ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN IPS MEDIANTE HERRAMIENTAS DE DEVOPS

IMPLEMENTACIÓN DE SERVICIOS DE SEGURIDAD EN REDES MEDIANTE DEVOPS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

ANGÉLICA VANESSA GUERRA GONZÁLEZ

DIRECTOR: ING. FERNANDO VINICIO BECERRA CAMACHO

DMQ, Agosto 2024

CERTIFICACIONES

Yo, ANGELICA VANESSA GUERRA GONZALEZ declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ANGELICA GUERRA

angelica.guerra@epn.edu.ec

guerravanessa1999@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por ANGELICA VANESSA GUERRA GONZALEZ, bajo mi supervisión.

FERNANDO BECERRA

DIRECTOR

fernando.becerrac@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

ANGELICA GUERRA

DEDICATORIA

Queridos Dora Piedad y Luis Guerra,

A mis queridos hermanos y sobrino,

A mi tía Margoth González,

A toda mi familia materna González y abuelos maternos,

Y finalmente, a Lana, mi querida mascota,

Esta tesis es el resultado de un viaje lleno de retos y logros, y está dedicada a cada uno de ustedes con profundo agradecimiento y amor. A lo largo de mi carrera, cada uno ha sido un faro de apoyo y amor incondicional. A mis padres, por ser mis pilares inquebrantables y enseñarme la verdadera determinación. A mis hermanos y sobrino, por inspirarme con su pasión y alegría contagiosa. A mi tía Margoth, cuyo constante aliento y apoyo han sido una luz en los momentos más oscuros. A toda mi familia materna y abuelos, por su amor incondicional y creencia en mi capacidad para alcanzar mis sueños. A Lana, mi compañera fiel, quien con su lealtad me ha recordado que la perseverancia y el amor son la clave para superar cualquier obstáculo.

Gracias a cada uno de ustedes por ser mi motivación, mi fortaleza y parte esencial de mi camino. Este logro no solo es mío, sino de todos nosotros, quienes hemos crecido y triunfado juntos.

Con profundo cariño y gratitud eterna,

Vanessa Guerra

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a la Escuela Politécnica Nacional, especialmente a los ingenieros docentes de la ESFOT (Escuela de Formación de Tecnólogos), y a mi dedicado tutor, Fernando Becerra. Su guía experta, paciencia y apoyo incondicional han sido fundamentales en mi desarrollo académico. Agradezco también a mis amigos incondicionales, Jordan, Kevin, Gabriel, Gary, Geomaira y Benedict, por su constante apoyo y compañerismo durante esta travesía académica. Gracias a todos ustedes, mis queridos docentes y amigos, por creer en mí y ser parte esencial de este importante logro en mi vida académica y personal.

ÍNDICE DE CONTENIDOS

RESUMEN

El presente trabajo de integración curricular se centró en implementar un sistema avanzado de seguridad de redes utilizando Snort 3.0 como un Sistema de Prevención de Intrusos (IPS), automatizando todo el proceso mediante Ansible. Se elaboró un playbook detallado que cubrió desde la instalación de los requisitos previos hasta la configuración detallada de reglas usando archivos con formato YAML. La parte teórica exploró conceptos clave de DevOps y la función de Snort en el ámbito de seguridad de redes, así como también conceptos básicos sobre seguridad informática, opciones de automatización, etc.

En la sección de metodología se aseguró que se cumplieran los objetivos del proyecto, describiendo paso a paso cada fase de implementación y prueba. Los resultados obtenidos demostraron la efectividad de las reglas configuradas en Snort mediante pruebas de ataques DoS, escaneo de red e intentos de acceso a puertos como SSH y Telnet, evidenciando la capacidad para detectar y mitigar amenazas tanto conocidas como emergentes.

Es así, como este proyecto combinó teoría y práctica para ofrecer una solución automatizada y robusta que fortalece la seguridad de las redes. El uso de herramientas como Ansible y Snort no solo simplificó la configuración y gestión del sistema de prevención, sino que también mejoró significativamente la capacidad de respuesta frente a posibles vulnerabilidades y ataques cibernéticos.

PALABRAS CLAVE: DevOps, Playbook, Snort, Ansible

ABSTRACT

The present curriculum integration project focused on implementing an advanced network security system using Snort 3.0 as an Intrusion Prevention System (IPS), automating the entire process through Ansible. A detailed playbook was developed, covering everything from installing prerequisites to configuring rules using YAML format files. The theoretical part explored key DevOps concepts and Snort's role in network security, alongside fundamental aspects of cybersecurity, automation options, and more.

In the methodology section, ensuring project objectives were met involved describing each implementation and testing phase step-by-step. Results demonstrated the effectiveness of Snort rules in detecting and mitigating known and emerging threats, including DoS attacks, network scans, and attempts to access ports like SSH and Telnet.

Thus, this project seamlessly combined theory and practice to deliver an automated, robust solution enhancing network security. Leveraging tools such as Ansible and Snort not only streamlined the setup and management of the prevention system but also significantly bolstered responsiveness against potential vulnerabilities and cyberattacks.

KEYWORDS: *DevOps, Playbook, Snort, Ansible*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto se enfoca en implementar Snort como un Sistema de Prevención de Intrusiones (IPS) utilizando Ansible dentro de un entorno DevOps. Ansible juega un papel importante al automatizar la instalación y configuración de Snort en múltiples clientes. La implementación con Ansible asegura una configuración coherente de Snort en múltiples clientes, además de facilitar la gestión centralizada y la escalabilidad del IPS. Los playbooks pueden adaptarse fácilmente para añadir nuevas reglas de detección o actualizar configuraciones, mejorando así la capacidad de respuesta ante amenazas emergentes según se requiera.

1.1 Objetivo general

Implementar servicios de seguridad en redes mediante DevOps.

1.2 Objetivos específicos

- Analizar la herramienta de DevOps que soluciona cada componente del proyecto de titulación.
- Diseñar la solución para cada servicio de seguridad en red mediante herramientas de DevOps.
- Implementar las soluciones mediante DevOps para el despliegue de los servicios de seguridad en red.
- Verificar el funcionamiento de cada servicio de seguridad en una red implementada mediante DevOps.

1.3 Alcance

Este proyecto se enfoca en la utilización de DevOps y el despliegue de soluciones en seguridad informática. El uso de herramientas de DevOps es fundamental para ser capaz de automatizar la implementación de los servicios propuestos y adicionalmente probar las soluciones mediante el análisis de prueba y error. Es así, como en el proyecto se planea incorporar el siguiente componente:

- Implementación de un IPS mediante herramientas de DevOps.

1.4 Marco Teórico

La automatización y la ciberseguridad en la actualidad son un tema relevante en el ámbito de la tecnología, es así como existen varias opciones y herramientas que facilitan

el proceso extenso de instalación, la detección de intrusos y protección en los equipos. Dado que las empresas controlan de forma aislada las vulnerabilidades de sus sistemas de protección, elementos y dispositivos; es sumamente importante incorporar el software y las herramientas adecuadas para evitar dificultades en el entorno y sobre todo llevar pruebas extensas para garantizar un correcto funcionamiento de detección y protección.

Tomando en cuenta lo anteriormente expuesto, existen soluciones de ciberseguridad que también implican la remisión de ciberataques, entre ellos, medidas para asegurar el acceso remoto, proteger contra phishing y spam, monitorear y restringir el tráfico, así como segmentar las redes. [1].

DevOps y Seguridad

Cuando se habla de DevOps, está referido a prácticas y herramientas que conjuntamente hacen el Desarrollo de software (Dev) y operaciones de tecnologías de loa información (Ops). La intersección entre DevOps y Seguridad actualmente destaca en el modo en que las empresas u organizaciones abordan la automatización para acelerar el tiempo de despliegue, desarrollo y mejorar de calidad del Software [2]. A pesar de que DevOps puede ofrecer grandes ventajas en el aspecto de seguridad en las redes, los desarrolladores tienen un enfoque más profundo hacia la velocidad de detección amenazas o ataques, mas no a la seguridad o aseguramiento de datos.

DevOps y Seguridad son dos conceptos importantes que componen a *DevSecOps,* una evolución y colaboración de aplicaciones de software que incluyen la capacidad en tiempo real de analizar datos con operaciones de seguridad [3]. La implementación de herramientas y procesos automatizados fomenta la adopción de prácticas de codificación segura y al mismo tiempo la realización de pruebas de seguridad de forma permanente y regular. Una base importante sobre *DevSecOps* es fomentar la cultura de seguridad en el entorno, así, la seguridad es convertida en una responsabilidad compartida por todos quienes integran la organización.

Linux en Seguridad Informática

En el ámbito de la seguridad cibernética, Linux ocupa una posición destacada, es un sistema operativo ampliamente destacado en funciones de estabilidad, flexibilidad con una amplia gama de herramientas de seguridad accesibles y disponibles para el usuario. Linux entre varias de sus ventajas ofrece en su naturaleza de código abierto una delantera para la seguridad, lo cual permite adaptar opciones de encriptación, medidas

de seguridad, protección de datos, etc., al sistema operativo según sean las necesidades objetivas.

Es así, aunque Linux es considerado más seguro que Windows u otros sistemas operativos, también puede ser utilizado a menudo para necesidad informáticas como el material de estudio por quienes poseen conocimientos técnicos en servidores web y operaciones de red [4].

El bajo riesgo de ciberataques con Linux es una de las razones por la cual también es considerado más seguro puesto que facilita la protección de datos; no obstante, este sistema no evita amenazas internas o negligencias del personal por pérdida de datos, por ello, es sumamente importante las medidas de prevención por parte de las empresas independientemente del sistema operativo que elijan utilizar [5].

Herramientas de Protección y Detección de Intrusos

La supervisión y análisis de tráfico de red necesita de herramientas indispensables para detección de actividades sospechosas o maliciosas. Algunas de ellas son dispositivos o software diseñado para identificar y prevenir intrusiones no autorizadas en sistemas y redes, estos componen varias opciones que proporcionan una defensa amplia contra amenazas cibernéticas. Algunas de estas herramientas que actualmente destacan en el ámbito de detección y protección son los Firewalls, *Security Information and Event Management (SIEM), Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS)*, Antivirus y Antimalware, etc. [6]

Distintas empresas desarrolladoras destacan en las soluciones para protección y detección de intrusos garantizando seguridad y confianza cibernética en el entorno, tal como Cisco, Palo Alto Networks, Fortinet, IBM Security y McAfee.

A continuación, se presentan algunos de los IPS más conocidos en la industria de la ciberseguridad:

Snort 3.0

Snort es una herramienta diseñada para la detección de intrusiones que fue desarrollado por Cisco, gracias a su código abierto ofrece avanzadas opciones de monitoreo y análisis de tráfico de red referente a actividades sospechosas en tiempo real, así como también el análisis de protocoles y detección de ataques. La característica de esta herramienta son el uso de reglas personalizables por el usuario que acorde a las necesidades, estas se adaptan hacia la detección y protección del equipo.

Además, la comunidad de Snort que contribuye con reglas y *plugins* adaptables a las necesidades de los usuarios son un plus a su eficacia [7]. Aunque la versión 3.0 de Snort ofrece nuevas y amplias capacidades de detección de intrusos, su implementación tiene un requerimiento profundo de prácticas de seguridad, por ello, los administradores deben ser cuidadosos en cuanto a las políticas y reglas de Snort para evitar falsos positivos.

Las configuraciones de Snort hacen que tengan 3 modos de funcionalidad:

- **Modo pasivo*/sniffer:*** Los paquetes son capturados en tiempo real y son mostrados continuamente a través de la consola.
- **Modo registro de paquetes*/logger:*** Los paquetes capturados son guardados en un fichero.
- **Modo NIPS:** Los paquetes capturados son comparados con las reglas de detección establecidas por el usuario, que en efecto aparecerán alertas por consola en caso de coincidencia [8].

Reglas en Snort: tipos y función

Las reglas en Snort son de vital importancia para detectar posibles amenazas en el tráfico de red, por lo que es importante configurarlas correctamente para que el IDS/IPS funcione de manera óptima. Para crear reglas personalizadas, es esencial comprender su sintaxis, que se presenta así:

[acción][protocolo][IP origen][puerto origen] -> [IP destino][puerto destino] ([Opciones de regla])

Esta estructura abarca la acción de la regla, el protocolo utilizado, las direcciones IP de origen y destino, y los puertos de origen y destino involucrados.

