

# **ESCUELA POLITÉCNICA NACIONAL**

## **ESCUELA DE FORMACIÓN DE TECNÓLOGOS**

### **IMPLEMENTACIÓN DE SERVICIOS DE SEGURIDAD EN REDES MEDIANTE DEVOPS**

**Implementación de un IDS mediante herramientas de DevOps**

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

**ANDRÉS ALCIDES GUAMÁN PULLAS**

**DIRECTOR: FERNANDO VINICIO BECERRA CAMACHO**

**DMQ, 05 2024**

## **CERTIFICACIONES**

Yo, Andrés Alcides Guamán Pullas 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.

---

**Andrés Alcides Guamán Pullas**

**andres.guaman@epn.edu.ec**

**andyguaman01092002@gmail.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por Andrés Alcides Guamán Pullas, 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.

Andrés Alcides Guamán Pullas

## **DEDICATORIA**

A mis queridos padres, Alcides Guamán y María Pullas, cuyo amor, apoyo y sacrificio han sido la base sobre la que he construido mis sueños. Gracias por creer en mí incluso cuando yo dudaba, por enseñarme el valor del esfuerzo y la perseverancia, y por estar siempre a mi lado en cada paso de este camino. Este logro es tanto suyo como mío.

## **AGRADECIMIENTO**

Agradezco de corazón a mis padres y mi familia que siempre me apoyado en todo momento.

Quiero expresar mi más profundo agradecimiento a mi tutor de tesis, Fernando Becerra, cuya guía experta, paciencia y apoyo constante han sido fundamentales en la realización de este trabajo. Su dedicación y conocimientos han sido una fuente de inspiración y aprendizaje continuo.

Agradezco también a todos mis profesores, quienes con su sabiduría y entusiasmo me han brindado las herramientas necesarias para enfrentar este desafío. Cada clase, consejo y enseñanza han dejado una huella imborrable en mi formación académica y personal.

Finalmente, quiero agradecer a mis amigos, cuyo apoyo incondicional y compañía han sido esenciales a lo largo de este proceso. Sus palabras de aliento, momentos de distracción y comprensión en los momentos difíciles han hecho de este viaje algo más llevadero y gratificante.

Gracias a todos por su apoyo a este logro.

## ÍNDICE DE CONTENIDOS

CERTIFICACIONES .....	I
DECLARACIÓN DE AUTORÍA .....	II
DEDICATORIA .....	III
AGRADECIMIENTO .....	IV
ÍNDICE DE CONTENIDOS .....	V
RESUMEN.....	VII
<i>ABSTRACT</i> .....	VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO.....	1
1.1 Objetivo general .....	1
1.2 Objetivos específicos.....	1
1.3 Alcance .....	1
1.4 Marco Teórico .....	2
2 METODOLOGÍA.....	9
3 RESULTADOS .....	10
3.1 Análisis de las herramientas de DevOps e IDS que solucionan cada componente del proyecto de titulación e investigación sobre el manejo de DevOps. 11	
3.2 Diseño de la solución para el servicio de seguridad mediante herramientas de DevOps.....	17
3.3 Implementación de la solución mediante DevOps para el despliegue del servicio de seguridad en red. ....	18
3.4 Verificación de playbooks y funcionamiento del servicio de seguridad implementado mediante DevOps. ....	27
4 CONCLUSIONES.....	31
5 RECOMENDACIONES.....	32
6 REFERENCIAS BIBLIOGRÁFICAS.....	33

7 ANEXOS.....	34
ANEXO I: Certificado de Originalidad .....	i
ANEXO II: Enlaces .....	ii
ANEXO III: Códigos Fuente .....	iii

## RESUMEN

En el trabajo de titulación, se realiza la automatización de la implementación de un sistema de detección de intrusos utilizando Ansible para la instalación, configuración y agregación de reglas de Suricata. Todo esto permite mejorar rapidez, seguridad, eficacia y gestión de la implementación.

El proyecto está dividido por cinco secciones, en la descripción de la componente se muestra el objetivo general y los objetivos específicos que permite conseguir el resultado esperado. Luego, el alcance del proyecto que limita el resultado. Por último, se da a conocer la teoría y conceptos necesarios para el despliegue de un sistema de detección de intrusos con Ansible.

En la metodología de la investigación se detalla el despliegue de cada objetivo específico junto con su justificación.

En resultados se presenta una explicación paso a paso de lo obtenido durante cada fase, en la que se detalla el análisis de cada solución propuesta, el diseño para cada solución, la ejecución de las soluciones DevOps e IDS, y las pruebas necesarias para validar la funcionalidad y eficacia del sistema implementado.

En conclusiones, se detallan aspectos importantes en base a lo implementado en el proyecto de titulación.

En recomendaciones, se detallan sugerencias que ayudaran a interesados a realizar la implementación de un proyecto similar.

**PALABRAS CLAVE:** Ansible, DevOps, IDS, reglas, seguridad, automatización.



## **ABSTRACT**

*In the degree work, the automation of the implementation of an intrusion detection system using Ansible for the installation, configuration and aggregation of Suricata rules is performed. All this allows to improve speed, security, efficiency and management of the implementation.*

*The project is divided into five sections, the description of the component shows the general objective and the specific objectives to achieve the expected result. Then, the scope of the project that limits the result. Finally, the theory and concepts necessary for the deployment of an intrusion detection system with Ansible are presented.*

*In the research methodology, the deployment of each specific objective is detailed along with its justification.*

*In results, a step-by-step explanation of what was obtained during each phase is presented, detailing the analysis of each proposed solution, the design for each solution, the execution of the DevOps and IDS solutions, and the necessary tests to validate the functionality and effectiveness of the implemented system.*

*In conclusions, important aspects are detailed based on what was implemented in the degree project.*

*In recommendations, suggestions that will help interested parties to implement a similar project are detailed.*

**KEYWORDS:** *Ansible, DevOps, IDS, rules, security, automation*

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente tiene el objetivo de implementar soluciones de detección ante ataques cibernéticos en la red. Para ello, se realiza una previa investigación en busca de la mejor solución que más sea compatible con el IDS. Se enfoca en el uso de herramientas de código abierto específicas que ayuden a facilitar y eficientizar<sup>7</sup> la automatización en el proceso de instalación y configuración de las soluciones para la detección intrusos. Además, incluye procesos interactivos para que el administrador de red o variantes puedan gestionar los procesos de la mejor manera. Por último, se hacen pruebas implantando software malicioso en la red y que se pueda asegurar el funcionamiento óptimo de las soluciones sin fallos.

## 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

El presente proyecto se pretende utilizar DevOps y que se desplieguen soluciones en seguridad informática. Para realizar este propósito se necesita de herramientas de DevOps capaces de automatizar el despliegue de los servicios propuestos y además se puedan realizar pruebas de las soluciones. En el proyecto se pretende implementar la siguiente componente:

- Implementación de un IDS mediante herramientas de DevOps.

## 1.4 Marco Teórico

### DevOps:

DevOps es un conjunto de automatizaciones, que permite la comunicación y colaboración entre equipos de desarrollo software y tecnologías de la información; trata de una nueva forma de trabajar en equipo para lograr aumentar la velocidad y calidad en la implementación software [1].

DevOps utiliza una mejora continua en cada fase; lo que resulta que cada fase fluya de manera secuencial. Sin embargo, se necesita de una colaboración constante y de una mejora repetitiva en todo el ciclo de vida para lograr mejorar antiguas implementaciones [1].

El ciclo de DevOps se divide en ocho fases:

- Descubrir: Como preparación para el siguiente objetivo en la creación de software, los equipos deben realizar estudios para explorar, organizar y priorizar ideas [1].
- Panifica: Los equipos DevOps deben dividir la carga del trabajo en partes más pequeñas para que la entrega del proyecto sea rápido y eficiente [1].
- Compila: Tiene la funcionalidad de convertir el código escrito por los equipos DevOps en un formato ejecutable [1].
- Prueba: Se realizan pruebas automatizadas para garantizar la calidad y correcto funcionamiento del código antes y después de la integración [1].
- Implementa: Los equipos DevOps pueden publicar nuevas funciones en el código de manera automática después de integrarse; lo que permite mejorar la velocidad, productividad y sostenibilidad del desarrollo software [1].
- Opera: Gestión de los servicios de TI a los clientes e incluye prácticas de diseño, implementación y mantenimiento en la infraestructura de TI. Esto ayuda a respaldar los servicios de la organización [1].
- Observa: Se identifica de manera automática posibles incidencias que afectan el tiempo de actividad, velocidad y funcionalidad del producto. Luego, las incidencias encontradas deben resolverse para que los servicios sigan funcionando correctamente [1].

- Retroalimentación continua: De manera obligatoria, los equipos DevOps deben generar informes para mejorar en futuras implementaciones. Además, al obtener opiniones de los usuarios se podrá arreglar las versiones próximas que ofrecerán un servicio de calidad.

Las ventajas más importantes en implementar DevOps de manera empresarial son:

- Los equipos DevOps entregan el software con mayor frecuencia. Según el informe 2019 (Estado de DevOps 2019) de DORA, los equipos automatizados implementan software 106 veces más rápido que los equipos de bajo rendimiento. La entrega constante permite a los equipos dedicarse a la creación, comprobación e implementación de software; todo esto de manera automatizada. [1]
- El objetivo de DevOps es la colaboración entre desarrolladores software y equipos de tecnologías de la información, los cuales, comparten responsabilidades y juntan el trabajo. Lo que permite, entregas de trabajo más eficientes y en el menor tiempo [1].
- En relación con la seguridad, se implementa un parte activa llamada DevSecOps, la cual integra la seguridad en todas las etapas DevOps y el software en desarrollo tenga seguridad desde el principio; así se evita posibles brechas de vulnerabilidad [1].

## **Máquina virtual y Sistema Operativo**

### **Ubuntu:**

Ubuntu es un sistema operativo de código abierto para que cualquier persona pueda usarlo, distribuirlo o modificarlo. Ubuntu tiene una distribución basada en Unix, la cual tiene una versión con interfaz gráfica llamada GNOME.

Ubuntu se destaca por su interfaz gráfica intuitiva y fácil de usar, lo que facilita su manejo para usuarios de diversos niveles de experiencia. Además de su accesibilidad, Ubuntu es reconocido por su alto nivel de seguridad. Esto se debe a las actualizaciones y parches de seguridad periódicos que la comunidad de desarrolladores proporciona rápidamente en respuesta a vulnerabilidades encontradas en el sistema operativo.

Además de su seguridad robusta, Ubuntu es conocido por su amplia compatibilidad con hardware diverso, lo que significa que puede funcionar eficientemente en dispositivos simples con recursos limitados, así como en servidores y equipos más potentes. Esta

versatilidad en la compatibilidad contribuye a su popularidad y utilidad en una amplia gama de entornos informáticos. [2].

### **VirtualBox:**

VirtualBox es un software libre de virtualización, utilizado para desplegar máquinas virtuales dentro del sistema operativo principal o también llamado host. Los proveedores utilizan VirtualBox para reducir los costos y tiempo al implementar equipos de manera física.

VirtualBox se caracteriza por ejecutar cualquier tipo de sistema operativo, así mismo como sus versiones anteriores. Además, VirtualBox puede llegar a soportar cargas de trabajo masivas de hasta 32 procesadores virtuales en su versión gratuita [3].

