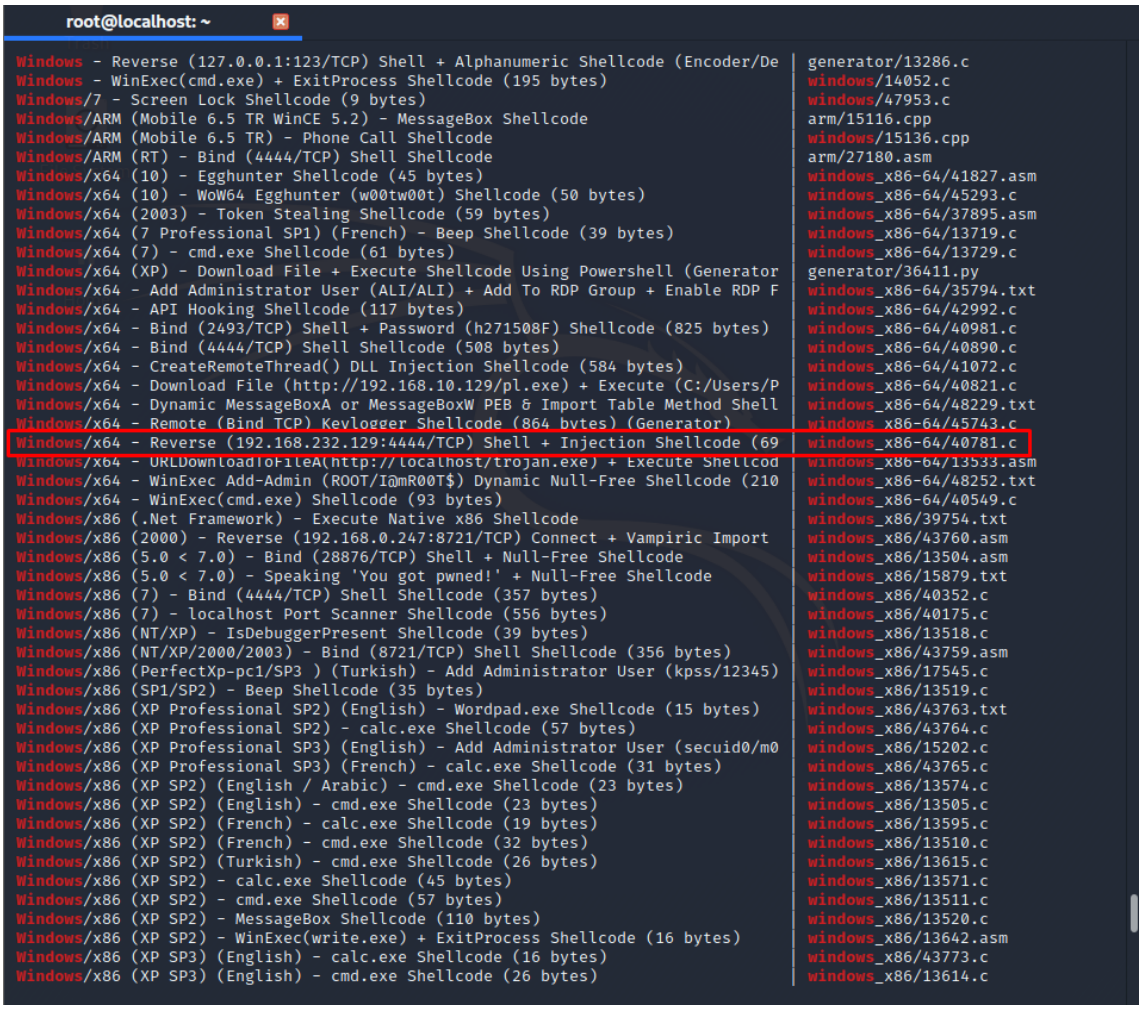


Figura 79. *Resultados de searchsploit*

Fuente: *Autoría propia*

Ahora, el propósito de *Metasploit* es averiguar el nombre de usuario que administra el ODL, esto se logra con la función citada en el anterior acápite (*reverse_tcp*) y consiste en colocar un archivo de payload en el ODL, mismo que se colocó en el host previo a que se ejecute esta herramienta. Además, se configura en Metasploit los parámetros señalados en la Tabla 26; una vez que el archivo payload sea abierto por el usuario administrador, se establecerá la comunicación entre el ODL y Kali permitiendo al atacante tener control sobre la víctima.

Tabla 26

Parámetros de configuración Metasploit

Parámetro	Función
use exploit/multi/handler	Indica que se usan funciones con payload.
set PAYLOAD windows/meterpreter/revers_tcp	Indicar la función que usa metasploit
s**et LHOST 192.168.10.117**	Es la dirección IP del atacante
s**et LPORT 4444**	Establece el puerto de comunicación
run	Inicia el proceso de metasploit

Fuente: *Autoría propia*

En la Figura 80 se indica el proceso para la configuración de los parámetros en la herramienta referida (Metasploit), con los parámetros configurados esta herramienta queda a la espera de que se abra el archivo payload para establecer la comunicación entre la víctima y el atacante.

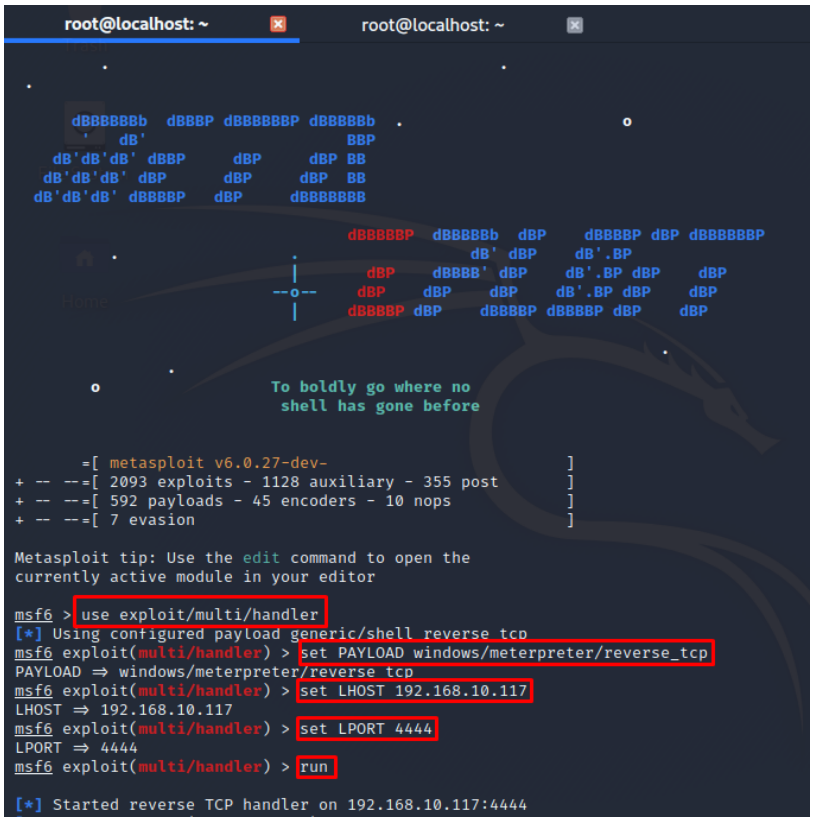


Figura 80. *Proceso de Metasploit*

Fuente: *Autoría propia*

Por otro lado, la Figura 81 denota los resultados de Metasploit, es decir, el proceso llevado a cabo posterior a que el archivo en el ODL se haya abierto. Una vez dentro de la consola de Windows con el comando `shell` basta con digitar `net user` para que se muestren los usuarios que posee este host. Finalmente, con los usuarios obtenidos, se escoge el usuario **Willams** para ser vulnerado, es decir, para obtener su contraseña.

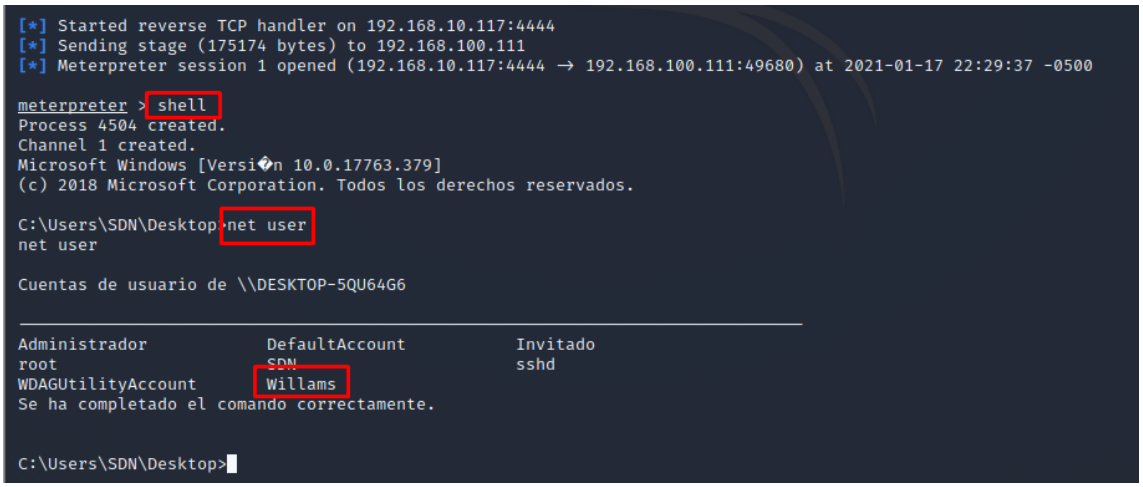


Figura 81. *Resultados de Metasploit*

Fuente: *Autoría propia*

En cuanto se tiene un nombre de usuario en específico y observando en la exploración que el puerto 22 del servicio SSH se encuentra vulnerable, se inicia un ataque para conocer su contraseña; el cual se consigue con dos herramientas: Crunch y Hydra. *Crunch* permite crear diccionarios basados en parámetros programados, mientras que *Hydra* usa estos diccionarios para intentar obtener credenciales (usuario y contraseña) del ODL.

En este caso se intenta obtener únicamente la contraseña para el acceso remoto al usuario llamado Willams, es así como se empieza con la creación de un diccionario denominado **crunch_passwords.txt**. El proceso de ello se encuentra en la Figura 82, únicamente se necesita una línea de comando en la que se indica que el diccionario posee palabras de 5 letras (`crunch 5 5`) usando todo el alfabeto (`abcdefghijklmnopqrstuvwxyz`) y este diccionario se guarda en un archivo de texto (`> crunch_password.txt`).

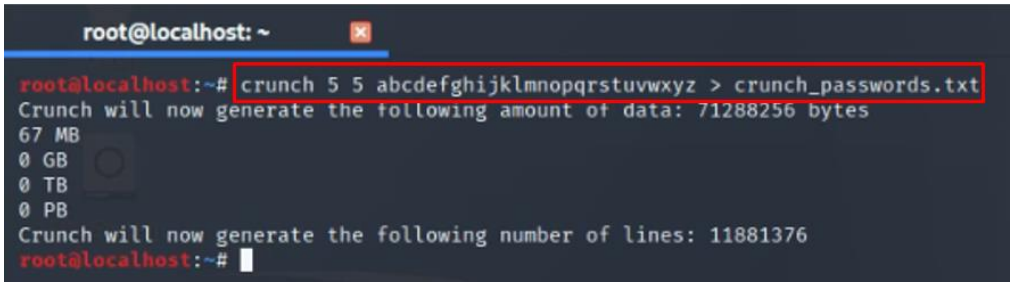


Figura 82. *Creación de diccionario con Crunch*

Fuente: *Autoría propia*

A continuación Hydra usa el diccionario **crunch_passwords.txt** para ejecutar un ataque de fuerza bruta y de esta manera conseguir el password, para ello es primordial ciertos configurar los parámetros que se establecen en la Figura 83; entre ellos están: el nombre del usuario de la víctima (`-l Willams`), el diccionario que se usara para encontrar la contraseña del usuario señalado (`-P /root/Crunch_password`), la dirección IP de la víctima (`192.168.100.111`), cuantos subprocesos se utilizan durante el ataque (`-t 16`), por último se escribe el nombre del servicio que será vulnerado (`ssh`). Al finalizar el ataque de Hydra, la contraseña obtenida con este ataque es **admin**.

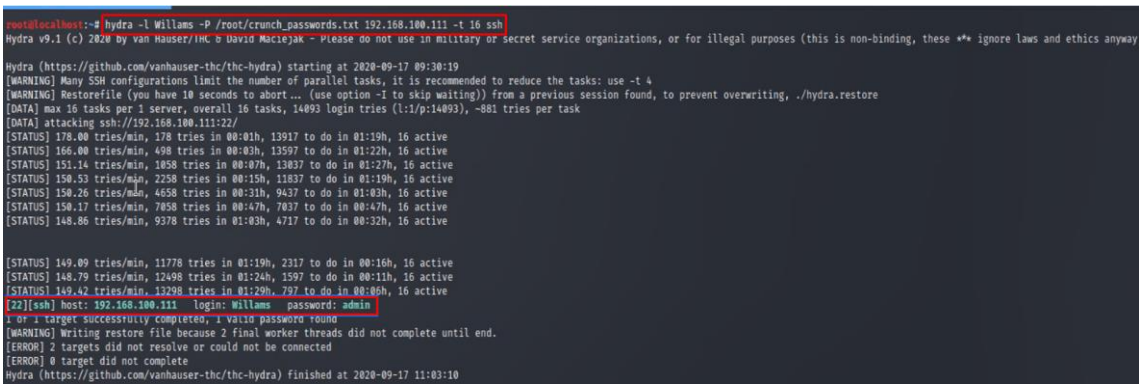


Figura 83. *Resultados de Hydra*

Fuente: *Autoría propia*

Como se observa en el capítulo 3, el controlador de la SDN permite una administración mediante una interfaz web, misma que usa el puerto 8181, convirtiéndose este puerto en otra vulnerabilidad, por este motivo se decide realizar un ataque DOS y para ello se utiliza la herramienta denominada *Hping3* ya que permite enviar gran

cantidad de peticiones de acceso llegando a truncar el servicio que usa el controlador y así evitando que el ingreso a la administración de este. El proceso de Hping3 se presenta en la Figura 84, en el que indica que desde la IP atacante (`-a 10.10.10.10`) se inunda de paquetes Sync al puerto 8181 (`-p 8181 -S --flood`) de la IP víctima (`192.168.100.111`).

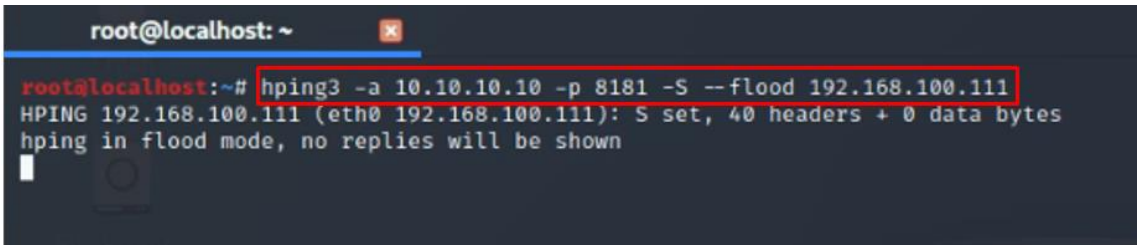


Figura 84. *Proceso de Hping3*

Fuente: *Autoría propia*

En la Figura 85 se demuestra que a pesar del ataque realizado con Hping3 el servidor ODL no presenta Denegación de Servicio (DoS) ya que sigue trabajando con normalidad permitiendo acceder al mismo y sin presentar retraso o lentitud en el acceso a su interfaz web. Sin embargo, el IDS logra detectar las anomalías que esta herramienta genera, los resultados obtenidos se presentan en la sección 5.1.3.5.

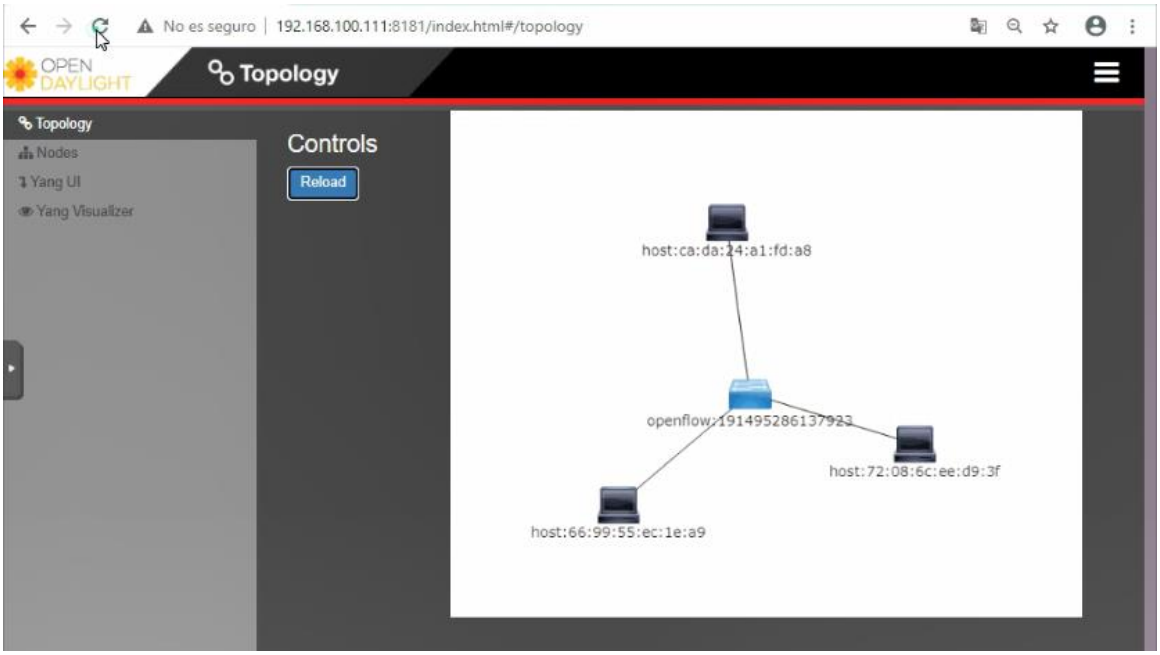


Figura 85. *Interfaz web disponible durante ataque de Hping3*

Fuente: *Autoría propia*

5.1.3.4. Mantener acceso

El propósito que tiene la cuarta fase es permitir al atacante (hacker) acceder a la información de la víctima cada vez que lo requiera y sin necesidad de volver a pasar por las fases anteriores, de modo que la herramienta indicada es un archivo de texto, ya que en él se almacenan las credenciales obtenidas del ataque de Hydra cumpliendo así con el objetivo de mantener acceso, puesto que al final se obtiene un archivo a su completa disponibilidad con las credenciales para el acceso remoto al ODL. Por esta razón no posee mucha complejidad más que transcribir las credenciales en un archivo de texto en este caso llamado credenciales.txt, similar a como se presenta en la Figura 86.