Algunos elementos tienen conjuntos predefinidos de valores, como la acción, que determina la respuesta de Snort cuando se encuentra un paquete que coincide con la regla. Las opciones disponibles incluyen alerta, registro, paso, eliminación, rechazo y eliminación segura. Además, el protocolo indica el tipo de protocolo que se está analizando: *TCP, UDP, ICMP o IP*. Las direcciones IP y los puertos de origen y destino, junto con el operador de dirección, especifican los orígenes y destinos del tráfico.

Las opciones finales en la regla definen los criterios de detección de Snort y se clasifican en cuatro categorías: generales, de carga útil, no de carga útil y de post-detección. Las opciones generales ofrecen información sobre la regla sin afectar la detección, mientras que las de carga útil buscan datos dentro del contenido del paquete. Por otro lado, las opciones no de carga útil se centran en parámetros como el tiempo de vida o el tamaño

del contenido del paquete, y las de post-detección están relacionadas con eventos específicos que ocurren después de la activación de una alerta.

Suricata

Suricata al igual que Snort, es una herramienta de detección de código abierto, sus modos de configuración actúan tanto como IDS/IPS, este detecta y bloquea paquetes que coincidan con reglas establecidas por el usuario en la red. Los modos de operación en los que opera son pasivo y activo. El modo pasivo actúa como observador no intrusivo, es válido para monitorizar y recopilar información sobre las posibles amenazas sin afectar el rendimiento del tráfico. Por otro lado, el modo activo previene y mitiga amenazas identificadas, además de que envía alertas como respuesta rápida en tiempo real [9].

OSSEC

OSSEC a diferencia de Suricata y Snort, es una plataforma centrada en la detección de amenazas a nivel de red, enfocada únicamente en la protección de sistemas individuales (HIDS). Los agentes que posee monitorean registros, archivos, actividad y otros aspectos en exploración de signos o sospechas maliciosas atacantes. Además, OSSEC brinda una interfaz intuitiva y opciones de personalización que se adaptan a las necesidades de cada usuario [10].

IPS (Intrusion Prevention Systems)

Un Intrusion Detection System (IDS) cumple la función de supervisar y detectar amenazas o actividades sospechosas, generando alertas que, al ser analizadas por un centro de operaciones de seguridad en respuesta, investiga el problema, únicamente limitándose a eso.

Los IPS son una extensión de los *Intrusion Detection Systems (IDS)*, que adicionalmente tienen la característica de bloquear tráfico y realizar tareas similares a los cortafuegos. Es así, como se define a un IPS como un software o dispositivo que detecta y actúa proactivamente en contra del tráfico malicioso en una red o equipo.

Tipos de IPS

Los IPS son clasificados según su ubicación [11]. Uno de los más comunes es el ***Host-based IPS (HIPS)***, únicamente aplica las funcionalidades en único Host. Dicho sistema detecta y previene todas las actividades sospechosas en el host.

El ***Network-based IPS (NIPS)*** es otra clasificación, esta previene posibles ataques a través de la monitorización de red o subred, toma medidas proactivas con el fin de bloquear o mitigar aquellas amenazas en tiempo real.

En semejanza al NIPS, el ***Wireless-network-based IPS (WIPS)*** se enfoca en redes inalámbricas, proporcionando una capa adicional de protección contra ataques dirigidos a dispositivos y puntos de acceso Wi-Fi.

Por último, el ***Network Behavior Analysis (NBA)*** es un tipo de IPS se centra en estudiar el tráfico de red en busca de desviaciones basadas en ciertos criterios, tales como la tasa de paquetes por segundo o la cantidad de conexiones por dispositivo.

Funcionamiento, ventajas y desventajas

La estructura del IPS, o Sistema de Prevención de Intrusiones, tiene la función de analizar el tráfico de red en tiempo real para posteriormente enviar los eventos detectados para su análisis adicional a un analizador designado. Este analizador se encarga de examinar y filtrar los eventos de registros generados por el IPS, y luego los transfiere al servidor de registros (*log server)* para su almacenamiento y referencia futura como se indica en la Ilustración 1 [12].

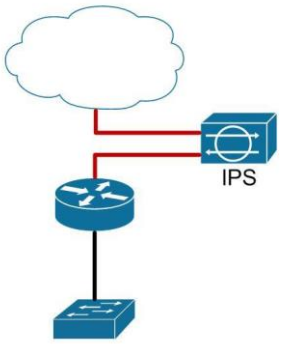

Figura 1.1. Estructura de funcionamiento de un IPS [12].

Entre las ventajas del IPS se destaca su capacidad para detener paquetes de datos en tiempo real, incluso interceptando ataques que consisten en un solo paquete. Además, puede emplear técnicas de normalización de red para optimizar el análisis y la detección de amenazas.

No obstante, el IPS también presenta algunas desventajas a considerar. Por ejemplo, los falsos positivos y la complejidad de configuración y mantenimiento, ya que al ser un sistema amplio con minuciosas configuraciones para la protección del equipo se requiere de conocimientos técnicos para su funcionamiento.

Oracle VM Virtual Box

Oracle VM Virtual Box es un software capaz de virtualizar máquinas ejecutables en diversos sistemas operativos, capacidad de disco duro, memoria RAM, entre otras características. Desarrollado por *Oracle Corporation*, actualmente su última versión 6.x ofrece grandes ventajas de virtualización, al ser un programa gratuito y tener una interfaz manejable para el usuario es una herramienta indispensable para simular aspectos de red en sistemas Linux, u otros SO sin afectar y comprometer el estado del equipo real [13].

Herramientas de Automatización

La automatización es un concepto tecnológico que se aplica en la realización de tareas con ausencia de intervención manual o humana. Es implementado en sectores donde se lleven a cabo tareas repetitivas o extremadamente largas, por ello, la automatización toma un papel fundamental en el sentido de eficiencia, productividad y flexibilidad que, en consecuencia, también reduce costos, tiempo y errores humanos.

Ansible

La plataforma de automatización Ansible es respaldada por RedHat, diseñada para configurar sistemas, desplegar software y realizar tareas de computación avanzadas. Además de abastecer máquinas virtuales, contenedores y redes, Ansible está pensado para todo tipo de usuarios, desde desarrolladores hasta administradores de sistemas.

Para utilizar Ansible, se necesitan dos tipos de ordenadores: el nodo de control, que ejecuta Ansible, y los nodos gestionados, que son los dispositivos gestionados por el nodo de control. Ansible opera conectándose a estos nodos en una red y enviando módulos Ansible, pequeños programas que representan un estado deseado del sistema, a cada nodo [14]. Estos módulos se ejecutan a través de *SSH* y se borran al finalizar, lo que simplifica su funcionamiento y gestión.

Playbook de Instalación

Los módulos Ansible definen el estado deseado de un sistema, permitiendo automatizar tareas en múltiples ordenadores de manera eficiente. Un *playbook* de Ansible, escrito en YAML, proporciona instrucciones sobre cómo poner un nodo gestionado en el estado deseado. Estos *playbooks* son fáciles de entender y autodocumentados, lo que facilita su uso incluso para usuarios sin experiencia en programación. Ansible ofrece una solución robusta y flexible para la gestión de infraestructuras de TI, contribuyendo al movimiento de *"infrastructure as code"* y permitiendo la automatización de tareas críticas en entornos empresariales [14].

El propósito de un *playbook* de instalación, como se ha mencionado anteriormente, es optimizar el tiempo utilizado en descargar archivos, ejecutar comandos de instalación y ajustar configuraciones adicionales. La automatización mediante un *playbook* facilita una instalación más eficiente y uniforme en múltiples sistemas.

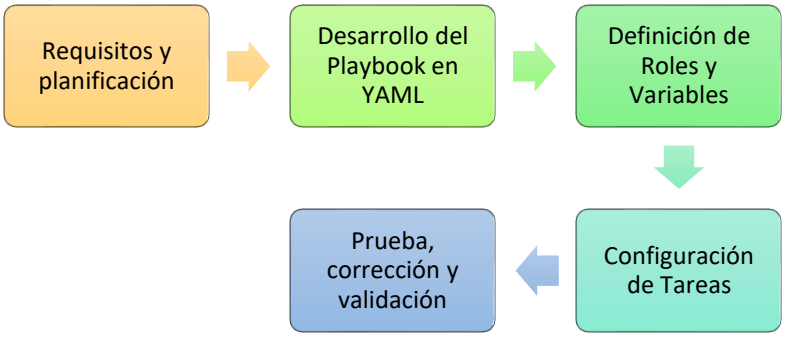

Figura 1.2. Proceso de creación de un *playbook* de instalación en Ansible.

YAML y DevOps

YAML es un formato de serialización de datos empleado en el diseño de archivos de configuración, valorado por su sencillez y facilidad de comprensión. Este lenguaje es popular por su legibilidad y capacidad para trabajar junto con otros lenguajes de programación. Ansible lo aprovecha para crear procesos de automatización mediante sus *playbooks*. Los archivos YAML, que suelen tener extensiones *.yml* o *.yaml*, siguen reglas de sintaxis específicas y permiten una estructuración clara y concisa de los datos.

Los *playbooks* de Ansible, escritos en YAML, organizan los procesos de TI mediante la definición de estados deseados del sistema y la ejecución de tareas automatizadas mediante módulos de Ansible. La sintaxis tiene mucho que ver con el proceso de automatización del *playbook* [15]. En el Cuadro 1 se detallan algunas de las funciones más empleadas en la creación de *playbooks* en Ansible.

Tabla 1.1. Funciones básicas de un archivo YAML para Ansible [15].

Lenguaje YAML	**Función**
hosts	Especifica los hosts sobre los que se ejecutará el playbook.
tasks	Define las tareas.
name	Etiqueta descriptiva de una tarea.
apt	Administra paquetes en sistemas derivados en Debian/Ubuntu.
yum	Administra paquetes en sistemas derivados en Red Hat/CentOS.
copy	Copia archivos locales a los hosts remotos.

file	Gestiona archivos y directorios en los hosts remotos.
user	Gestiona usuarios en el sistema remoto.
group	Gestiona grupos de usuarios en el sistema remoto.
shell	Ejecuta comandos de shell en los hosts remotos.

Puppet

Como herramienta de automatización, Puppet es utilizada para configuraciones y administraciones de sistemas de servidores según las necesidades del usuario. Se lo hace a través de un lenguaje de configuración declarativo, así, se prepara un archivo de configuración para luego ejecutarlo en el sistema por medio de un agente que aplica la configuración de forma automática. Su arquitectura permite automatizar la configuración en varios sistemas, de esta manera su implementación hace más fácil el monitoreo y control de problemas en el entorno de *TI* [16].

Tácticas, Técnicas y Procedimiento (TTP) de Atacantes

Los ciberdelincuentes tienen ciertos patrones y estructuras de ataque, lo que da lugar a combinaciones de TTP. Las tácticas son las conductas que los atacantes llevan a cabo para cumplir el objetivo de perpetrar a la red o cumplir con el ciberataque. Las técnicas son aquellas tareas para desarrollar para conseguir la táctica, estas pueden ser de varios tipos, tales como tareas maliciosas programadas, *Powershell*, ficheros y archivos maliciosos, etc. Por último, los procedimientos son los pasos premeditados que el ciberdelincuente despliega para conseguir el ataque, uno de los mas conocidos son el *Phishing*, basado en el envío de correos electrónicos maliciosos para obtener datos o información de la víctima [17].

Comprender estos Métodos, Tácticas y Procedimientos (TTP) brinda a los profesionales especializados en ciberseguridad una comprensión para estar listos ante las amenazas presentes y futuras, así como para responder y reducir su impacto de manera efectiva.

2 METODOLOGÍA

Según el plan de trabajo de titulación, el proyecto se organizó en secciones definidas con el fin de alcanzar los objetivos planteados. En primera instancia, se realizó una investigación sobre las herramientas de automatización DevOps y su aplicación en la seguridad de redes, especialmente en IPS. Esta investigación proporcionó el conocimiento necesario para entender cómo estas herramientas funcionan y cómo se pueden utilizar para automatizar servicios de seguridad en redes.

En la siguiente fase, se diseñó una solución para implementar Snort como un IPS en la red. Se empleó una máquina virtual en VirtualBox utilizando el Sistema Operativo Ubuntu como servidor y cliente y Kali Linux como atacante para realizar pruebas y asegurar el correcto funcionamiento de Snort como IPS dentro de un entorno realista. También se desarrolló un *playbook* dinámico que contiene reglas específicas para la implementación efectiva de Snor en archivos YAML.