### **Ciberseguridad**

#### **IDS:**

El sistema de detección de intrusos está diseñado para implementarse en distintos escenarios donde exista una red de equipos. IDS tiene la función es detectar cualquier tipo de anomalía en la red que se encuentre instalada, es decir va a encontrar cualquier software malicioso que entre a la red [4].

Existen dos tipos de IDS:

- Basado en host: También llamado HIDS, su función es proteger amenazas internas y externas. HIDS puede monitorear todo el tráfico hasta llegar a un equipo informático [4].
- Basado en la red: También llamado NIDS, su función es monitorear la red interna ante posibles amenazas. Pues, NIDS tiene una visibilidad completa de todo el tráfico que pasa por la red y crea las alertas ante amenazas generalizadas [4].

### **Herramientas IDS**

#### **Suricata:**

Suricata es una herramienta de ciberseguridad de código abierto que se utiliza para la detección y prevención de intrusos de red (IDS/IPS). Suricata examina el tráfico de la red en tiempo real, identificado patrones de origen malicioso y respondiendo ante posibles amenazas, lo cual permite realizar un análisis detallado después de un incidente [5].

El funcionamiento de suricata consiste en la ejecución de reglas, que permite definir patrones y comportamientos típicos de una amenaza que ya es conocida. De manera profunda, suricata puede decodificar los protocolos en capas, monitorea desde la capa de red hasta la aplicación en busca de posibles amenazas.

Suricata tiene dos posibles modos de operación:

- Modo pasivo: Actúa como espectador en la red. Analiza la red sin la necesidad de interferir en la entrega de paquetes. Además, recopila información sobre posibles amenazas sin que afecte el rendimiento del tráfico de la red [5].
- Modo activo: Suricata ya tiene un papel más importante en la red. Básicamente, al detectar anomalías en la red, toma medidas de prevención y puede llegar a bloquear las conexiones o enviar de manera urgente alertas ante las amenazas [5].

Los archivos de suricata deben ser configurados para que las reglas no puedan dar un falso positivo en la red o afecten en su el rendimiento al entregar de paquetes [5].

#### **OSSEC:**

OSSEC es una herramienta de código libre utilizado para la ciberseguridad. OSSEC tiene la función de monitorizar la red y filtrar el tráfico sospechoso que llega a la red interna y a los equipos informáticos.

OSSEC se caracteriza por llevar un seguimiento detallado y analítico sobre actividades en la red, además tiene una vista completa y en tiempo real sobre todo el tráfico que pasa por la red. Por último, OSSEC al detectar alguna amenaza, rápidamente genera alertas y las guarda en los registros llamados logs.

En lo que OSSEC destaca es la capacidad de detección de *rootkits*, ya que los atacantes lo utilizan para esconderse de los antivirus y los IDS/IPS [6].

#### **BRO IDS:**

BRO IDS es una herramienta de código abierto diseñada para detectar actividades anómalas o sospechosas en redes internas empresariales. Una de sus capacidades distintivas es la habilidad para desencapsular el contenido de los paquetes de red y luego analizarlos en profundidad.

BRO IDS se destaca por su capacidad para categorizar registros según el protocolo y el tipo específico de amenaza detectada. Por ejemplo, puede analizar y registrar sesiones HTTP junto con las URLs accedidas, solicitudes DNS, sesiones SMTP, intentos de fuerza bruta SSH, así como la extracción de archivos de sesiones HTTP. Este enfoque detallado permite a los administradores de red y equipos de seguridad identificar y responder rápidamente a diversas amenazas potenciales, mejorando así la seguridad global de la red empresarial. [7].

### **Security Onion:**

Security Onion es un software distribuido por Linux; se compone por la unión de varias herramientas IDS/IPS como: Snort, Suricata, Bro, entre otros. Security Onion tiene la función de monitorear la red y gestionar los registros log.

Security Onion es capaz de proporcionar alta visibilidad en el tráfico de la red, alertas y actividades de origen sospechosa. Sin embargo, es necesario de una gestión adecuada y muy asertiva por parte del administrador para que no existan falsos positivos [8].

### **Protocolo de servicio remoto y llavero ssh**

#### **SSH:**

SSH es un protocolo seguro de conexión remota que permite al administrador controlar y modificar servidores a través de la autenticación, además cuenta con un mecanismo de encriptación para que la conexión de extremo a extremo sea seguro [9].

#### **Llavero:**

Las llaves en ssh son una forma de autenticación que admite la conexión a varios servidores sin la necesidad de contraseñas. Para ello se tiene dos tipos de llave:

- Llave publica: La llave publica es compartida en el servidor para poder acceder a la llave privada del ordenador [10].
- Llave privada: La llave privada es la que se genera de forma automática en el ordenador y trabaja junto a la llave pública. La llave privada no debe ser compartida y debe estar bien resguardada [10].

### **Amenazas en la red**

#### **DOS:**

DOS es un ataque de denegación de servicios, diseñada para enviar solicitudes masivas a un sistema. De manera que, al servidor se le agoten sus recursos y así llegue a

colapsar. Entonces, los usuarios no podrán acceder al servidor que está siendo atacado porque está siendo interrumpido la efectividad del servicio [11].

### **Phishing:**

Phishing es un ataque de suplantación de identidad. El ciberdelincuente se hace pasar por personas o empresas de alto prestigio para poder robar información valiosa a su víctima. Lo que el mal actor hace, es enviar un enlace que lleva a la víctima a un sitio web con las mismas características de un sitio web reconocido, en donde la víctima da información privada como las contraseñas y el ciberdelincuente utiliza dicha información para robar dinero, manipular o extorsionar a la víctima [11].

### **Ransomeware:**

Ransomeware es un ataque de encriptación de datos. En la que, la víctima es obligada a pagar una suma de dinero para poder rescatar sus datos. El ataque puede ser infectada a varias computadoras a la vez, llegando a propagarse por todos los equipos de la red interna [11].

## **Herramientas de automatización y su funcionamiento**

### **Ansible:**

Ansible es una herramienta DevOps de código abierto que se utiliza para automatizar grandes sistemas informáticos. Además, ansible ayuda a automatizar el despliegue de aplicaciones, su configuración, actualización de equipos y servidores, entre otras.

La herramienta ansible funciona en sistemas operativos de Windows o Unix. Sin embargo, el servidor que ejecuta la automatización denomina nodo de control; de manera obligatoria debe estar instalada en un sistema operativo Unix/Linux. Pues, Ansible aprovecha de mejor manera varias características y herramientas nativas de Linux [12].

Los beneficios al tener ansible son: La reducción de recursos para la gestión de las tecnologías de la información, fácil manipulación y aprendizaje ya que utiliza un lenguaje de programación llamado YAML y no afecta el rendimiento de los servidores pues lo único que requiere para funcionar es la conexión remota por ssh.

Por otra parte, *Ansible* necesita la creación de *playbooks* (se ejecutan en el nodo de control). Los *playbooks* son un conjunto de tareas que se envían a las maquinas destinadas. Además, los *playbooks* tienen un mecanismo de verificación; lo que significa



que antes de que se ejecuten se comprueba las tareas y los equipos, ya que si no está correctamente gestionado no realizara ningún cambio [13].

Acerca del lenguaje de programación yaml, es utilizado gracias a su fácil comprensión. También, yaml es compatible con otros lenguajes de programación y su sintaxis proviene de XML, JSON y HTML [14].

### **Puppet:**

Puppet es una herramienta de automatización en desarrollo por open source, por lo que se tendrá que acceder a su código para hacer uso de él. Puppet trabaja de manera interna con el lenguaje de programación DSL (parecido a JSON). Además, Puppet trabaja con el modo maestro-esclavo, el cual ya conocido por otras herramientas de automatización. El diseño del código de Puppet funciona como una lista que pueden ayudar a simplificar o complicar la situación [15] [16].

Puppet tiene dos modos:

- Puppet maestro: Puppet maestro es el servidor central encargado de controlar almacenar y distribuir los archivos de configuración a Puppet agente [15].
- Puppet agente: Puppet agente es el software que se ejecuta en cada sistema que se necesite configurar y verifica periódicamente si existe actualizaciones enviadas por Puppet maestro [15].

De manera profunda Puppet tiene tres principales fases:

- Autorización: Puppet autoriza la comunicación de agente a maestro mediante un sistema de certificados; una vez realizada la comunicación, el agente puede recibir y realizar cambios en las configuraciones [15].
- Recolección de información: El agente guarda la información sobre un sistema llamado *facts*. Los *facts* sirven para personalizar la configuración de Puppet y ayuda a tener gestionada la información [15].
- Configuración: El agente recibe los archivos de configuración. Luego, los archivos de configuración deseados describen su estado. Por último, el agente compara su estado actual con el estado deseado para realizar las acciones necesarias como instalar, desinstalar o configurar [15].

## **Chef:**

Chef es una herramienta de código abierto que sirve para automatizar y gestionar usuarios en específico. Chef trabaja con el modelo maestro-esclavo, basado en otras herramientas de automatización y en las que se requiere asignar un nodo central [16] [17].

Chef muestra las tareas de manera transparente al ejecutarse, sin embargo, su sintaxis es complicada y muy estricta de manejar; se recomienda escribir de forma determinada [16].

Chef cuenta con tres componentes que ayudaran en su gestión:

- Maestros: Los maestros son el epicentro donde se inician los cambios a los usuarios, se crean los libros de cocina, recetas y políticas para después enviarlos al servidor Chef e implementarlos en los nodos del trabajador [17].
- Servidor Chef: El servidor Chef permite una buena conexión entre todos los dispositivos de la red [17].
- Nodos de trabajador: Los nodos de trabajador son dispositivos administrados por el servidor Chef y en donde cada nodo tiene un Chef preinstalado para poder permitir la comunicación en toda la infraestructura [17].

## **2 METODOLOGÍA**

En el proyecto de titulación se realizó una investigación profunda para recopilar información importante sobre el manejo de soluciones DevOps e IDS, así como los requisitos necesarios para lograr una implementación sin fallos. La información recopilada sobre IDS permitió una mejor comprensión de su funcionamiento y ayudó a determinar la mejor manera de automatizarlo.

Una vez claro el panorama de cómo implementar DevOps e IDS, se procedió con el diseño de *playbooks* que automatizarán todo el proceso del IDS. Los *playbooks* incluyen tareas específicas que permiten realizar todo el proceso de instalación y configuración del IDS. Luego, se utiliza un *playbook* dinámico que asigna las reglas necesarias.

Con la información recopilada y el diseño de los *playbooks*, se lleva a cabo la implementación de la solución DevOps, seguida de la ejecución de los *playbooks* que contienen el servicio IDS. La metodología se entiende de la siguiente manera:

Se descarga e instala el programa VirtualBox para correr el sistema operativo Ubuntu Linux con la finalidad de implementar máquinas virtuales para DevOps, IDS y Ataques. En las máquinas virtuales se tiene el sistema operativo Ubuntu Linux con la versión 18.04.6 de Ubuntu Linux.

En el nodo central se instala el servicio de automatización, el cual controla las demás máquinas virtuales. Es decir, es la única máquina que se utiliza para realizar todos los procesos de implementación del sistema IDS, sin la necesidad de hacerlo manualmente en cada máquina virtual.