Figura 86. *Almacenamiento de credenciales*

Fuente: *Autoría propia*

5.1.3.5. Borrar huellas

Esta última fase consiste en eliminar los rastros que el atacante pudo dejar a lo largo del proceso efectuado por el circulo hacker; para ello se usan comandos respectivos de Kali Linux, mismos que tienen el propósito de eliminar los registros de los comandos usados en la ejecución de cada una de las herramientas mencionadas a lo largo del proceso del circulo hacker, estos comandos se los describe en la Tabla 27, la cual señala el comando y su función.

Tabla 27

Comandos para borrar huellas

Comando	Función
rm ~/.bash_history -rf	Borrar historial de comandos usados.
history -c	Borrar registro de sesión actual.
kill -9 $$	Cerrar sesiones actuales.

Fuente: *Autoría propia*

5.1.3.5. Resultados del Snort+RNA

En esta última sección de la etapa verificar se plasman los resultados obtenidos con cada una de las herramientas utilizadas dentro del circulo hacker, los cuales serán usados posteriormente para determinar el funcionamiento del Snort+RNA. Cabe aclarar que los resultados expuestos en la Tabla 28 son los obtenidos de los paquetes que circulan a través del Snort+RNA durante los diferentes análisis.

Tabla 28

Resultados del Snort+RNA

	Nslookup	Nmap	Sparta	Metasploit	Hydra	Hping3
Paquetes Recibidos	8	267994	713540	185	95791	4671833
Paquetes analizados	4	267973	560663	185	95791	825335
Paquetes rechazados	0	0	152877	0	0	3846498
Paquetes en espera	4	21	0	0	0	0
Alertas	0	166	307	0	57	516

Fuente: *Autoría propia*

Basados en los resultados presentados anteriormente, es posible concluir que los ataques por parte de las herramientas *nslookup* y *metasploit* no generaron alertas puesto que no presentaron alteraciones en la red. Por otro lado, los ataques de *nmap*, *sparta*,

hydra y *hping3* si generaron alertas ya que estas herramientas provocan un aumento en el tráfico de la red. Otra manera de verificarlo es a través de las alertas generadas por el Snort+RNA, estas se clasifican como reales debido a que indican un 99% de certeza, se toma a la Figura 87 como ejemplo de las notificaciones que se generan durante el ataque de la herramienta nmap.

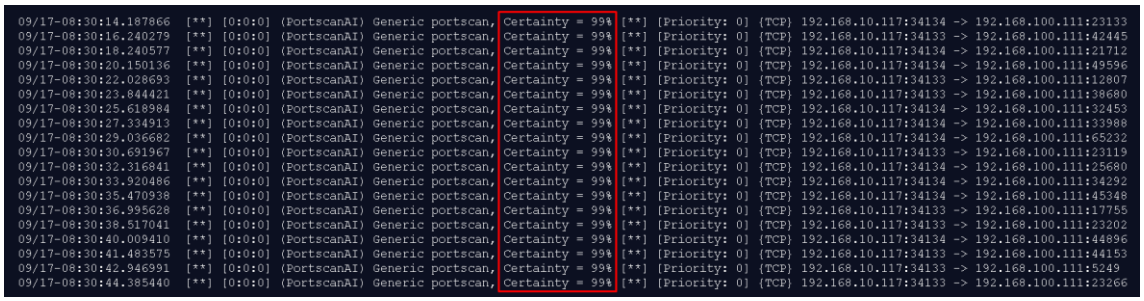


Figura 87. *Alertas de Snort+RNA para nmap.*

Fuente: *Autoría propia*

Cabe agregar lo que menciona la sección 4.5, Snort+RNA posee una interfaz web para observar el registro del tráfico capturado durante los ataques realizados con las herramientas en mención, en los cuales se verifican si las alertas generadas se tratan de ataques reales o de falsos positivos, teniendo como punto de referencia la información que brinda la Tabla 24 del capítulo 4.

El registro para nslookup y metasploit son iguales en cuanto a rangos de valores normales se refiere, sin embargo, se toma como ejemplo al registro de nslookup (Figura 88), en él se observa que los números de paquetes en hits_as_src y hits_as_dst son inferiores a 10^3, igualmente los tiempos en av_rcv_time y av_snd_time son superiores a 10^{-3}, aclarando las razones por el cual el Snort+RNA no genera alertas con este ataque.

Figura 88. *Logs del ataque Nslookup*

Por último, los registros para nmap, sparta, hydra y hping3, muestran valores fuera de los rangos establecidos, de esta manera se acude como ejemplo al registro de hydra (Figura 89), mismo que exponen a hits_as_src y hits_as_dst con números de paquetes superiores a 10^3, es necesario aclarar que para la IP víctima (192.168.100.111) se toma en cuenta a win_count debido a que indica que las mediciones se hicieron 31 veces y en cada medición hubo 1599 hits_as_dst (número de paquetes como destino), asimismo, av_rcv_time y av_snd_time contiene tiempos inferiores a 10^{-3}, y de este modo se verifican que las alertas generadas por el Snort+RNA son de ataques reales.

Figura 89. *Logs del ataque Hydra*

5.1.4. Actuar

La última etapa del modelo PDCA brinda una retroalimentación sobre la eficiencia de la propuesta presentada en la etapa de planificación, en este caso el Snort+RNA en SDN; para ello se analizan los resultados obtenidos en la sección 5.1.3.5 y es posible determinar que el sistema no es capaz de examinar todos los paquetes que recibe cuando estos empiezan a transmitirse en gran cantidad, tomando como ejemplo a los resultados de la herramienta Hping3 y de Hydra.

En el primer caso con Hping3, el Snort+RNA captura un total de 4’671.833 paquetes, de los cuales se analizan 825.335 paquetes (equivalente al 17.67%) mientras que 3’846.498 paquetes (82.33%) se rechazan, dando a entender que el sistema no se encuentra listo para brindar un análisis completo de los paquetes que recibe a partir de herramientas destinadas a ataques DDoS. Por otro lado, están los resultados de Hydra, aquí el Snort+RNA logra analizar 95.791 paquetes capturados (100%), lo cual indica que ante este tipo de ataques de fuerza bruta el sistema es totalmente eficiente en lo que respecta la parte del análisis hacia los paquetes recibidos.

De acuerdo con el anterior acápite, es necesario dejar claro que, aunque el Snort+RNA no puede analizar todo el tráfico que se dirige hacia la SDN; si genera alertas cuando se presentan ataques de tipo activo, los cuales se caracterizan por ocasionar un incremento inusual de tráfico.

Es así como para obtener resultados diferentes es posible ejecutar tres alternativas; la primera alternativa es realizar el mismo diseño con otro set de entrenamiento, mismo que permitirá al sistema mejorar la eficiencia ante ataques de DDoS, la segunda alternativa es hacer un cambio en el tipo de RNA y de esta manera observar los cambios que el IDS presente, en tanto que la tercera alternativa es cambiar de IDS y adaptarlo a la RNA que se propone en este trabajo de titulación.

CAPÍTULO 6.

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

El análisis bibliográfico permite centrar los tópicos más importantes bajo los cuales se encuentra sustentado el presente trabajo de titulación, tanto la parte de hardware como software que conforman el mismo.

La infraestructura hiperconvergente de Proxmox del data center en la FICA, es capaz de apalancar infraestructuras virtuales como la SDN propuesta y de esta manera optar por métodos más eficientes de administración en lo que respecta a los dispositivos de red que se alojan allí.

En base a lo mencionado por Reddy & Annapurani Panaiyappan (2018), se concluye que el controlador (ODL) es parte fundamental de la SDN ya que en él se centra la completa administración de su infraestructura, convirtiéndose en un flanco de ataque para personas mal intencionadas.

La política open source bajo la cual se encuentra diseñado Snort facilitó el uso de esta herramienta puesto que se adaptó sin inconvenientes a esta propuesta que fue la integración con el preprocesador RNA y con este nuevo diseño brindar seguridad al controlador (ODL) de la SDN.

El diseño de Snort+RNA automatizó en cierta forma el proceso de configuración del IDS debido a que el administrador de la red no necesitó configurar reglas para permitir la detección de futuras anomalías suscitadas en la red.

A diferencia del trabajo de Manso, Moura, & Serrão (2019) esta propuesta de Snort+RNA fue poco eficiente al momento de analizar los paquetes recibidos en gran cantidad como por ejemplo, los producidos por ataques DoS; sin embargo, al momento

de analizar paquetes generados mediante ataques de fuerza bruta, la propuesta resultó totalmente eficiente.

Aunque el Snort+RNA no logró analizar todos los paquetes recibidos durante las pruebas de funcionamiento, sí generó alertas cuando se incrementó el tráfico en la red, demostrando que la herramienta propuesta protege al ODL.

El controlador ODL de la SDN fue capaz de soportar un ataque de DoS generado por la herramienta Hping3 puesto que no presentó retardos ni impedimentos en el acceso a su interfaz web durante la ejecución del ataque.

El modelo PDCA de la ISO/IEC 27001 propuesto en este trabajo de titulación resultó una opción flexible al momento de ofrecer seguridad de la información debido a que se acopló perfectamente en la protección de Snort+RNA en ODL.

El proceso del circulo hacker se ajustó a las diferentes herramientas que se plantearon en esta propuesta, ya que se logró cumplir con el objetivo en cada fase (reconocimiento, exploración, obtener acceso, mantener acceso, borrar huellas) que lo conforma.

6.2. Recomendaciones

En caso de que la infraestructura física del data center no disponga dispositivos de red compatibles con el protocolo Openflow, existe la posibilidad de configurar dispositivos de red dentro de una máquina virtual con el propósito de integrarlos a la SDN.

El escenario de la SDN implementado en esta propuesta sirve como base para desarrollar futuros proyectos como: la integración a otras SDN que trabajen bajo el mismo protocolo openflow, implementación de API para balancear la carga en este tipo de red, entre otros.

El informe de registros de tráfico que ofrece esta propuesta se puede mejorar al hacerlo en tiempo real puesto que Snort+RNA necesita detener su análisis para poder tener acceso a dichos registros.

Al momento de brindar seguridad de la información a una red, ninguna herramienta es 100% segura, es por esta razón que no se debe confiar en una sola herramienta para proteger la información de la infraestructura de comunicación contra ataques informáticos.

Esta propuesta puede ser puesta a prueba bajo otras normativas de seguridad de seguridad de la información diferente al de la ISO/IEC con el propósito abarcar otras áreas dentro del ámbito de la seguridad.

La etapas del circulo hacker son un proceso en el cual puede usarse herramientas de hacking distintas a la del presente trabajo de titulación, siempre y cuando cumplan la función de cada una de sus fases.

Las pruebas de funcionamiento de Snort+RNA pueden realizarse bajo diferentes conjuntos de procesos diferentes a los propuestos por el circulo hacker.

BIBLIOGRAFÍA

Acevedo, E., Serna, A., & Serna, E. (2017). *Principios y características de las redes neuronales artificiales*. (March 2019), 348–353.

Albowarab, M. H., Zakaria, N. A., & Abidin, Z. Z. (2018). Software Defined Network: Architecture and Programming Language Survey. *International Journal of Pure and Applied Mathematics*, *119*(18), 561–572.

Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbeh, M. (2017). Evaluation of machine learning algorithms for intrusion detection system. *SISY 2017 - IEEE 15th International Symposium on Intelligent Systems and Informatics, Proceedings*, (Iv), 277–282. https://doi.org/10.1109/SISY.2017.8080566

Añas, E. (2018). *Ataques Informáticos*. San Juan Bautista Cuicatlán.

Bedon, C. (2020). Snort-AI. Retrieved from https://sourceforge.net/p/snort-ai/wiki/Home/

Bholebawa, I. Z., & Dalal, U. D. (2018). Performance analysis of SDN/openflow controllers: POX versus floodlight. *Wireless Personal Communications*, *98*(2), 1679–1699. https://doi.org/10.1007/s11277-017-4939-z

Bhushan, K., & Gupta, B. B. (2019). Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*, *10*(5), 1985–1997. https://doi.org/10.1007/s12652-018-0800-9

Blial, O., Ben Mamoun, M., & Benaini, R. (2016). An Overview on SDN Architectures with Multiple Controllers. *Journal of Computer Networks and Communications*, *2016*. https://doi.org/10.1155/2016/9396525

Bricata. (2020). What is Bro IDS [Zeek]? And Why IDS Doesn't Effectively Describe It [Overview and Resources]. Retrieved from https://bricata.com/blog/what-is-bro-ids/

Caicedo B, E. F., & López S, J. A. (2017). Una aproximación práctica a las redes neuronales artificiales. In *Una aproximación práctica a las redes neuronales artificiales*. https://doi.org/10.25100/peu.64

Calvo, D. (2017). Definición de red neuronal artificial. Retrieved from Definición de red neuronal artificial website: https://www.diegocalvo.es/definicion-de-red-neuronal/

Calvo, D. (2018). Perceptrón Multicapa – Red Neuronal. Retrieved from https://www.diegocalvo.es/perceptron-multicapa/

Camacho, B. (2017). *Sistema de Detección y Prevención de Intrusos para el Control de Vulnerabilidades de la Red Corporatia de la Universidad Regional Autónoma de las Andes "UNIANDES."* UNIANDES.

Cárdenas, R. (2016). *Propuesta de una Solución de Seguridad que Provea una Respuesta Activs frente a Ataques de Denegación de Servicios Basada en la Integración de Herramientas Open Source sobre un Prototipo de Red Jerárquica*.

Carvajal, C. (2015). *Sistema de Detección de Intrusos en Redes mediante el Método Heurístico para el Gobierno Municipal de la Ciudad de Otavalo*. UNIANDES.

CCN. (2018). *Recomendaciones de seguridad para CDN*. Retrieved from http://www.seguridad.unam.mx/documento/?id=1927

CentOS. (2020). CentOS Product Specifications. Retrieved from https://wiki.centos.org/About/Product

Chávez, J. (2016). *Análisis y Modelos de Datos de Redes para Seguridad Informática*. Universidad de Chile.

Cisco. (2019a). *OpenFlow*. Retrieved from https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1611/b_1611_programmability_cg/OpenFlow.pdf

Cisco. (2019b). Software-Defined Networking (SDN) Definition - Cisco. Retrieved November 10, 2019, from https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html?dtid=osscdc000283#~services

Cloudflare. (2020). SYN Flood DDoS Attack | Cloudflare. Retrieved May 15, 2020, from https://www.cloudflare.com/es-la/learning/ddos/syn-flood-ddos-attack/

Cox, B. (2018). Open Source Tripwire. Retrieved from https://github.com/Tripwire/tripwire-open-source

Cuevas-Tello, J. C. (2018). *Apuntes de Redes Neuronales Artificiales*. 1–20. Retrieved from http://arxiv.org/abs/1806.05298

Denazis, S., Hadi Salim, J., Meyer, D., & Koufopavlou, O. (2015). *Software-Defined Networking (SDN): Layers and Architecture Terminology* (E. Haleplidis & K. Pentikousis, Eds.). https://doi.org/10.17487/rfc7426

Díaz, M. (2016). Redes neuronales recurrentes y series temporales. Retrieved from http://software-tecnico-libre.es/es/articulo-por-tema/todas-las-secciones/todos-los-temas/todos-los-articulos/redes-neuronales-recurrentes-y-series-de-tiempo

Domínguez L., H., Maya O., E., Peluffo O., D., & Crisanto Ñ., C. (2017). Aplicación de técnicas de fuerza bruta con diccionario de datos, para vulnerar servicios con métodos de autenticación simple “Contraseñas”, pruebas de concepto con software libre y su remediación. *Maskana*, *7*(Supl.), 87–95.

Domínguez, M., Maya, E., Meneses, S., Rosero, P., Narvaez, S., Zambrano, M., & Peluffo, D. (2020). Design and Tests to Implement Hyperconvergence into a DataCenter: Preliminary Results. *Advances in Intelligent Systems and Computing*,

1066, 54–66. https://doi.org/10.1007/978-3-030-32022-5_6

El computador 2020. (2020). Que es servidores web? - El computador 2020. Retrieved May 26, 2020, from https://elcomputador.org/tecnologia/que-es-servidores-web/

ESAN. (2016). La norma ISO 27001 y la mejora continua en la gestión de seguridad de la información | Tecnología | Apuntes empresariales | ESAN. Retrieved May 19, 2020, from https://www.esan.edu.pe/apuntes-empresariales/2016/05/norma-iso-27001-mejora-continua-en-la-gestion-de-seguridad-informacion/

Grediaga, A., Ibarra, F., Ledesma, B., & Brotons, F. (2016). Utilización de redes neuronales para la detección de intrusos. *Primer Congreso Iberoamericano de Seguridad*, (February), 13. Retrieved from http://www.ua.es/tia%0ASee%09discussions,%09statshttps://www.researchgate.net/publication/26740053 8

Guerrero, D. (2019). *Metodología para desarrollar un sistema de gestión de la calidad aplicado al data center de la Facultad de Ingeniería en Ciencias Aplicadas bajo la norma ISO 9001:2015*. 282.

Herrera, T. (2015). Sistema de detección de intrusos IDS. Retrieved from https://slideplayer.es/slide/5412864/

Iren, I., Fonseca, L., Lau Fernández, R., Superior, I., José, P., & Echeverría, A. (2017). *Detección de Intrusos en la red utilizando redes neuronales. Resultados Preliminares*. Retrieved from https://www.researchgate.net/publication/242286517_Deteccion_de_Intrusos_en_la_red_utilizando_redes_neuronales_Resultados_Preliminares

ISO. (2013). ISO/IEC 27001:2013. *Information Technology — Security Techniques — Information Security Management Systems — Requirements*, *2014*(ISO/IEC

27001:2013), 38.