En la tercera fase, que se centra en la implementación de soluciones utilizando herramientas DevOps, se procedió con la instalación de Ansible en el servidor y la configuración de las herramientas de programación necesarias. Se ajustó el archivo de *host* y se verificó el acceso SSH entre el servidor y el cliente para preparar la implementación del IPS.

Posteriormente, en la cuarta fase, se ejecutó la implementación del *playbook* dinámico diseñado para desplegar Snort como IPS. Se asegurará la correcta comunicación SSH entre todas las partes involucradas y se ejecutarán los comandos necesarios para implementar Snort de manera eficiente y segura.

Para finalizar, se llevaron a cabo pruebas con el fin de verificar el funcionamiento correcto de Snort como IPS en el cliente. Se probaron diversos ataques hacia el cliente y se registraron y evaluaron los resultados con éxito de todo el proceso y las elecciones realizadas a lo largo de su desarrollo.

3 RESULTADOS

Este apartado expone la implementación y gestión automatizada de un Sistema de Prevención de Intrusiones (IPS) usando Ansible dentro del ámbito de la seguridad en redes. En un entorno práctico, se simula un ataque cibernético en una red local, generando tráfico malicioso desde una máquina atacante. El IPS, configurado con Snort 3.0, monitorea y analiza este tráfico, produciendo alertas sobre actividades sospechosas. La utilización de Ansible para gestionar y actualizar el IPS garantiza una respuesta rápida y eficiente ante las amenazas detectadas. Este método no solo fortalece la seguridad de la red, sino que también ilustra cómo la automatización puede revolucionar la gestión de la seguridad en redes actuales.

A continuación, se analiza el funcionamiento del playbook de instalación con Ansible, los pasos necesarios para automatizar la configuración y despliegue de las herramientas y dependencias esenciales. Seguidamente, se aborda el diseño del Sistema de Prevención de Intrusiones (IPS) utilizando Snort 3.0, describiendo la arquitectura del

sistema y las reglas de detección diseñadas para identificar actividades sospechosas en la red. Luego, se detalla la implementación y despliegue del IPS en un entorno Linux, explicando cómo se configura Snort y cómo se integra con otros componentes de seguridad. Por último, se realiza la comprobación del funcionamiento del IPS mediante diversas pruebas, garantizando que el sistema detecte y responda eficazmente a los intentos de intrusión y al tráfico malicioso simulado.

3.1 Análisis de la herramienta de DevOps con Ansible y Snort

El objetivo de DevOps va enfocado a la producción de nuevos servicios/productos con el fin de optimizar el uso del tiempo y los recursos. Automatizar la red de datos implica hacer que la gestión, configuración, pruebas e implementación de dispositivos en la red sean automáticas. Esto significa que no existirá acciones manuales en cada proceso. Esto trae ventajas, como que los servicios se activen más rápido, visualizar todas las aplicaciones de manera clara y que los errores que suelen ocurrir por culpa del personal reduzcan.

Ansible

Basado en el marco teórico, Ansible es caracterizado por ser una herramienta libre y sencilla, además de partir del ejecutamiento de archivos YAML para compilación de *playbooks*, un lenguaje con sintaxis sencillo de entender y aprender. La sobrecarga de SSH lo hace menos escalable, por lo que no es eficiente para gestionar números grandes de nodos.

En contraste con Puppet, que es otra herramienta muy utilizada en el ámbito de herramientas DevOps, maneja el lenguaje de configuración declarativo, así, se prepara un archivo de configuración para luego ejecutarlo en el sistema por medio de un agente que aplica la configuración de forma automática [16]. Puppet resulta ser más complejo de usar, pues al ser un modelo agente-servidor, existen nodos únicamente gestionados por Puppet lo que lo hace limitado y dependiente. No obstante, esta herramienta es conocida por ser escalable y capaz de administrar grandes unidades de servidores eficientemente.

Así, los principales beneficios de usar Ansible son:

- Reduce el uso de recursos para la gestión de TI al permitir el control de múltiples máquinas desde un único nodo controlador.

- Facilita la automatización a través de *playbooks* escritos en formato YAML, un lenguaje comprensible para los usuarios.
- Emplea el protocolo SSH para establecer conexiones seguras y eficientes con los hosts remotos.

Snort 3.0

Snort es un sistema de prevención de intrusiones (IPS) que funciona bajo un modelo de código abierto que se especializa en detectar y responder rápidamente a amenazas en tiempo real mediante el análisis del tráfico de red. Con su habilidad para procesar grandes volúmenes de datos con baja latencia, Snort utiliza reglas personalizables para identificar patrones de comportamiento malicioso y emitir alertas precisas [8].

Por otro lado, OSSEC es una plataforma de detección de intrusos y prevención de amenazas (IDPS) que se centra en la monitorización exhaustiva de logs, archivos y actividades del sistema. Utiliza agentes instalados en los hosts para recopilar datos, los cuales son enviados a un servidor central donde se analizan en busca de comportamientos anómalos y posibles intrusiones [10]. Snort y OSSEC son dos herramientas ampliamente utilizadas en el campo de la seguridad informática, cada una con enfoques distintos pero complementarios.

En términos de funcionalidad, Snort se destaca por su capacidad para examina el tráfico de red en busca de firmas particulares y comportamientos sospechosos, proporcionando una defensa efectiva para entornos donde la seguridad del tráfico de red es sumamente importante. Por otro lado, OSSEC ofrece una visión integral al detectar amenazas a nivel de host y proporcionar capacidades avanzadas de correlación de eventos y gestión de incidentes.

La elección entre Snort y OSSEC depende de las necesidades específicas de seguridad. Snort es ideal para entornos que requieren una respuesta rápida y precisa ante amenazas en el tráfico de red, mientras que OSSEC es más adecuado para entornos donde se necesita una monitorización detallada de la integridad del sistema y la detección de amenazas a nivel de host.

3.2 Diseño del IPS mediante Snort 3.0 y Ansible

Red e interfaces en máquinas virtuales

En el marco de un entorno virtualizado en VirtualBox, se lleva a cabo una configuración altamente técnica de la red para facilitar la comunicación eficiente y segura entre las diversas máquinas virtuales, en este caso, el servidor, el cliente y el atacante. Este

proceso es fundamental para establecer un ambiente controlado que permita realizar pruebas de seguridad y monitorear la red de forma efectiva.

Para diseñar un sistema de prevención de intrusiones (IPS) con Ansible y Snort 3.0, se deben establecer configuraciones específicas para cada máquina virtual. Tanto el servidor como el cliente ejecutarán Ubuntu 22.0 de 64 bits con 4096 MB de RAM asignados cada uno. Esta asignación de memoria se justifica por la necesidad de gestionar eficazmente las reglas de Snort y las tareas administrativas controladas por Ansible, garantizando un rendimiento óptimo durante la detección y respuesta ante intrusiones.

Además, se debe incluir una máquina virtual adicional para el que deberá realizar los ataques de prueba (atacante), utilizando Kali Linux como sistema operativo y configurada también con 2048 MB de RAM. Esta configuración uniforme de recursos asegura que los ataques simulados sean realistas y efectivos contra el IPS implementado en el servidor Ubuntu.

A continuación, se presentan las características de las máquinas virtuales en VirtualBox para su debida implementación:

