UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Análisis de seguridad en una nube privada orquestada por OpenStack

TESIS

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A

Iker Alberto Cedillo Martínez

DIRECTOR DE TESIS

Dr. Luis Francisco García Jiménez

Ciudad Universitaria, Cd. Mx., 2025

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado ANALISIS DE SEGURIDAD EN UNA NUBE PRIVADA ORQUESTADA POR OPENSTACK que presenté para obtener el titulo de INGENIERO EN TELECOMUNICACIONES es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

IKER ALBERTO CEDILLO MARTINEZ
Número de cuenta: 114004660

Agradecimientos

A mi familia

Quiero agradecer profundamente a mi familia, cuya inspiración ha sido el motor que me ha impulsado a seguir sus pasos y esforzarme por convertirme en un ingeniero extraordinario. Su ejemplo y los valores que me inculcaron durante mi crecimiento han guiado mi camino, motivándome a servir a la sociedad con dedicación y compromiso.

A mis amigos

Quiero expresar mi más profundo agradecimiento a las maravillosas personas que me acompañaron durante la carrera: Luis Ernesto Del Arco Campo, Joshua Adair García Benítez, Mariana Guadalupe Rojas Hernández y Sebastián Yetzcani Suárez Caballero. Aunque nuestra convivencia fue limitada por la pandemia, su impacto en mi vida ha sido invaluable. Siempre serán una parte fundamental de quien soy, y no importa cuánto tiempo pase ni lo que el futuro nos depare, siempre llevaré conmigo el privilegio y la alegría de haberlos conocido.

A la Universidad Nacional Autónoma de México

En primer lugar, quiero expresar mi más sincero agradecimiento a la Máxima Casa de Estudios de México, la UNAM. Durante más de 10 años, esta noble institución me ha brindado conocimiento, amistades invaluables y un profundo sentido de responsabilidad social. Gracias a ello, he podido forjarme como el profesionista y ser humano que hoy soy.

Quiero expresar mi gratitud a todos los profesores que he tenido a lo largo de mi formación. Muchos de ellos me han impulsado a superarme, a aspirar a ser más, y me han abierto las puertas para descubrir quién puedo llegar a ser y cómo lograrlo. En especial, deseo agradecer profundamente al Dr. Luis Francisco García Jiménez, cuyo apoyo incansable ha sido incomparable y fundamental en este camino.

Finalmente, agradezco el apoyo dado por el proyecto PAPIIT-IN109624.

Índice general

Índice de figuras

Índice de Consola

Acrónimos

1. IP: *Internet Protocol*, protocolo de Internet.
2. API: *Application Programming Interface*, interfaz de programación de aplicaciones.
3. CPU: *Central Processing Unit*, unidad central de procesamiento.
4. MB: Megabyte.
5. VM: *Virtual Machine*, máquina virtual.
6. TCP: *Transport Control Protocol*, protocolo de control de transporte.
7. DNS: *Domain Name Server*, servidor de nombre de dominios.
8. IES: *Institución de Educación Superior*.
9. CA: *Certificate Authority*, unidad certificadora.
10. HTTPS: *Hypertext Transfer Protocol Secure*, protocolo de transferencia de hipertexto seguro.
11. GUI: *Graphical User Interface*, interfaz gráfica de usuario.
12. IDPS: *Intrusion Detection and Prevention System*, sistema de detección y prevención de intrusiones.
13. SMTP: *Simple Mail Transfer Protocol*, protocolo simple de transferencia de correo.
14. SSH: *Secure Shell*.

Resumen

Internet le ha permitido a muchas organizaciones empresariales acceder a sus recursos de cómputo, almacenamiento y servicios desde cualquier parte del mundo. Este modelo se denomina infraestructura como servicio (IaaS). Actualmente, existen diversos proveedores de estas soluciones que operan bajo tarifas que varían según las necesidades del cliente. A pesar de que las soluciones IaaS ofrecidas en el mercado suelen tener muchas ventajas, como no adquirir infraestructura física (servidores, routers, switches, etc.), numerosas organizaciones optan por montar sus propios servicios utilizando plataformas de código abierto. Esto les permite gestionar y personalizar sus recursos de manera privada, dado que sus datos no deben ser compartidos.

Existen múltiples plataformas IaaS de código abierto y una de las más confiables es OpenStack, respaldada por Red Hat. Esta plataforma utiliza diversos módulos que permiten a los administradores orquestar su propia nube privada de forma flexible. Sin embargo, es común que los administradores no suelen aplicar las herramientas de seguridad que esta plataforma ofrece, lo que puede crear vulnerabilidades que la hacen susceptible a diversos ataques como fuerza bruta, escaneo de puertos, entre muchos más. Estos ataques no solo comprometen la integridad y confidencialidad de la información almacenada, sino que también representan un riesgo significativo para los usuarios que confían en estos entornos.

Este proyecto de tesis se estructura en dos etapas. En la primera fase, se implementa una nube privada OpenStack con múltiples medidas de seguridad, de tal suerte que esté protegida frente a diversas amenazas cibernéticas. En la segunda fase, se evalúa la fortaleza de la nube a través de la ejecución de pruebas controladas de vulnerabilidad mediante ataques de pentesting, y se compara con una nube sin medidas de seguridad. A través de esta comparación, se determina el impacto de las implementaciones de seguridad con base al estándar internacional CVSS 3.0. Finalmente, se proponen recomendaciones para mejorar la seguridad en este tipo de sistemas.