Dentro del servicio Ansible se crean y se ejecutan los *playbooks* que implementaran el sistema IDS automáticamente.

Finalmente, se verifica el funcionamiento del servicio IDS realizando un ataque a la red, la cual estaría siendo monitoreado por el sistema de detección de intrusos.

### **3 RESULTADOS**

En la siguiente parte se detalla aspectos importantes para implementar el sistema IDS con herramientas DevOps. En primer lugar, se presenta un análisis y comparación de herramientas de automatización como: Ansible, Puppet y Chef. También, se presenta el análisis y comparación de herramientas IDS como Suricata, OSSEC, BRO y Security Onion. Luego, de tener claro el panorama de como implementar DevOps e IDS se diseña el *playbook* que empleará tareas propias para implementar IDS. Después, se ejecuta el *playbook* dinámico que contiene el servicio IDS y sus reglas para hacer posible la detección. Por último, se lleva a cabo la debida verificación que garantiza el correcto funcionamiento del servicio IDS para la seguridad de red mediante DevOps.

### **3.1 Análisis de las herramientas de DevOps e IDS que solucionan cada componente del proyecto de titulación e investigación sobre el manejo de DevOps.**

Para realizar el objetivo del trabajo de titulación se lleva a cabo un análisis de las herramientas DevOps disponibles en el mercado, se consideran requerimientos dependiendo del uso de recursos, implementación y gestionamiento.

A continuación, se explican las principales características de las soluciones DevOps: Ansible, Puppet y Chef.

Ansible trabaja con el modelo maestro-esclavo por lo que, necesita de un nodo central dedicado para poder controlar los demás equipos. Ansible es una herramienta fácil de comprender, ya que utiliza el lenguaje de programación yaml para crear sus *playbooks*. Ansible tiene gran compatibilidad con sistemas operativos Unix, pues aprovecha sus características y herramientas nativas tales como: ssh, scripts, Shell. Ansible no necesita aplicaciones adicionales para funcionar porque envía comandos a los esclavos mediante la conexión remota ssh. Ansible es fácil de instalar y configurar ya que requiere un comando y agregar sus esclavos al inventario.

Para Puppet es necesario acceder a su código fuente para hacer uso de él, se debe tener conocimientos medios en los lenguajes de programación Ruby y DSL; lo que se convierte en un código incontrolable y difícil de modificar. Sin embargo, Puppet trabaja con el modelo maestro-esclavo; ya conocido por otras herramientas de DevOps. Puppet se caracteriza por tener una interfaz de uso mucho mejor para su gestión. Además, Puppet tiene gran soporte de su comunidad y de su compañía desarrolladora.

En relación con Chef, también trabaja con el modelo maestro-esclavo lo que significa que requiere de un equipo independiente para su uso. La configuración y creación de recetas de chef tiene una sintaxis muy estricta, así que se debe considerar escribir de manera adecuada. Los administradores necesitan de previos conocimientos en el lenguaje de programación Ruby para poder crear, corregir o modificar líneas en el código de la receta. Chef cuenta con una herramienta llamada *knife*, lo que aliviara las cargas al implementar trabajos pesados. Chef ofrece una interfaz web con una gran colección de módulos y configuraciones específicas para la automatización.

Ahora bien, en la Tabla 3.1 se resume las ventajas y desventajas de Ansible, Puppet y Chef.

**Tabla 3.1 Ventajas y desventajas de herramientas DevOps**

	<b>Ventajas</b>	<b>Desventajas</b>
<b>Ansible</b>	<ul style="list-style-type: none"> <li>• Utiliza el modelo maestro-esclavo.</li> <li>• No requiere de más aplicaciones para funcionar, simplemente el protocolo de conexión remota ssh.</li> <li>• Rápido aprendizaje ya que únicamente debe añadir sus esclavos en el inventario y para la automatización utiliza <i>playbooks</i>.</li> <li>• Gran compatibilidad con el sistema operativo de distribución Unix porque aprovecha características y herramientas nativas, así que no requiere de muchos recursos del servidor.</li> </ul>	<ul style="list-style-type: none"> <li>• El nodo central dedicado y necesariamente debe utilizar como sistema operativo la distribución Unix.</li> <li>• No es apto para trabajos demasiado pesados.</li> <li>• La velocidad de la red puede ser afectada por su conexión ssh.</li> </ul>
<b>Puppet</b>	<ul style="list-style-type: none"> <li>• Utiliza el modelo maestro-esclavo.</li> <li>• Soporte de la comunidad y de la compañía desarrolladora.</li> <li>• Interfaz web mejorada para la gestión de esclavos.</li> </ul>	<ul style="list-style-type: none"> <li>• Es necesario conocimientos previos de los lenguajes de programación Ruby y DSL ya que es necesario entrar en su código fuente para su uso y creación de tareas.</li> <li>• Es necesario la instalación de aplicaciones terceras para realizar trabajos pesados.</li> <li>• Requiere de recursos altos ya que cuenta con interfaz web, herramientas y módulos de gestión.</li> </ul>

	<b>Ventajas</b>	<b>Desventajas</b>
<b>Chef</b>	<ul style="list-style-type: none"> <li>• Utiliza el modelo maestro-esclavo.</li> <li>• Gran colección de módulos y ajustes en su interfaz web.</li> <li>• Cuenta con una herramienta llamada <i>knife</i> para evitar sobrecargas al momento de implementar trabajos pesados.</li> </ul>	<ul style="list-style-type: none"> <li>• Dificultad de aprendizaje ya que necesita de conocimientos altos con el lenguaje de programación Ruby y tienen una sintaxis estricta.</li> <li>• Necesita de un Nodo central dedicado.</li> <li>• Requiere de recursos altos ya que cuenta con interfaz web, herramientas y módulos de gestión.</li> </ul>

Por otro lado, se lleva a cabo el análisis detallado de herramientas IDS disponibles, considerando su fácil implementación, fácil aprendizaje y configuración.

A continuación, se explican las principales características de las soluciones IDS: Suricata, OSSEC, BRO y Security Onion.

Suricata puede funcionar tanto para IDS y para IPS. El tráfico de la red lo analiza en tiempo real, ya que obtiene una visualización completa de la red interna. Suricata optimiza los recursos de la red, pues cuenta con técnicas avanzadas de procesamiento en multihilo. Suricata soporta una amplia gama de protocolos, lo que permiten la correcta detección de amenazas.

Suricata puede acceder a las reglas de la comunidad, esto permite a los administradores ahorrar tiempo en crear y evaluar nuevas reglas, además de no correr el peligro de implementar mal una regla y que en consecuencia de falsos positivos. Suricata puede ser difícil de implementar, especialmente para administradores que no tengan experiencia o una investigación previa de IDS dado que, suricata requiere habilidades analíticas y de interpretación.

OSSEC permite detección tanto de red como de host, tratando de analizar registros o también llamados logs y archivos del sistema que tengan comportamientos sospechosos. OSSEC genera alertas en tiempo real al detectar anomalías, lo que facilita la respuesta rápida de los administradores. OSSEC cuenta con un gran soporte de la comunidad y de sus desarrolladores, así que va mejorando continuamente. OSSEC no tiene la capacidad de analizar el tráfico de una red de alta capacidad, pues esta más

orientado a detección mediante host. OSSEC puede ser difícil de adaptar por lo que requiere de un conocimiento profundo de la configuración y políticas de seguridad.

BRO proporciona un análisis detallado acerca del tráfico de la red a nivel de protocolos, mediante la detección de patrones inusuales; lo que ayuda a identificar y examinar amenazas en la red.

La configuración de BRO no requiere de conocimientos avanzados ni de gran experiencia con herramientas IDS, lo que permite a los administradores aptarlo en su red fácilmente, sin embargo, BRO requiere de conocimientos profundos de protocolos de red y técnicas de interpretación de las alertas.

BRO tiene interfaz gráfica que ayuda a la visualización y gestión de alertas, el cual mejora la eficiencia operativa de los administradores. BRO analiza entornos de red de gran velocidad, pero necesita de recursos de hardware para poder funcionar correctamente, específicamente en implementaciones de gran escala.

Security Onion combina varias herramientas IDS en una sola e incluso simplificando la implementación y mejorando la gestión de alertas para ayudar a los administradores a dar una respuesta rápida ante incidentes.

Security Onion proporciona una interfaz gráfica que ayuda a los administradores a supervisar su red. Security Onion tiene un soporte activo de la comunidad y sus desarrolladoras, así que cualquier problema puede ser solventado rápidamente. Security Onion puede necesitar de hardware adicional, necesariamente en entornos de alta carga de red porque el monitoreo y respuesta ante amenazas es en tiempo real.

Ahora bien, en la Tabla 3.2 se resume las ventajas y desventajas de Suricata, OSSEC, BRO y Security Onion.

**Tabla 3.2 Ventajas y desventajas de los sistemas de detección de intrusos**

	<b>Ventajas</b>	<b>Desventajas</b>
<b>Suricata</b>	<ul style="list-style-type: none"> <li>• Monitoreo en tiempo real de toda la red.</li> <li>• Utiliza pocos recursos en el servidor ya que está bien optimizado.</li> <li>• Se puede implementar IDS e IPS al mismo tiempo en una sola distribución.</li> <li>• Basado en protocolos de red, lo que significa que detecta amenazas mediante patrones inusuales.</li> <li>• Acceso a la comunidad para reglas y soporte</li> </ul>	<ul style="list-style-type: none"> <li>• Difícil de implementar si no se tiene experiencia con otras herramientas IDS o una previa investigación.</li> <li>• El aprendizaje podría ser complicado al momento de implementar nuevas reglas.</li> <li>• Es necesario de técnicas de análisis para poder interpretar las alertas</li> </ul>
<b>OSSEC</b>	<ul style="list-style-type: none"> <li>• Monitoreo en tiempo real de toda la red</li> <li>• Basado en protocolos de red, lo que significa que detecta amenazas mediante patrones inusuales.</li> <li>• Acceso al soporte de la comunidad y de sus desarrolladores.</li> <li>• Puede implementar Host IDS y Network IDS</li> </ul>	<ul style="list-style-type: none"> <li>• No puede analizar el tráfico en redes de alta velocidad.</li> <li>• Requiere de un conocimiento profundo de la configuración y políticas de seguridad.</li> <li>• Orientado a Host IDS</li> </ul>



	<b>Ventajas</b>	<b>Desventajas</b>
<b>BRO</b>	<ul style="list-style-type: none"> <li>• Monitoreo en tiempo real de toda la red</li> <li>• Basado en protocolos de red, lo que significa que detecta amenazas mediante patrones inusuales.</li> <li>• No requiere de conocimientos avanzados ni de gran experiencia con herramientas IDS.</li> <li>• Tiene una interfaz gráfica que ayuda a la visualización y gestión de alertas, el cual mejora la eficiencia operativa de los administradores.</li> </ul>	<ul style="list-style-type: none"> <li>• Requiere de conocimientos profundos de protocolos de red y técnicas de interpretación de las alertas.</li> <li>• Necesita de hardware dedicado para implementaciones a gran escala.</li> </ul>
<b>Security Onion</b>	<ul style="list-style-type: none"> <li>• Monitoreo en tiempo real de toda la red</li> <li>• Basado en protocolos de red, lo que significa que detecta amenazas mediante patrones inusuales.</li> <li>• Combina varias herramientas IDS en una sola e incluso simplificando la implementación y mejorando la gestión de alertas.</li> <li>• Proporciona una interfaz gráfica que ayuda a los administradores a supervisar su red.</li> <li>• Tiene un soporte activo de la comunidad y sus desarrolladoras</li> </ul>	<ul style="list-style-type: none"> <li>• Necesita de hardware adicional, necesariamente en entornos de alta carga de red.</li> <li>• Requiere de conocimientos profundos de protocolos de red y técnicas de interpretación de las alertas.</li> </ul>

## 3.2 Diseño de la solución para el servicio de seguridad mediante herramientas de DevOps.

Para el diseño del servicio de seguridad en la red, se requiere de una herramienta de virtualización que se adapte a las necesidades del sistema de detección de intrusos; en el marco teórico se menciona VirtualBox por su facilidad de uso. Por lo descrito antes, se puede simular una red interna aislada de la red principal.