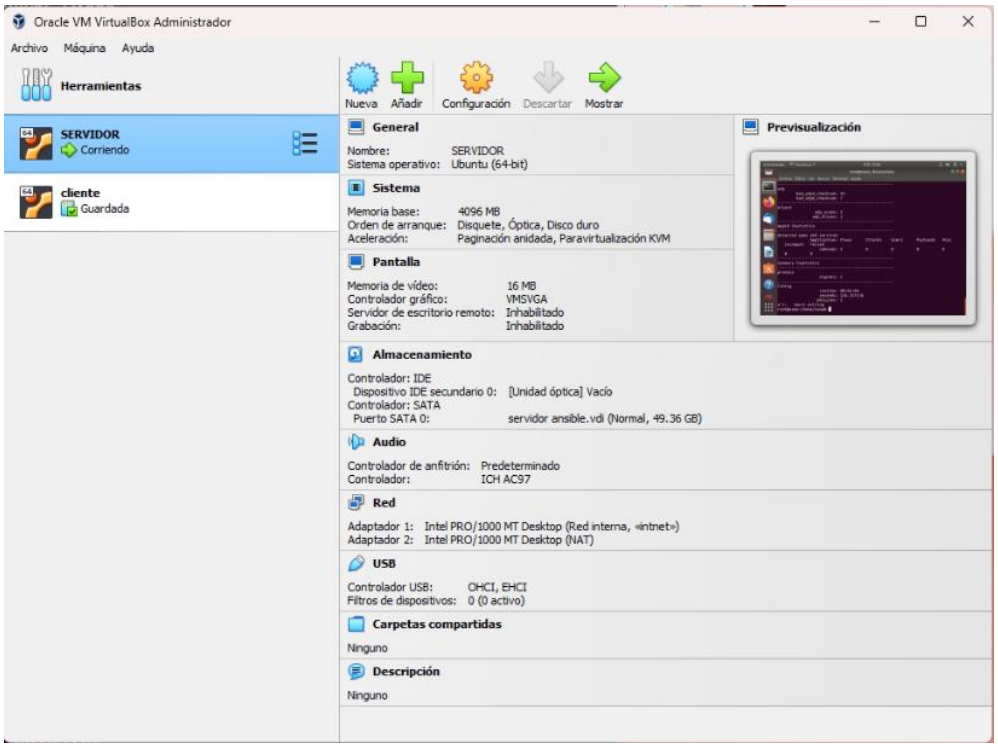

Figura 3.1. Características de máquina virtual en el servidor

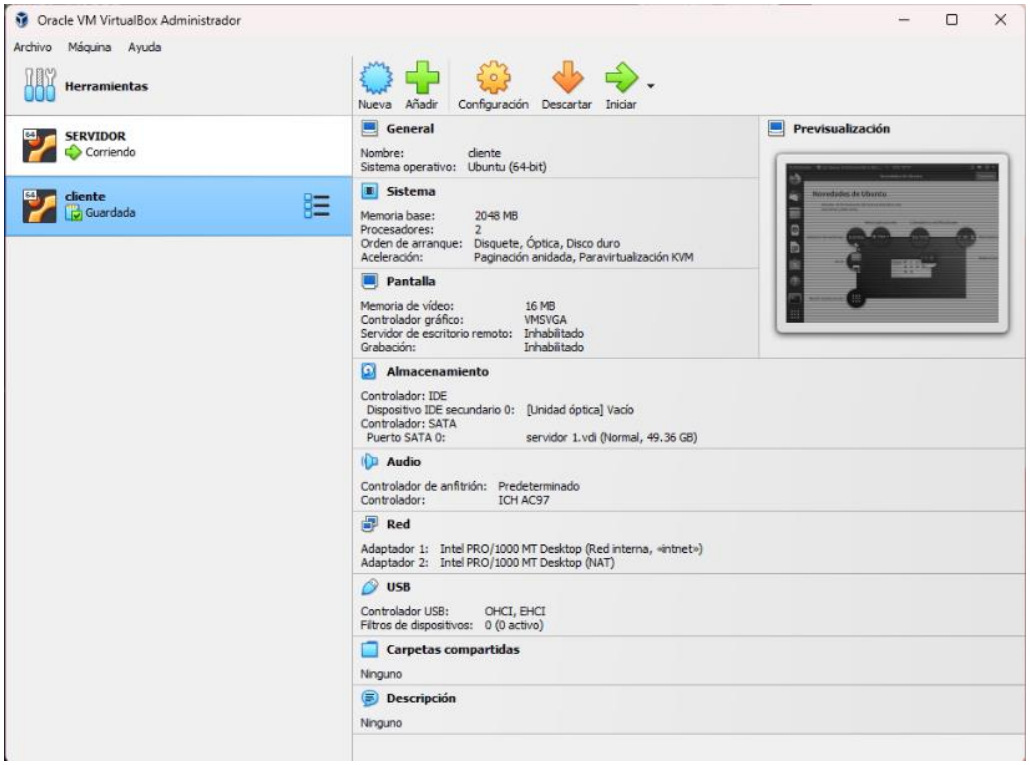

Figura 3.2 Características de máquina virtual en el cliente

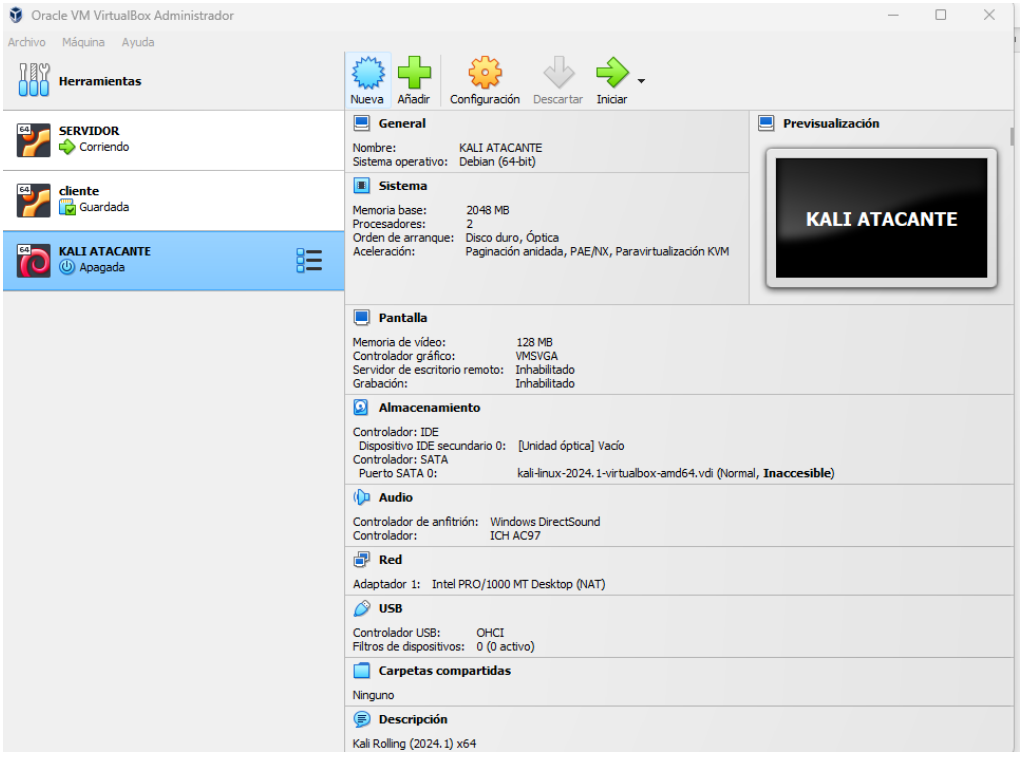

Figura 3.3. Características de Máquina Virtual del atacante

Seguidamente, se necesita configurar dos adaptadores en cada máquina virtual. En la **Tabla 3.1** Modos de Red en Máquinas Virtuales: Configuraciones y Descripciones.**Tabla 3.1** se justifican el tipo de red que se utilizan:

Tabla 3.1 Modos de Red en Máquinas Virtuales: Configuraciones y Descripciones.

Modo de Red	Descripción
Adaptador NAT	Conecta la máquina virtual directamente a la red física, permitiendo la comunicación directa con otros dispositivos en la misma red y acceso a Internet.
Red Interna	Crea una red privada entre las máquinas virtuales, aislada del entorno externo, facilitando la comunicación interna exclusiva.

3.3 Implementación y despliegue del IPS en Linux mediante herramientas de DevOps

A continuación, se procede con la configuración detallada de las interfaces de red y las direcciones IP en el servidor Ubuntu 18.02 utilizando el archivo Netplan de la siguiente forma:

Tabla 3.2. Configuración de Interfaz de Red en Máquinas Virtuales

Interfaz	Dirección IP	Descripción
enp0s3 (Ubuntu Desktop1) enp0s8	192.168.10.10/24 DHCP (Red NAT, salida a internet)	Interfaz de Red Interna para comunicación interna. Servidor.
enp0s3 (Ubuntu Desktop2) enp0s8	192.168.10.11/24 DHCP (Red Nat salida a internet)	Interfaz de Red Interna para comunicación interna. Cliente.
enp0s3 (Kali Linux)	192.168.10.22/24 DHCP (Red Nat salida a internet)	Interfaz de Red Externa para prueba de ataques y DoS. Atacante.

Esta infraestructura técnica IP permite simular un escenario realista, de manera que la comunicación entre los tres (Ubuntu Desktop1 como servidor, Ubuntu Desktop2 como cliente, y Kali Linux como atacante) pueda ser gestionada internamente a través de una red privada con dirección IP 192.168.10.0/24. Esto facilita la interacción y prueba de diferentes configuraciones de red, desde la comunicación interna exclusiva entre máquinas virtuales hasta la simulación de escenarios de ataque desde una red externa controlada.

Configuración de Ansible

Cuando se trabaja con sistemas operativos basados en Linux como nodo administrador de Ansible, se debe tener presente que los paquetes necesarios de Ansible y SSH no vienen preinstalados por defecto. Por lo tanto, es fundamental realizar la instalación de estos paquetes de manera manual.

Para instalar Ansible, se ejecuta el comando `*sudo apt install ansible*` en el terminal. Es importante destacar que esta instalación debe hacerse con privilegios de administrador para asegurar el acceso completo a los recursos del sistema. De igual forma, para instalar el servicio SSH se ejecuta el comando *'sudo apt install ssh'.*

```
root@vane:/etc/ansible/roles# sudo apt install ansible
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
ansible ya está en su versión más reciente (2.5.1+dfsg-1ubuntu0.1).
Los paquetes indicados a continuación se instalaron de forma automática y ya no
 son necesarios.
  courier-authdaemon courier-authlib courier-authlib-userdb courier-base
  expect gir1.2-goa-1.0 gir1.2-snapd-1 libcourier-unicode4 libfam0 libtcl8.6
  tcl-expect tcl8.6
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 3 no actualizados.
```

Figura 3.4. Instalación de Ansible mediante comandos

```
root@vane:/etc/ansible/roles# sudo apt install ssh
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
ssh ya está en su versión más reciente (1:7.6p1-4ubuntu0.7).
Los paquetes indicados a continuación se instalaron de forma automática y ya no
 son necesarios.
  courier-authdaemon courier-authlib courier-authlib-userdb courier-base
  expect gir1.2-goa-1.0 gir1.2-snapd-1 libcourier-unicode4 libfam0 libtcl8.6
  tcl-expect tcl8.6
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 3 no actualizados.
```

Figura 3.5 . Instalación de SSH mediante comandos

Los directorios principales de Ansible se localizan en la ruta `*/etc/ansible*`. En estos directorios es donde se almacenan los archivos de configuración y los roles que se utilizan para automatizar las tareas de administración de sistemas. Estos directorios son

importantes para el correcto funcionamiento de Ansible y deben ser manejados con precaución para evitar posibles errores o pérdida de datos.

Configuración de SSH

Ansible hace uso de un archivo de inventario de hosts, ubicado típicamente en */etc/ansible/hosts*. Para acceder y editar este archivo, se emplea el comando *sudo nano /etc/ansible/hosts*. Este archivo detalla los dispositivos a gestionar, comenzando con divisiones de secciones entre corchetes, seguido de la especificación de nombres para identificar los hosts, tal como se ilustra en la Figura 3.6. Además, dentro de estas secciones, se definen variables necesarias para el control de los dispositivos, como contraseñas, nombres de usuario, tipo de comunicación, entre otros. En este caso se definen las variables y host.

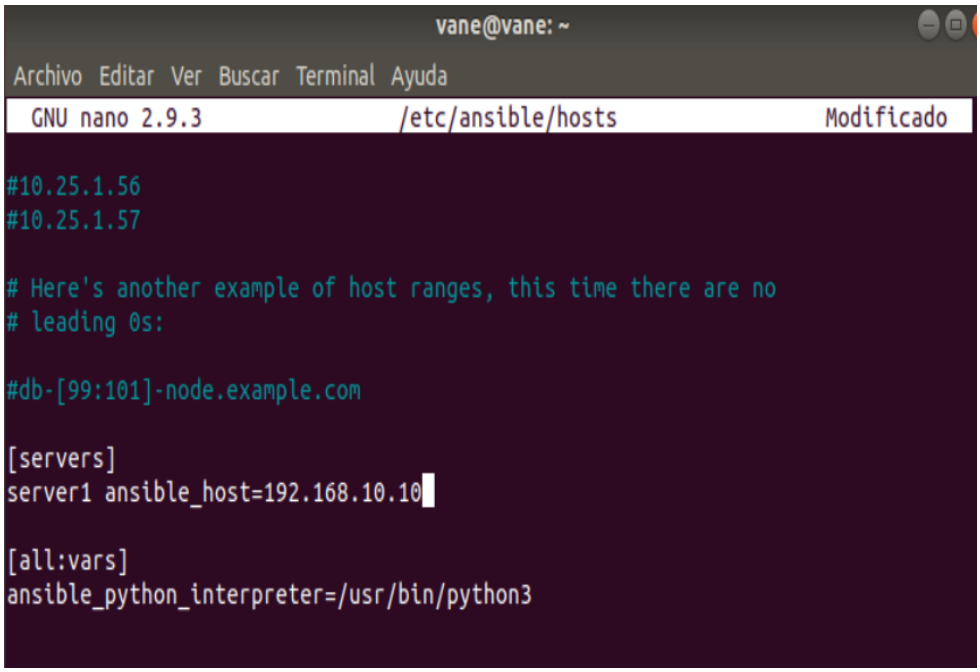

Figura 3.6. Configuración del Archivo de Inventario de Hosts de Ansible y Definición de Variables

Antes de seguir con la configuración del archivo *ssh_config,* es importante comprobar si el servicio de SSH está activo y funcionando. Una vez instalado con el comando *apt-get install SSH*, se accede al archivo *sudo nano /etc/ssh/sshd_config* para confirmar que ciertas líneas estén habilitadas. Esto es importante para que el servidor de Ansible pueda autenticar mediante claves.

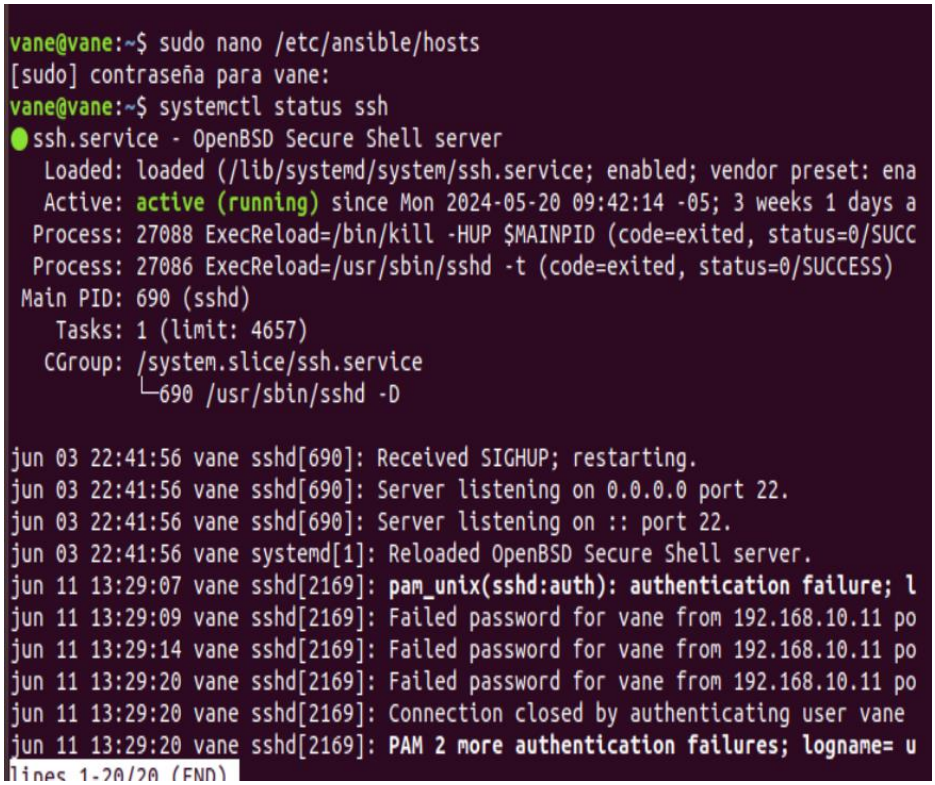

Figura 3.7. Verificación y Configuración del Servicio SSH antes de la Configuración de sshd_config

En el host, es muy necesario activar el protocolo SSH para establecer la comunicación con el servidor de Ansible. El nodo controlador debe tener las claves SSH. Para generarlas, se utiliza el comando *ssh-keygen,* el cual genera un par de claves RSA por defecto. El algoritmo RSA emplea un cifrado asimétrico basado en la dificultad de factorizar grandes números primos, asegurando un alto nivel de seguridad. Estas claves se guardan en el archivo *id_rsa,* ubicado en */home/vane/.ssh/id_rsa.* El resultado de este proceso se muestra en la Figura 3.8.