Capítulo 1

Introducción

En la actualidad, Internet juega un papel fundamental en la vida de las personas, ya que permite el acceso a una cantidad prácticamente ilimitada de recursos y servicios, muchos de los cuales se han vuelto indispensables en nuestra vida cotidiana. La mayoría de estos servicios son ofrecidos por empresas que gestionan su propia infraestructura tecnológica, compuesta por servidores, routers, switches, entre otros dispositivos. No obstante, para algunas empresas adquirir infraestructura propia no siempre es justificable, dado que el acceso a sus recursos es fluctuante. Y por esta razón, estas empresas optan por rentar los recursos de cómputo, pagando únicamente por lo que utilizan en lugar de realizar grandes inversiones en equipos que podrían permanecer subutilizados si la demanda disminuye.

Una solución que ha ganado popularidad para abordar estas necesidades es la computación en la nube o cloud computing. Esta tecnología permite el acceso bajo demanda a recursos informáticos, servicios, almacenamiento y respaldo de datos proporcionados por un proveedor de servicios mediante diferentes tarifas. Existen diversos modelos de computación en la nube, entre los cuales destacan: Infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS).

Aunque la computación en la nube elimina la necesidad de adquirir infraestructura propia, muchas empresas y organizaciones optan por no utilizar proveedores externos debido a preocupaciones relacionadas con la privacidad de los datos. En entornos donde la confidencialidad es esencial, como en hospitales, instituciones financieras o educativas, el acceso del proveedor a la información almacenada puede representar un riesgo significativo. En este contexto, una alternativa viable es la creación de una nube privada, que permite a las organizaciones gestionar sus propios servicios de manera autónoma. Aunque esta opción implica adquirir y mantener infraestructura física, ofrece mayores garantías de privacidad, así como elimina la dependencia de terceros y brinda mayor flexibilidad en la implementación y personalización del servicio en la nube.

Entre las herramientas disponibles para diseñar nubes privadas, OpenStack se destaca como una solución de código abierto ampliamente utilizada para la implementación de modelos IaaS. OpenStack, desarrollado inicialmente por Red Hat, facilita la gestión de almacenamiento y creación de máquinas virtuales, además ha sido adoptada por importantes compañías como IBM, Dell, VMware, Verizon, Oracle, Ericsson, entre muchas más [1].

A pesar de su facilidad de uso, uno de los desafíos asociados con OpenStack es que, en muchos casos, no se aplican todas las configuraciones de seguridad que la plataforma ofrece, ni se implementan medidas adecuadas en los servidores que la alojan. Esto puede dejar expuesta la información de los usuarios y representar un riesgo para las organizaciones que confían en estos entornos. Por ejemplo, los atacantes pueden aprovechar estas vulnerabilidades para comprometer datos sensibles, afectando tanto a las organizaciones como a sus clientes.

En este trabajo de tesis, se implementa una nube privada utilizando OpenStack sobre Red Hat Enterprise Linux 9 (RHEL 9); una distribución reconocida por su robustez y seguridad. Además, se añade un conjunto de medidas de seguridad tanto en la plataforma como en el servidor que la aloja, con el objetivo de reforzar la protección de los datos. Posteriormen-

te, se evalúa la robustez de estas medidas mediante pruebas controladas de ataques utilizando varias herramientas de pentesting. Para ello, se va a comparar nuestra nube con otra nube implementada en Ubuntu Server 22.04 que cuenta únicamente con la configuración básica de OpenStack sin medidas de seguridad. A través de esta comparación, se cuantifica el nivel de vulnerabilidad con base en el estándar internacional CVSS 3.0. Finalmente, se proponen recomendaciones para mejorar la seguridad en implementaciones futuras.

1.1. Definición del problema

Una nube privada en una organización representa una herramienta fundamental para ofrecer servicios a través de la red. Entre estos servicios se encuentra el almacenamiento, correo electrónico, plataformas de desarrollo, despliegue de aplicaciones para uso comercial, entre otros. Esta infraestructura es especialmente útil en contextos donde el acceso físico a los servidores está limitado, como ocurrió durante la pandemia del COVID-19, que evidenció la necesidad de soluciones tecnológicas más robustas y accesibles. Sin embargo, la implementación de una nube privada conlleva muchos desafíos relacionados con la seguridad. Las organizaciones manejan grandes volúmenes de información sensible como datos personales de sus trabajadores y clientes, lo que las convierte en un objetivo atractivo para atacantes malintencionados. Por lo tanto, cualquier brecha en la seguridad podría comprometer la integridad, confidencialidad y disponibilidad de esta información, afectando tanto la reputación de la organización como la confianza de sus usuarios, por lo que garantizar la seguridad de una nube privada es una tarea fundamental.

1.2. Alcance de la tesis

El presente trabajo de tesis hace un análisis de seguridad en una nube privada orquestada por OpenStack. Específicamente, se enfoca en tres vectores de ataque críticos: la interfaz web para la administración de OpenStack llamada Horizon, el servicio de acceso remoto SSH y las conexiones internas entre los módulos de OpenStack. Los ataques abordados en esta tesis son ataques de fuerza bruta automatizados y de escaneo de puertos.