ISO 27001. (2020). ISO 27001 - Certificado ISO 27001 punto por punto - Presupuesto Online. Retrieved May 19, 2020, from https://normaiso27001.es/

ISOTools. (2016). 12 Beneficios de Implantar un SGSI de acuerdo a ISO 27001 - ISOTools Chile. Retrieved May 19, 2020, from https://www.isotools.cl/12-beneficios-de-implantar-un-sgsi-de-acuerdo-a-iso-27001/

Jácome, V. (2019). *PLAN DE SEGURIDAD PARA LA GESTIÓN DE RIESGOS EN EL DATACENTER DE LA FACULTAD DE INGENIERA EN CIENCIAS APLICADAS CON LA METODOLOGÍA MAGERIT V3.0*.

Jirón, I. (2015). *Redes Neuronales Artificiales Aplicadas a la Seguridad de la Información*. 1–28.

Kairi, A. (2017). *Study on Intrusion Detection System*. Retrieved from https://www.researchgate.net/publication/316854204_Study_on_Intrusion_Detection_System

Krishnan, P., Duttagupta, S., & Achuthan, K. (2020). SDN/NFV security framework for fog-to-things computing infrastructure. *Software - Practice and Experience*, *50*(5), 757–800. https://doi.org/10.1002/spe.2761

Kumar, G. (2015). Evaluation Metrics for Intrusion Detection Systems-A Study. *International Journal of Computer Science and Mobile Applications*, *2*(11), 11–17. Retrieved from www.ijcsma.com

Lovato, J. (2019). OpenDaylight, Most Pervasive Open Source SDN Controller, Celebrates Sixth Anniversary with Neon Release - The Linux Foundation. Retrieved May 18, 2020, from https://www.linuxfoundation.org/press-release/2019/03/opendaylight-most-pervasive-open-source-sdn-controller-

celebrates-sixth-anniversary-with-neon-release/

Manso, P., Moura, J., & Serrão, C. (2019). SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. *Information (Switzerland)*, *10*(3). https://doi.org/10.3390/info10030106

Mardini, J. (2016). *Instruders detection computer systems model, chisquare, training and classification using shsom, features selection criteria*. (número 1), 24–35.

Moodle. (2020). Acerca de Moodle - MoodleDocs. Retrieved May 12, 2020, from https://docs.moodle.org/all/es/Acerca_de_Moodle#Mundialmente_probado_y_de_confianza

Narváez, C. (2016). *DISEÑO DE LA INFRAESTRUCTURA FÍSICA DE UN DATA CENTER TIER I BASADO EN EL ESTÁNDAR TIA 942, PARA LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS DE LA UNIVERSIDAD TÉCNICA DEL NORTE*.

NMAP.ORG. (2020). Nmap Network Scanning. Retrieved from https://nmap.org/man/es/index.html

Novas Otero, P. (2018). *Redes neuronales aplicadas al criptoanálisis del Advanced Encryption Standard*. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81610/8/pnovasTFG0618memoria.pdf

Novillo, C., & Guaño, M. (2012). *Implementación de un sistema de detección de intrusos utilizando inteligencia artificial*. Escuela Politécnia Nacional.

OBS Business School. (2020). Seguridad de la información, un conocimiento imprescindible. Retrieved May 19, 2020, from https://obsbusiness.school/int/blog-investigacion/sistemas/seguridad-de-la-informacion-un-conocimiento-

imprescindible

Ocampo, C. A., Viviana, Y., Bermúdez, C., & Solarte Martínez, G. R. (2017). Sistema de detección de intrusos en redes corporativas Intrusion Detection System in Corporate Networks. *Scientia et Technica Año XXII*, *22*(1), 122–170.

Oladunjoye, O. (2017). *Software Defined Networking– The Emerging Paradigm To Computer Networking*. 38. Retrieved from https://www.theseus.fi/bitstream/handle/10024/125665/Oladunjoye_Olanrewaju.pdf?sequence=1&isAllowed=y

OnApp. (2020). Install OpenDaylight Controller. Retrieved from https://docs.onapp.com/agm/6.0/network-settings/sdn-management/install-opendaylight-controller#id-.InstallOpenDayLightControllerv6.0-prerequisities

Open Networking Foundation. (2020, March 9). Software-Defined Networking (SDN) - Open Networking Foundation. Retrieved March 9, 2020, from https://www.opennetworking.org/sdn-definition/

OpenDaylight. (2019). OpenDaylight. Retrieved from https://wiki.opendaylight.org/view/Main_Page

OpenDaylight Project. (2016). Welcome to the OpenDaylight Handbook! — OpenDaylight Documentation Beryllium documentation. Retrieved May 26, 2020, from https://docs.opendaylight.org/en/stable-beryllium/

Palacios, F. (2019). La neurona artificial. Retrieved November 13, 2019, from https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x38.html

Pereira, G., & Gamess, E. (2017). Lineamientos para el Despliegue de Redes SDN/OpenFlow. *Revista Venezolana de Computación*, *4*(2), 21–33.

Pérez, E. (2018). Un acercamiento a las Redes Definidas por Software. Arquitectura y beneficios. Retrieved November 10, 2019, from https://www.researchgate.net/publication/326271018_Un_acercamiento_a_las_Redes_Definidas_por_Software_Arquitectura_y_beneficios

Pérez, J. (2017). Artificial Neural Networks Java. Retrieved November 13, 2019, from https://github.com/sidlors/ANNs

PMG SSI. (2016). ISO 27001 Política de Seguridad de la Información. Retrieved May 19, 2020, from https://www.pmg-ssi.com/2016/06/que-debe-incluir-en-su-politica-de-seguridad-de-la-informacion-basado-en-la-norma-iso-27001/

Proxmox. (2019). Open-Source Virtualization Platform. Retrieved from https://www.proxmox.com/en/proxmox-ve

Pública, A., Manuel, J., & Viñas, R. (n.d.). *Ciclo Formativo De Grado Superior: Administración Y Finanzas*. Retrieved from http://www.juntadeandalucia.es/averroes/centros-tic/41008970/helvia/sitio/upload/Admon_Publica.pdf

Quincozes, S. E., Soares, A. A. Z., Oliveira, W., Cordeiro, E. B., Ferreira, V. C., Lopes, Y., … Passos, D. (2019). *Survey and Comparison of SDN Controllers for Teleprotection and Control Power Systems*.

Radware. (2020). Radware: Open Daylight Partnership. Retrieved March 11, 2020, from https://www.radware.com/partners/technologypartners/open-daylight/

Reddy, T. N., & Annapurani Panaiyappan, K. (2018). Intrusion detection on software defined networking. *International Journal of Engineering and Technology(UAE)*, *7*(3.12 Special Issue 12), 330–332. https://doi.org/10.14419/ijet.v7i3.12.16052

Rodriguez, A., & Santibáñez, A. (2018). *Redes de Computadores 1 Ataques en redes*.

Romero, M. I., Figueroa, G. L., Vera, D. S., Álava, J. E., Parrales, G. R., Álava, C. J., … Castillo, M. A. (2018). Mecanismo Correctivos en seguridad informática. In *Introducción a la seguridad informática y el análisis de vulnerabilidades*. https://doi.org/10.17993/ingytec.2018.46

Sáenz, K., & Martínez, K. (2018). *Implementación de una herramienta de detección de intrusiones en la red de área local del Sistema Naciona de Inversión Pública (SNIP)*. 155. Retrieved from http://ribuni.uni.edu.ni/1488/1/80713.pdf

Sahoo, K. S., Mohanty, S., Tiwary, M., Mishra, B. K., & Sahoo, B. (2016). A Comprehensive Tutorial on Software Defined Network. *Proceedings of the International Conference on Advances in Information Communication Technology & Computing - AICTC '16*, 1–6. https://doi.org/10.1145/2979779.2983928

Salunkhe, U. R., & Mali, S. N. (2017). Security Enrichment in Intrusion Detection System Using Classifier Ensemble. *Journal of Electrical and Computer Engineering*, *2017*. https://doi.org/10.1155/2017/1794849

Sánchez Lorente, O. (2015). Detección de intrusiones con Snort. *Universitat Oberta de Catalunya*. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/43090/6/osanchezloTFM0715memoria.pdf

Security Onion. (2020). About Security Onion. Retrieved from https://securityonion.net

Snort. (2020). Snort FAQ. Retrieved from https://www.snort.org/faq

Stalling, W. (2016). *Founfation of Modern Networking SDN, NFV, QoE, IoT and Cloud*.

Stephen, M. (2015). *Machine Learning An Algorithmic Perspective Second Edition*.

Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2019). Survey on SDN based

network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, *12*(2), 493–501. https://doi.org/10.1007/s12083-017-0630-0

Tayyebi, Y., & Bhilare, D. S. (2018). *A Comparative Study of Open Source Network Based Intrusion Detection Systems*. *9*(2), 23–26. https://doi.org/10.1016/S0169-5347(00)02077-2

Tecon. (2020). La Seguridad de la Información - Tecon. Retrieved May 19, 2020, from https://www.tecon.es/la-seguridad-de-la-informacion/

Torruella, J. (2017). *Conocimientos Fundamentales en Ciberseguridad Orientados a desenvolverse de forma*.

Vallejo, C. (2018). Sistemas de Prevención de Intrusos (IDS) en la Gestión de la Información. https://doi.org/978-9942-792-39-6

Varanasi, V. R., & Razia, S. (2019). Intrusion Detection using Machine Learning and Deep Learning. *International Journal of Recent Technology and Engineering*, *8*(4), 9704–9719. https://doi.org/10.35940/ijrte.d9999.118419

Yazdinejad, A., Bohlooli, A., & Jamshidi, K. (2018). Efficient design and hardware implementation of the OpenFlow v1.3 Switch on the Virtex-6 FPGA ML605. *Journal of Supercomputing*, *74*(3), 1299–1320. https://doi.org/10.1007/s11227-017-2175-7

ANEXOS

Anexo 1. Instalación de controlador ODL

Los requisitos para la correcta instalación de controlador ODL son; tener instalado Java Runtime Environment (JRE), que se lo puede obtener del siguiente enlace https://www.java.com/es/download/ y tener el software del controlador en sí, esto se lo obtiene desde este enlace https://bit.ly/2Na9gB2, finalmente se descarga la versión y formato señalados en la Figura 90.

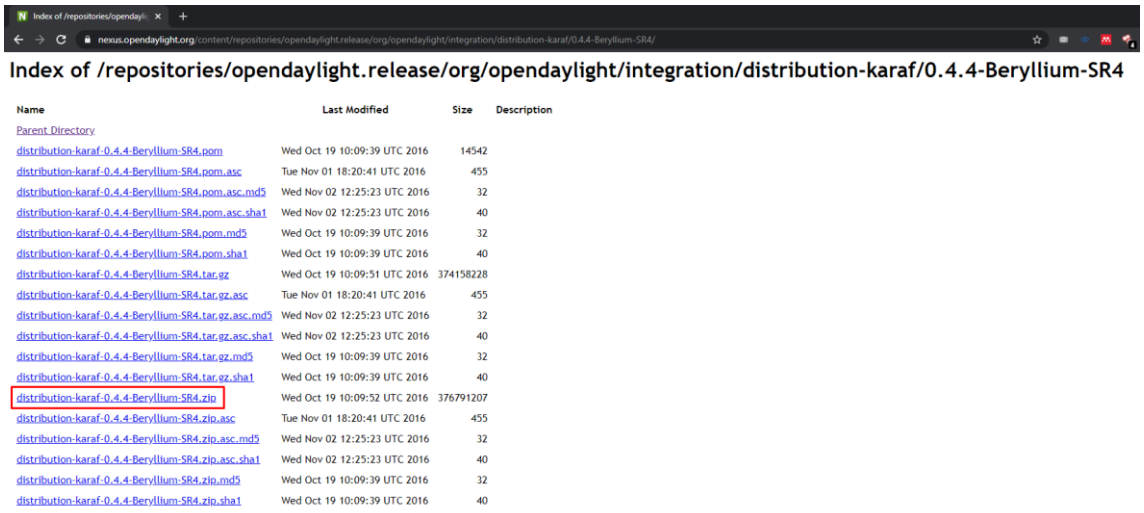


Figura 90. *Página de descarga ODL.*

Cabe recalcar que los diferentes formatos de compresión van a depender del Sistema Operativo en el que se realice la instalación del ODL. Al ser la instalación en Windows 10, se descomprime el archivo distribution-karaf-0.4.4-Beryllium-SR4.zip, quedando los archivos que se presentan en la Figura 91.

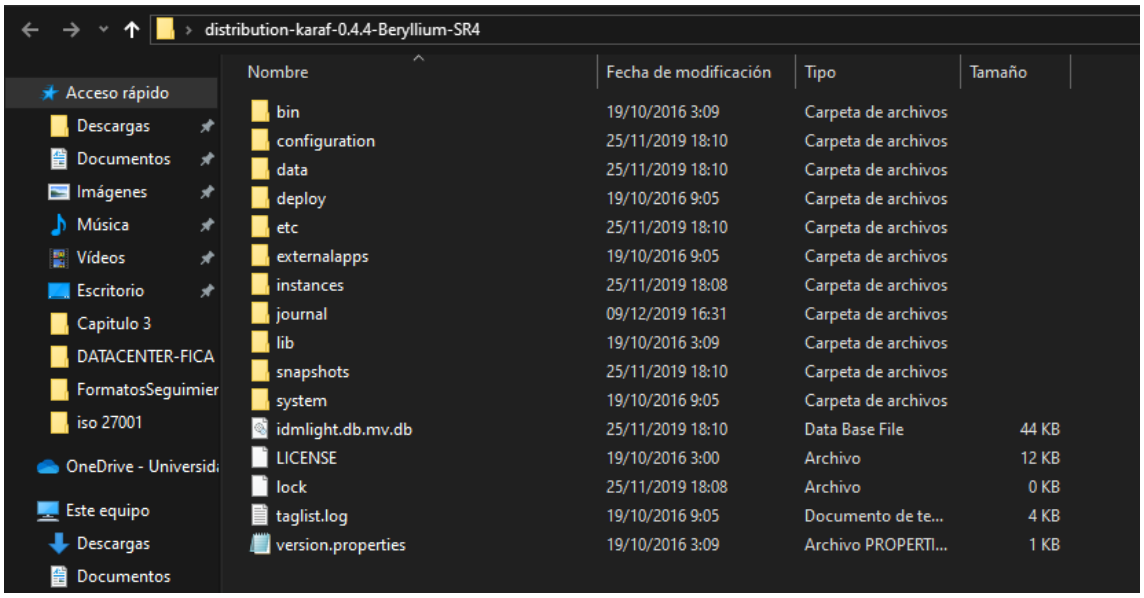


Figura 91. *Archivos de ODL descomprimido*

Para que el controlador funcione en Windows 10 hace falta agregar una liena de comando en el archivo llamado **system.properties**, el cual se encuentra dentro de la carpeta **etc**; el comando que se debe agregar al final del archivo es:

```
org.osgi.framework.os.name = Win32
```

Ahora ya es posible abrir la consola del ODL, para ello se abre como Administrador el archivo **karaf.bin** que se encuentra en la carpeta **bin,** desde la consola se instalarán los paquetes (features) necesarios dependiendo de las necesidades del administrador de la red, en este caso los paquetes que se instalaran son los descritos en la Tabla 13, y el comando lo que permite es:

```
feature:install odl-restconf odl-mdsal-apidocs odl-dlux-all odl-dlux-node
odl-dlux-core odl-dlux-yangui odl-l2switch-switch odl-openflowplugin-all
```

Para acceder a la interfaz web del ODL, es necesario escribir la siguiente dirección; http://localhost:8181/index.html, dando como resultado lo que está en la Figura 92.

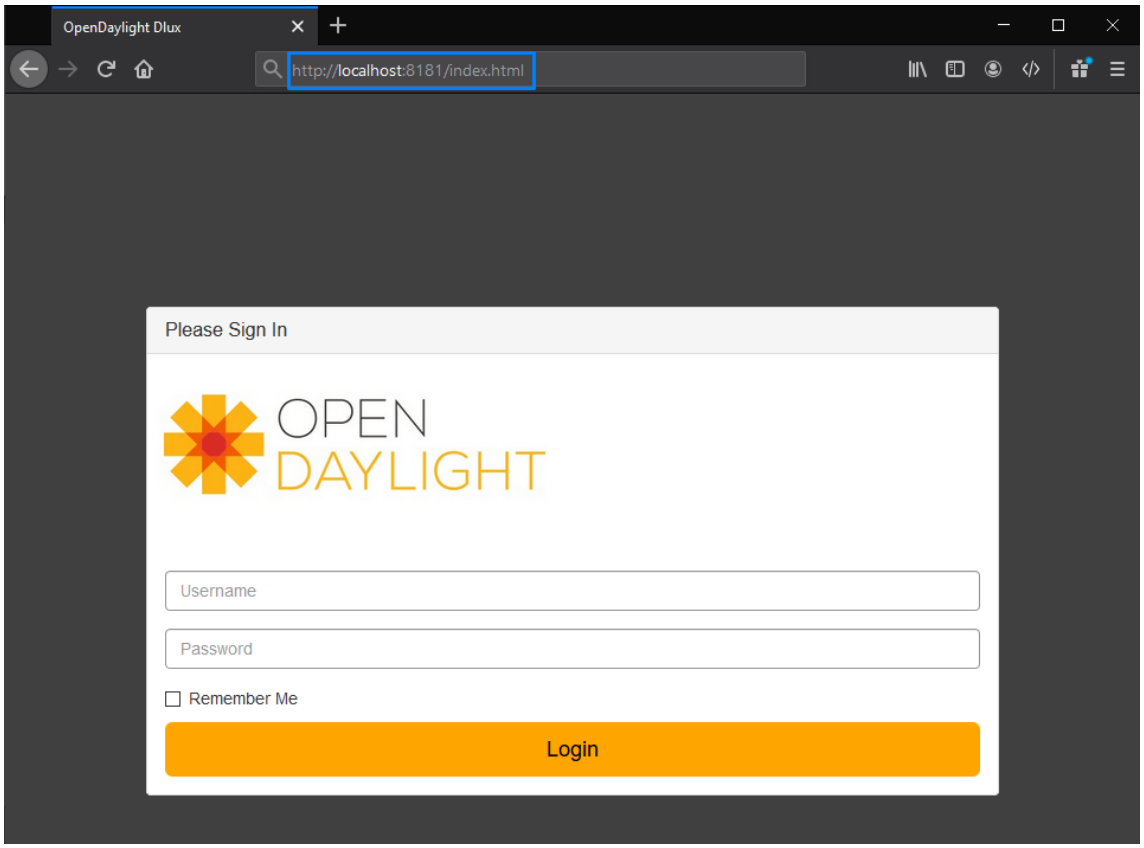


Figura 92. *Inicio de sesión en interfaz web de ODL*

Las credenciales por defecto para acceder al panel de control del ODL son **admin**, tanto en username como en password. Y si la instalación fue correcta, se presenta una pantalla igual a la de la Figura 93.