La versión 7.0.14 de VirtualBox y la imagen del sistema operativo Ubuntu Linux versión 18.04.6 son necesarias porque cuentan con soporte y con más funciones que las antiguas no.

En cada máquina virtual se establece recursos recomendados para evitar posibles fallos o que no cuenten con los recursos suficientes al momento de ejecutar las soluciones DevOps e IDS, es decir, que las máquinas virtuales se traben frecuentemente o en el peor escenario que los servicios no se habiliten.

De las soluciones DevOps e IDS expuestas en el marco teórico y en el análisis, se utilizará la más adecuada para automatizar los procesos IDS. La de herramienta DevOps que se utilizará debe ser fácil de implementar, tener gran compatibilidad con Linux y su sintaxis debe ser fácil de entender. Por otra parte, el sistema IDS que se utilizará debe requerir pocos recursos, tener monitoreo de paquetes en tiempo real y poder acceder a reglas de su comunidad.

En las máquinas virtuales se añadirá una interfaz de red extra que permitirá conectarse a Internet para poder descargar paquetes del repositorio de Linux, mientras que la otra interfaz de red estará conectada a la red interna para la implementación del sistema de detección de intrusos.

A continuación, será necesario utilizar Ansible para automatizar el proceso de implementación del sistema IDS. Ansible requiere de la creación de *playbooks* para ejecutar procesos a sus servidores esclavos. Por otra parte, se necesita tener conocimientos previos sobre la sintaxis yml, puesto que Ansible lo requiere para definir los procesos que se desea ejecutar. Ahora bien, la automatización debe ser completa sin la necesidad de entrar al servidor IDS; los *playbooks* deberán realizar los procesos de: instalación, configuración y agregación.

Ahora bien, no se puede crear los *playbooks* sin tener conocimientos previos sobre la implementación de Suricata en un servidor común. Por esto, es necesario investigar

acerca del sistema Suricata como: la instalación, la configuración y la agregación de reglas que serán necesarias para detectar los intrusos.

En VirtualBox se añadirá una máquina intrusa para poder realizar pruebas y saber si realmente está funcionando el sistema IDS. En la máquina intrusa se deberá instalar una herramienta que permita enviar ataques mediante DOS a un servidor web, luego el servidor IDS deberá enviar alertas. Finalmente, se podrá garantizar el funcionamiento de un sistema IDS mediante herramientas DevOps.

### **3.3 Implementación de la solución mediante DevOps para el despliegue del servicio de seguridad en red.**

En el siguiente apartado se implementan las soluciones del proyecto de tal manera que empieza con la creación y configuración de máquinas virtuales en VirtualBox para establecer la red interna virtual. Después, la actualización de paquetes de Ubuntu para que no instale paquetes con versiones antiguas, las cuales podrían causar fallos en la implementación del proyecto. Luego, se instala y configura ssh para que el nodo central envíe las tareas a sus esclavos. A continuación, se procede con la instalación y configuración de Ansible; la herramienta DevOps que permite desplegar el servicio IDS en la red. Por último, se crean y ejecutan los *playbooks* que permitan automatizar el proceso de implementación IDS.

#### **Creación y configuración de máquinas virtuales en VirtualBox**

En VirtualBox se crea tres máquinas virtuales con la imagen de Ubuntu; las máquinas virtuales se llaman: Ubuntu Ansible, Ubuntu Suricata y Ubuntu Intruso. Los recursos dados a cada máquina virtual son: 2 GB de RAM y 1 CPU virtual excepto Ubuntu Ansible con 4 GB de RAM para correr Ansible sin problemas y será el nodo central.

Luego, las interfaces de red de las máquinas virtuales se configuran como red interna. Seguidamente se añade una Interfaz de red adicional como NAT para tener conexión a Internet y poder descargar los paquetes del repositorio de Linux. Por último, se inician las máquinas virtuales y se asigna en la interfaz de la red interna de cada máquina virtual la dirección IP que este en la misma red: 192.168.110.0/24 y a la interfaz de red NAT solo se habilita porque ya automáticamente se conecta a internet.

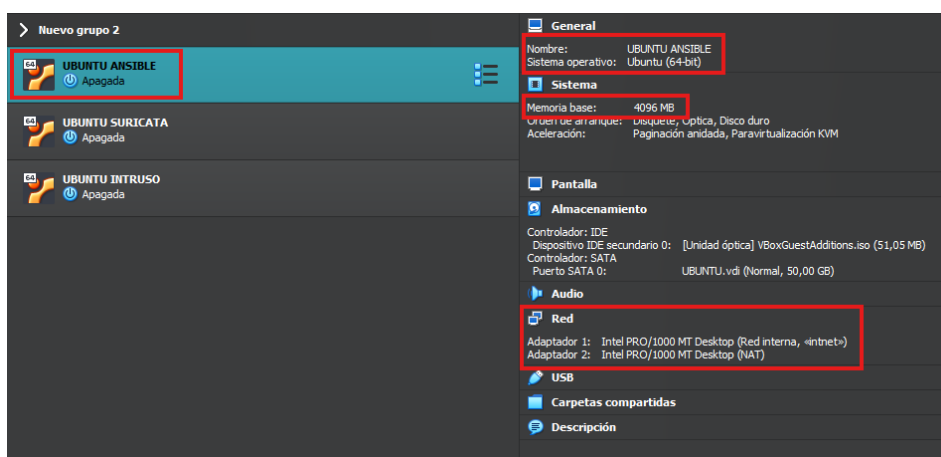
En la Tabla 3.3 se muestra el nombre de las máquinas virtuales, recursos, su respectiva dirección IP y sus interfaces de red respectivas.

**Tabla 3.3 Descripción de las máquinas virtuales**

	Recursos	IP	Interfaces
UBUNTU ANSIBLE	RAM: 4GB	192.168.110.183/24	enp0s3: red interna enp0s8: internet
UBUNTU SURICATA	RAM: 2GB	192.168.110.184/24	enp0s3: red interna enp0s8: internet
UBUNTU INTRUSO	RAM: 2GB	192.168.110.185/24	enp0s3: red interna enp0s8: internet

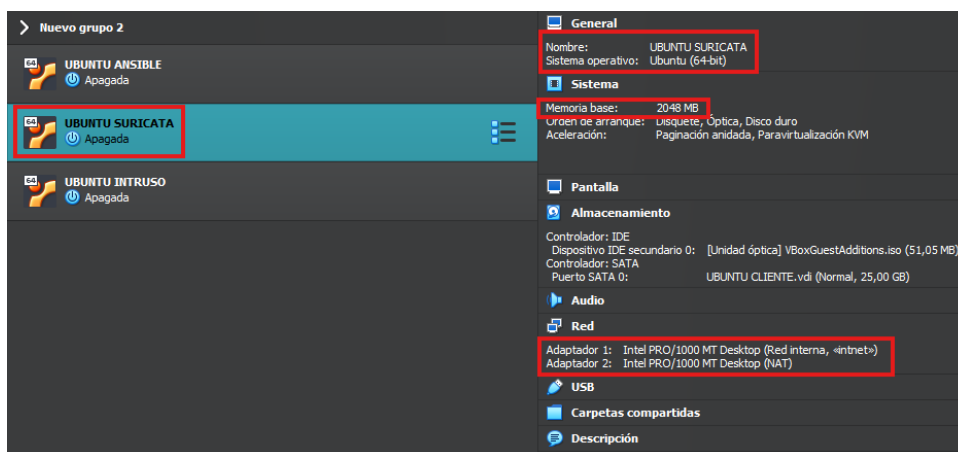
A continuación, se muestran figuras de las máquinas virtuales creadas y con las configuraciones dichas anteriormente.

En la Figura 3.1 se muestra la máquina virtual “Ubuntu Ansible” y sus especificaciones.



**Figura 3.1** Especificaciones máquina virtual “Ubuntu Ansible”.

En la Figura 3.2 se muestra la máquina virtual “Ubuntu Suricata” y sus especificaciones.



**Figura 3.2** Especificaciones máquina virtual “Ubuntu Suricata”.

En la Figura 3.3 se muestra la máquina virtual “Ubuntu Intruso” y sus especificaciones

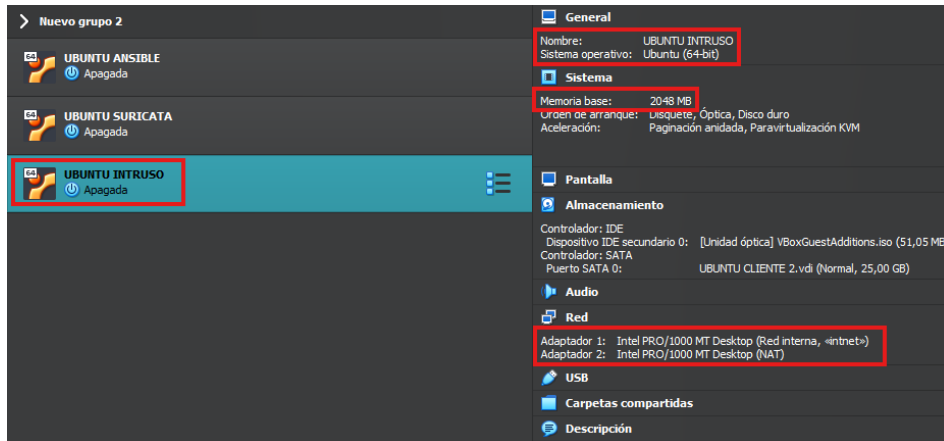


Figura 3.3 Especificaciones máquina virtual “Ubuntu Intruso”.

Finalmente, se muestran las figuras con las máquinas virtuales iniciadas y con la dirección ip de la red interna configurada.

En la Figura 3.4 se muestra la maquina “Ubuntu Ansible” iniciada y con la dirección ip 192.168.110.183 de la interfaz de red enp0s3 configurada.