1.3. Objetivo

Desarrollar e implementar medidas de seguridad en una nube privada orquestada con OpenStack, evaluando su robustez a través de pruebas de penetración y análisis de vulnerabilidades, con el propósito de fortalecer la protección de la plataforma frente a posibles ataques cibernéticos.

1.4. Hipótesis

La implementación de medidas de seguridad en una IaaS orquestada con OpenStack enfocadas en la restricción de exposición de puertos, mitigación de ataques por fuerza bruta, cifrado de conexiones internas, así como la creación de imágenes ISO con medidas de seguridad preconfiguradas, reducirá significativamente las puertas de ataque y fortalecerá la protección tanto de la nube como del servidor que la aloja.

1.5. Justificación

La creciente demanda de infraestructuras en la nube ha generado nuevos desafíos en materia de seguridad, especialmente en entornos privados, donde la configuración y protección de los sistemas dependen directamente de los administradores. OpenStack, como plataforma de orquestación de nubes privadas, ofrece una gran flexibilidad y control. Sin embargo,

también introduce riesgos si no se implementan adecuadamente las medidas de seguridad necesarias, ya que su instalación por defecto, no cuenta con una configuración segura. Este trabajo de tesis propone estrategias de protección enfocadas en los principales vectores de ataque tanto de OpenStack como de los servidores que la alojan. Se busca que estas soluciones sean efectivas, de fácil implementación y compatibles con distintos entornos, sin requerir una inversión significativa, facilitando así su adopción en organizaciones con recursos limitados. Además, a través de pruebas de penetración y análisis de vulnerabilidades, este trabajo permite evaluar la efectividad de las medidas propuestas, proporcionando un enfoque práctico y replicable para mejorar la seguridad de las nubes basadas en OpenStack. Los resultados beneficiarán tanto a administradores de sistemas como a organizaciones que buscan adoptar infraestructuras en la nube con un mayor nivel de seguridad, privacidad y resiliencia frente a ciberataques comunes.

1.6. Metodología

El desarrollo de este proyecto de tesis se llevará a cabo en las siguientes etapas:

- Configuración de dos servidores virtuales utilizando QEMU/KVM con RHEL 9 como sistema operativo, con el fin de crear los nodos de OpenStack (Compute y Controller).
- Instalación, configuración, optimización y adaptación de los componentes de OpenStack para reforzar la seguridad interna, así como la protección del Compute y del Controller.
- Creación de imágenes en la nube que incluyan medidas de seguridad básicas y servicios mínimos esenciales.
- Análisis de seguridad mediante ataques de penetración en la nube privada configurada con RHEL 9, así como a una nube privada implementada en Ubuntu Server sin medidas de seguridad.
- Comparación y evaluación de vulnerabilidades bajo el estándar CVSS 3.0 entre ambas nubes privadas.

1.7. Estructura de la tesis

El contenido de esta tesis se organiza de la siguiente manera:

- Capítulo 2: Se presentan los conceptos básicos necesarios para la implementación de las medidas de seguridad. Además, se ofrece una explicación general de los programas y herramientas que se utilizaron, así como su propósito en el contexto del proyecto.
- Capítulo 3: Se describe la estructura de OpenStack, se detallan los componentes y configuraciones relevantes. También, se especifican las medidas de seguridad que se implementaron en cada uno de los nodos.
- Capítulo 4: Se detalla la implementación de las mejoras de seguridad, tanto para la nube privada como para los servidores que la albergan.
- Capítulo 5: Se explica el proceso de creación de máquinas virtuales dentro de la nube, incorporando las implementaciones de seguridad necesarias para garantizar un entorno seguro.
- Capítulo 6: Se realiza un análisis de seguridad exhaustivo, evaluando la nube privada implementada en esta tesis contra una nube privada insegura (no implementada en esta tesis). Los resultados obtenidos se analizan en detalle.
- Capítulo 7: Se presentan las conclusiones generales, así como las posibles mejoras basándose en los resultados obtenidos durante las pruebas de seguridad.

Capítulo 2

Antecedentes

En este capítulo se exponen los conceptos fundamentales necesarios para comprender el significado de una nube privada y los elementos asociados a su implementación. Asimismo, se abordan los conceptos clave relacionados con la seguridad informática y se presentan los programas que se utilizan a lo largo del desarrollo de este trabajo.

2.1. Computación en la nube

La computación en la nube es un modelo de servicio que proporciona recursos de procesamiento, almacenamiento y gestión de datos a cambio de un costo determinado. Este enfoque permite a los clientes evitar la necesidad de adquirir y mantener infraestructura informática propia, optimizando costos y recursos [2].

2.1.1. Infraestructura como servicio (IaaS)

Este modelo ofrece servicios de infraestructura tecnológica, incluyendo herramientas para la gestión de redes, almacenamiento y virtualización. Proporciona al cliente un control elevado sobre cómo manejar los recursos que alquila, lo que lo hace ideal para usuarios que necesitan flexibilidad y personalización [3]. Algunos ejemplos destacados de `IaaS` son `Amazon Web Services (AWS)`, `Microsoft Azure`, `Google Cloud`, y `OpenStack`.