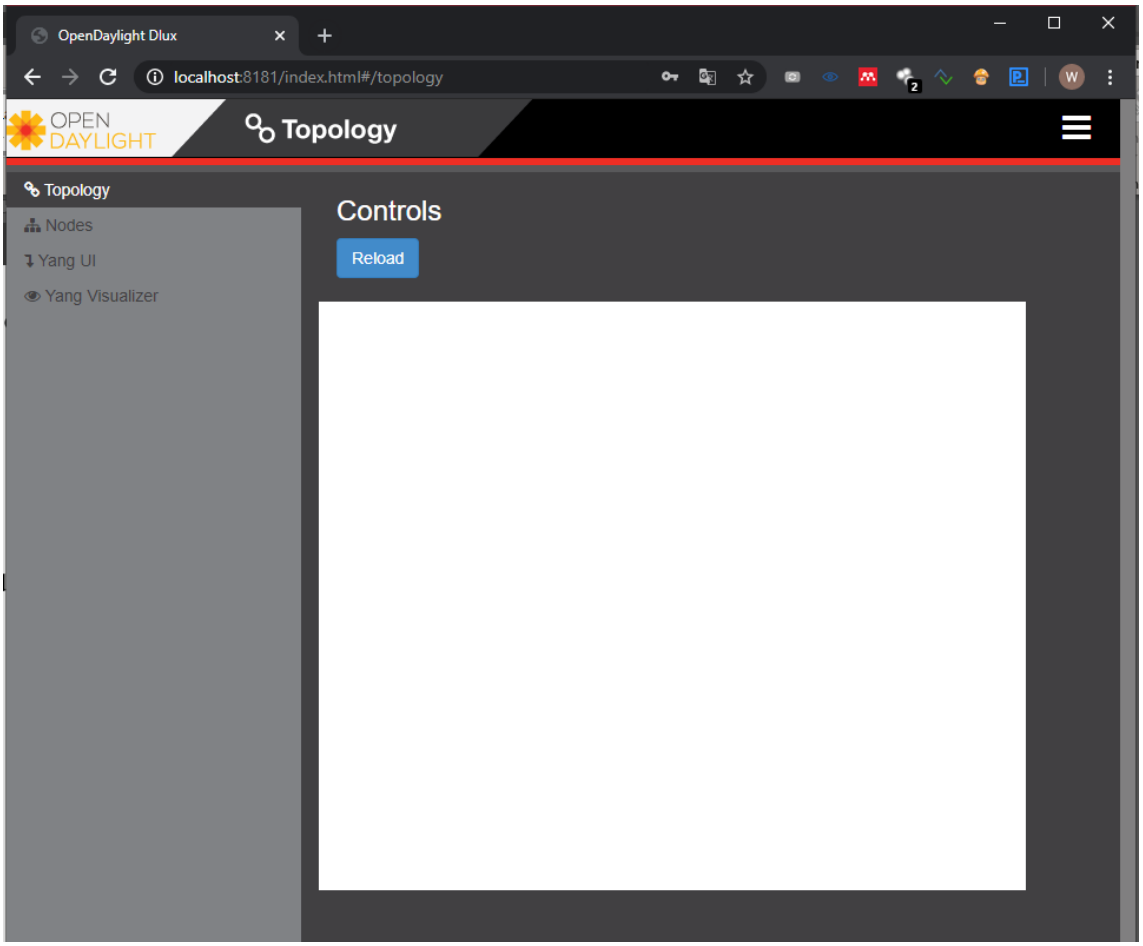


Figura 93. *Panel de control de ODL.*

Anexo 2. Instalación de Open vSwitch

Open vSwitch permite que los equipos finales se comuniquen con el controlador, para empezar con su implementación es necesario instalar una herramienta con esta línea de comando:

```
yum install -y https://repos.fedorapeople.org/repos/openstack/EOL/openstack-juno/epel-7/openvswitch-2.3.1-2.el7.x86_64.rpm
```

Ahora ya es posible iniciar con el servicio, ejecutando estos comandos

```
systemctl enable openvswitch

systemctl start openvswitch
```

Lo siguiente es configurar la interfaz que permite la comunicación del OVS con el controlador, para ello se crea una interfaz virtual que funciona como puente a la cual es posible agregar varias interfaces tanto virtuales como físicas, la creación de esta interfaz es con el comando:

```
ovs-vsctl add-br ovs-br0
```

Para que la interfaz ovs-br0 comience a funcionar debe contener a otra interfaz, en este caso será la interfaz ens34, además es necesario modificar los archivos de configuración de ambas interfaces, estos se alojan dentro de **/etc/sysconfig/network-scripts/**, la Figura 94 representa la configuración de ens34, mientras que la Figura 95 indican los datos que deben ir dentro de ovs-br0.

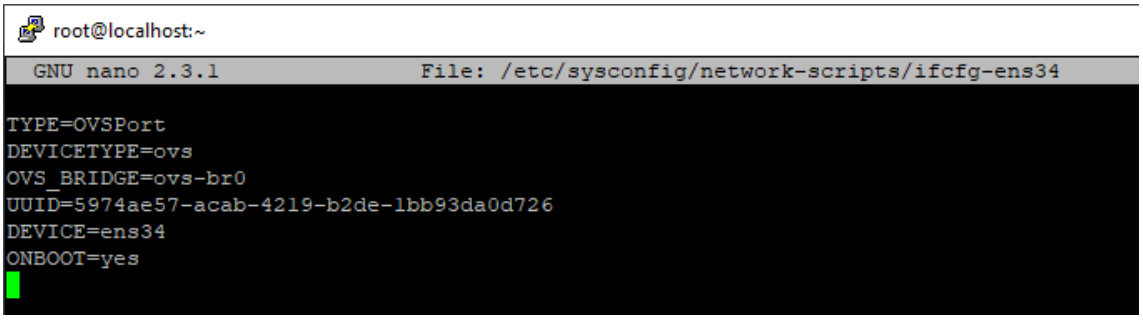


Figura 94. *Configuración de ifcfg-ens34*

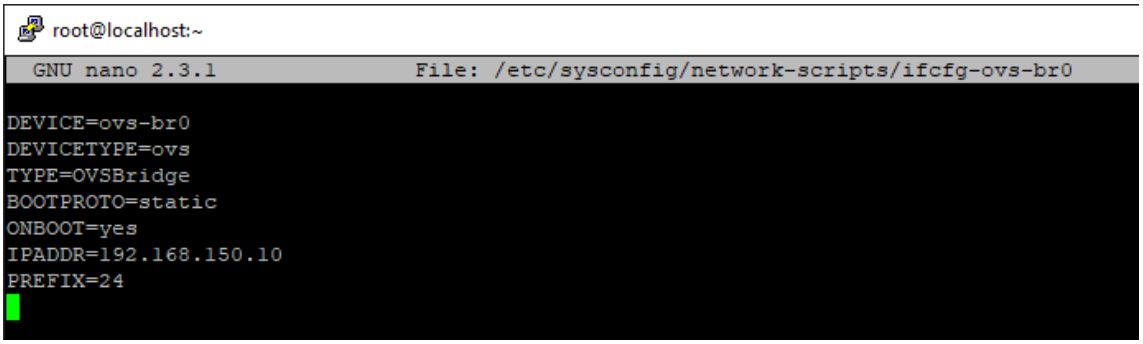


Figura 95. *Configuración de ifcfg-ovs-br0*

Finalmente, se conecta a OVS con el controlador ejecutando el siguiente comando:

```
ovs-vsctl set-controller ovs-br0 tcp:192.168.100.111:6653
```

En el cual se indica la interfaz que se conecta con el controlador (`ovs-br0`), el protocolo de comunicación (`tcp`), la IP del controlador (`192.168.100.111`) y el puerto de comunicación del controlador (`6653`).

Anexo 3. Instalación de servidores

Anexo 3.1. Instalación de servidor web

En lo que respecta al servidor web, es necesario empezar por la instalación del servicio denominado **httpd**, con el comando.

```
yum -y install httpd
```

Una vez instalado el servicio, se debe configurar el mismo para su correcta ejecución, para ello se elimina el archivo que contiene la información de la página de bienvenida, esta acción se logra con el siguiente comando.

```
rm -f /etc/httpd/conf.d/welcome.conf
```

Lo siguiente es editar el archivo de configuración llamado **httpd.conf**, el cual se encuentra dentro de esta dirección **/etc/httpd/conf/**. El editor usado para este caso es ejecutado con el comando `vi`; dentro del archivo citado se modifican las líneas 95, 151, y 164, además de agregar una línea de comando al final de este archivo. Estas configuraciones son representadas en desde la Figura 96 hasta la Figura 99.

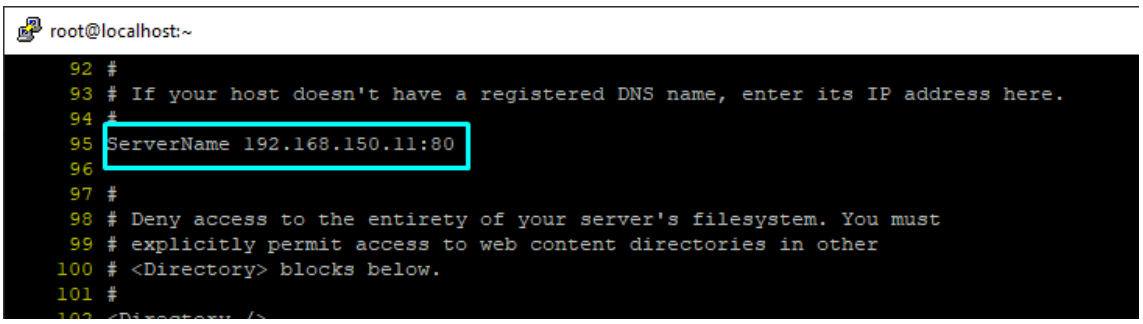


Figura 96. *Asignación de IP de servidor web*

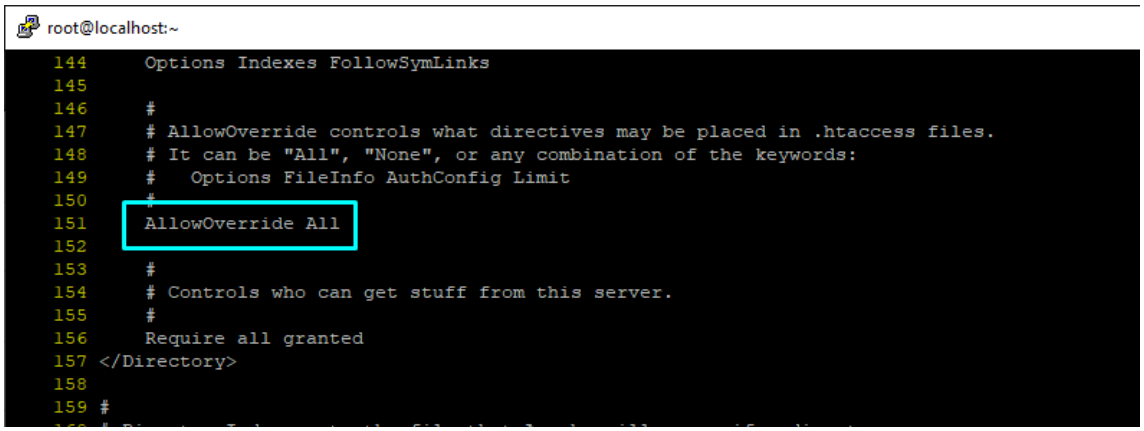


Figura 97. *Permisos para administración del servidor web*

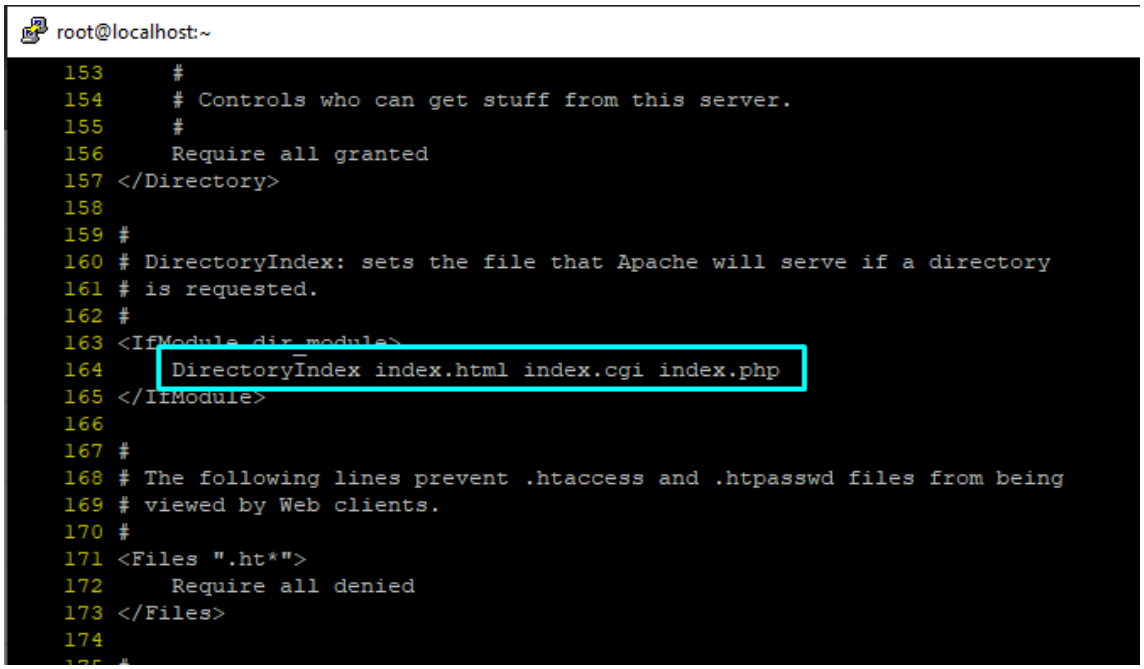


Figura 98. *Formatos que el servidor web permite*

Figura 99. *Información que el servidor comparte*

Ahora se configura la página de inicio en el servidor, para ello se crear un archivo llamado **index.html** dentro de **/var/www/html/**, la información que contiene este archivo se la presenta en la Figura 100.

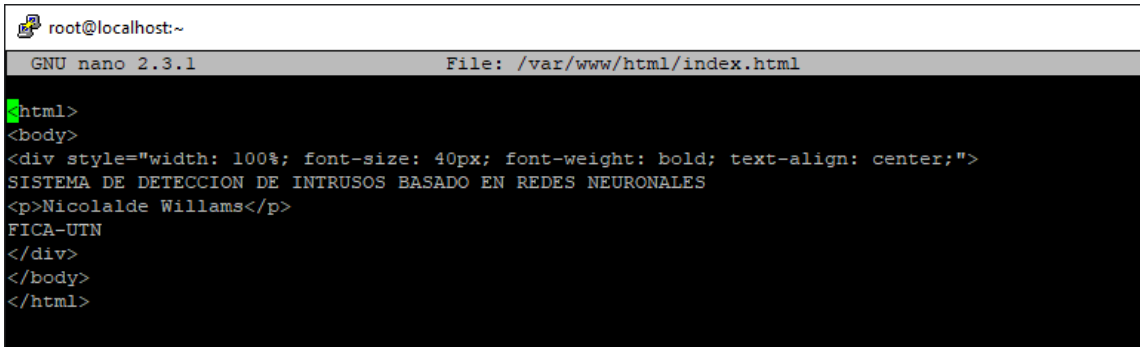


Figura 100. *Código de index.html*

Finalmente se inicia el servicio, pero antes es necesario habilitar su inicio automático. Los comandos necesarios para este proceso se indican luego del presente párrafo. Siendo el primero de ellos el que permite su inicio automático, con el propósito de que este servicio se ejecute de manera automática cada vez que el servidor se encienda; mientras que el segundo inicia el servicio en mención.

```
systemctl enable httpd
```

```
systemctl start httpd
```

Para ingresar al servidor web se escribe IP de este en un navegador, y el resultado es tal como lo indica la sección 3.5.2.

Anexo 3.2. Instalación de servidor ftp

Al igual que el servidor web, este servidor también necesita los servicios respectivos para su funcionamiento que en este caso se denominan **vsftpd**; estos son instalados con el comando:

```
yum -y install vsftpd
```

A continuación se configura el archivo **vsftpd.conf** alojando en **/etc/vsftpd/**, los cambios hechos en este archivo se los indica en las Figura 101 y Figura 102.