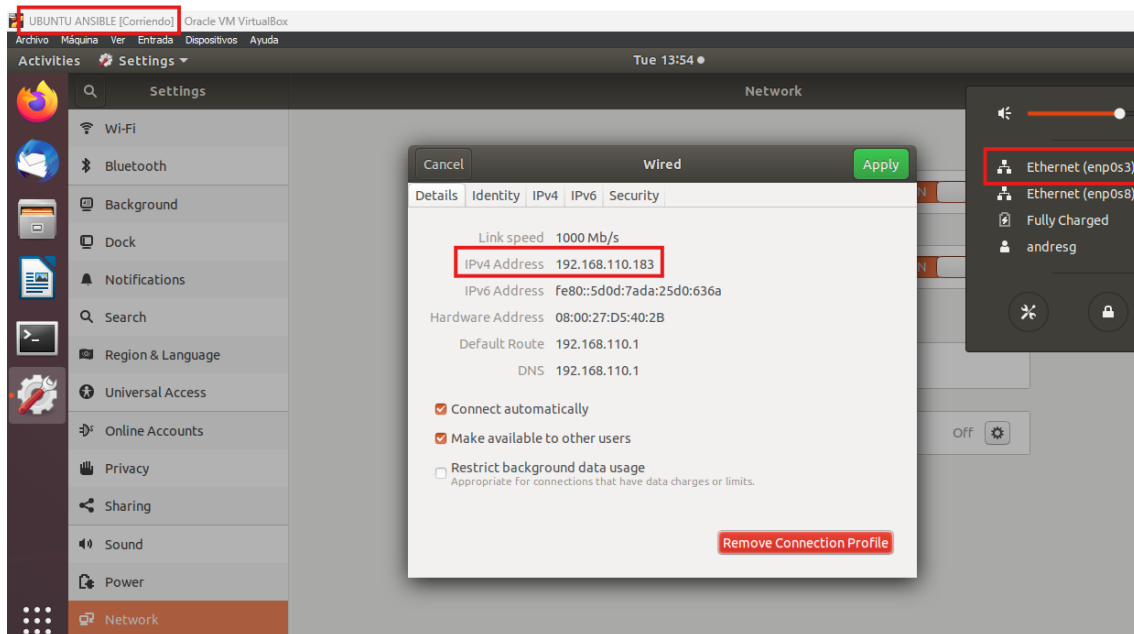
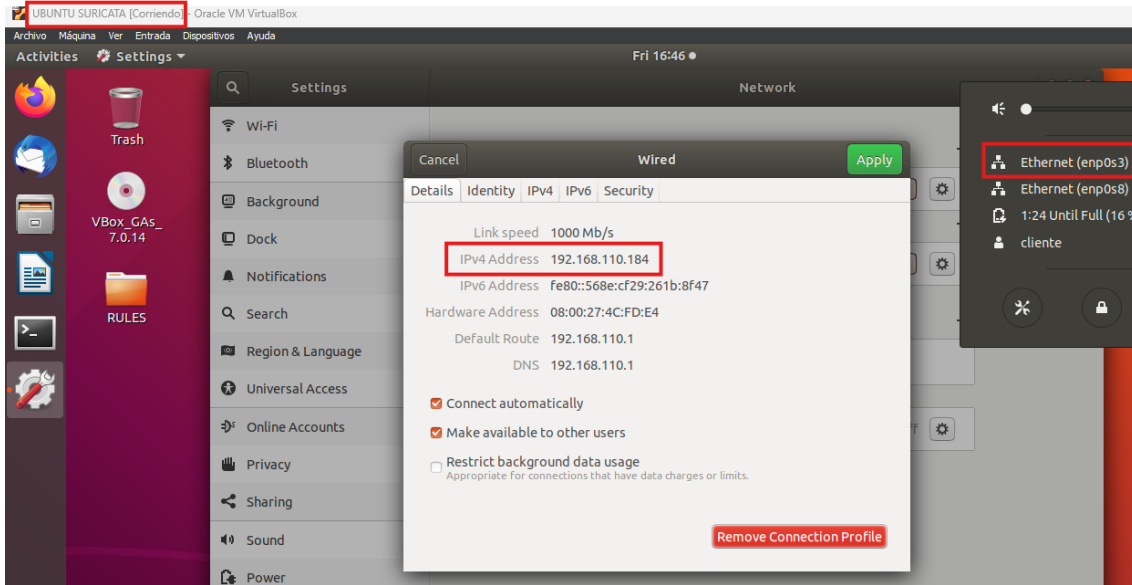


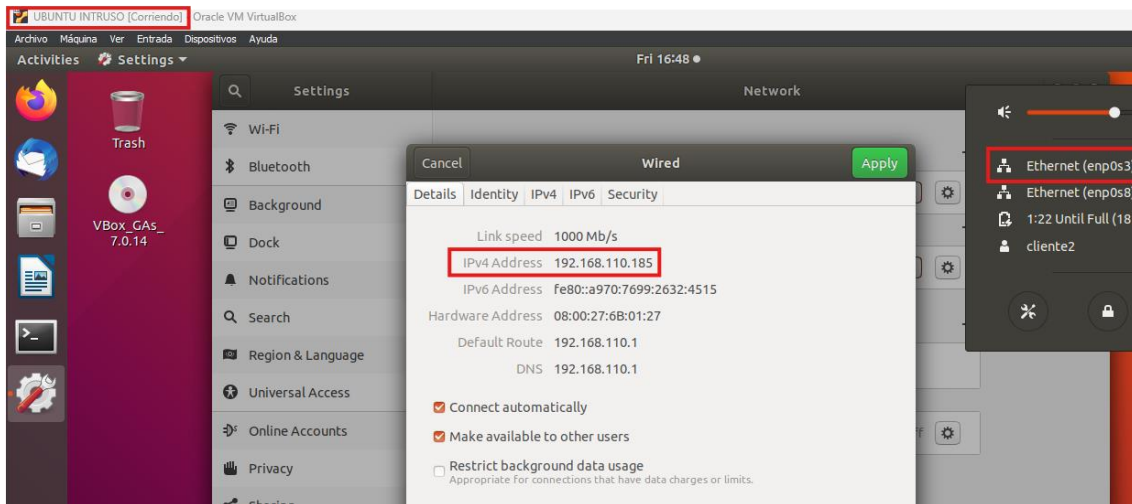
Figura 3.4 Configuración ip máquina virtual “Ubuntu Ansible”.

En la Figura 3.5 se muestra la maquina “Ubuntu Suricata” iniciada y con la dirección ip 192.168.110.184 de la interfaz de red emp0s3 configurada.



**Figura 3.5** Configuración ip máquina virtual “Ubuntu Suricata”.

En la Figura 3.6 se muestra la maquina “Ubuntu Intruso” iniciada y con la dirección ip 192.168.110.185 de la interfaz de red emp0s3 configurada.



**Figura 3.6** Configuración ip máquina virtual “Ubuntu Intruso”.

### **Actualización de paquetes de Ubuntu en máquinas virtuales: “Ubuntu Ansible” y “Ubuntu Suricata”.**

Antes de proceder con la instalación de Ansible, es importante actualizar la lista de paquetes disponibles en Linux. Para hacerlo, se utiliza el comando "sudo apt update". En la Figura 3.7 se muestra la ejecución de este comando para actualizar la lista del repositorio de Linux.

```

root@andresg:/home/andresg# apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [102 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [102 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [1,666 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3,044 kB]
0% [4 Packages store 0 B] [5 Packages 44.4 kB/3,044 kB 1%] 140 kB/s 60s

```

**Figura 3.7** Actualización de lista del repositorio.

Después, es necesario instalar y actualizar los paquetes del sistema. Para ello, se utiliza el comando “sudo apt upgrade”, como se muestra en la Figura 3.8

```

root@andresg:/home/andresg# apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
 gir1.2-goa-1.0 gir1.2-snapd-1
Use 'sudo apt autoremove' to remove them.
The following security updates require Ubuntu Pro with 'esm-infra' enabled:
 libpam0g bluez libwebp6 libkrb5-3 libgssapi-krb5-2 libpython3.6-minimal

```

**Figura 3.8** Actualización paquetes del repositorio.

### Instalación y configuración del protocolo ssh en las máquinas virtuales: “Ubuntu Ansible” y “Ubuntu Suricata”.

Se instala el protocolo de conexión retoma ssh para que el maestro pueda enviar tareas a sus esclavos. Para ello, se utiliza el comando “sudo apt install ssh -y”, como se muestra en la Figura 3.9

```

root@andresg:/home/andresg# apt install ssh -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
ssh is already the newest version (1:7.6p1-4ubuntu0.7).
The following packages were automatically installed and are no longer required:
 gir1.2-goa-1.0 gir1.2-snapd-1

```

**Figura 3.9** Instalación ssh.

Después, es necesario acceder al archivo de configuración de SSH para habilitar la conexión mediante root. Para ello, se utiliza el comando "sudo nano /etc/ssh/sshd\_config", como se muestra en la Figura 3.10.

```
GNU nano 2.9.3 /etc/ssh/sshd_config
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
PermitRootLogin yes
```

Figura 3.10 Permisos root ssh.

Después, se procede a cambiar la contraseña para actualizar los permisos de SSH. Para hacerlo, se utiliza el comando "sudo passwd". El comando solicitará escribir la nueva contraseña y repetirla para confirmar los cambios exitosamente. Este proceso se muestra en la Figura 3.11.

```
root@andresg:/home/andresg# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@andresg:/home/andresg#
```

Figura 3.11 Cambio de contraseña.

### Creación de Llavero en la máquina virtual "Ubuntu Ansible"

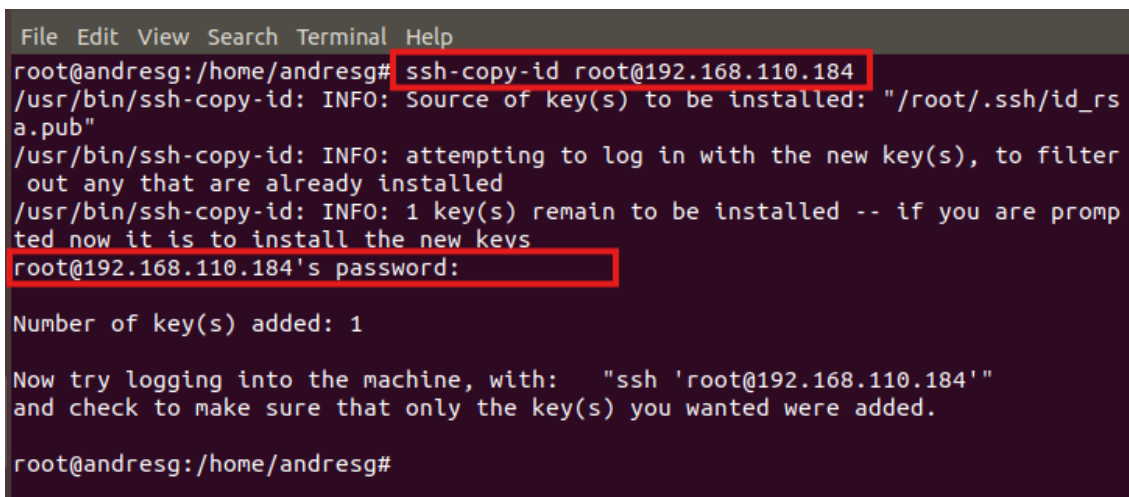
A continuación, se crea el llavero que ayudará a conectarnos con una sola contraseña a cualquier servidor de la red interna sin la necesidad de memorizarse muchas contraseñas, por lo tanto, se utiliza el comando "sudo ssh-keygen". El llavero pedirá un nombre y una contraseña para poder generar dos claves: una pública y otra privada. Este proceso se muestra en la Figura 3.12.

```
root@andresg:/home/andresg# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:VBtQbjyuPUK38+IhhYpe4s4qFVvt27u4W7rs6jL6WbA root@andresg
The key's randomart image is:
+---[RSA 2048]-----+
|          .o+       |
|           + o      |
|            . . *   |
|       . . . . + .  |
|    .+ . S +       |
|   oo ..o = .      |
| .E + oo= *        |
| . o* +.+o.=       |
| .+*Bo0o+...      |
+---[SHA256]-----+
```

Figura 3.12 Generación del llavero ssh.