2.1.2. Plataforma como servicio (PaaS)

Este modelo ofrecen recursos de `hardware` y `software` para desarrollar aplicaciones directamente en la nube [3]. A diferencia de `IaaS`, las empresas no administran los recursos tecnológicos subyacentes, sino que se centran en utilizar las herramientas proporcionadas para el desarrollo de sus aplicaciones. Ejemplos populares de `PaaS` incluyen `Google App Engine`, `Microsoft Azure App Service`, e `IBM Cloud Foundry`.

2.1.3. Software como servicio (SaaS)

Este modelo proporciona aplicaciones completamente gestionadas como servicios accesibles en la nube [3]. Incluye toda la infraestructura necesaria para su funcionamiento, así como la gestión de actualizaciones y resolución de errores, permitiendo al usuario final enfocarse únicamente en el uso de la aplicación sin preocuparse por su mantenimiento interno. Ejemplos comunes de `SaaS` incluyen `Microsoft Office 365`, `Dropbox`, y `Zoom`.

2.2. Red Hat Enterprice Linux

`Red Hat Enterprice Linux` (`RHEL`) es una distribución de Linux desarrollada por `Red Hat`, diseñada específicamente para entornos empresariales [4]. Es ampliamente reconocida por su

robustez, alto nivel de seguridad, estabilidad y flexibilidad, lo que la convierte en una elección preferida para organizaciones que necesitan soluciones confiables y escalables.

RHEL está optimizada para implementaciones críticas, como servidores, infraestructuras de la nube y plataformas de contenedores, ofreciendo soporte para arquitecturas modernas y tecnologías avanzadas. Además, proporciona una gestión eficiente de recursos y herramientas integradas que facilitan la automatización, el monitoreo y la administración del sistema.

Para aprovechar al máximo las características de esta distribución, es necesario adquirir una suscripción a RHEL. Esto permite acceder a actualizaciones, parches de seguridad y soporte técnico especializado, incluyendo asistencia profesional 24/7, certificaciones de hardware y software, y herramientas exclusivas como Red Hat Insights para la gestión proactiva de sistemas. Sin embargo, Red Hat también ofrece una opción gratuita para desarrolladores, que brinda acceso a RHEL 9 sin soporte técnico ni mantenimiento, ideal para fines de aprendizaje y pruebas en entornos no críticos.

2.3. OpenStack

OpenStack es una plataforma de código abierto basada en el modelo IaaS de la computación en la nube [5]. Aunque es respaldada y utilizada ampliamente por empresas como Red Hat, su desarrollo está liderado por una comunidad global. Esta plataforma permite a las organizaciones construir y administrar su propia nube privada, otorgándoles la flexibilidad de gestionar los recursos según sus necesidades específicas y las aplicaciones que deseen ofrecer o crear. OpenStack está compuesta por una arquitectura modular, con componentes diseñados para cubrir diferentes aspectos de la infraestructura en la nube. Entre los principales módulos se encuentran:

- **Nova**: Responsable de la gestión de instancias virtuales, como máquinas virtuales y contenedores. Proporciona escalabilidad, lo que permite a las empresas adaptarse dinámicamente a las necesidades de carga.
- **Neutron**: Administra redes virtuales, facilitando la creación de redes flexibles y personalizables que soportan conectividad interna y externa.
- **Keystone**: Maneja la autenticación y autorización en la plataforma, garantizando la seguridad en el acceso a los recursos y módulos de OpenStack mediante políticas de control de acceso.
- **Horizon**: Ofrece una interfaz web intuitiva que permite a los usuarios gestionar fácilmente sus recursos y servicios, desde la creación de instancias hasta la administración de redes.
- **Glance**: Se encarga de almacenar y gestionar imágenes de máquinas virtuales, ofreciendo soporte para múltiples formatos y la posibilidad de que los usuarios personalicen sus propias imágenes.
- **Placement**: Se encarga de gestionar los recursos disponibles en la nube, como CPU, memoria y almacenamiento, y proporciona esta información a otros servicios, como Nova, para optimizar la asignación de recursos en las máquinas virtuales.

Otros módulos complementarios incluyen Cinder para el almacenamiento en bloque, Swift para el almacenamiento de objetos, y Heat, que automatiza el despliegue de aplicaciones utilizando plantillas.

OpenStack también se auxilia de varias herramientas, de las cuales se destacan 3, etcd, MySQL, RabbitMQ.

- **etcd**: Actúa como un almacén de claves/valores distribuido y altamente disponible. OpenStack lo utiliza principalmente para gestionar configuraciones distribuidas y proporcionar sincronización entre servicios en clústeres (por ejemplo, para Neutron en entornos de alta disponibilidad).

- **MySQL**: Es el backend más comúnmente utilizado para almacenar los datos estructurados de los servicios de OpenStack. Cada módulo (Keystone, Nova, Glance, etc.) guarda información en tablas específicas dentro de la base de datos.

- **RabbitMQ**: Facilita la comunicación entre los diferentes servicios de OpenStack mediante un sistema de mensajería basado en colas. Es el intermediario que asegura que los módulos puedan enviarse y recibir mensajes de forma asíncrona.