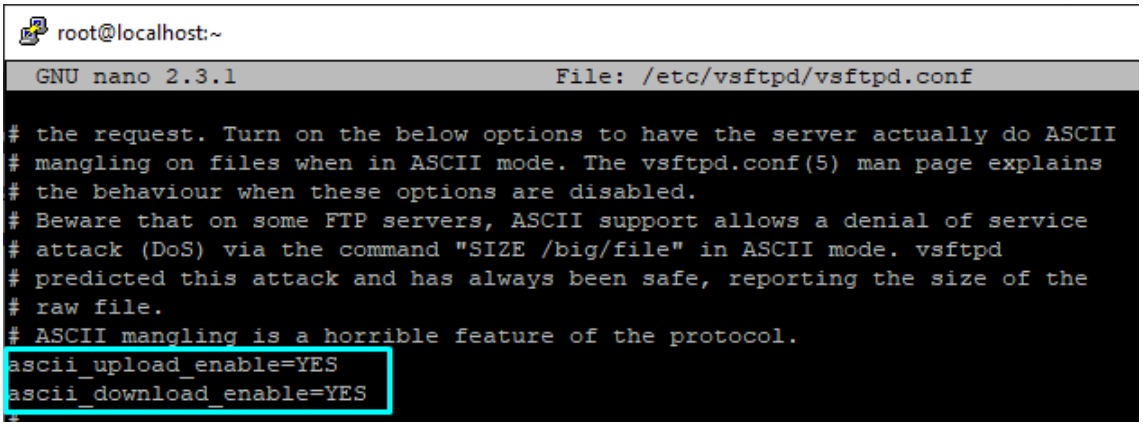


Figura 101. *Permite el uso de código ASCII*

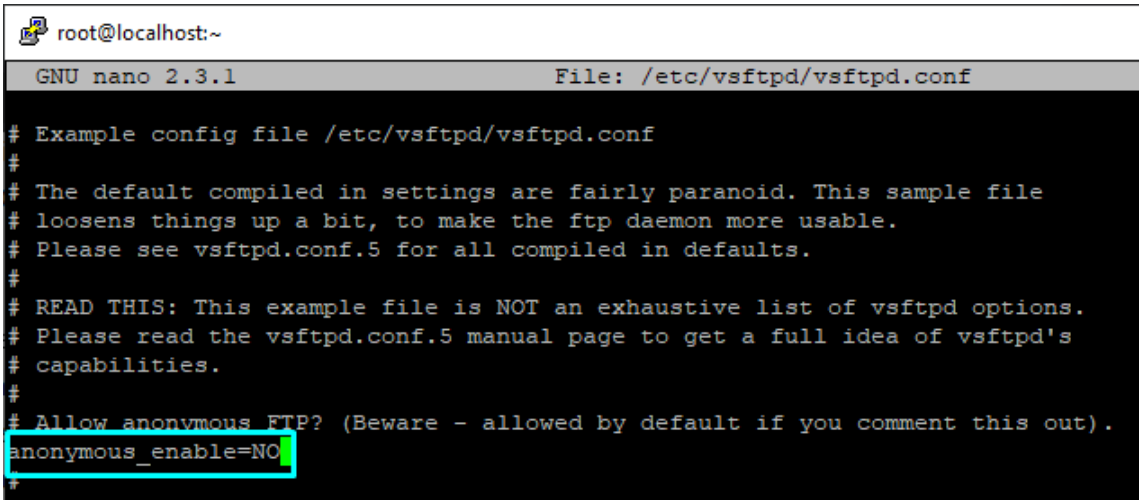


Figura 102. *Deshabilita acceso anónimo al servidor ftp*

Para iniciar el servicio se emplean los comandos señalados bajo este párrafo, de igual manera que en el servidor web, el primer comando permite la ejecución automática del servicio, mientras que el segundo comando inicia al servicio.

```
systemctl enable vsftpd

systemctl start vsftpd
```

Finalmente se establecen las credenciales que serán usadas para el acceso a este servidor, en este caso se creó un usuario llamado **Willams**, mismo que cuenta con su

respectiva contraseña, los comandos usados para la creación de las credenciales se presentan a continuación. En tanto que la sección 3.5.2 presenta los resultados de este servidor.

```
useradd Willams

passwd Willams
```

Anexo 3.3. Instalación de servidor moodle

Para la instalación del servicio de moodle es necesario algunos requisitos, entre ellos están: el repositorio EPEl, PHP 7.0, Apache y MariaDB. Para agregar el repositorio EPEl, se necesitan los comandos:

```
yum install -y epel-release yum-utils

rpm -Uvh https:// mirror.webtatic.com/yum/el7/webtatic-release.rpm
```

En lo que respecta a PHP 7.0 y Apache, se requieren de ciertas herramientas, la siguiente línea de comando es usada para su instalación.

```
yum install -y php70w php70w-curl php70w-gd php70w-intl php70w-ldap php70w-
mysql php70w-pspell php70w-xml php70w-xmlrpc php70w-zip php70w-common php70w-
opache php70w-mbstring php70w-soap
```

Ahora se configura el archivo **httpd.conf** que se encuentra en **/etc/httpd/conf/**; dicha configuración se indica en la Figura 103.

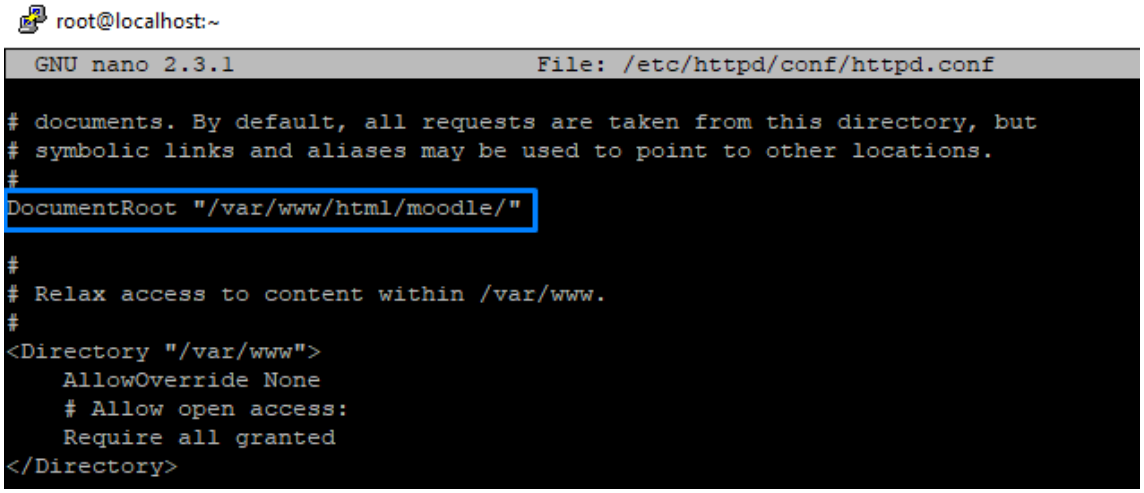


Figura 103. *Asignación de la carpeta de moodle*

Posterior a la configuración se inicia el servicio de Apache, estos son los comandos necesarios para esto propósito.

```
systemctl start httpd

systemctl enable httpd
```

El siguiente comando instala MariaDB que es otra herramienta primordial en este servidor.

```
yum install mariadb-server -y
```

Para habilitar el inicio automático e inicio manual del servicio de MariaDB se escriben los comandos respectivamente:

```
systemctl enable mariadb

systemctl start mariadb
```

En la herramienta MariaDB hace falta agrega ciertas líneas de comando tal como se exhibe en la Figura 104 en el archivo **server.cnf** que se aloja en **/etc/my.cnf.d/**. Para que estos cambios surtan efecto se debe reiniciar el servicio de MariaDB con el comando.

```
systemctl restart mariadb
```

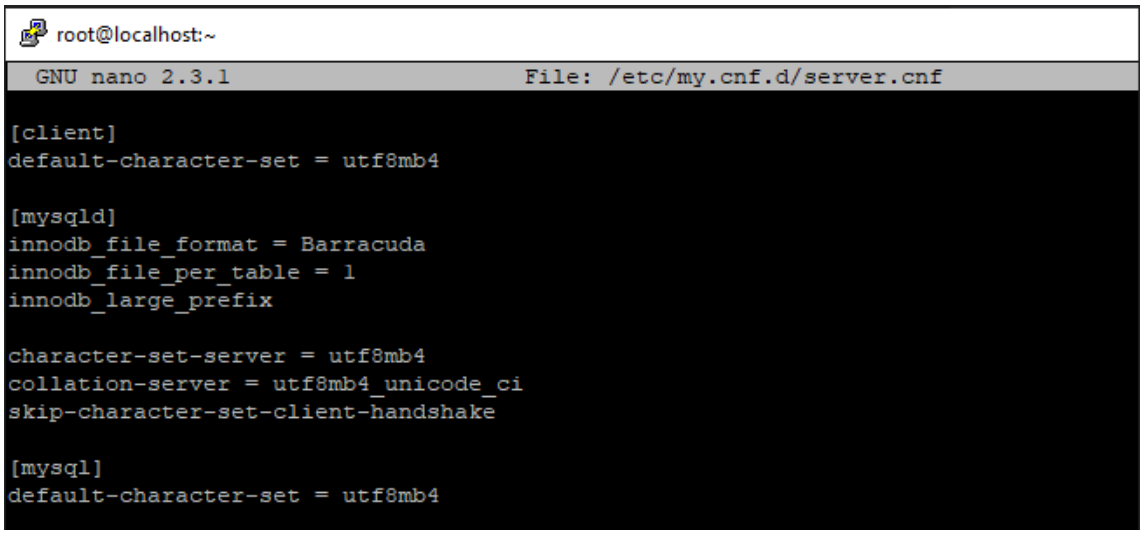


Figura 104. *Configuración de server.cnf*

Para finalizar con los pre requisitos, se configura la base de datos en MariaDB, misma que es indispensable en la instalación de moodle y la cual se logra ejecutando estas líneas de comandos.

```
mysql -u root -p

create database moodle;

grant all privileger on moodle.* to ‘root’@‘localhost’ indentified by
‘moodle’;

quit
```

Antes de empezar con la instalación de moodle, se descarga el paquete que contiene los archivos necesarios en su instalación, este comando ejecuta el proceso en mención.

```
wget https://download.moodle.org/estable34/moodle-latest-34.tgz
```

A continuación se descomprime el paquete **moodle-latest-34.tgz**, para ello se usa el comando; `tar xvzf moodle-latest-34.tgz -C /var/www/html/`. Luego de eso se concede permisos al grupo **apache** escribiendo el comando.

```
chown -R apache:apache /var/www/
```

La instalación de moodle es mediante su interfaz web, a la cual se accede escribiendo la IP del servidor en un navegador, este proceso es intuitivo razón por la cual todos los valores deben quedar tal como se indican desde la Figura 105 hasta la Figura 110.