Después, es necesario copiar la llave pública a los demás servidores para poder establecer la conexión con la clave privada del servidor ssh, por lo tanto, se utiliza el comando “ssh-copy-id ”(hostname)@(ipcliente)“. El llavero pedirá la contraseña del esclavo y se copiará exitosamente; diciendo que ha añadido una nueva contraseña al llavero. Este proceso se muestra en la Figura 3.13.



```
File Edit View Search Terminal Help
root@andresg:/home/andresg# ssh-copy-id root@192.168.110.184
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@192.168.110.184's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@192.168.110.184'"
and check to make sure that only the key(s) you wanted were added.

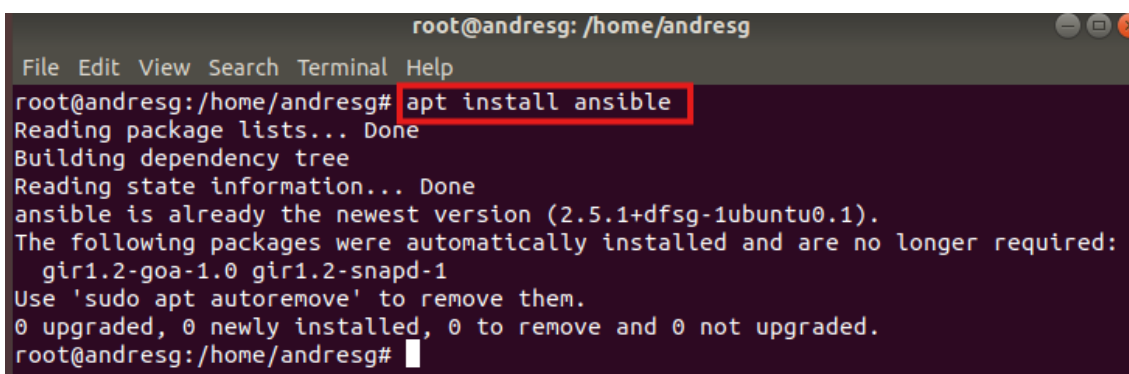
root@andresg:/home/andresg#
```

**Figura 3.13** Adición cliente al llavero ssh.

### **Instalación y configuración de Ansible en la máquina virtual “Ubuntu Ansible”**

Una vez instalado y configurado las máquinas virtuales; se empieza a implementar la herramienta de automatización Ansible en el nodo principal.

En la Figura 3.14 se muestra la instalación de Ansible, por lo tanto, se utiliza el comando “sudo apt install ansible”.

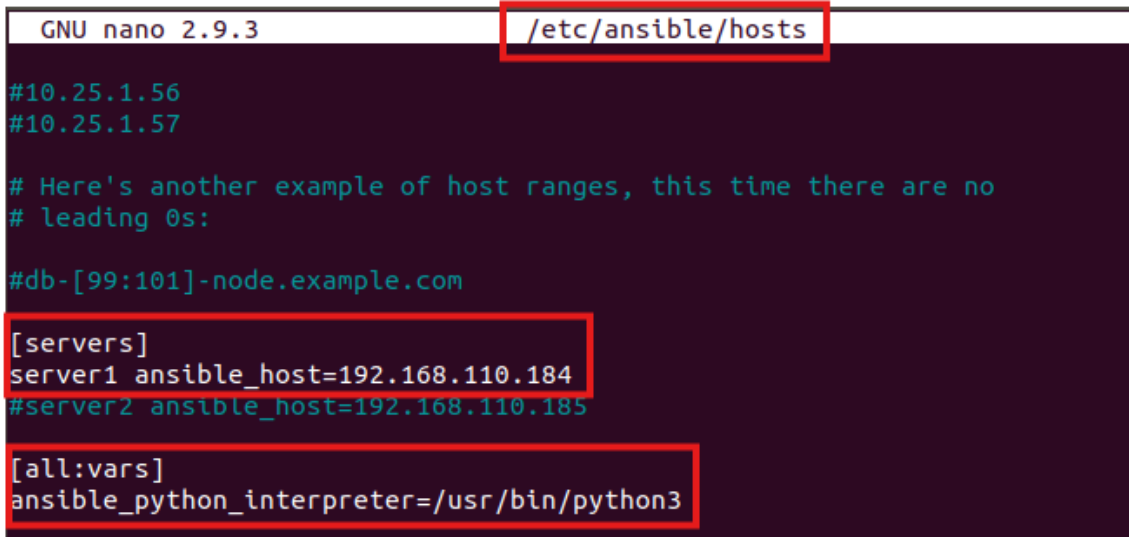


```
root@andresg: /home/andresg
File Edit View Search Terminal Help
root@andresg:/home/andresg# apt install ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
ansible is already the newest version (2.5.1-dfsg-1ubuntu0.1).
The following packages were automatically installed and are no longer required:
 gir1.2-goa-1.0 gir1.2-snapd-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@andresg:/home/andresg#
```

**Figura 3.14** Instalación Ansible.

Ahora bien, se procede con la configuración de los hosts al inventario de suricata para saber que servidor va a utilizar como esclavo, por lo tanto, en el terminal ir a la ruta del archivo hosts con el comando “sudo nano /etc/ansible/hosts”. En el archivo se define un grupo llamado “[servers]”, luego se añade los servidores junto con un nombre y la

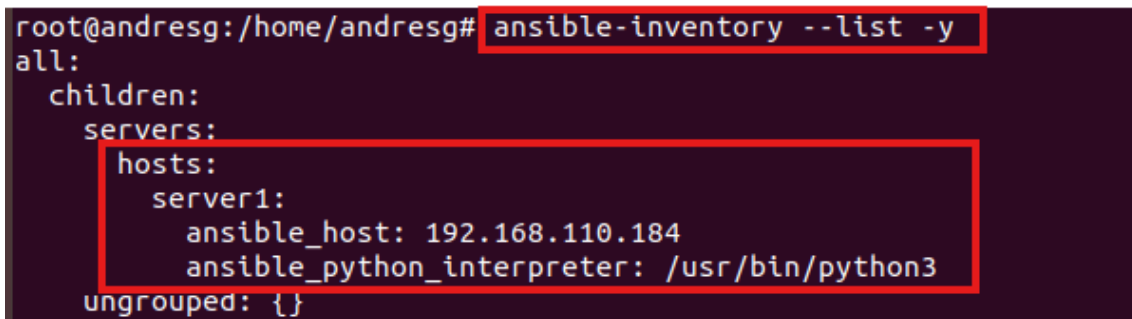
dirección ip del esclavo que va a controlar Ansible. Por último, se define otro grupo llamado [all:vars] para establecer el parámetro “ansible\_python\_interpreter”. Este parámetro garantiza que el servidor remoto utilice la versión de Python 3. Este proceso se muestra en la Figura 3.15.



```
GNU nano 2.9.3 /etc/ansible/hosts
#10.25.1.56
#10.25.1.57
# Here's another example of host ranges, this time there are no
# leading 0s:
#db-[99:101]-node.example.com
[servers]
server1 ansible_host=192.168.110.184
#server2 ansible_host=192.168.110.185
[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

**Figura 3.15** Adición esclavos al inventario de Ansible.

A continuación, es necesario consultar el inventario de Ansible para observar los servidores que están activos actualmente, por lo tanto, se utiliza el comando “ansible-inventory --list -y”. Este proceso se muestra en la Figura 3.16.



```
root@andresg:/home/andresg# ansible-inventory --list -y
all:
  children:
    servers:
      hosts:
        server1:
          ansible_host: 192.168.110.184
          ansible_python_interpreter: /usr/bin/python3
      ungrouped: {}
```

**Figura 3.16** Verificación del inventario de Ansible

Por último, Se debe verificar si Ansible es capaz de conectarse a los servidores de su inventario y ejecutar comandos ssh, por lo tanto, se utiliza el comando “ansible all -m ping -u root”, el comando pedirá la contraseña del llavero que se ha creado anteriormente y mostrará que la conexión al esclavo “Ubuntu Suricata” fue exitosa. Este proceso se muestra en la Figura 3.17.

```
root@andresg:/home/andresg# ansible all -m ping -u root
Enter passphrase for key '/root/.ssh/id_rsa':
server1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
root@andresg:/home/andresg#
```

Figura 3.17 Comprobación de la conexión maestro a esclavo.

### Creación de *playbooks*

Una vez establecida la conexión desde "Ubuntu Ansible" al esclavo "Ubuntu Suricata", se puede proceder a ejecutar *playbooks* para automatizar tareas. Para crear un *playbook*, se utiliza la sintaxis YAML. En el terminal, se crea un *playbook* utilizando el editor de texto nano mediante el comando "sudo nano (nombre del archivo).yml". Una vez creado el archivo, se inicia la redacción del *playbook* que automatizará los procesos necesarios.

Para automatizar la implementación del IDS, se crean tres *playbooks* que facilitarán la instalación, configuración y agregación de reglas en la máquina virtual "Ubuntu Suricata":

#### **Playbook "tarea.yml"**

El *playbook* tarea.yml, instala los paquetes de suricata en su última versión, de igual manera el servicio web apache2 para realizar pruebas en él y paquetes adicionales como *curl* y *jq* con el fin de observar de mejor manera las alertas de suricata. Finalmente, habilita todos los servicios instalados. El código se muestra en Anexo III.I.

#### **Playbook "configuracion.yml"**

El *playbook* configuracion.yml busca y abre el archivo de configuración de Suricata, luego busca y reemplaza líneas de código específicas para lograr automatizar la configuración. Por último, se reinicia el servicio para que se apliquen las configuraciones. El código se muestra en Anexo III.II.

#### **Playbook "interactivo.yml"**

El *playbook* "interactivo.yml" muestra una bienvenida y un menú con reglas que se desean instalar y activar. Después de seleccionar una opción, saltará el proceso de las demás reglas e irá directamente al seleccionado para habilitarlo en la ruta /etc/suricata/rules. Luego, ejecutara comandos Shell en la máquina virtual "Ubuntu

Suricata” para actualizar la lista de reglas, verificar la lista de reglas y reiniciar el servicio para que se apliquen los cambios. El código se muestra en Anexo III.III.

### 3.4 Verificación de playbooks y funcionamiento del servicio de seguridad implementado mediante DevOps.

En el terminal de la máquina virtual "Ubuntu Ansible", para verificar y ejecutar los *playbooks*, se utiliza el comando "sudo ansible-playbook (nombre del archivo).yml". A continuación, se solicitará la contraseña del llavero para iniciar la ejecución del *playbook*. Finalmente, en pantalla se mostrarán los resultados de los procesos completados o los cambios realizados. Si alguno de los procesos no se cumple, se mostrarán resultados fallidos.

A continuación, se presentan las figuras que evidencian la ejecución correcta de los *playbooks*: tarea.yml, configuración.yml e interactivo.yml.

En la Figura 3.18 se muestra la ejecución del *playbook* tarea.yml y de las tareas que logro completar correctamente.