2.4. Certificados SSL/TLS

Un certificado es una herramienta clave para verificar la identidad de un sistema y establecer una conexión de red cifrada con un cliente mediante el protocolo Secure Socket Layer/Transport Layer Security (SSL/TLS) [6]. Estos certificados se basan en un sistema criptográfico de clave pública (PKI), que permite a los sistemas verificar la identidad de un servidor confiando en una autoridad certificadora (CA), encargada de firmar los certificados para validar su autenticidad.

El funcionamiento de los certificados SSL/TLS en páginas web es el siguiente:

1. El servidor genera un par de claves (pública y privada) y solicita a la CA que firme su certificado.

2. La CA firma el certificado, asociándolo con la identidad del servidor, y devuelve un archivo que incluye la clave pública del servidor.

3. Durante una conexión HTTPS, el servidor envía este certificado firmado al navegador del cliente.

4. El navegador valida la autenticidad del certificado utilizando la clave pública de la CA.

5. Si la validación es exitosa, se establece un canal cifrado entre el servidor y el cliente utilizando la clave privada del servidor para descifrar los datos.

Esta tecnología permite que la comunicación entre una página web y un cliente esté cifrada, y a su vez garantiza la autenticidad y el no repudio de los datos transmitidos.

No siempre es necesario solicitar la firma de una CA. Es posible crear una autoridad certificadora interna y utilizarla para firmar el propio certificado, lo que se conoce como un certificado auto-firmado. Sin embargo, es importante destacar que muchos navegadores y aplicaciones alertarán al usuario de que el sitio no es seguro, ya que no pueden confiar completamente en un certificado auto-firmado debido a la falta de una entidad de confianza reconocida que lo respalde.

2.5. Ataques comunes dirigidos a servidores y páginas web

En el ámbito de la ciberseguridad, existen innumerables formas en las que un sistema puede ser atacado. En muchos casos, los atacantes combinan diferentes tipos de ataques, dando lugar a técnicas altamente sofisticadas y difíciles de detectar, sin embargo, existen ataques mucho más simples que pueden afectar gravemente un servidor o servicio en la red. A continuación, se presentan algunos de los ataques más comunes dirigidos a servidores y páginas web.

- **Ataques de fuerza bruta automatizados**: Un ataque de fuerza bruta es un método en el que se realizan intentos repetidos y sistemáticos para adivinar o encontrar un valor desconocido. Se basa en la prueba y error, explorando todas las combinaciones posibles hasta encontrar la correcta [7]. Este tipo de ataques cuenta con muchas ramas, una de ellas es un ataque de fuerza bruta para la obtención de credenciales de inicio de sesión y un ataque de fuerza bruta para realizar una búsqueda de rutas ocultas en paginas web.

- **Escaneo de puertos**: Un ataque de escaneo de puertos utiliza programas especiales que detectan y recopilan información sobre los puertos abiertos en un sistema que son usados por diferentes servicios para detectar posibles puntos de entrada y vulnerabilidades asociadas a los servicios expuestos [8].

- **Ataque de sondeo de red**: Un ataque de sondeo de red consiste en capturar y analizar paquetes de datos que circulan a través de un sistema de comunicaciones, con el objetivo de extraer información sensible contenida en ellos [9]. Este tipo de ataque suele aprovechar la falta de cifrado en las comunicaciones para acceder a datos como credenciales, mensajes o información privada.

2.6. Herramientas de análisis de tráfico y seguridad en redes

En `RHEL 9`, así como en otras distribuciones de Linux, existen varias herramientas que permiten analizar el tráfico de red y obtener información sobre los recursos y servicios compartidos por el servidor. Estas herramientas son fundamentales para identificar posibles vulnerabilidades en la red y en el sistema, lo que facilita la implementación de medidas correctivas.

2.6.1. Nmap

`Nmap` es una herramienta de código abierto utilizada principalmente para explorar redes y realizar auditorías de seguridad [10]. Es conocida por su capacidad para descubrir dispositivos y servicios activos dentro de un segmento de red, así como por su capacidad para identificar puertos abiertos en servidores. Además, de escanear puertos, `Nmap` permite realizar un análisis detallado de la red por lo que es una herramienta esencial en auditorías de seguridad, que permite a los administradores de red identificar puntos débiles en sus infraestructuras y fortalecer la protección de sus sistemas. Para conocer los principales parámetros del comando `Nmap` y sus principales aplicaciones, puede consultar el apéndice B.1.

2.6.2. Wireshark

`Wireshark` es una herramienta de código abierto ampliamente utilizada para el análisis de protocolos de red, que captura y examina los paquetes transmitidos entre un cliente y un servidor, o entre servidores [11]. Gracias a su interfaz gráfica intuitiva, `Wireshark` se ha convertido en una de las herramientas más populares entre los analistas de seguridad, quienes la emplean para diagnosticar problemas de conectividad y detectar vulnerabilidades en las redes. Su capacidad para proporcionar detalles precisos sobre cada paquete capturado facilita la resolución de incidencias en la red y la identificación de posibles amenazas. `Wireshark` está diseñado principalmente para usarse en sistemas con entorno gráfico, lo que limita su implementación en distribuciones de Linux sin interfaz gráfica. Como alternativa en estos casos, existe `Tshark`, una versión de línea de comandos que realiza las mismas funciones que `Wireshark`.