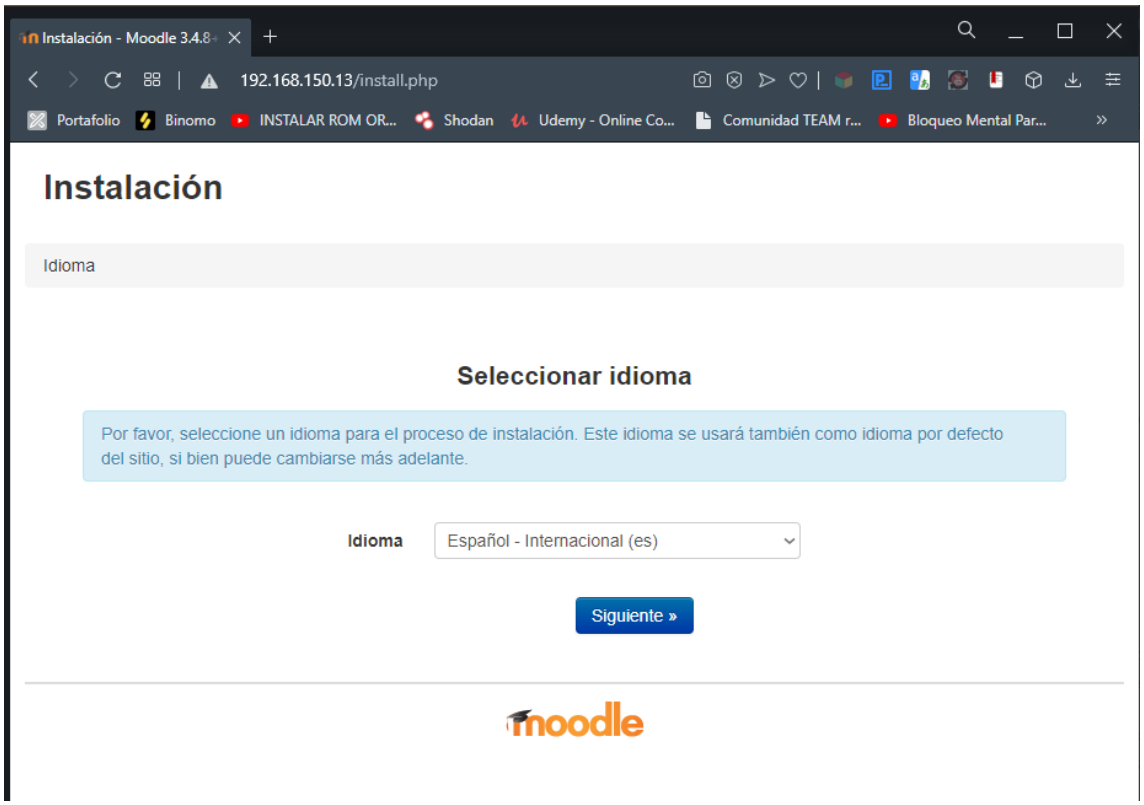


Figura 105. *Interfaz web para instalación de moodle*

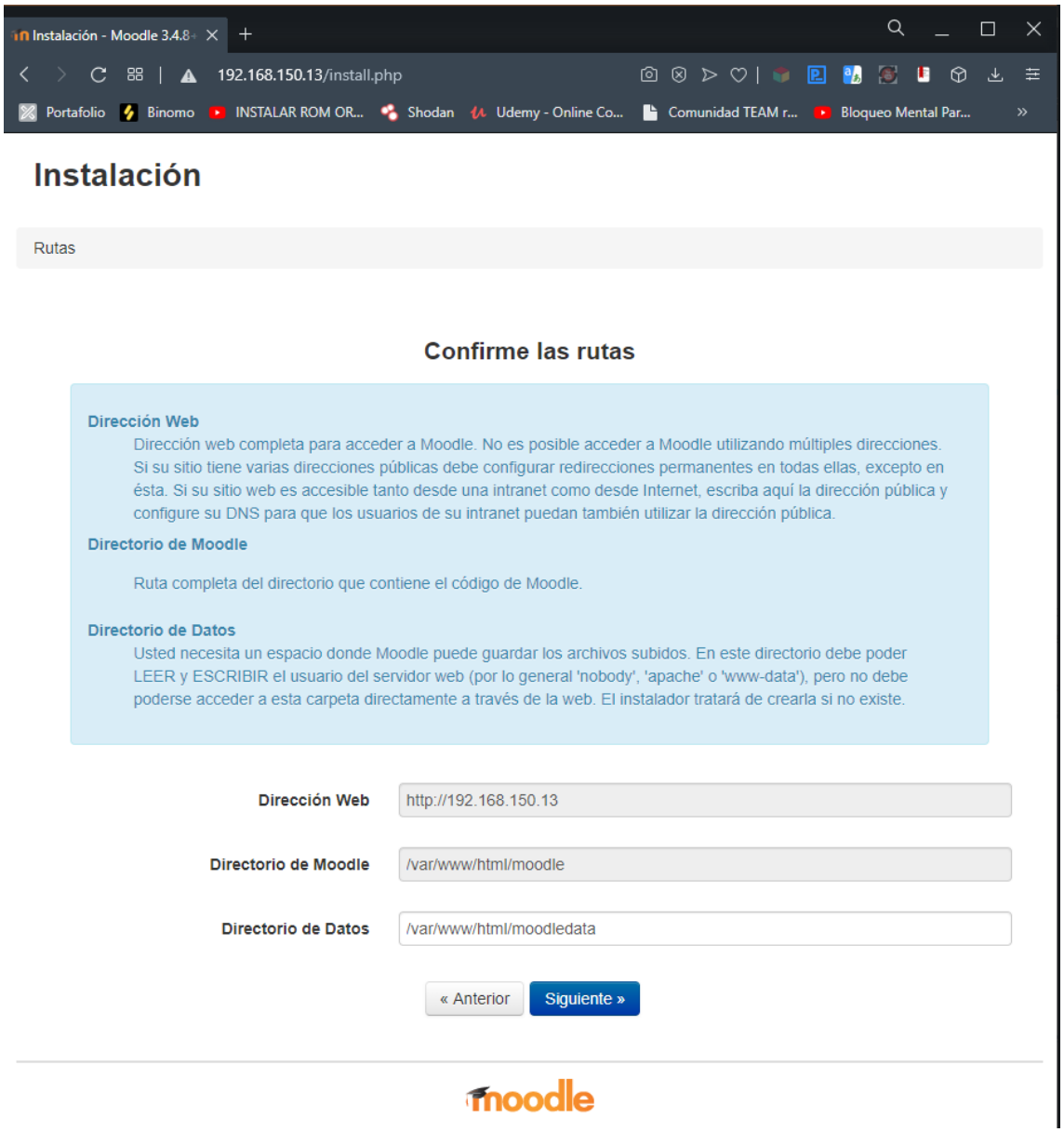


Figura 106. *Configuración de los directorios de moodle*

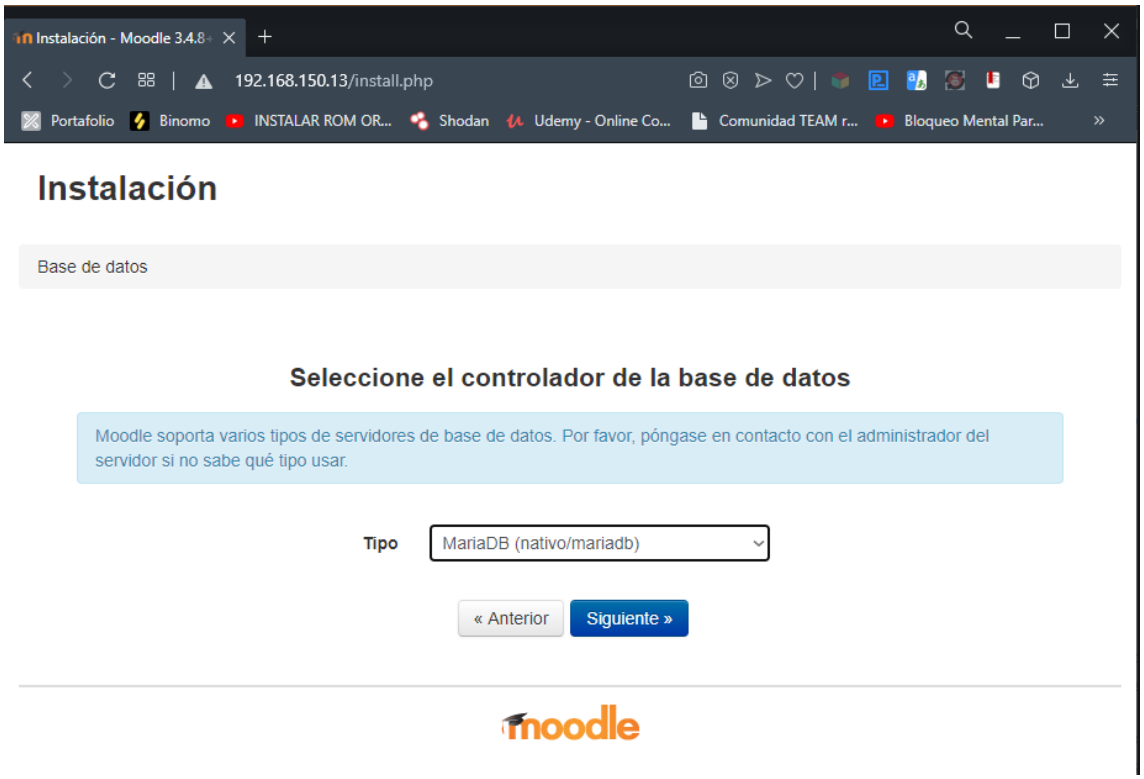


Figura 107. *Configuración del gestor de base de datos*

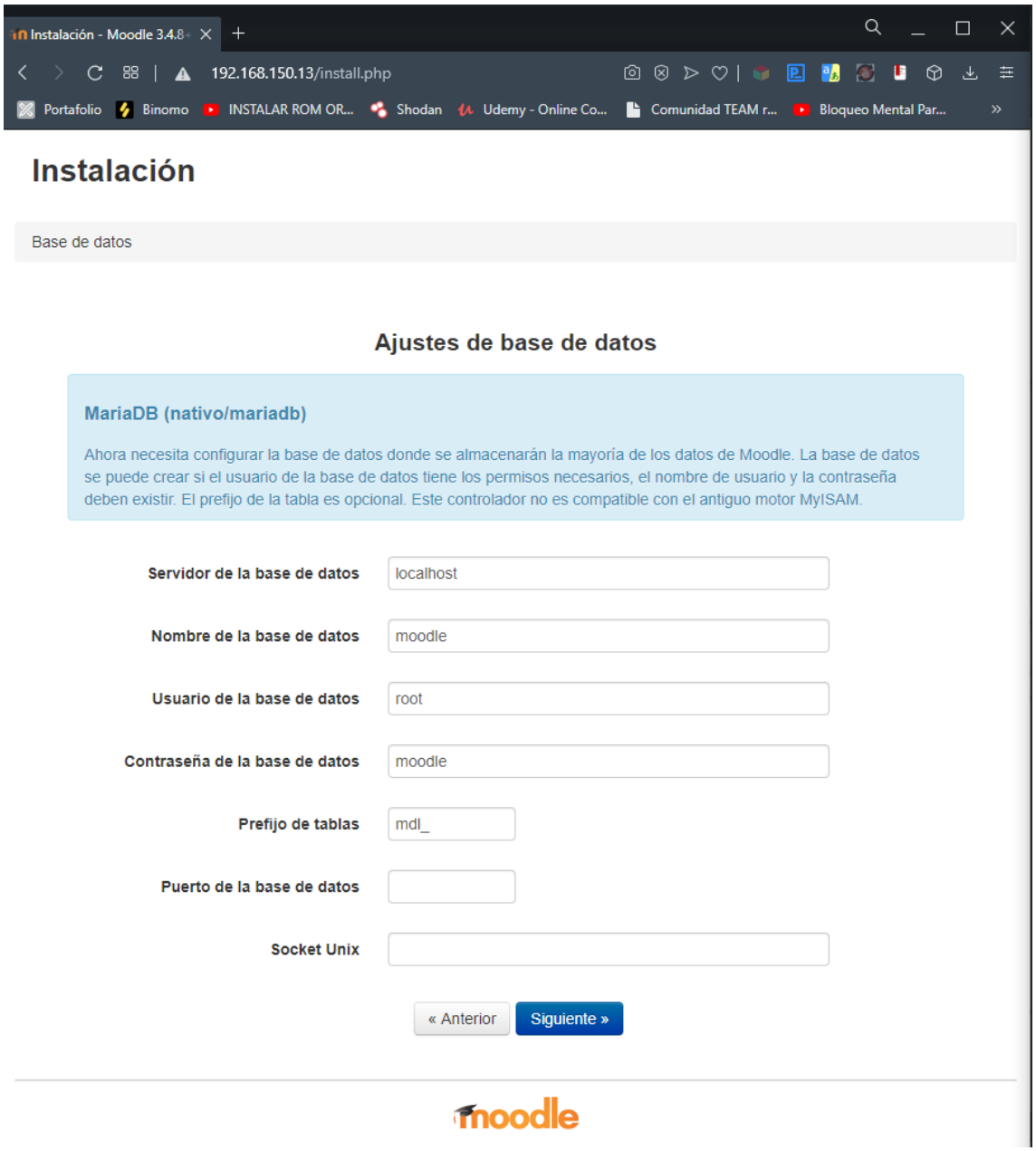


Figura 108. *Configuración de base de datos*

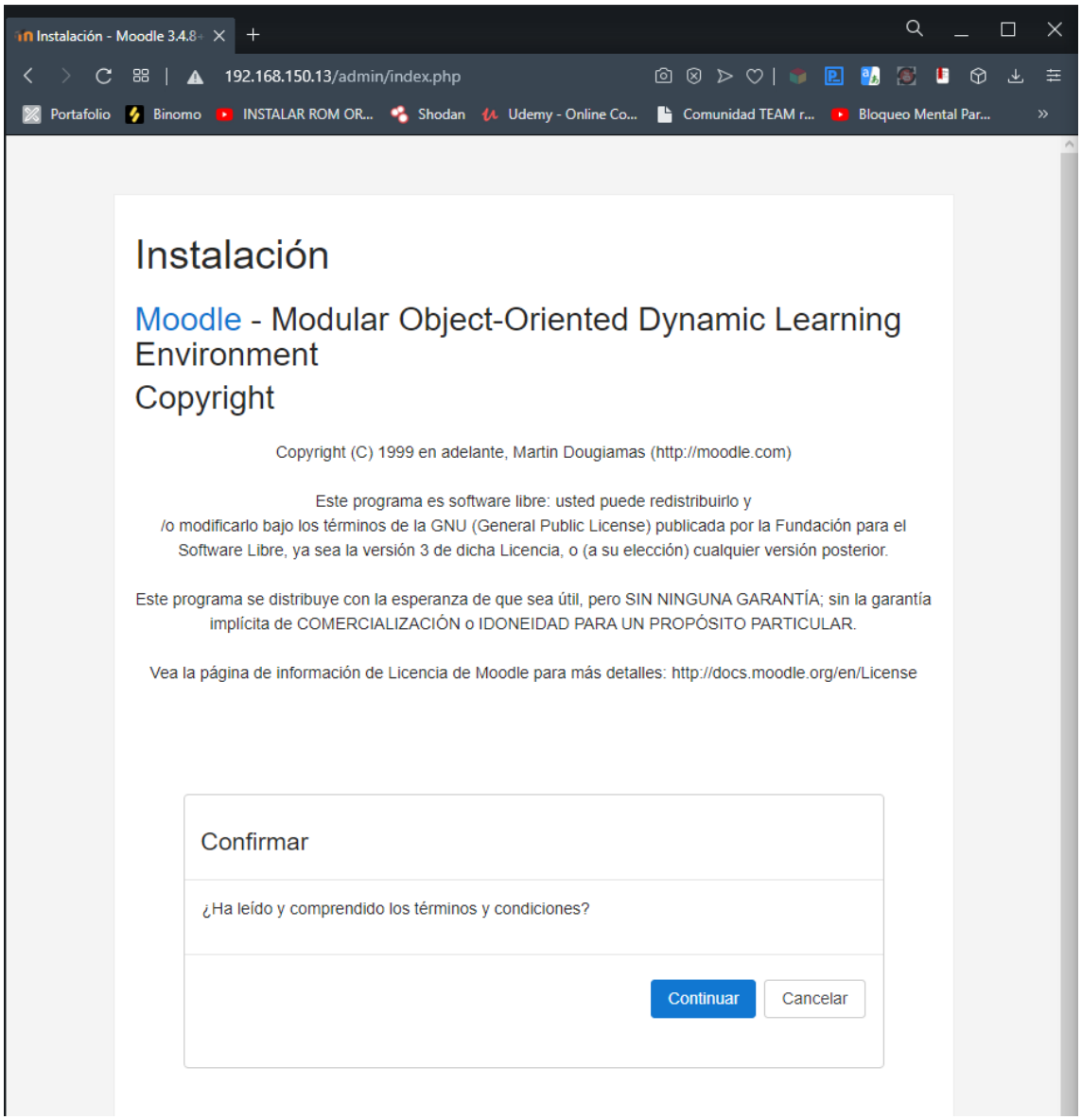


Figura 109. *Términos y condiciones de moodle*

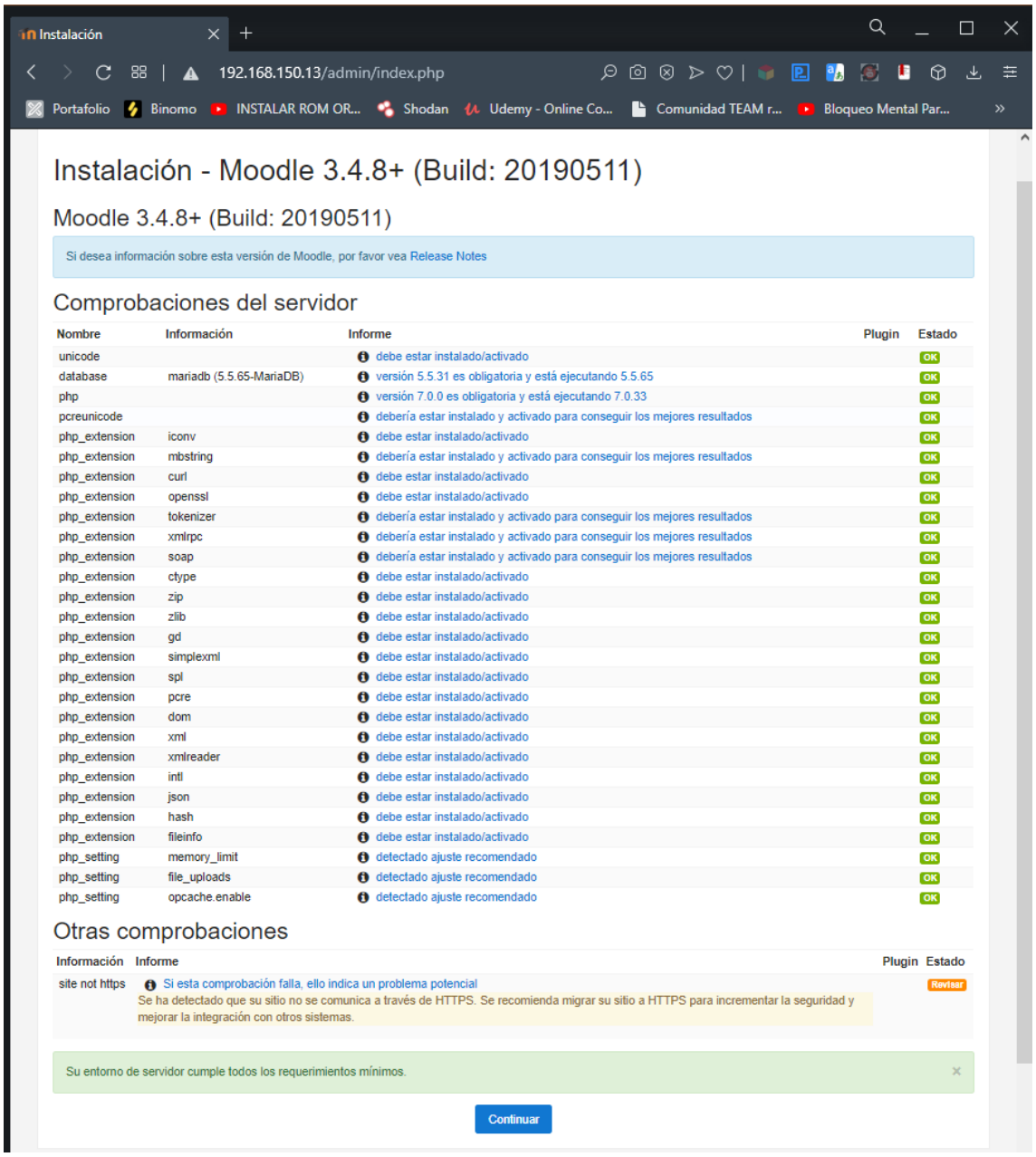


Figura 110. *Verificación de los requisitos de moodle*

La información de la Figura 111 y la Figura 112 en la instalación del servidor de moodle varían de acuerdo con la información que proporcione el administrador.

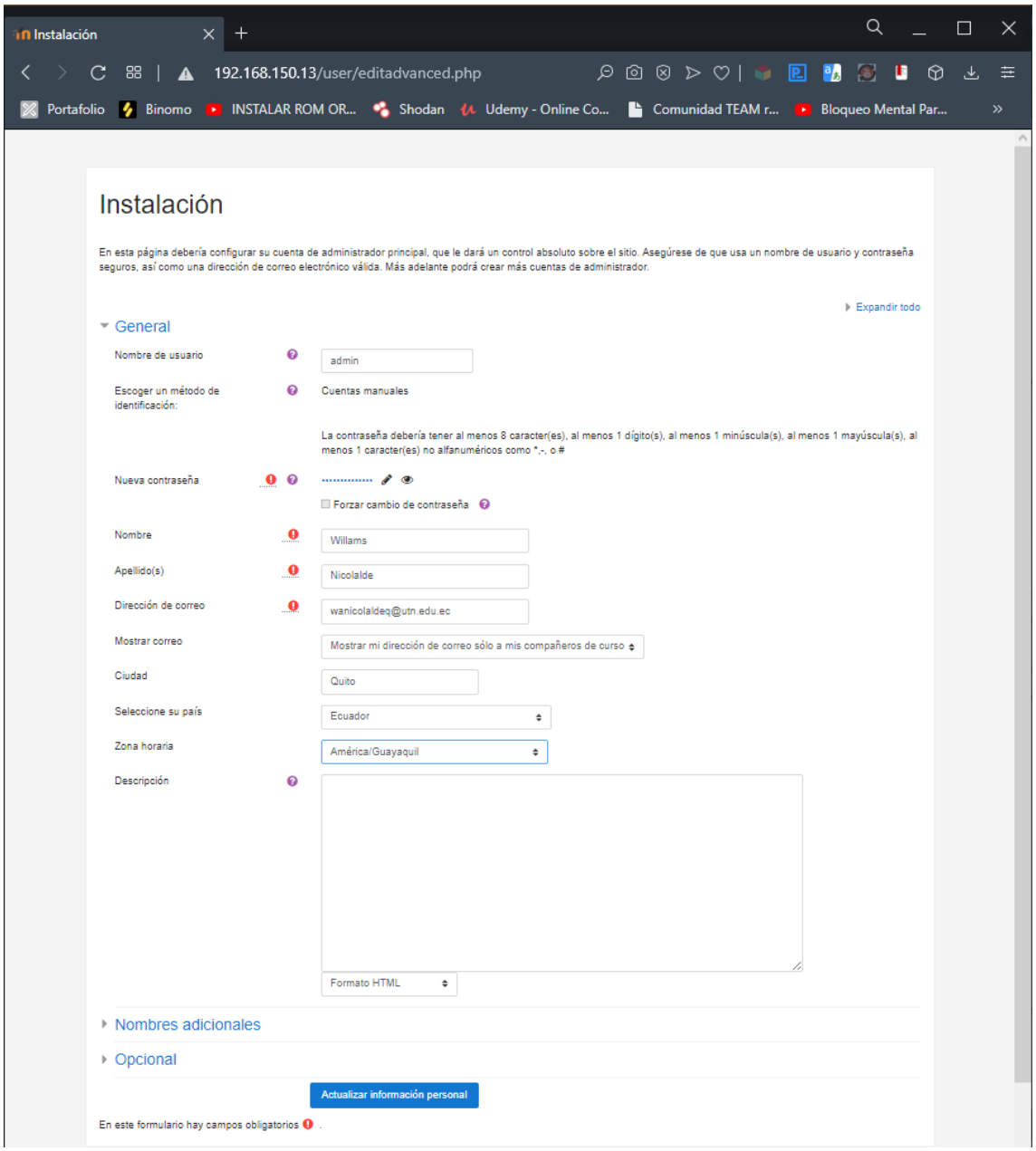


Figura 111. *Configuración de datos del administrador de moodle*

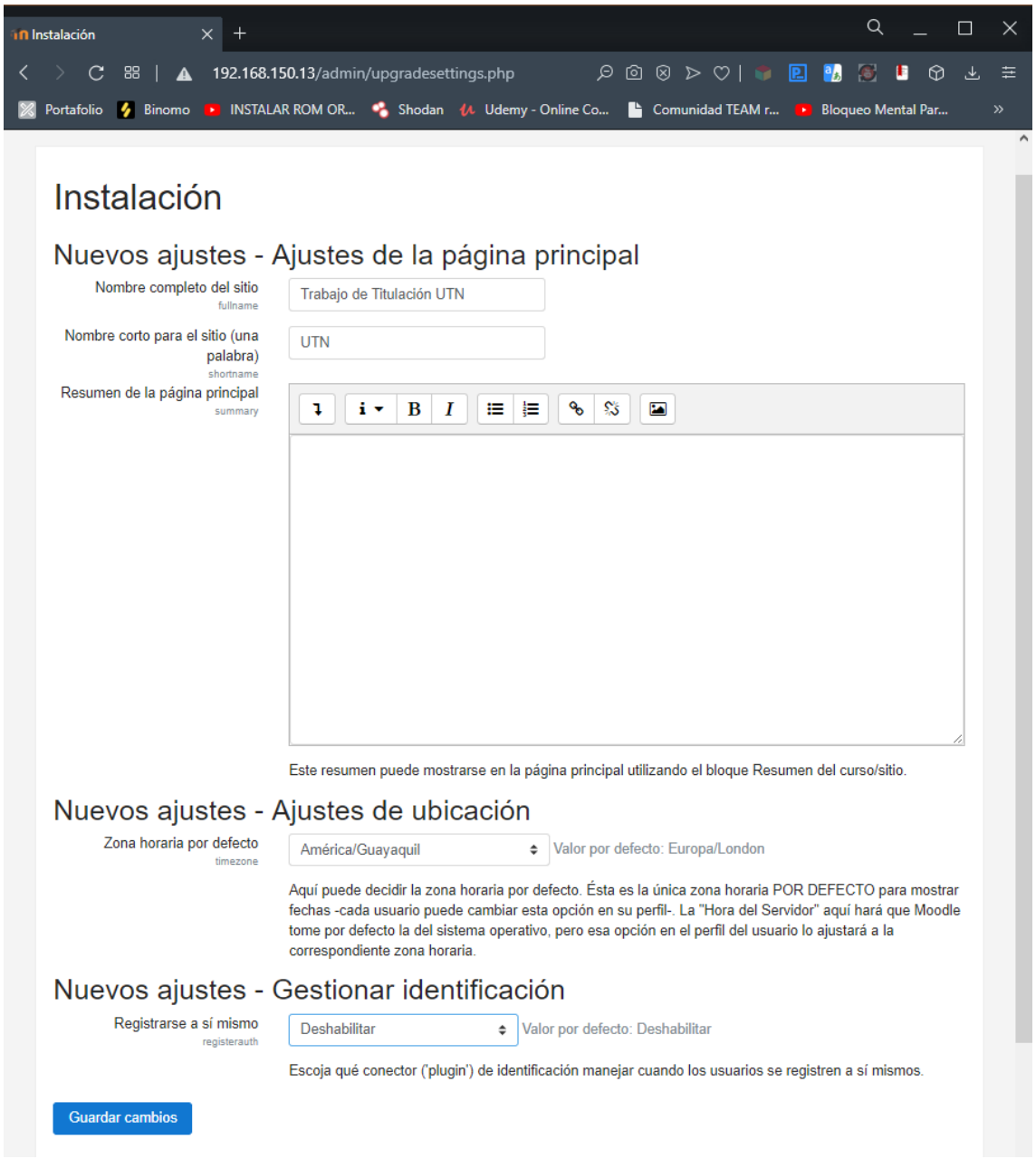


Figura 112. *Configuración de la página principal de moodle*

Finalmente se observan los resultados del servidor moodle ya instalado y configurado en la sección 3.5.2.