Figura 3.8. Generación de Claves SSH RSA para Establecer Comunicación con el Servidor de Ansible

Para concluir la configuración de la conexión SSH entre el nodo controlador y el cliente, se procedió a transferir la clave pública SSH generada en el nodo controlador al cliente mediante el comando *ssh vane2@192.168.10.11.* En la Figura 3.9 se puede observar la ejecución de este comando y la confirmación del inicio de sesión exitoso en el dispositivo.

Figura 3.9. Transferencia de Clave SSH y Confirmación de Inicio de Sesión Exitoso

Por último, en Ansible, se verifica que si la conexión se ha establecido correctamente, el envío de un paquete de ping al host remoto generará una respuesta llamada "pong", confirmando así que la solicitud fue recibida y procesada adecuadamente con el comando *ansible all -m ping -u vane2.*

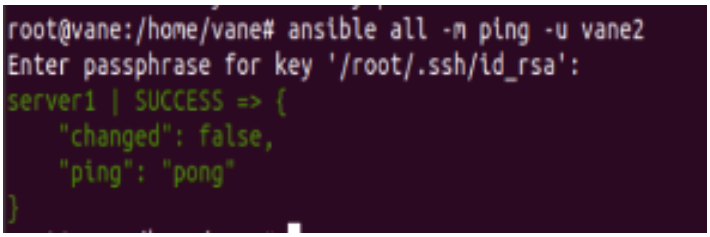

Figura 3.10. Verificación de Conexión y Respuesta 'pong' en Ansible

Playbook de Instalación de Snort 3.0 con Ansible

El *playbook* de Ansible permite la automatización completa de la configuración inicial del IPS [14]. Este proceso incluye la instalación de dependencias esenciales como bibliotecas de soporte y herramientas adicionales necesarias para el funcionamiento óptimo de Snort 3.0. La automatización con Ansible no solo acelera significativamente la configuración, sino que también reduce el tiempo necesario de horas a minutos.

Para lograr esto, se definen pasos en un archivo YAML dentro de la máquina en donde se ejecutará el sistema, especificando todas las tareas necesarias, desde la actualización del sistema hasta la instalación de paquetes específicos. Cada tarea en el *playbook* se válida para asegurar que se complete con éxito antes de proceder a la siguiente, garantizando así una configuración robusta y libre de errores.

A continuación, se presenta un diagrama de flujo creado en PSeInt sobre la instalación automatizada con Ansible en Linux, el mismo pseudocódigo que se adjuntarán en la sección de Anexos.:

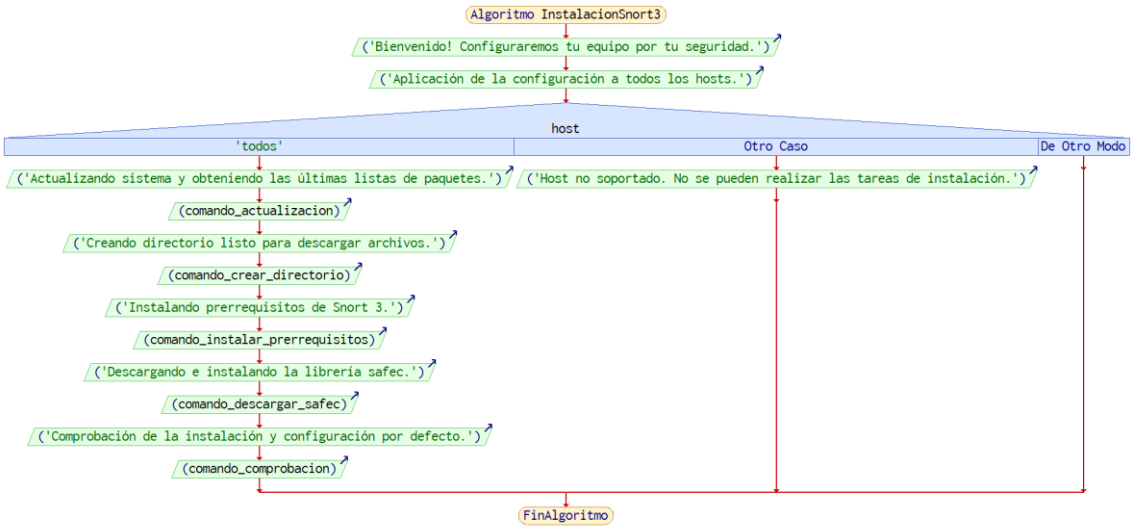

Figura 3.11. Diagrama de bloque del *playbook* de instalación

El despliegue de Snort 3.0 y sus dependencias mediante Ansible se realiza de manera estructurada y eficiente. El *playbook* incluye tareas para la descarga e instalación de Snort, descompresión de archivos esenciales, creación de directorios, instalación de bibliotecas, etc. Este enfoque estructurado asegura que todas las herramientas y dependencias estén configuradas correctamente y listas para su uso inmediato.

Playbook de instalación de Snort 3.0

El *playbook* que se detallará a continuación describe el proceso completo para instalar Snort 3.0, una herramienta avanzada de detección y protección contra intrusiones. Este *playbook* está diseñado específicamente para sistemas basados en Debian/Ubuntu y automatiza todo el proceso de instalación, asegurando que todas las dependencias necesarias para Snort se configuren adecuadamente. A través de una serie de pasos detallados, desde la instalación de prerrequisitos hasta la configuración y verificación final, se garantizará que Snort esté preparado para proteger redes contra amenazas de seguridad de manera efectiva y eficiente.

Para empezar, se crea un archivo YAML llamado "*main.yaml*" en el directorio del servidor */etc/ansible/roles* donde se ejecutará el *playbook* por medio del comando ***ansible-playbook main.yaml*** para la correcta instalación en los host que sean necesarios, en este caso, el cliente con IP *192.168.10.11/24.* Es así como se empieza con la elaboración del *playbook* a través del comando **"*nano main.yaml*".**

Primero, una de las principales características y propósito de cada sección diseñado del playbook es la definición de un nombre (**name)** descriptivo para la tarea principal del *playbook,* dando la bienvenida al usuario para darle un toque interactivo. Seguidamente

la sección **“hosts**” especifica que las tareas definidas se aplicarán a todos los hosts listados en el inventario de Ansible y **“become”** que indica que las tareas se ejecutarán en modo superusuario (sudo). En la sección de "**tasks",** se localiza una lista detallada de pasos que deben llevarse a cabo en cada host de forma secuencial. Cada uno de estos pasos utiliza el módulo *shell* de Ansible para ejecutar comandos en los servidores remotos.

La Figura 3.12 muestra el archivo ***main.yaml***, donde se detallan las tareas y la estructura principal del *playbook* utilizado para desplegar Snort 3.0 en los sistemas operativos de los hosts remotos.

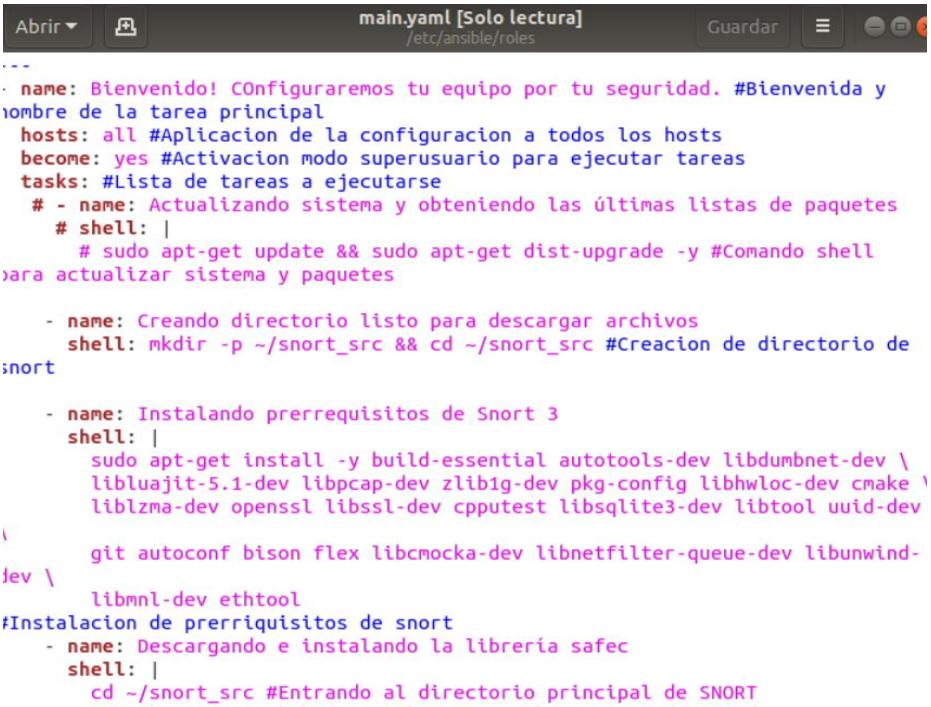

Figura 3.12. Contenido de archivo main.yaml para la ejecución del playbook de instalación de Snort 3.0

En la Figura 3.13 se muestra la primera parte del playbook de instalación en donde comienza con la tarea de bienvenida aplicada a todos los hosts, con el modo super usuario y un mensaje de “WARNING” que Ansible anticipa antes de ejecutar la creación de directorios.

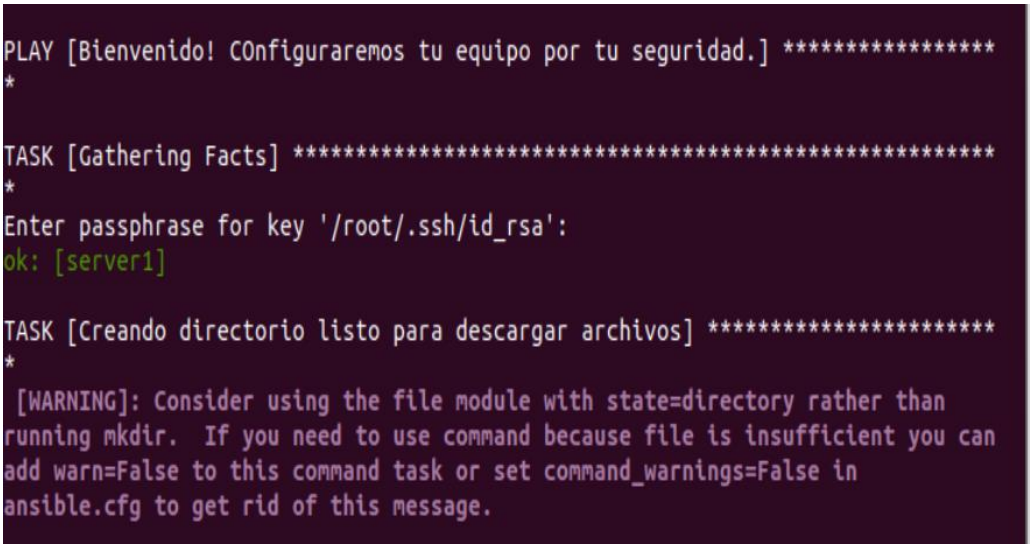

Figura 3.13. Primera ejecución del playbook de instalación

Durante la elaboración y desarrollo del playbook, se proponen tareas de descarga e instalación de pre-requisitos para que Snort tenga una ejecución exitosa. La librería *Safec* asegura que el software opere establemente, reduce la probabilidad de fallos relacionados con errores de programación en C. Seguidamente se añade la tarea de instalación de *PCRE (Perl Compatible Regular Expressions)* que proporciona las herramientas necesarias para la detección de actividad sospechosa en el tráfico de red. Luego, la tarea de descarga de herramientas de rendimiento *gperftools* desarrollada por Google, que cumple la función de optimizar el uso de memoria durante la ejecución del programa. Después, la instalación de *Ragel,* que proporciona capacidades de análisis de protocolos mediante la generación de analizadores de estados finitos.