2.7. Herramientas de pruebas de penetración

Dentro del ecosistema Linux existen diversas herramientas de código abierto diseñadas específicamente para realizar pruebas de penetración en servidores, con el objetivo de evaluar su robustez y resistencia frente a posibles intrusiones. Estas herramientas deben ser utilizadas exclusivamente en entornos controlados y con los permisos adecuados, ya que su uso sin autorización puede acarrear consecuencias legales para quienes las empleen.

2.7.1. Gobuster

Gobuster es una herramienta de código abierto que permite identificar el contenido web, específicamente los directorios o archivos expuestos de un sitio web [12]. Gobuster realiza una serie de solicitudes HTTP utilizando un diccionario de subdirectorios comunes y detecta si estos existen en el servidor, proporcionando información sobre accesos directos disponibles. Para conocer los principales parámetros del comando Gobuster y sus principales aplicaciones, se puede consultar el Apéndice B.2.

2.7.2. Hydra

Hydra es una herramienta de código abierto utilizada para realizar ataques de fuerza bruta en aplicaciones web con formularios de inicio de sesión y servidores SSH [13]. Es conocida por su rapidez y flexibilidad, ya que trabaja con módulos fácilmente configurables. Su propósito principal es evaluar la vulnerabilidad de una aplicación web o servidor frente a ataques de fuerza bruta, probando diversas combinaciones de credenciales hasta encontrar una válida. Para conocer los principales parámetros del comando Hydra y sus principales aplicaciones, se puede consultar el Apéndice B.3.

2.8. Herramientas de detección y protección de intrusos

Existen diversas herramientas diseñadas para proteger nuestros servidores frente a amenazas en la red. En particular, para los servidores web que disponen de un formulario de inicio de sesión, es crucial implementar alertas que detecten posibles ataques de fuerza bruta.

2.8.1. CSRF

El Cross-Site Request Forgery (CSRF) es una técnica de seguridad fundamental para proteger aplicaciones web como OpenStack de ataques maliciosos, como los de fuerza bruta [14]. Al exigir un token CSRF único en cada solicitud que modifica datos o realiza acciones sensibles, OpenStack garantiza que solo los usuarios legítimos puedan realizar acciones y que los ataques automatizados, como los de fuerza bruta, sean bloqueados. La protección CSRF está habilitada de manera predeterminada en Django, el cual se encarga de gestionar las vistas, los formularios y las interacciones con los usuarios, y también proporciona el marco para el sistema de autenticación y autorización que Horizon usa para interactuar con los usuarios de OpenStack.

2.8.2. Firewalld

Firewalld es una herramienta de gestión de firewall dinámica y de código abierto incluida en RHEL 9, diseñada para proporcionar control sobre el tráfico de red. A diferencia de herramientas más tradicionales, firewalld usa zonas y servicios para gestionar las reglas del firewall, lo que permite una configuración más flexible y sencilla [15]. Con firewalld, los administradores pueden definir reglas específicas según el contexto, como qué tráfico se permite o se bloquea en función de la red a la que está conectado el servidor. Su principal ventaja es que no requiere de un lenguaje complejo para escribir las reglas y cuenta con una interfaz gráfica que facilita el manejo de las zonas y las reglas. Además, firewalld permite realizar cambios sin necesidad de reiniciar el servicio, lo que facilita su uso en entornos de producción. Soporta tanto IPv4 como IPv6, y se integra con otras herramientas de seguridad del sistema para proteger servidores y redes contra accesos no autorizados. De manera predeterminada esta herramienta se encuentra activa en RHEL 9.

2.9. Herramientas para la personalización de imágenes ISO

La personalización de imágenes de Linux, es una práctica usual en entornos donde se necesitan sistemas operativos adaptados a requisitos específicos. Esto permite incluir paquetes, configuraciones y `scripts` predefinidos en la imagen base, reduciendo el tiempo de configuración y asegurando una uniformidad en su ejecución. Existen diferentes métodos para poder realizar esta personalización, no obstante, en este proyecto de tesis se utiliza `Cubic`.

2.9.1. Cubic

`Cubic` (`Custom Ubuntu ISO Creator`) es una herramienta gráfica que permite crear imágenes ISO personalizadas de `Ubuntu` y sus derivados, como `Linux Mint` [16]. Con `Cubic`, se puede modificar una instalación base de `Ubuntu` de manera sencilla, añadiendo o eliminando paquetes, configuraciones, y personalizaciones antes de generar una nueva imagen ISO sin necesidad de tener que realizar procesos que involucren el comando `chroot` manualmente.

2.10. Herramientas adicionales

2.10.1. Nginx

`Nginx` es un servidor web de alto rendimiento, conocido por su eficiencia y capacidad para manejar un gran número de conexiones simultáneas de manera eficiente [17]. `Nginx` es usado principalmente como servidor web y también como servidor proxy inverso, balanceador de carga y caché `HTTP`. Un proxy inverso es un tipo de servidor que recibe las solicitudes de los clientes y las dirige a uno o más servidores backend para procesarlas, y luego devuelve las respuestas al cliente. El proxy inverso se coloca entre los clientes y los servidores backend para manejar las solicitudes de manera más eficiente.