Anexo 4. Diseño de red MLP

Para el diseño de la red MLP es necesario el archivo de la red neuronal artificial que contiene las capas de entrada, ocultas y salida, también el set de entrenamiento, ambos archivos se descargan del siguiente enlace; https://sourceforge.net/projects/snort-

ai/files/PortscanAI%20Neural%20Networks/. Con los archivos descargados, se los abre en JavaNNS, tal como indican la Figura 113 y la Figura 114.

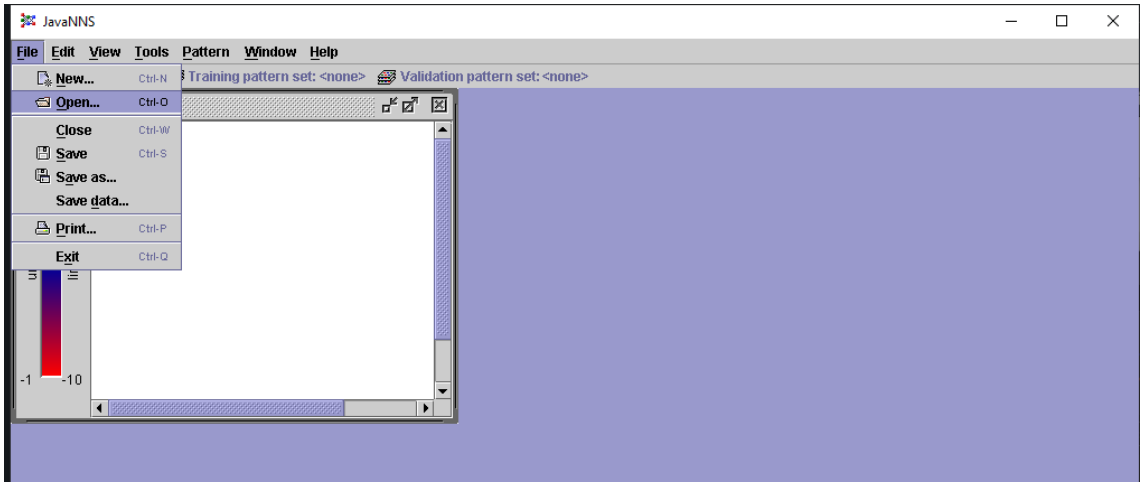


Figura 113. *Abrir archivos necesarios para red MLP*

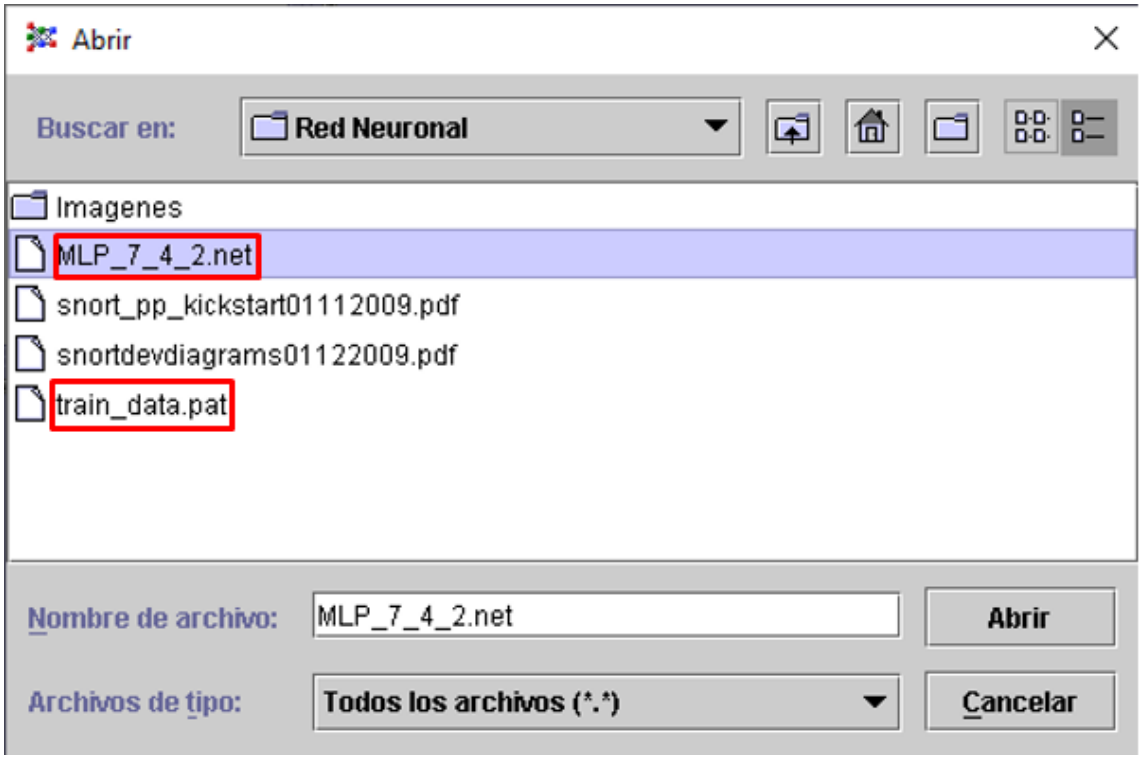


Figura 114. *Archivos para la red MLP*

Se verifican que ambos archivos se cargaron correctamente y se encuentran listos para iniciar el proceso de entrenamiento de la red MLP si se observa algo similar a la Figura 115 con cierta variación en los valores de las capas.

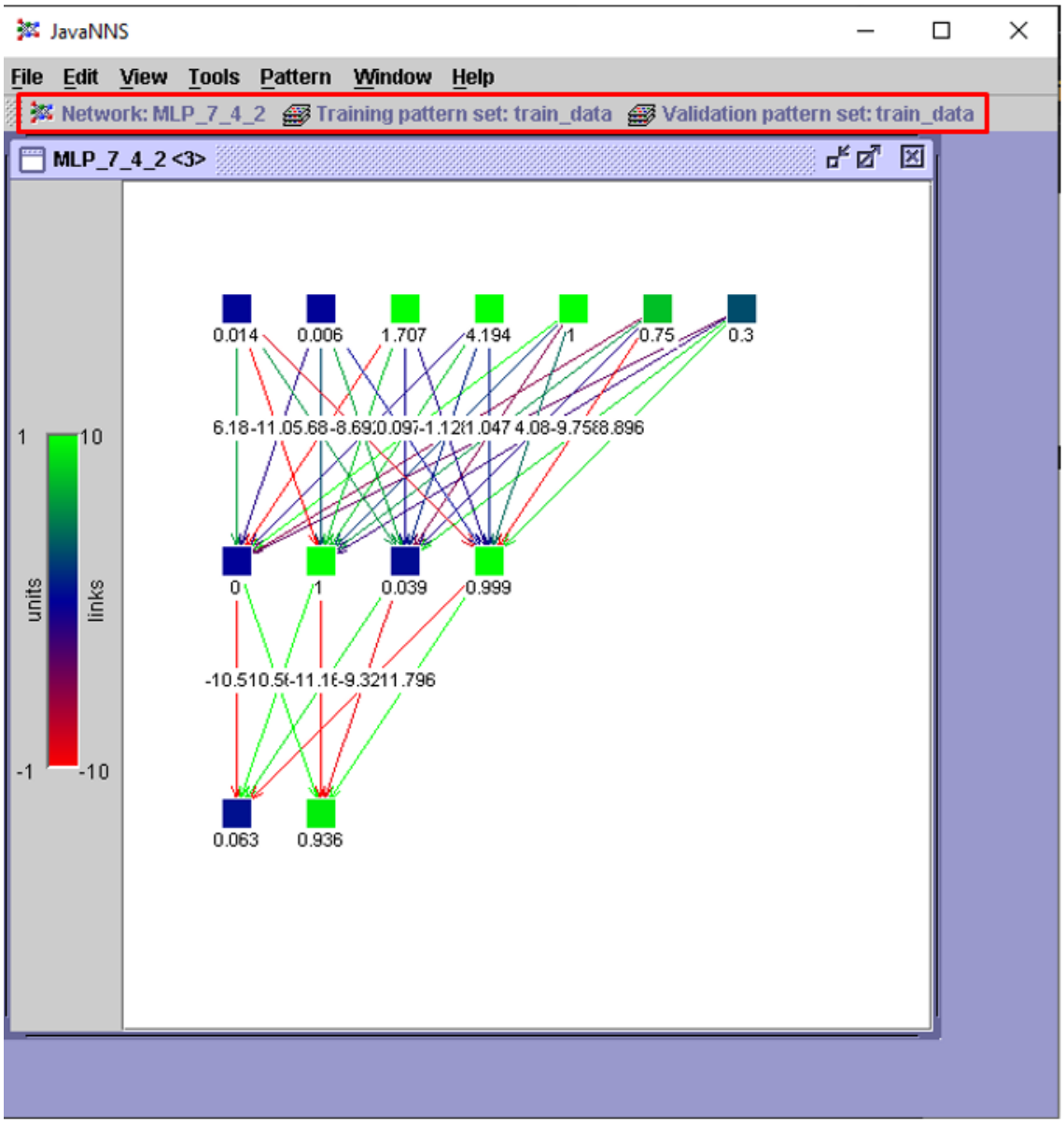


Figura 115. *Red MLP sin entrenar*

Antes de empezar a entrenar la red MLP se comprueban que los parámetros de entrenamiento se encuentren igual a los de la Figura 116 y la Figura 117.

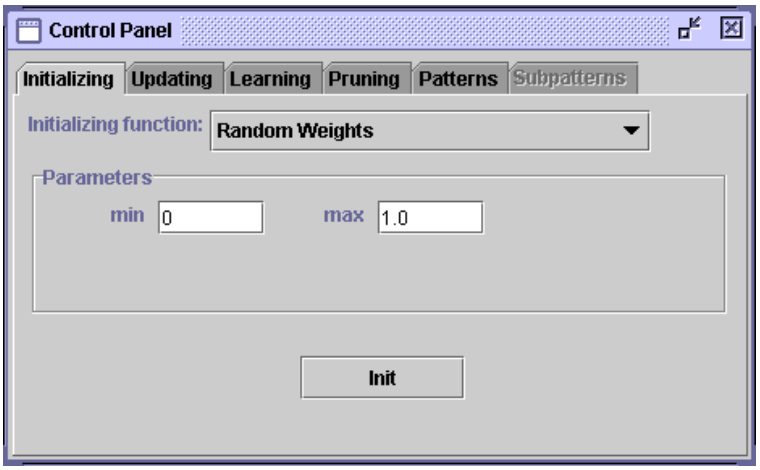


Figura 116. *Rangos de pesos establecidos*

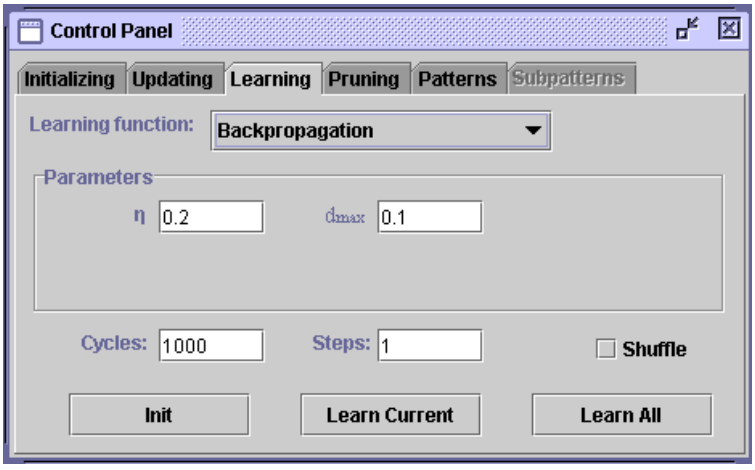


Figura 117. *Aprendizaje con retroalimentación configurada*

El entrenamiento de la red MLP empieza únicamente dando clic en **Learn All** hasta obtener en su capa de salida valores de 0 y 1 o aproximados a estos, tal como se obtuvo en la Figura 118.

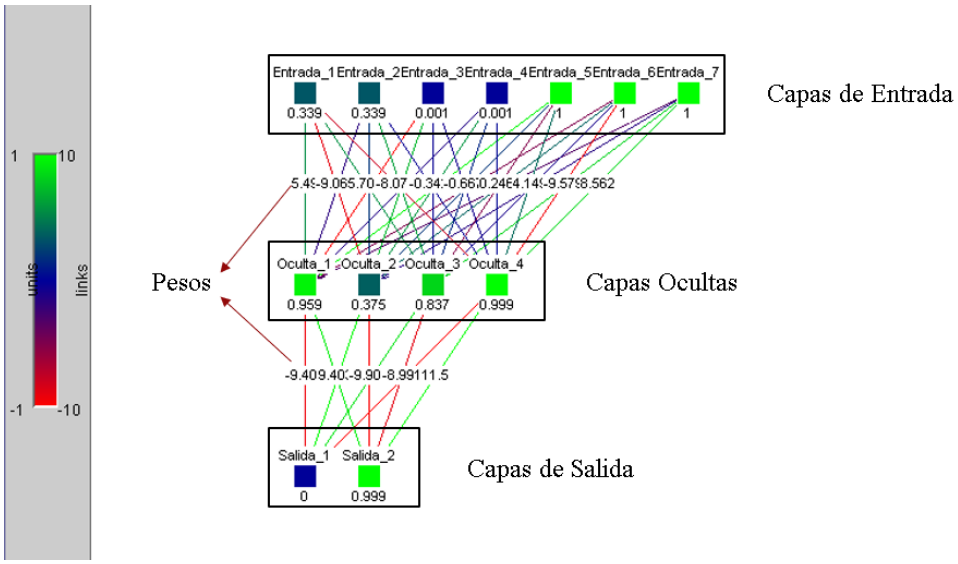


Figura 118. *Red MLP entrenada*

Finalmente, se guarda el archivo con formato .net y posteriormente observar los pesos que llevaron a este resultado, además de que estos serán útiles para el diseño del preprocesador.

Anexo 5. Instalación de Snort+RNA

La instalación de Snort+RNA necesita como requisito ciertas herramientas, las cuales deben ser instaladas con los siguientes comandos:

```
yum install -y epel-release

yum install -y libnghttp2

yum install -y httpd gcc pcre-devel php-gd gd mod_ssl glib2-devel gcc-c++
bison flex php-pear zlib pcre libnet tcpdump mariadb-server mysql-devel
```

A continuación se vincula la librera libnet con las siguientes líneas de comandos.

```
ln -s /usr/lib64/libdnet.so.1.0.1 /lib64/libdnet.1

ldconfig
```

Otra herramienta necesaria previa a la instalación de Snort, es la instalación de las librerías que permite el análisis del tráfico, este proceso se indica bajo el presente apartado.

```
wget https://www.tcpdump.org/release/libpcap-0.9.8.tar.gz

tar xvfz libpcap-0.9.8.tar.gz

cd libpcap-0.9.8

./configure && make && make install
```

En este caso la instalación de Snort+RNA no será automática, dicho en otras palabras, la instalación usa los archivos proporcionados un enlace externo y el proceso a seguir es el siguiente:

```
wget https://sourceforge.net/code-snapshots/svn/s/sn/snort-ai/code/snort-ai-code-r55-trunk-snortai-stable.zip

unzip snort-ai-code-r55-trunk-snortai-stable.zip

mv snort-ai-code-r55-trunk-snortai-stable snortia

cd /root/snortia

./configure --with-mysql --with-mysql-libraries=/usr/lib64/mysql/ --with-mysql-includes=/usr/include/

make && make install
```

Para continuar la instalación se crean las carpetas que alojaran los archivos de Snort+RNA, esto se realiza con estos comandos.

```
mkdir /etc/snort

mkdir /etc/snort/rules

mkdir /var/log/snort

mkdir /etc/snort/preproc_rules
```

Dentro de las carpetas creadas se colocan los archivos de configuracion de Snort+RNA, para ellos se escriben estas líneas de comandos, primero indicando el origen de los archivos y luego su destino.

```
cp /usr/local/bin/snort /usr/sbin

cp /root/snortia/etc/snort.conf /etc/snort/

cp /root/snortia/etc/classification.config /etc/snort/

cp /root/snortia/etc/reference.config /etc/snort/

cp /root/snortia/etc/unicode.map /etc/snort/

cp /root/snortia/rpm/snortd /etc/init.d/

cp /root/snortia/preproc_rules/* /etc/snort/preproc_rules/
```

Por último se configura el archivo **snort.conf** que se aloja en **/etc/snort/**, en el cual se indica la red interna y red externa, la primera red será la que protege Snort+RNA, mientras que la segunda red no será tomada en cuenta por Snort+RNA. Las líneas de comando que establecen estas redes son estas:

```
var HOME_NET 192.168.100.0/24

var EXTERNAL_NET ¡HOME_NET
```