```
root@andresg:/home/andresg# ansible-playbook tarea.yml
PLAY [all] *****
TASK [Gathering Facts] *****
Enter passphrase for key '/root/.ssh/id_rsa':
ok: [server1]
TASK [instala apache2] *****
ok: [server1]
TASK [instala suricata] *****
ok: [server1]
TASK [corre suticata] *****
ok: [server1]
TASK [instala curl] *****
ok: [server1]
TASK [instala jq] *****
ok: [server1]
PLAY RECAP *****
server1 : ok=6   changed=0   unreachable=0   failed=0
```

Figura 3.18 Ejecución correcto del *playbook* “tarea.yml”.

En la Figura 3.19 se muestra la ejecución del *playbook* configuracion.yml y las tareas que logro completar correctamente.

```

root@andresg:/home/andresg# ansible-playbook configuracion.yml
PLAY [all] *****
TASK [Gathering Facts] *****
Enter passphrase for key '/root/.ssh/id_rsa':
ok: [server1]
TASK [configura ip] *****
ok: [server1]
TASK [configura reglas de comunidad] *****
ok: [server1]
TASK [configura interface] *****
ok: [server1]
TASK [configura ruta de reglas] *****
ok: [server1]
TASK [reinicia servicio] *****
changed: [server1]
PLAY RECAP *****
server1 : ok=6  changed=1  unreachable=0  failed=0

```

Figura 3.19 Ejecución correcto del *playbook* “configuracion.yml”.

En la Figura 3.20 se muestra la ejecución del *playbook* interactivo.yml y de las tareas que logro completar correctamente.

```

root@andresg:/home/andresg# ansible-playbook interactivo.yml
PLAY [all] *****
TASK [Gathering Facts] *****
Enter passphrase for key '/root/.ssh/id_rsa':
ok: [server1]
TASK [BIENVENIDO/A] *****
[BIENVENIDO/A]
¿Cual es tu nombre?:
Andres
ok: [server1]
TASK [Mostrando usuario] *****
ok: [server1] => {
  "msg": "Buen dia, Andres."
}
TASK [Reglas para instalar] *****
[Reglas para instalar]
¿Que reglas paquetes deseas instalar? 1. Deteccion de Movimiento lateral 2. Deteccion de malware 3. Deteccion DOS Seleccione la opcion:
3
ok: [server1]
TASK [procesando opción seleccionada] *****
ok: [server1]
TASK [Deteccion de Movimiento lateral] *****
skipping: [server1]
TASK [añadiendo paquete de reglas moviniento lateral] *****
changed: [server1]
TASK [Deteccion de malware] *****
skipping: [server1]
TASK [añadiendo paquetes de reglas malware] *****
changed: [server1]
TASK [Deteccion DOS] *****
ok: [server1] => {
  "msg": "Se instalara el paquete de reglas para la deteccion de DOS"
}
TASK [añadiendo el paquete de reglas DOS] *****
changed: [server1]
TASK [actualizando la reglas] *****
changed: [server1]
TASK [verificando errores en las reglas] *****
changed: [server1]
TASK [reinciando servicio suricata] *****
changed: [server1]
PLAY RECAP *****
server1 : ok=12  changed=6  unreachable=0  failed=0

```

Figura 3.20 Ejecución correcto del *playbook* “interactivo.yml”.

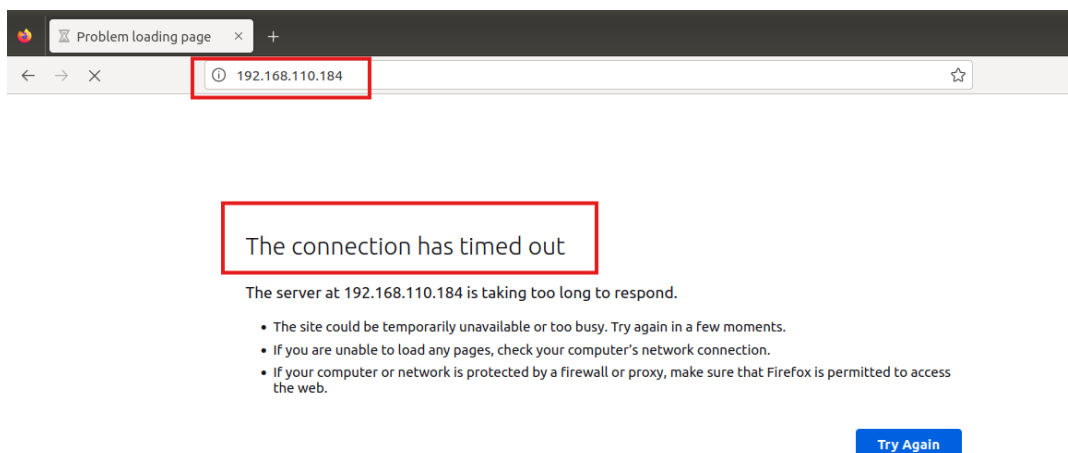
En la verificación del sistema IDS, se utiliza la máquina virtual “Ubuntu Intruso” que está conectada a la red interna. En el terminal se descarga un paquete llamado *slowhttptest* para realizar pruebas DOS al servicio web, lo que significa denegar el servicio web enviando una infinidad de solicitudes de acceso a la web hasta que los recursos del servidor se agoten, por lo tanto, se utiliza el comando “*slowhttptest -c 8000 -H -g -o slowhttp -i 10 -r 1600 -t GET -u http://192.168.110.184 -x 24 -p 3*”. A continuación, se mostrará información del número de solicitudes que se están haciendo al servicio y si está disponible o no. Este proceso se muestra en la Figura 3.21.

```
root@cliente2: /home/cliente2
File Edit View Search Terminal Help
slowhttptest version 1.6
- https://code.google.com/p/slowhttptest/ -
test type: SLOW HEADERS
number of connections: 8000
URL: http://192.168.110.184/
verb: GET
Content-Length header value: 4096
Follow up data max size: 52
interval between follow up data: 10 seconds
connections per seconds: 1600
probe connection timeout: 3 seconds
test duration: 240 seconds
using proxy: no proxy

Fri Jul 12 18:02:03 2024:
slow HTTP test status on 25th second:
initializing: 0
pending: 3919
connected: 1038
error: 0
closed: 150
service available: NO
```

**Figura 3.21** Ataque DOS con *slowhttptest*

Ahora, al intentar entrar al servicio web con la dirección ip: 192.168.110.184, el servicio no funcionara correctamente mostrando información que el servicio a tardado demasiado tiempo en responder. Este proceso se muestra en la Figura 3.22.



**Figura 3.22** Verificación del ataque DOS en la ip 192.168.110.184

Luego, en la máquina virtual "Ubuntu Suricata", es posible revisar las alertas que se están guardando automáticamente en los `logs` del sistema. En el terminal, se utiliza el comando `"tail -f /var/log/suricata/fast.log"` para visualizar las alertas DOS en tiempo real que Suricata detecta. Este comando mostrará las alertas detallando la hora, fecha, nombre del ataque, protocolo de transporte y la IP del atacante con una flecha hacia la IP del servidor que está siendo atacado, tal como se muestra en la Figura 3.23.

```
root@cliente:/etc/suricata/rules# tail -f /var/log/suricata/fast.log
07/12/2024-18:12:34.467781 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:47764 -> 192.168.110.184:80
07/12/2024-18:13:36.727337 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:41224 -> 192.168.110.184:80
07/12/2024-18:14:09.972846 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:38204 -> 192.168.110.184:80
07/12/2024-18:14:39.581603 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:53788 -> 192.168.110.184:80
07/12/2024-18:15:11.327377 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:51726 -> 192.168.110.184:80
07/12/2024-18:16:07.202825 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:37206 -> 192.168.110.184:80
07/12/2024-18:16:14.227464 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:37108 -> 192.168.110.184:80
07/12/2024-18:17:13.953734 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:52666 -> 192.168.110.184:80
07/12/2024-18:17:18.938345 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:52606 -> 192.168.110.184:80
07/12/2024-18:18:01.823762 [**] [1:2014103:7] ET WEB_SERVER Unusually Fast HTTP Requests With Referer Url Matching DoS Tool [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.110.185:58476 -> 192.168.110.184:80
```

Figura 3.23 Alertas DOS en "Ubuntu Suricata"

El sistema Suricata continuará lanzando alertas hasta que el ataque se detenga. Para detener el ataque, en la máquina virtual "Ubuntu Intruso", se utiliza `Ctrl+C`. La herramienta `slowhttptest` mostrará que las conexiones se han cerrado y que el servicio está nuevamente disponible. Este proceso se muestra en la Figura 3.24

```
Fri Jul 12 18:21:11 2024:
slow HTTP test status on 240th second:

initializing:      0
pending:           0
connected:        85
error:             0
closed:            7915
service available: YES