2.10.2. OpenSSL

`OpenSSL` es una biblioteca de software de código abierto que proporciona herramientas para implementar las funciones de seguridad necesarias en aplicaciones y sistemas [18]. Se utiliza para implementar protocolos de comunicación seguros, como `SSL` y `TLS`, y para generar, administrar y verificar certificados, claves y firmas digitales. `OpenSSL` se usa ampliamente en aplicaciones web, servidores y otros sistemas que necesitan comunicaciones seguras.

2.11. Mejoras en la seguridad de servidores Linux: Buenas prácticas y recomendaciones

La mayoría de los servidores desarrollados en distribuciones de Linux distintas a `RHEL` suelen contar con medidas de seguridad limitadas o incluso inexistentes al momento de la instalación. Esto contrasta con `RHEL`, que está diseñado específicamente para entornos empresariales, donde la seguridad es una prioridad fundamental. Sin embargo, es posible implementar medidas adicionales en otras distribuciones de Linux para fortalecer la seguridad de los servidores y hacerlos más robusto frente a posibles amenazas.

2.11.1. Mejoras de seguridad en SSH

Un servidor suele contar con el servicio de `SSH` para poder conectarse a él de manera remota, si no es necesario, se recomienda deshabilitar este servicio y cerrar el puerto 22. Este puerto en particular es uno de los más atacados, dado que es sabido que en su mayoría los servidores de manera predeterminada tiene el servicio de `SSH` habilitado [19]. Algunas de las recomendaciones de seguridad, si se requiere acceder al servicio, se listan a continuación.

- **Desactivar el acceso del `root`**: Esta es, sin duda, una de las prácticas más recomendadas al trabajar con `SSH`. Al deshabilitar el acceso directo del usuario `root`, se dificultan los intentos de acceso por fuerza bruta, ya que los atacantes no solo necesitarían adivinar la contraseña del `root`, sino también identificar el nombre de un usuario válido, lo cual es considerablemente más complejo. Además, también tendrían que descifrar la contraseña de dicho usuario. Sin embargo, es importante destacar que esta medida no elimina la posibilidad de que un atacante, tras obtener acceso con las credenciales de un usuario, escale privilegios utilizando comandos como `sudo -i` o `su -`.

- **Modificar configuraciones de accesos**: Es fundamental limitar la cantidad de intentos permitidos para que un cliente ingrese sus credenciales correctas. Si bien esto no elimina la posibilidad de un ataque de fuerza bruta, sí logra ralentizarlo, dificultando su ejecución. Asimismo, es recomendable establecer un tiempo límite para que el cliente introduzca sus credenciales; si no lo hace dentro de este período, la conexión debe cerrarse automáticamente. Además, se sugiere definir un número máximo de conexiones `SSH` simultáneas permitidas, lo que ayuda a evitar una sobrecarga innecesaria del servidor y reduce el riesgo de aceptar conexiones no deseadas.

- **Eliminar el acceso por contraseña**: Por defecto, la autenticación por contraseña está habilitada en `SSH`, lo que lo convierte en un objetivo frecuente de ataques de fuerza bruta. Sin embargo, deshabilitar el acceso por contraseña y habilitar la autenticación mediante pares de claves ofrece una solución mucho más segura. Este método requiere que el cliente genere un par de claves, compuesto por una clave pública y una clave privada. La clave pública se almacena en el servidor `SSH`, y cada vez que un cliente intente acceder, el servidor verificará la clave correspondiente, permitiendo el acceso sin necesidad de ingresar una contraseña. Aunque esta práctica no es comúnmente implementada, brinda un nivel más alto de protección para el servidor.

- **Cambiar el puerto predeterminado**: Una medida eficaz para dificultar los intentos de ataque es configurar el servicio `SSH` para que opere en un puerto diferente al predeterminado (22). Aunque los atacantes pueden detectar este nuevo puerto, no podrán determinar con certeza si corresponde a un servicio `SSH`, lo que añade una capa adicional de confusión y complica los ataques automatizados.

Estas medidas de seguridad combinadas pueden mejorar la protección de nuestro servidor sin la necesidad de bloquearlo por medio de un `firewall` o deshabilitarlo.

2.11.2. Limitaciones para superusuarios

En muchos servidores, los usuarios tienen la posibilidad de utilizar el comando `sudo -i` para obtener los mismos privilegios que el usuario `root`. Esto puede ser riesgoso, ya que les otorga acceso total a la configuración del servidor, incluyendo la capacidad de cambiar la contraseña del `root`. Para mitigar este riesgo, es posible ajustar la configuración de `sudo` para imponer reglas que restrinjan el uso de este comando. A continuación, se presentan algunas limitaciones recomendadas para fortalecer la seguridad.

- **No permitir el cambio de contraseña del `root`**: Con esta medida, los usuarios solo podrán cambiar su propia contraseña, evitando que la contraseña del `root` sea modificada por otra persona y garantizando que el acceso administrativo permanezca protegido.

- **No permitir que el usuario pueda usar `sudo -i`**: En este caso, el usuario no podrá acceder al `shell` del `root` como superusuario utilizando `sudo -i`. Esto representa una gran ventaja, ya que permite aplicar reglas más estrictas a los usuarios, quienes solo podrán ejecutar comandos con privilegios elevados utilizando `sudo` antes de cada comando y únicamente si tienen autorización previa del `root` para hacerlo.