Boost se emplea para mejorar el rendimiento de funciones críticas dentro de Snort, mientras que DAQ se encarga de la captura de datos de redes. *Hyperscan* proporciona una forma eficiente de realizar análisis de expresiones regulares, esencial para el procesamiento rápido de patrones en el tráfico de red. Snort, por su parte, es la herramienta principal para la detección y protección de intrusiones en sistemas de red. Y finalmente, la biblioteca *libtcmalloc-minimal4* mejora la gestión de la memoria, optimizando el rendimiento general de Snort durante su funcionamiento.

Es así, como el *playbook* termina con la ejecución del comando que servirá para verificar la correcta instalación y funcionamiento en el sistema.

Configuración de Snort 3.0

El archivo *Snort.lua* ubicado en el directorio */usr/local/etc/snort* contiene la configuración general de Snort 3.0, mismo archivo que debe ser configurado con el fin de adaptar los

hosts remotos para llegar a la automatización requerida y ejecutar los *playbooks.* Existen parámetros importantes como la sección de "*configure defaults"* que contiene las variables "HOME_NET" y "EXTERNAL_NET" en el que se ajusta la red intena y la red externa respectivamente según sea la necesidad del usuario, en este caso, el cliente con dirección IP "192.168.10.11/24" para que así, los *playbook* de instalación y de reglas de Snort se realicen en el host. En la variable "EXTERNAL_NET" se añade la línea "$HOME_NET" o "any" para referirnos a que cualquier dirección de red externa que no pertenezca a la entidad, sea reconocida. En la Figura 3.14 se muestra la sección de configuración ya mencionada:

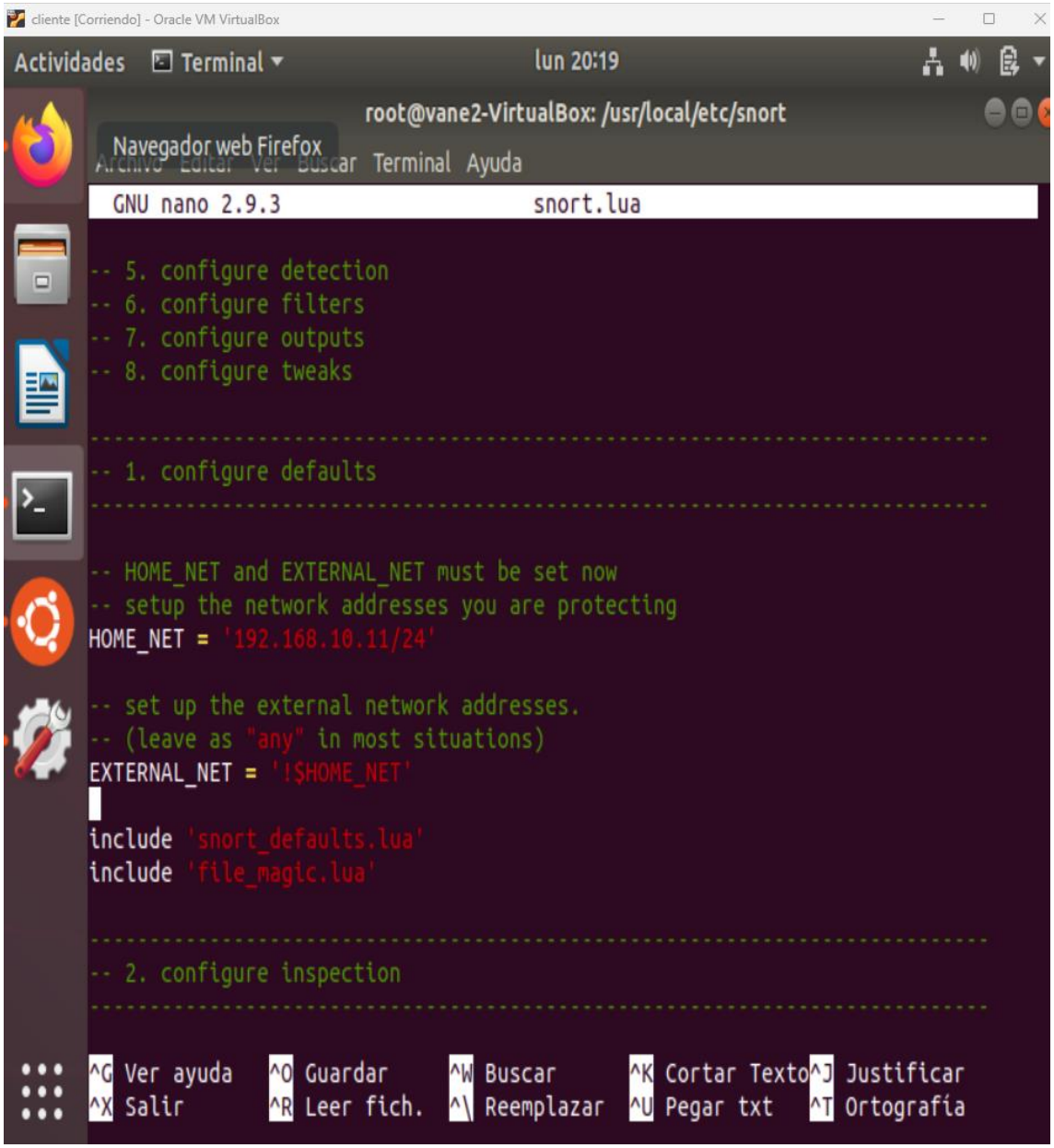

Figura 3.14. Archivo snort.lua con la configuración de direcciones de red.

Las líneas en el archivo que comienzan con “include” se refieren a una directiva que permite incluir otros archivos de configuración dentro del archivo principal, en este caso, se incluyeron los archivos por defecto “*snort_defaults.lua”* que contiene las funciones por defecto recomendadas para la detección de intrusiones, procesamiento de paquetes, configuración de alertas, etc. *y “file_magic.lua”* que permitirá la detección de archivos mediante firmas o patrones determinados dentro del tráfico de red.

Playbook de creación de reglas en Snort 3.0

El *playbook* está configurado para ejecutarse en todos los hosts (hosts: all) y utilizar privilegios de superusuario (become:yes) al igual que el *playbook* de instalación. Usa el módulo “blockinfile” de Ansible para añadir o modificar reglas en el archivo */usr/local/etc/rules/local.rules.*

Estas reglas abarcan la detección y bloqueo de diversas actividades maliciosas:

- Alerta y bloquea ataques ICMP y TCP dirigidos a la red local ($HOME_NET).
- Detecta conexiones SSH y Telnet que no están permitidas.
- Maneja posibles ataques SYN hacia un servidor web particular.

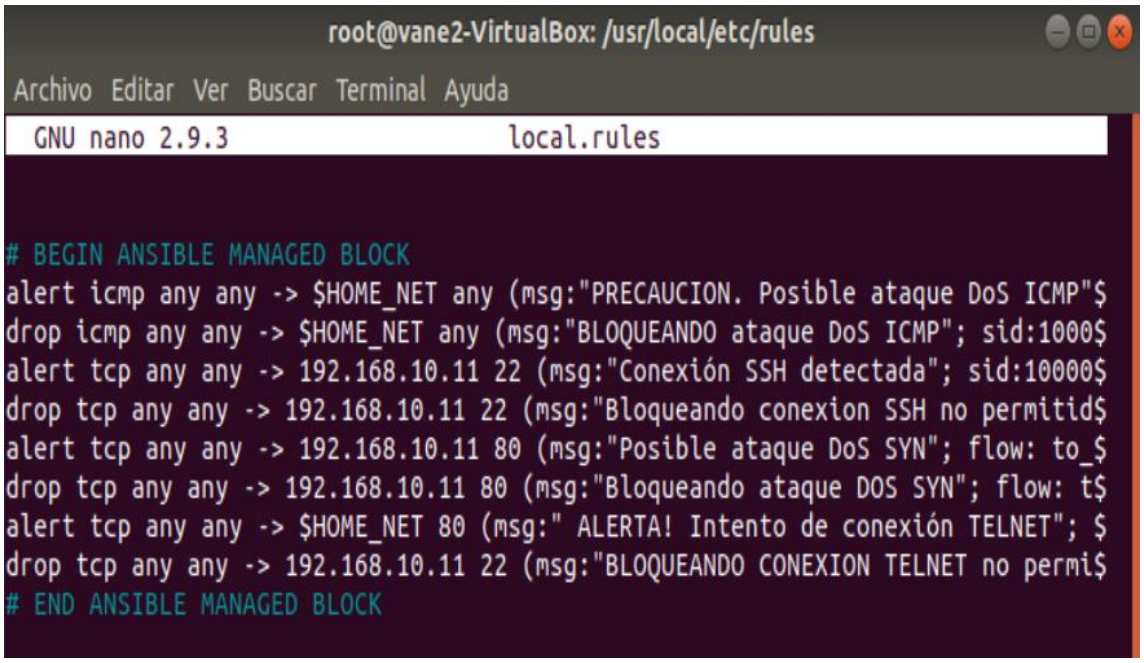

Figura 3.15. Archivo local.rules modificado por el modulo “blockinfile” en el cliente

3.4 Comprobación de funcionamiento del IPS

Ejecución del playbook de instalación y creación de reglas de Snort

Para ejecutar el *playbook* se utiliza el comando "*ansible-playbook main.yaml"* en modo superusuario, donde "*main.yaml"* es el nombre del archivo en formato *.yaml* creado para la instalación de Snort 3.0 y sus dependencias. Para asegurar la correcta implementación de todas las tareas, se incluyen pasos de verificación en el *playbook* de Ansible. Estos pasos consisten en la validación de la correcta instalación de los paquetes, la comprobación de los servicios en ejecución y la verificación de la funcionalidad de Snort. En la siguiente figura se muestra el proceso de tareas exitosas que se obtuvo del *playbook* y la comprobación de instalación de Snort versión 3.0 con el comando "*Snort -V*".

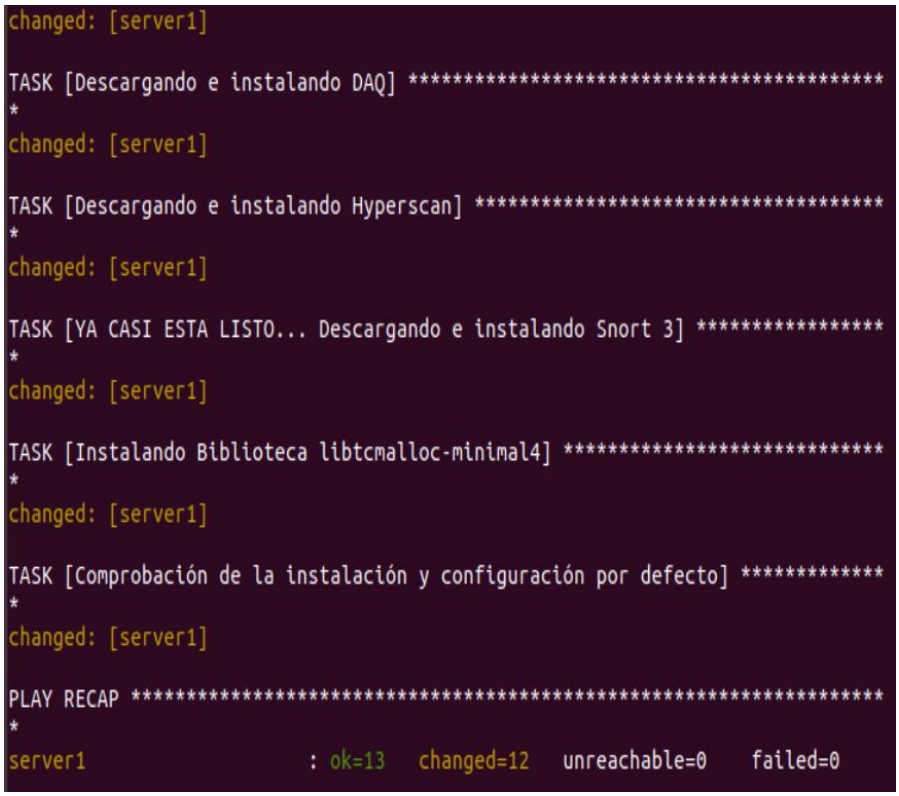

Figura 3.16. Playbook de instalación ejecutado correctamente en el cliente

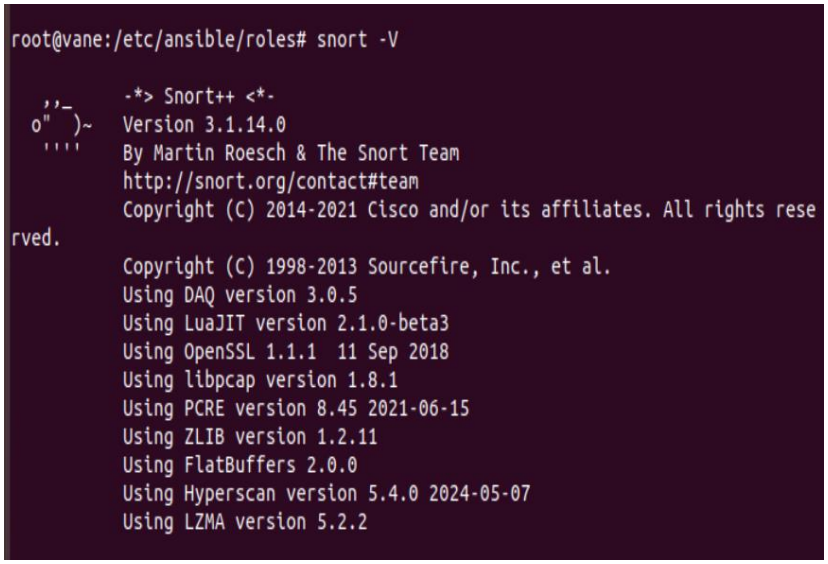

Figura 3.17. Comprobación de instalación de Snort versión 3.0 usando Snort -V

De igual forma, se ejecuta el *playbook* de creación de reglas de Snort creadas para el IPS. La Figura 3.18 muestra el proceso de ejecución correcta y culminada del *playbook.*

Figura 3.18. Ejecución del playbook de Reglas de Snort 3.0 con Ansible

Prueba mediante Ataques

Iniciar las pruebas del IPS con Snort 3.0 es un paso importante para verificar la efectividad de las políticas de seguridad implementadas y garantizar que el sistema esté correctamente configurado para detectar y mitigar los distintos tipos de ataques.