Fri Jul 12 18:21:12 2024:
Test ended on 241th second
Exit status: Hit test time limit
CSV report saved to slowhttp.csv
HTML report saved to slowhttp.html
```

Figura 3.24 Finalización del ataque DOS

Finalmente, la máquina virtual "Ubuntu Suricata" dejará de lanzar alertas porque el ataque hacia el servicio web ha finalizado.

## 4 CONCLUSIONES

- La solución DevOps utilizada en el proyecto fue Ansible, por su gran compatibilidad con Linux, ya que no requiere de aplicaciones terceras para funcionar, sino que utiliza funciones nativas de Linux como lo es ssh.
- Puppet y Chef no se utilizaron en el proyecto porque requieren más recursos, son más difíciles de implementar y necesitan el uso de varios lenguajes de programación, como Ruby y DSL, para crear tareas. Ansible, en comparación, ofrece una solución más sencilla y eficiente para la automatización y gestión de configuraciones en entornos Linux.
- La solución IDS utilizada para detectar intrusos en la red, implementada de manera automática, es Suricata. Esta elección se debe a que Suricata requiere pocos recursos para funcionar. Además, cuenta con soporte y reglas proporcionadas por la comunidad, lo que permite a los administradores ahorrar tiempo en la creación de nuevas reglas.
- OSSEC, BRO y Security Onion no se utilizan como sistema IDS en el proyecto porque carecen de un gran soporte comunitario. Además, están basados en host IDS, lo que limita su capacidad para analizar el tráfico en redes de alta velocidad, volviéndolos más complicados de implementar. Estas limitaciones hacen que estas soluciones sean menos adecuadas para los requisitos del proyecto en comparación con Suricata que es más eficiente y fácil de implementar.
- El diseño de los *playbooks* se realizó en un archivo YAML, que consiste en escribir una lista de tareas que se ejecutarán en los nodos esclavos. Cada tarea debe describir claramente lo que se está realizando en ese momento, de manera que el usuario pueda saber qué tareas se ejecutan correctamente y cuáles no.
- La automatización ayuda a los administradores desplegar nuevos proyectos rápidamente, permitiendo mejoras constantes y evitando la fatiga de tener que configurar o instalar paquetes manualmente en cada servidor. Gracias a la automatización, los administradores pueden gestionar múltiples servidores de manera eficiente, asegurando que las configuraciones sean consistentes y reduciendo significativamente el tiempo y esfuerzo necesarios para el mantenimiento y la implementación de nuevos servicios.
- Las reglas de la comunidad son cruciales porque pueden utilizarse como plantillas para seleccionar e instalar las que permitan al sistema detectar intrusos de manera efectiva. A partir de estas plantillas, es posible modificar las reglas según los requerimientos específicos, lo que evita tener que empezar desde cero



nuevas reglas. Esto es especialmente útil para aquellos que no tienen un gran conocimiento en la creación de reglas personalizadas, ya que proporciona una base sólida sobre la cual trabajar y adaptarse a las necesidades particulares del sistema y de la red. La disponibilidad de estas reglas comunitarias facilita la implementación y mejora continua de la seguridad del sistema.

- Para verificar el funcionamiento de la regla IDS, se utilizó un ataque DoS que bloqueó el servicio web. Con esto, Suricata pudo detectar el ataque correctamente, demostrando su efectividad en la identificación de amenazas y la protección del sistema contra ataques de denegación de servicio

## 5 RECOMENDACIONES

- Para implementar más servidores en el inventario de Ansible, será necesario configurar el acceso SSH para el usuario root y crear un llavero que incluya todos a los servidores. Esto permitirá automatizar todos los servidores simultáneamente sin necesidad de ingresar individualmente el usuario y la contraseña, los cuales suelen ser distintos para cada servidor.
- En YAML, es crucial tener bien definidas las reglas gramaticales, ya que cualquier error en la sintaxis puede provocar fallos en la ejecución del *playbook*. Por esto, se debe asegurar que la sintaxis esté correctamente estructurada para que el sistema funcione como se espera.
- Es importante asegurarse de que la regla implementada esté bien estructurada de lo contrario, el sistema IDS podría generar falsos positivos. Una regla mal configurada puede interpretar actividades legítimas como amenazas, lo que podría llevar a alertas innecesarias y a una posible pérdida de confianza en el sistema de detección de intrusos.
- Es fundamental tener en cuenta las versiones del sistema operativo Linux que se utilizan, ya que esto puede afectar en la disponibilidad de paquetes o que existan errores que aún no tienen solución por parte de la comunidad o del desarrollador.
- Se debe tener en cuenta que las direcciones IP se configuran de manera estática en los servidores para mantener un control preciso sobre la infraestructura. Esto evita la necesidad de reconfigurar los servidores si se utiliza DHCP. Al configurar las direcciones IP estáticamente, se asegura que cada servidor tenga una dirección específica, facilitando la gestión y asegurando la consistencia en la conectividad dentro de la red.

## 6 REFERENCIAS BIBLIOGRÁFICAS

- [1] Atlassian, «¿Qué es DevOps?», Atlassian. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.atlassian.com/es/devops>
- [2] «Ubuntu: qué es y cómo instalarlo», Neolo Blog. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.neolo.com/blog/ubuntu-que-es-y-como-instalarlo.php>
- [3] «Oracle VM VirtualBox». Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.oracle.com/es/virtualization/virtualbox/>
- [4] «¿Qué es un Sistema de Detección de Intrusos (IDS)? - Software Check Point», Check Point Software. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.checkpoint.com/es/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/>
- [5] C. Cilleruelo, «¿Qué es Suricata en ciberseguridad? | KeepCoding Bootcamps». Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://keepcoding.io/blog/que-es-suricata-en-ciberseguridad/>
- [6] C. Cilleruelo, «¿Qué es OSSEC? | KeepCoding Bootcamps». Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://keepcoding.io/blog/que-es-ossec/>
- [7] «Herramientas open source de detección de intrusión», OpenWebinars.net. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://openwebinars.net/blog/las-8-mejores-herramientas-open-source-de-deteccion-de-intrusion/>
- [8] D. A. C. Rodríguez, «DISEÑO DE UN SISTEMA DE SEGURIDAD PARA LA PROTECCIÓN Y PREVENCIÓN DE INTRUSOS IDS/IPS EN LA RED EMPRESARIAL DE PUNTOQOM MINIMIZANDO EL RIESGO Y ASEGURANDO LOS ACTIVOS DE INFORMACIÓN DE LA ORGANIZACIÓN».
- [9] D. A., «¿Cómo funciona el SSH?», Tutoriales Hostinger. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.hostinger.es/tutoriales/que-es-ssh>
- [10] «¿Qué son las llaves SSH?», Linube. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://linube.com/ayuda/articulo/306/que-son-las-llaves-ssh>
- [11] «¿Qué es un ciberataque y los tipos de ataques en la red?», Fortinet. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.fortinet.com/lat/resources/cyberglossary/types-of-cyber-attacks.html>
- [12] S. Manjaly, «Qué es Ansible: la herramienta DevOps para automatizar tareas de IT». Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://blog.invgate.com/es/ansible>
- [13] «Introducción a los *Playbooks* en Ansible», OpenWebinars.net. Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://openwebinars.net/blog/playbooks-en-ansible/>
- [14] «¿Qué es YAML?» Accedido: 2 de junio de 2024. [En línea]. Disponible en: <https://www.redhat.com/es/topics/automation/what-is-yaml>
- [15] O. M. F. Alzate, «¿Qué es puppet?», <http://codigoelectronica.com>. Accedido: 15 de junio de 2024. [En línea]. Disponible en: <http://codigoelectronica.com/blog/que-es-puppet>
- [16] «Automatizar tareas con Ansible, Chef y Puppet | OpenWebinars», OpenWebinars.net. Accedido: 15 de junio de 2024. [En línea]. Disponible en: <https://openwebinars.net/blog/automatizar-tareas-con-ansible-y-puppet/>
- [17] «Best DevOps Tools in 2024», staragile.com. Accedido: 15 de junio de 2024. [En línea]. Disponible en: <https://staragile.com/blog/what-is-chef-in-devops>

## **7 ANEXOS**

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Códigos Playbooks

# ANEXO I: Certificado de Originalidad

## CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 23 de julio de 2024

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad del Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE SERVICIOS DE SEGURIDAD EN REDES MEDIANTE DEVOPS asociado a la IMPLEMENTACIÓN DE UN IDS MEDIANTE HERRAMIENTAS DE DEVOPS, elaborado por la estudiante ANDRÉS ALCIDES GUAMÁN PULLAS 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 12%.

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.

[https://epnecuador-my.sharepoint.com/:f:/g/personal/fernando\\_becerrac\\_epn\\_edu\\_ec/EjFNusJ1-cZGg5jufx2rSMIB0a-EeyNgVvfQzEO6GVYR8A?e=dM7p2I](https://epnecuador-my.sharepoint.com/:f:/g/personal/fernando_becerrac_epn_edu_ec/EjFNusJ1-cZGg5jufx2rSMIB0a-EeyNgVvfQzEO6GVYR8A?e=dM7p2I)

Atentamente,

FERNANDO VINICIO BECERRA CAMACHO

Docente

Escuela de Formación de Tecnólogos

## ANEXO II: Enlaces



## ANEXO III: Códigos Fuente

### Anexo III.I Código *playbook* "tarea.yml"

```
---
#llama a todos los servidores de mi inventario
- hosts: all
#dar permisos de superusuario
  become: yes
#empiezan las tareas
  tasks:
#nombre de las tareas
  - name: instala apache2
#llama a instalar los paquetes en su ultima version
    apt: name=apache2 state=latest
#nombre de la tarea
  - name: instala suricata
#llama a instalar los paquetes en su ultima version
    apt: name=suricata state=latest
#nombre de la tarea
  - name: corre suticata
#habilita e inicia suricata
    service: name=suricata state=started enabled=yes
#nombre de la tarea
  - name: instala curl
#llama a instalar los paquetes en su ultima version
    apt: name=curl state=latest
#nombre de la tarea
  - name: instala jq
#llama a instalar los paquetes en su ultima version
    apt: name=jq state=latest
```

### Anexo III.II Código *playbook* “configuracion.yml”

```
---
#llama a todos los servidores que esten en mi inventario
- hosts: all
#habilita superusuario
  become: yes
#tareas
  tasks:
    - name: configura ip
#para reemplazar el codigo de una linea
      replace:
#ubicacion de un archivo a reemplazar su codigo de linea
        path: /etc/suricata/suricata.yaml
#codigo de linea a reemplazar
        regexp: '192.168.0.0/16,10.0.0.0/8,172.16.0.0/12'
#reemplazo
        replace: '192.168.110.184'
    - name: configura reglas de comunidad
      replace:
        path: /etc/suricata/suricata.yaml
        regexp: 'community-id: false'
        replace: 'community-id: true'
    - name: configura interface
      replace:
        path: /etc/suricata/suricata.yaml
        regexp: 'eth0'
        replace: 'enp0s3'
    - name: configura ruta de reglas
      replace:
        path: /etc/suricata/suricata.yaml
        regexp: '/var/lib/suricata/rules'
        replace: '/etc/suricata/rules/'
    - name: reinicia servicio
#comandos utilizar comandos shell en el servidor por medio de ssh
      shell: systemctl restart suricata
```

### Anexo III.III Código *playbook* "interactivo.yml"

```
---
#llama a todos los servidores que se encuentren en mi inventario
- hosts: all
#da permisos de superusuario
  become: yes
#para añadir tareas
  tasks:
  - name: BIENVENIDO/A
#para el proceso del playbook
  pause:
#imprime en pantalla
  prompt: "¿Cual es tu nombre?"
#guarda lo que el usuario indica
  register: user_input
  - name: Mostrando usuario
#muestra un mensaje al usuario y el nombre que se haya guardado
  debug:
    msg: "Buen dia, {{ user_input.user_input }}."
  - name: Reglas para instalar
  pause:
#imprime en pantalla un el menu
  prompt:
    ¿Que reglas paquetes deseas instalar?
    1. Deteccion de Movimiento lateral
    2. Deteccion de malware
    3. Deteccion DOS
    Seleccione la opcion
#registra la opcion del usuario
  register: menu_input
  - name: procesando opcion seleccionada
#procesa la informacion del usuario y la opcion
  set_fact:
    option: "{{ menu_input.user_input }}"
  - name: Deteccion de Movimiento Lateral
#para añadir un subbloque
```



```

block:
  - name: Deteccion de Movimiento lateral
#muestra un mensaje al usuario
  debug:
    msg: "Se instalara el paquete de reglas para la deteccion de MOVIMIENTO
LATERAL"
#mientras la opcion es igual a 1 se ejecutara el proceso del subbloque
  when: option == "1"
  - name: añadiendo paquete de reglas movimiento lateral
#ejecuta comandos shell del servidor
  shell: suricata-update enable-source stamus/lateral
- name: Deteccion de malware
block:
  - name: Deteccion de malware
  debug:
    msg: "Se instalara el paquete de reglas para la deteccion de MALWARE"
  when: option == "2"
  - name: añadiendo paquetes de reglas malware
  shell: suricata-update enable-source scwx/malware
- name: Deteccion DOS
block:
  - name: Deteccion DOS
  debug:
    msg: "Se instalara el paquete de reglas para la deteccion de DOS"
  when: option == "3"
  - name: añadiendo el paquete de reglas DOS
  shell: suricata-update enable-source oisf/traffid
- name: actualizando la reglas
  shell: suricata-update -o /etc/suricata/rules
- name: verificando errores en las reglas
  shell: suricata -T -c /etc/suricata/suricata.yaml -v
- name: reiniciando servicio suricata
  service: name=suricata state=restarted enabled=yes

```