Anexo 6. Configuración de interfaz web de Snort+RNA

La sección 4.6 muestra las pruebas que se realizaron con Snort+RNA ya configurado, sin embargo, hace falta aclarar el proceso que realiza para la visualización de registros que genera al finalizar cada análisis, este proceso se indica con los comandos bajo este acápite. Además, es necesario cuatro archivos que se alojan en **/var/www/html/tesis/file/**, los nombres de estos archivos son: **config.php, index.php, graph.php** y **graph_wrap.php**; el contenido que debe tener cada archivo se presenta en los siguientes anexos.

```
wget https://jpgraph.net/download/download.php?p=46

mv download.php?p=46 jpgraph-4.3.2.tar.gz

tar xvzf jpgraph-4.3.2.tar.gz

cp -r jpgraph-4.3.2 /var/www/
```

Anexo 6.1. Archivo config.php

```
<?php

$log_path = '/var/log/snort/portscanai';

$jpgraph_path = '/var/www/jpgraph-4.3.0/src';

?>
```

Anexo 6.2. Archivo index.php

```
<?php

include('config.php');

include('header.html');

$filter_limit=$_GET['iph_limit'];

switch($_GET['iph_filter']){

case 'hsrc':

       $part=1;

       break;

       case 'hdst':

       $part=2;

       break;

       case 'nresp':

       $part=5;

       break;

}
```

```
echo '<center><table><tr><td>Log_method: File</td>';
$lines = file($log_path.'/ips_hash.log');
$parts = explode('.',$lines[0]);
echo '<td style="text-align:right">Logs generated:
'.$parts[0].'</td></tr></table></center>';
sort($lines);
$ip_counter = $rel_counter = 0;
echo '<h2>IPs Hash table</h2><center><table><tr><th style="color: aliceblue;">IP</th>
<th><a href="help.html" title="hits as source – hits como origen">
hits_as_src</a></th>
<th><a href="help.html" title="hits as destination – hits como destino">
hits_as_dst</a></th>
<th><a href="help.html" title="average receive time – tiempo promedio de recepción">
av_rcv_time</a></th>
<th><a href="help.html" title="average send time – tiempo promedio de envío">
av_snd_time</a></th>
<th><a href="help.html" title="negative_resp (negative responses – respuestas
negativas)">
ack_rst_resp</a></th>
<!--<th><a href="help.html"> RST resp</a></th>
<th><a href="help.html">     win_count</a></th></tr>-->';
foreach($lines as $line){
if(strstr($line, '/') == FALSE) {
              $parts = explode(' ',$line);
              $ip_counter++;
              if($parts[$part] >= $filter_limit)
```

```
                echo '<tr><td><a href="graph_wrap.php?ip='.$parts[0].'&tipo=1">'.
                $parts[0].'</a></td><td>'.$parts[1].
                '</td><td>'.$parts[2].'</td><td>'.$parts[3].
                '</td><td>'.$parts[4].'</td><td>'.$parts[5].
                '</td><!--<td>'.$parts[6].'</td><td>'.$parts[7].'</td>--></tr>';
        }
}
$lines = file($log_path.'/dir_rel_hash.log');
$filter_limit = $_GET['dir_relh_limit'];
sort($lines);
echo '</table></center><h2>Direct Relation Hash table</h2>
<center><table><tr><th style="color: white;">IP src</th><th style="color: white;">IP
dst</th>
<th><a href="help.html" title="relation hits – hits de relación">rel_hits</a></th>
</tr>';
foreach($lines as $line){
$parts = explode(' ',$line);
$rel_counter++;
if($parts[2] >= $filter_limit)
echo '<tr><td>'.$parts[0].'</td><td><a href="graph_wrap.php?ip='.
                $parts[0].'_'.$parts[1].'&tipo=2">'.$parts[1].
                '</a></td><td>'.$parts[2].'</td></tr>';
}
echo '</table></center>';
echo '<b style="font-size: 15px;">Registered IPs in IPs Hash table:
</b>'.$ip_counter.'<br><b style="font-size: 15px;">Registered Direct Relations:
</b>'.$rel_counter;
```

```
	unset($lines);
?>
```

Anexo 6.3. Archivo graph.php

```
<?php
include('config.php');
 include($jpgraph_path.'/jpgraph.php');
 include($jpgraph_path.'/jpgraph_line.php');
 include($jpgraph_path.'/jpgraph_bar.php');
$file = $log_path.'/graph_info/'.$_GET['ip'];
switch($_GET['tipo']){
	case '1':
	display_graph($file, 'Hits como destino Vs N Flows',0,1,'');
	break;
	case '2':
	display_graph($file, 'Hits directos Vs Hits como destino',0,1,'');
	break;
	case '3':
	display_graph($file, 'diff_rcv_time Vs Tiempo (Ms)',2,3,'');
	break;
	case '4':
	display_graph($file, 'diff_snd_time Vs Tiempo (Ms)',2,4,'');
	break;
	case '5':
	display_graph($file, 'diff_rcv_time Vs N Flows',0,3,'');
	case '6':
	display_graph($file, 'diff_snd_time Vs N Flows',0,4,'');
	break;
}
function display_graph($file, $tit, $xindex, $yindex, $legend) {
```

```
	$lines = file($file);
	foreach($lines as $line){
		$toks=explode(' ',$line);
		if(($toks[$yindex]+0)!=0){
			$datay[]=$toks[$yindex]+0;
			$datax[]=$toks[$xindex]+0;
		}
	}
	$graph = new Graph(850,500,"auto");
	$graph->SetScale("linlin");
	$graph->yaxis->scale->ticks->Set(100);
	$graph->SetMargin(70,70,40,30);
	$graph->SetMarginColor('gray');
	$graph->SetFrame(true,'blue',3);
	$graph->title->Set($tit);
	$graph->title->SetFont(FF_FONT2,FS_BOLD);
	$graph->title->SetAlign("center");
	$p1 = new BarPlot(array_slice($datay,0,250000));
	$p1->SetFillColor("red");
	$graph->Add($p1);
	$graph->Stroke();
} ?>
```

Anexo 6.4. Archivo graph_wrap.php

```
<html>
<head>
<title>Gr&aacute;ficas estad&iacute;sticas para PortscanAI</title>
<style>
	body{
		font-family:Cambria, serif;
		font-size:24px;
```

```
                background-color: #F7F5E6;
                color:black;
        }
             a{
                text-decoration:none;
                        font-size: 15px;
                        color: #873AFC;
             }
             a:hover{
                        text-decoration:underline;
                        color:#1515E6;
             }
        table{
                width:95%;
                align:center;
                color:aliceblue;
        }
        td{
                text-align:center;
                font-size:20px;
                background-color:#52658f;
        }
</style>
</head>
<body>
<center>
<?php
include('config.php');
$tipo = $_GET['tipo'];
$ip = $_GET['ip'];
$GRAPH_DIR = $log_path.'/graph_info/';
```

```
echo '<table><tr><td style="font-family:Cambria, serif;font-size:24px;
color:black;"><h2>Graficos de: '.$ip.'</h2></td></tr>

</table>';

if(is_file($GRAPH_DIR.$ip)){ ?>

       <table>

        <tr><td><img src="graph.php?ip=<?php echo $_GET['ip'] ?>&tipo=<?php echo
        $tipo ?>" /></td>

       <?php

       if($tipo == '1') { ?>

              <td><img src="graph.php?ip=<?php echo $_GET['ip'] ?>&tipo=3" /></td>

              <tr><td><img src="graph.php?ip=<?php echo $_GET['ip'] ?>&tipo=5" /></td>

              <td><img    src="graph.php?ip=<?php    echo    $_GET['ip']    ?>&tipo=4"
              /></tr></td></table>

       <?php

       }?>

<?php }

else

       echo '<h2>La IP requerida no ha sido encontrada</h2>

       Por favor, verifique que el archivo de log exista ('.$GRAPH_DIR.$ip.

       '). Es probable que esta IP no haya sido registrada como de

        destino(&uacute;nicamente actu&oacute; como origen).';

       ?>

<table><tr><td><a href="index.php" style="font-family:Cambria, serif;font-size:24px;
color:#F7F5E6;" title="Regresar a consola de
PortscanAI">Regresar</a></td></tr></table>

</center>

</body>

</html>
```

Anexo 7. Configuración de interfaces red

Anexo 7.1. Configuración en Proxmox

Para la convergencia entre la SDN e Snort+RNA es necesario configurar la interfaz física de red eno2 en proxmox PV2, este proceso se divide en dos etapas la primera es en la interfaz gráfica tal como se muestran en la Figura 119 y Figura 120.

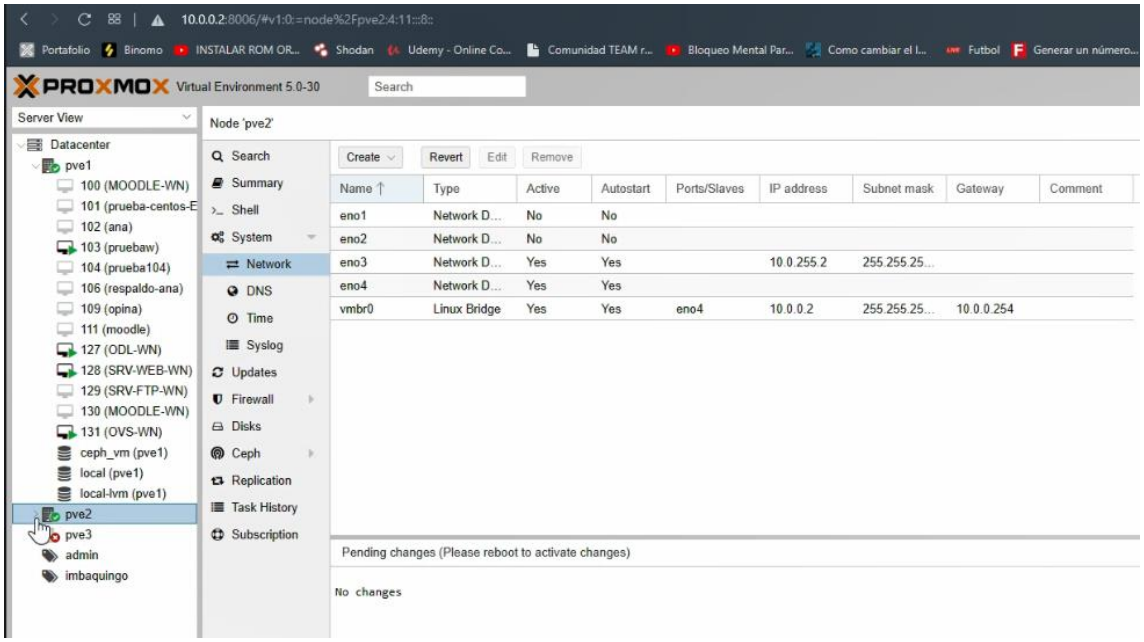


Figura 119. *Selección del servidor proxmox PV2*

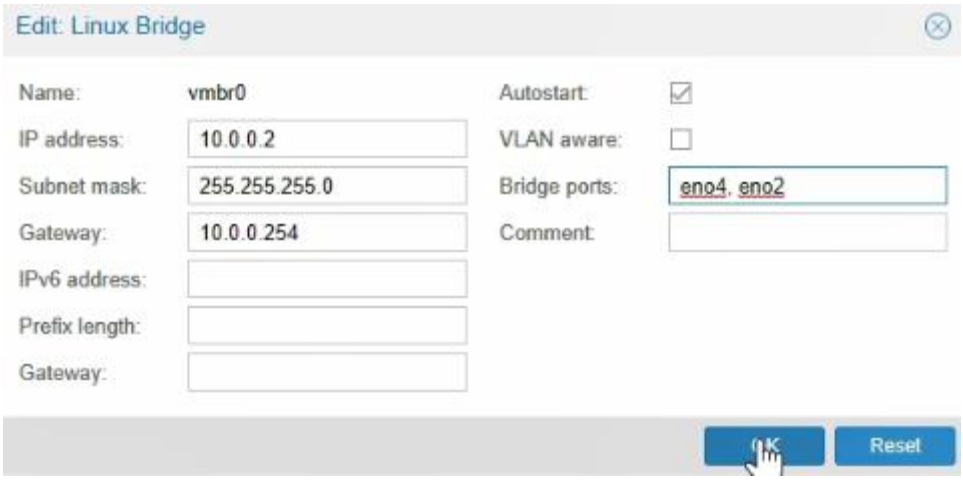


Figura 120. *Parámetros para la interfaz vmbr0 de PV2*

La segunda etapa es la configuración del archivo **interfaces** dentro de la carpeta **/etc/network/**, dentro de este archivo se agregan estas líneas de comando:

```
auto eno2

iface eno2 inet manual
```

Con los cambios realizados solo queda reiniciar al servidor proxmox PV2 para que surtan efecto.

Anexo 7.2. Configuración en VMware

Al igual que en proxmox se configurar una interfaz en VMware con el propósito de que el Snort+RNA en la máquina virtual, se comunique con la SDN del servidor físico PV2; los pasos de este proceso se señalan desde la Figura 121 hasta la Figura 123.

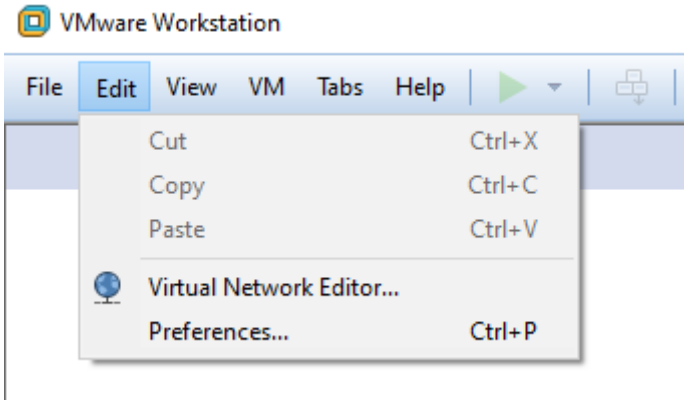


Figura 121. *Configuración del Virtual Network Editor*

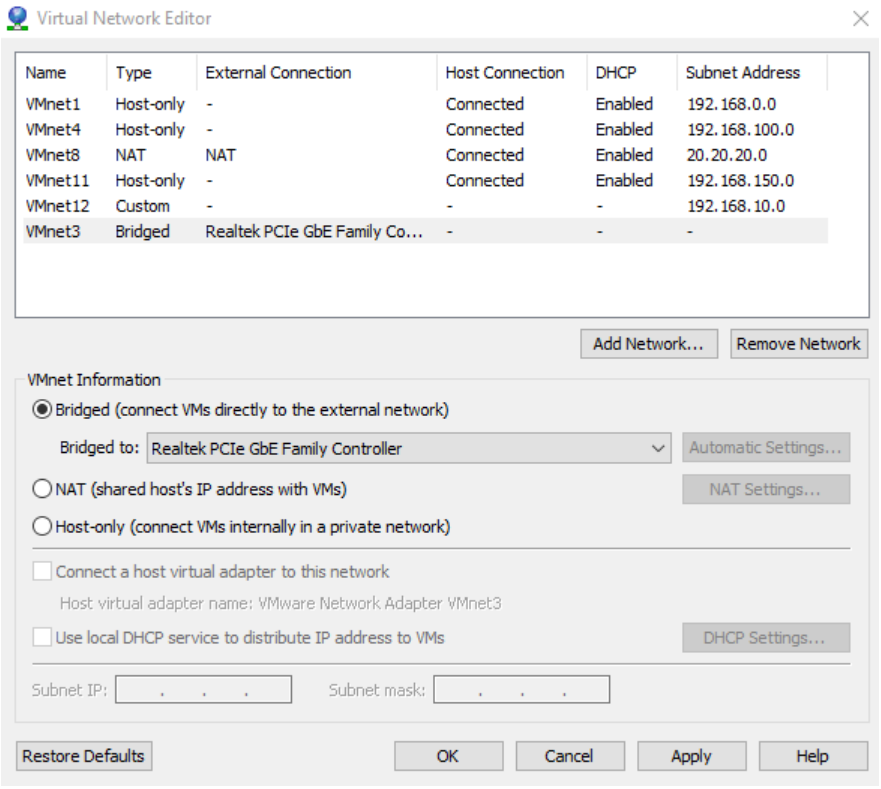


Figura 122. *Parámetros del VMnet3 de VMware*

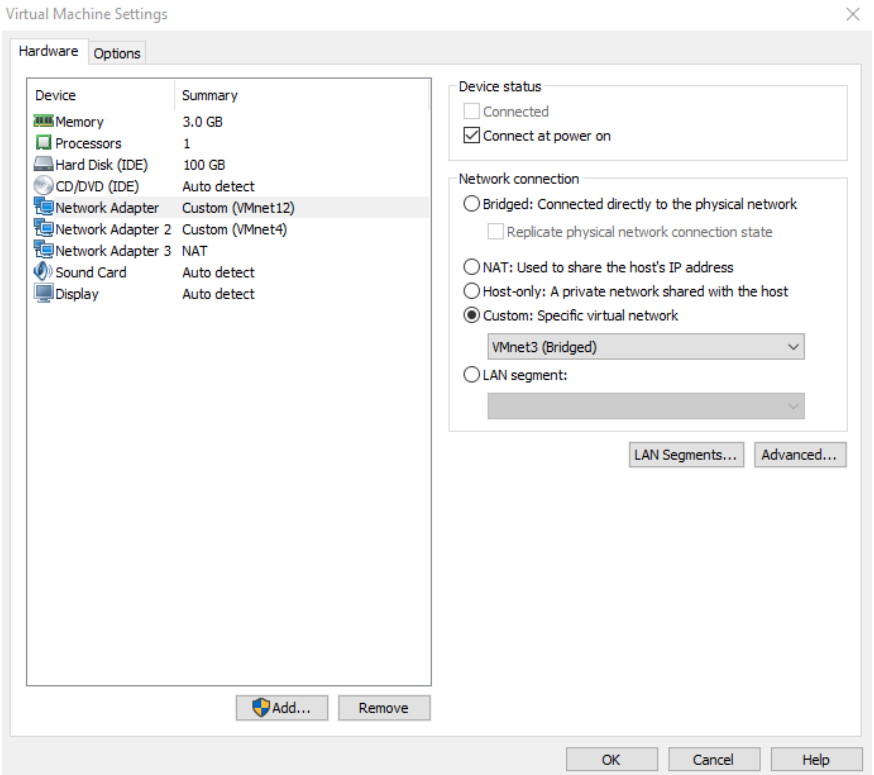


Figura 123. *Asignación de VMnet3 en Snort+RNA*