Es así como se llevará a cabo la prueba utilizando las primeras dos reglas, diseñadas para detectar y bloquear posibles intentos de ataques o exploraciones a través de ICMP

dirigidos hacia los sistemas internos, en este caso, el cliente. Para ejecutar la prueba, se simulará un ataque de Denegación de Servicio (DoS) utilizando el comando `***sudo hping3 --icmp --flood 192.168.10.11***` desde la máquina del atacante. Esto enviará paquetes ICMP a la máxima velocidad posible, simulando un ataque DoS, como se ilustra en la Figura 3.19. Posteriormente, en la Figura 3.20 se puede observar cómo se detecta y bloquea el ataque, generando alertas correspondientes como respuesta inmediata.

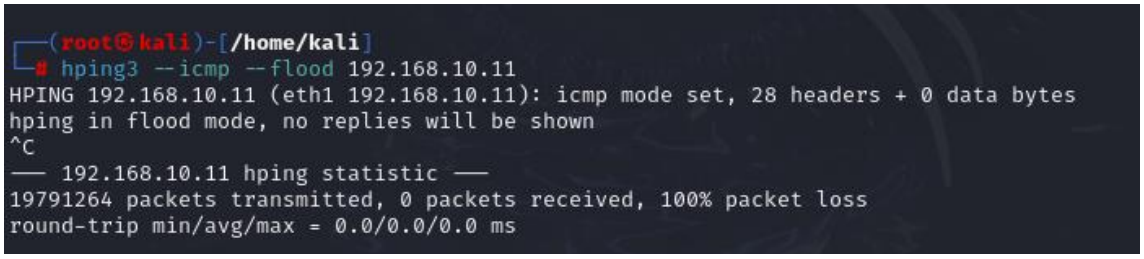

Figura 3.19. Simulación de ataque DoS enviando paquetes múltiples ICMP al cliente.

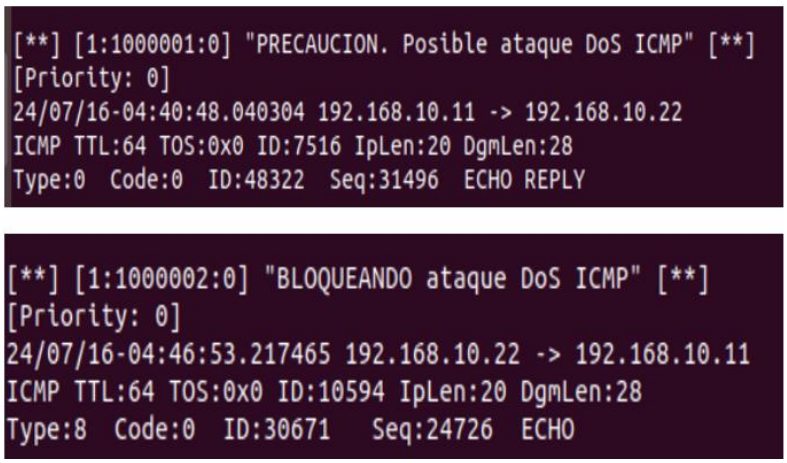

Figura 3.20. Alerta y bloqueo de ataque generado por Snort en el cliente.

A continuación, se llevará a cabo la prueba de un ataque de penetración SSH dirigido a la máquina del cliente, utilizando las reglas tres y cuatro. Estas reglas están configuradas para detectar y bloquear cualquier intento de conexión SSH no autorizado. El ataque se ejecutará mediante el comando ***"ssh vane2@192.168.10.11"*** desde la máquina del atacante, como se muestra en la Figura 3.21. Como respuesta al ataque, se verifica que Snort ha generado correctamente alertas y bloqueado el acceso, como se detalla en la Figura 3.22.

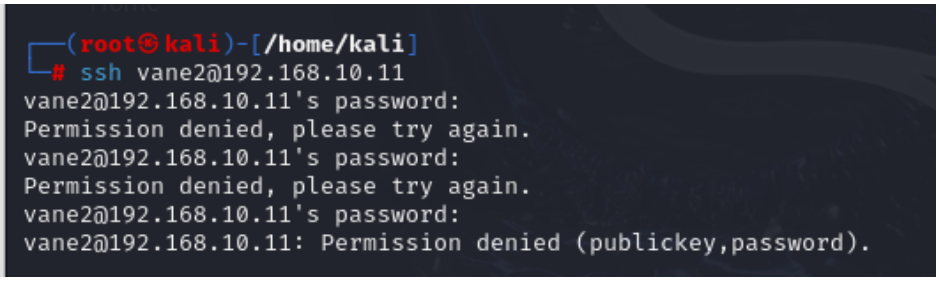

Figura 3.21. Prueba de Ataque de Penetración SSH

Figura 3.22. Detección y Bloqueo de Intento de Conexión SSH no Autorizado por Snort

Finalmente, para llevar a cabo pruebas de ataques mediante conexión vía Telnet, se utiliza el comando "telnet 192.168.10.11", como se ejemplifica en la Figura 3.23. Además, en la Figura 3.24 se observa cómo Snort detecta la actividad, generando alertas y bloqueando el acceso al puerto para prevenir la conexión exitosa al cliente de manera efectiva.

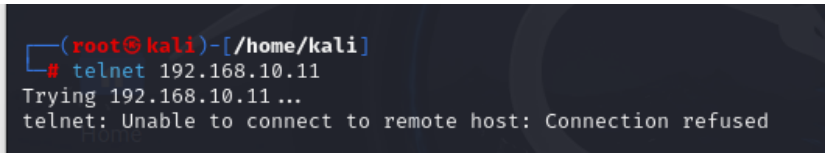

Figura 3.23. Prueba de Conexión vía Telnet a 192.168.10.11

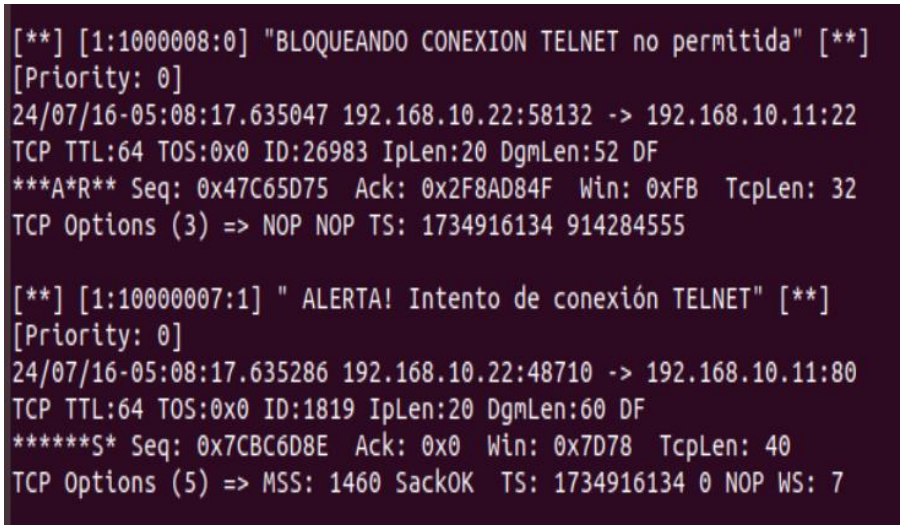

Figura 3.24. Detección y Bloqueo de Intento de Conexión Telnet por Snort

4 CONCLUSIONES

- El estudio de DevOps ha sido fundamental para la implementación eficiente de sistemas de prevención de intrusos (IPS) en infraestructuras de red. DevOps no solo optimiza la velocidad y eficiencia en el despliegue de IPS, sino que también fortalece la seguridad al integrar prácticas continuas de desarrollo, pruebas y despliegue. Esto permite una gestión ágil y adaptable de las políticas de seguridad, asegurando que las redes puedan responder de manera efectiva a las amenazas emergentes y a los cambios en el entorno cibernético.
- La implementación de sistemas de prevención de intrusiones (IPS) como Snort se beneficia enormemente del enfoque de DevOps, donde herramientas como Ansible destaca por su capacidad de desplegar rápidamente y su compatibilidad amplia con diversos sistemas operativos, lo que simplifica la configuración y gestión de la seguridad en redes.
- Snort como IPS proporcionó una capa adicional de seguridad, detectando y mitigando intrusiones como la detección de ataques DoS, intentos de acceso no autorizado a través de SSH y Telnet antes de que puedan comprometer la integridad de la red.
- Los *playbooks* desarrollados para la instalación de Snort mediante Ansible no solo automatizaron el proceso, sino que también aseguraron una implementación coherente y reproducible en múltiples entornos.
- Los archivos YAML utilizados para definir las reglas de Snort ofrecieron una estructura flexible y modular para la configuración del IPS. Esto permite ajustar

y personalizar las reglas según los requisitos específicos de seguridad de la red, adaptándose rápidamente a nuevos patrones de tráfico o amenazas emergentes.

- La utilización del protocolo SSH fue esencial para la instalación de Ansible, ya que asegura una comunicación segura y encriptada. Esto fue evidente durante la transferencia de la clave SSH, la cual estableció un canal protegido con el cliente host, garantizando la confidencialidad y la integridad de los datos durante todo el proceso de instalación.
- Se pudo comprobar que la gestión de dispositivos finales es viable a través de Ansible, aunque requiere un enfoque secuencial. El proceso comienza con la generación de claves SSH, compartidas con el host que se va a administrar para la autenticación. Esto facilita la ejecución de playbooks que contienen tareas específicas adaptadas a cada dispositivo final.
- Ubuntu se destaca como una excelente opción para implementar sistemas de prevención de intrusiones (IPS) como Snort, gracias a su fiabilidad, flexibilidad y soporte constante. La capacidad de Ubuntu para adaptarse fácilmente a entornos de servidor y su compatibilidad con diversas herramientas de seguridad permiten configuraciones avanzadas y adaptadas a las necesidades específicas de seguridad. Esto asegura que los sistemas IPS puedan ser desplegados de manera eficiente y optimizados para proteger redes contra intrusiones maliciosas de manera efectiva.

5 RECOMENDACIONES

- Es recomendable que antes de configurar la conexión SSH entre el servidor y el cliente, asegurarse de que los servicios de SSH estén activos y funcionando correctamente en ambos sistemas.
- Para configurar SSH adecuadamente, se recomienda ajustar los parámetros necesarios en el archivo de configuración ssh para permitir un intercambio de claves sin errores, según las necesidades específicas del entorno.
- Considerar ejecutar Snort con privilegios mínimos utilizando herramientas como “sudo” o configurando adecuadamente los permisos de archivo y directorio. Esto ayuda a mitigar vulnerabilidades potenciales y protege la integridad del sistema.
- Antes de implementar nuevas reglas en Snort, validar su efectividad y precisión en un entorno de prueba. Se recomienda utilizar herramientas como ***snort -T -c /path/to/snort.conf*** para verificar la sintaxis y la lógica de las reglas sin

activarlas. Esto ayuda a evitar falsos positivos y asegura que las reglas funcionen según lo esperado.

- Antes de ejecutar un *playbook,* se recomienda realizar pruebas utilizando la opción ***--syntax-check*** para verificar la sintaxis de los *playbooks* sin ejecutarlos, como por ejemplo: ***ansible-playbook --syntax-check vane.yaml***

6 BIBLIOGRAFÍA

[1] A. S. Iranzo, 7 diciembre 2021. [En línea]. Available: https://m.riunet.upv.es/bitstream/handle/10251/178959/Soucase%20-%20Implementacion%20de%20un%20Sistema%20de%20Prevencion%20de%20Intrusiones%20IPS%20en%20un%20modelo%20de%20red%20indus....pdf?sequence=1&isAllowed=y. [Último acceso: 21 mayo 2024].

[2] N. H. J. &. K. G. Forsgren, Accelerate: The Science of Lean Software and DevOps. IT Revolution Press., 2018.

[3] «Keeper,» [En línea]. Available: https://www.keepersecurity.com/es_ES/resources/glossary/what-is-devops-security/. [Último acceso: 26 Mayo 2024].

[4] «kaspersky,» [En línea]. Available: https://latam.kaspersky.com/resource-center/definitions/linux. [Último acceso: 26 mayo 2024].

[5] C. Moldovan, «endpointprotector,» 24 mayo 2022. [En línea]. Available: https://www.endpointprotector.es/blog/linux-y-la-seguridad-de-los-datos-los-mitos-retos-y-soluciones/. [Último acceso: 27 mayo 2024].

[6] «BECOMIT,» [En línea]. Available: https://becomit.com/sistemas-de-deteccion-y-prevencion-ids-e-ips/#:~:text=Tanto%20los%20Sistemas%20de%20Detecci%C3%B3n,datos%2C%20para%20detectar%20patrones%20sospechosos.. [Último acceso: 27 mayo 2024].

[7] «CISCO,» Comprensión de las reglas de Snort3, 26 octubre 2022. [En línea]. Available: https://www.cisco.com/c/es_mx/support/docs/security/ios-intrusion-prevention-system-ips/218349-understand-snort3-rules.html#:~:text=Snort%20es%20el%20motor%20Cisco,de%20contenido%20y%20detectar%20ataques.. [Último acceso: 27 mayo 2024].

[8] adastra, «The hacker way,» 1 julio 2011. [En línea]. Available: https://thehackerway.com/2011/07/01/conceptos-basicos-y-configuracion-de-snort-%E2%80%93-parte-i/. [Último acceso: 27 mayo 2024].

[9] C. Cilleruelo, «Keepcoding,» 8 septiembre 2022. [En línea]. Available: https://keepcoding.io/blog/que-es-suricata-en-ciberseguridad/. [Último acceso: 27 mayo 2024].

[10] A. Chema, «UN INFORMÁTICO EN EL LADO DEL MAL,» 11 Noviembre 2020. [En línea]. Available: https://www.elladodelmal.com/2020/11/ossec-el-mundo-del-ids-intrusion.html. [Último acceso: 27 mayo 2024].

[11] Albert, «A2SECURE,» 12 febrero 2019. [En línea]. Available: https://www.a2secure.com/blog/ids-ips-hids-nips-siem-que-es-esto/. [Último acceso: 27 mayo 2024].

[12] «Tech for progress,» [En línea]. Available: https://techforprogress.blogspot.com/2014/04/ips-sistema-de-prevencion-de-intrusos.html#:~:text=Ventajas%3A%20Se%20puede%20parar%20paquetes,el%20rendimiento%20de%20la%20red.. [Último acceso: 27 Mayo 2024].

[13] «KEEPcoding,» 22 agosto 2022. [En línea]. Available: https://keepcoding.io/blog/que-es-virtualbox/. [Último acceso: 27 mayo 2024].

[14] «DataScientest,» [En línea]. Available: https://datascientest.com/es/ansible-la-herramienta-de-automatizacion-preferida-por-los-devops#:~:text=Ansible%20es%20una%20plataforma%20de,permanentes%20sin%20tiempo%20de%20inactividad.. [Último acceso: 27 Mayo 2024].

[15] «RedHat,» 3 marzo 2023. [En línea]. Available: https://www.redhat.com/es/topics/automation/what-is-yaml. [Último acceso: 27 mayo 2024].

[16] O. Fernandez, «Codigo Electronica,» 21 octubre 2022. [En línea]. Available: http://codigoelectronica.com/blog/que-es-puppet. [Último acceso: 27 mayo 2024].

[17] «Cyber Zaintza,» [En línea]. Available: https://acortar.link/BDjLsN. [Último acceso: 27 mayo 2024].

7 ANEXOS

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. XX de XXXX de 2022

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UN IPS MEDIANTE HERRAMIENTAS DE DEVOPS asociado al IMPLEMENTACIÓN DE SERVICIOS DE SEGURIDAD EN REDES MEDIANTE DEVOPS elaborado por la estudiante ANGÉLICA VANESSA GUERRA GONZÁLEZ de la carrera en Tecnología Superior en Redes y Telecomunicaciones, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 11%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

4.- Vanessa-turnitin.pdf

Atentamente,

ING. Fernando Becerra

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

Pseudocódigo de PSeint de instalación de Snort

```
Algoritmo InstalacionSnort3
    Escribir ('Bienvenido! Configuraremos tu equipo por tu seguridad.')
    Escribir ('Aplicación de la configuración a todos los hosts.') // Bienvenida y nombre de la tarea principal
    // Tareas de instalación
    Según host Hacer
        'todos':
            Escribir ('Actualizando sistema y obteniendo las últimas listas de paquetes.')
            Escribir (comando_actualizacion)
            Escribir ('Creando directorio listo para descargar archivos.')
            Escribir (comando_crear_directorio)
            Escribir ('Instalando prerrequisitos de Snort 3.')
            Escribir (comando_instalar_prerrequisitos)
            Escribir ('Descargando e instalando la librería safec.')
            Escribir (comando_descargar_safec)
            // Continuar con las demás tareas de instalación
            Escribir ('Comprobación de la instalación y configuración por defecto.')
            Escribir (comando_comprobacion)
        Otro Caso:
            Escribir ('Host no soportado. No se pueden realizar las tareas de instalación.')
    FinSegún
FinAlgoritmo
```

Playbook de Instalación de Snort

```
---

- name: Bienvenido! COnfiguraremos tu equipo por tu seguridad. #Bienvenida y nombre de la tarea principal

  hosts: all #Aplicacion de la configuracion a todos los hosts

  become: yes #Activacion modo superusuario para ejecutar tareas

  tasks: #Lista de tareas a ejecutarse

   # - name: Actualizando sistema y obteniendo las últimas listas de paquetes

     # shell: |

      # sudo apt-get update && sudo apt-get dist-upgrade -y #Comando shell para actualizar sistema y paquetes


    - name: Creando directorio listo para descargar archivos

      shell: mkdir -p ~/snort_src && cd ~/snort_src #Creacion de directorio de snort


    - name: Instalando prerrequisitos de Snort 3

      shell: |

        sudo apt-get install -y build-essential autotools-dev libdumbnet-dev \

        libluajit-5.1-dev libpcap-dev zlib1g-dev pkg-config libhwloc-dev cmake \

        liblzma-dev openssl libssl-dev cpputest libsqlite3-dev libtool uuid-dev \

        git autoconf bison flex libcmocka-dev libnetfilter-queue-dev libunwind-dev \

        libmnl-dev ethtool
#Instalacion de prerriquisitos de snort
    - name: Descargando e instalando la librería safec

      shell: |

        cd ~/snort_src #Entrando al directorio principal de SNORT

        wget https://github.com/rurban/safeclib/releases/download/v02092020/libsafec-02092020.tar.gz #Descarga de paquete de libreria
```

```
        tar -xzvf libsafec-02092020.tar.gz #Compilacion del archivo
        cd libsafec-02092020.0-g6d921f
        ./configure #Instalacion del paquete
        make
        sudo make install
#Comandos shell para descargar, compilar e instalar la librería safec
    - name: Instalación de PCRE
      shell: |
        cd ~/snort_src/
        wget https://ftp.exim.org/pub/pcre/pcre-8.45.tar.gz #Redireccionamiento al link de descarga
        tar -xzvf pcre-8.45.tar.gz #Compilacion del archivo
        cd pcre-8.45
        ./configure #Instalacion y configuracion del paquete por defecto
        make
        make install

    - name: Instalación de gperftools
      shell: |
        cd ~/snort_src
        wget https://github.com/gperftools/gperftools/releases/download/gperftools-2.9.1/gperftools-2.9.1.tar.gz #Redireccionamiento del link de descarga
        tar xzvf gperftools-2.9.1.tar.gz #Compilacion del archivo de la herramienta
        cd gperftools-2.9.1
        ./configure #Instalacion y configuracion de la herramienta por defecto
        make
        make install
```

```
- name: Instalación de Ragel
  shell: |
    cd ~/snort_src
    wget http://www.colm.net/files/ragel/ragel-6.10.tar.gz #Redireccionamiento del link de descarga
    tar -xzvf ragel-6.10.tar.gz #Ejecucion del paquete
    cd ragel-6.10
    ./configure #Configuracion e instalacion de Ragel
    make
    make install
# Continuar con la descarga, compilación e instalación de las dependencias restantes y de Snort 3.
# Asegurarse de ajustar las rutas de los comandos según sea necesario por actualizacion de los link de descarga.

- name: Descargando librerías Boost
  become: true
  shell: |
    cd ~/snort_src
    wget https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.tar.gz
    tar -xvzf boost_1_76_0.tar.gz

- name: Descargando e instalando DAQ
  become: true
  shell: |
    cd ~/snort_src
```

```
      git clone https://github.com/snort3/libdaq.git
      cd libdaq
      ./bootstrap
      ./configure
      make
      sudo make install

  - name: Descargando e instalando Hyperscan
    become: true
    shell: |
      cd ~/snort_src
      wget https://github.com/intel/hyperscan/archive/refs/tags/v5.4.0.tar.gz
      tar -xvzf v5.4.0.tar.gz
      mkdir ~/snort_src/hyperscan-5.4.0-build
      cd hyperscan-5.4.0-build/
      cmake -DCMAKE_INSTALL_PREFIX=/usr/local \
      -DBOOST_ROOT=~/snort_src/boost_1_76_0/ ../hyperscan-5.4.0
      make
      sudo make install

  - name: YA CASI ESTA LISTO... Descargando e instalando Snort 3
    shell:
      cd ~/snort_src
      wget https://github.com/snort3/snort3/archive/refs/tags/3.1.14.0.tar.gz -O snort3-
3.1.14.0.tar.gz
      tar -xzvf snort3-3.1.14.0.tar.gz
      cd snort3-3.1.14.0
```

```
    ./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc
    cd build
    make
    sudo make install

- name: Instalando Biblioteca libtcmalloc-minimal4
  shell:
   sudo apt install libtcmalloc-minimal4

- name: Comprobación de la instalación y configuración por defecto
  shell:
   /usr/local/bin/snort -c /usr/local/etc/snort/snort.lua
```

Playbook de creación de reglas de Snort

```
---
- name: Creando reglas y accediendo a directorios de SNORT 3.0
  hosts: all
  become: yes
  tasks:
    - name: Accediendo al archivo de configuracion de reglas de Snort 3.0
      blockinfile:
        path: /usr/local/etc/rules/local.rules
        block: |
          alert icmp any any -> $HOME_NET any (msg:"PRECAUCION. Posible ataque DoS ICMP"; sid:1000001;)
          drop icmp any any -> $HOME_NET any (msg:"BLOQUEANDO ataque DoS ICMP"; sid:1000002;)
          alert tcp any any -> 192.168.10.11 22 (msg:"Conexión SSH detectada"; sid:1000003;)
          drop tcp any any -> 192.168.10.11 22 (msg:"Bloqueando conexion SSH no permitida"; sid:1000004;)
          alert tcp any any -> $HOME_NET 80 (msg:" ALERTA! Intento de conexión TELNET"; sid:10000007; rev:001;)
          drop tcp any any -> 192.168.10.11 22 (msg:"BLOQUEANDO CONEXION TELNET no permitida"; sid:1000008;)
```