- **Crear un archivo `log` de auditoria**: Si es necesario monitorizar cómo los usuarios utilizan el comando `sudo`, se puede configurar un archivo de registro que almacene cada

acción realizada por usuarios con privilegios elevados. Para garantizar la integridad de este archivo de auditoría, se recomienda protegerlo, evitando que cualquier usuario, excepto el `root`, pueda modificarlo o eliminarlo.

2.11.3. Firewalls

Los `firewalls` son, sin duda, una herramienta indispensable para proteger servidores de ataques. Su función principal es bloquear accesos no autorizados controlando el tráfico entrante y saliente. El `firewall` impide que los atacantes puedan acceder a los puertos o interfaces abiertas o en funcionamiento en un servidor, limitando así las posibles vías de entrada para realizar ataques. Además, permite establecer reglas específicas para permitir o denegar conexiones basadas en direcciones IP, protocolos o puertos, brindando un control granular sobre el tráfico y protegiendo los recursos del servidor de posibles amenazas externas. Se recomienda bloquear todo el acceso a nuestro servidor a excepción de los puertos que explícitamente sean necesarios (como los puertos de los servicios `SSH` y `HTTPS`).

2.11.4. Actualizaciones automáticas

Es fundamental mantener siempre el sistema actualizado para garantizar su seguridad y estabilidad. Las actualizaciones no solo incluyen nuevas funciones o mejoras de rendimiento, sino que también corrigen vulnerabilidades de seguridad que podrían ser explotadas por atacantes. Al mantener el sistema y las aplicaciones al día, se reducen significativamente los riesgos de sufrir intrusiones, ya que los parches de seguridad abordan amenazas conocidas y evitan que los atacantes aprovechen fallos en el software.

2.12. Metodología para la identificación, explotación y cuantificación de vulnerabilidades

Para realizar los ataques en esta tesis, a continuación se muestra la ruta a seguir.

1. **Identificación de puntos débiles**: La primera etapa de un ataque consiste en identificar los puntos débiles de un servidor. Para este propósito, se deben analizar los puertos expuestos utilizando la herramienta `Nmap`. A partir de esta información, se determina el tipo de ataque más adecuado. En este caso, los objetivos principales a atacar son el servicio de `SSH` y la interfaz de administración `Horizon`.

2. **Ataque a los puntos débiles**: Basándose en los puntos débiles identificados en la etapa previa y en las posibles formas de explotarlos, se selecciona la herramienta más adecuada para realizar un ataque. El objetivo de este ataque es el acceso al servidor o a la información confidencial, evaluando así las vulnerabilidades detectadas.

3. **Evaluación de resultados del ataque**: Una vez realizado el ataque, es fundamental analizar la información obtenida y determinar el alcance de la explotación realizada. Esto incluye identificar qué datos fueron comprometidos, los niveles de acceso alcanzados, y cualquier impacto potencial sobre la seguridad del servidor.

2.12.1. Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) es un marco internacional usado para evaluar y cuantificar la vulnerabilidad en sistemas informáticos. Fue desarrollado por el Foro de Infraestructura Abierta (FIRST) y es ampliamente adoptado en la industria para gestionar riesgos de seguridad. Existen diversas versiones de este marco, de la cual destaca la versión 3.0, ya que contiene mejoras significativas y permite cuantificar las vulnerabilidades de manera simple. El `CVSS` produce un puntaje numérico que va del 0.0 al 10.0, como se muestra en la tabla 2.1. Es posible calcular el puntaje de vulnerabilidad de un entorno utilizando la calculadora disponible en `https://www.first.org/cvss/calculator/3.0`. Esta herramienta

proporciona el puntaje a partir de un cuestionario que evalúa aspectos clave de seguridad relacionados con el ataque planteado a un sistema, basándose en las respuestas proporcionadas. Los aspectos considerados son los siguientes:

Grado de vulnerabilidad	Puntaje establecido por CVSS V3.0	Explicación
Crítica	9.0 - 10.0	La vulnerabilidad es extremadamente severa, con un alto impacto en los sistemas afectados y una facilidad de explotación muy alta. Puede comprometer completamente la seguridad del sistema o de la red.
Alta	7.0 - 8.9	La vulnerabilidad es significativa y puede ser explotada fácilmente en muchas condiciones. Impacta considerablemente la confidencialidad, integridad o disponibilidad.
Media	4.0 - 6.9	La vulnerabilidad puede ser explotada con impacto moderado en el sistema afectado. Podría causar daño, pero no es crítico para la operación o la seguridad general.
Baja	0.1 - 3.9	La vulnerabilidad tiene un impacto limitado o difícil de explotar. Requiere condiciones específicas para ser aprovechada y tiene consecuencias mínimas.
Nula	0	La vulnerabilidad no tiene ningún impacto en la confidencialidad, integridad o disponibilidad del sistema.

Tabla 2.1: Escala de puntuación CVSS V3.0 e interpretación.

- **Vector de ataque (AV)**: El vector de ataque describe el origen desde donde puede llevarse a cabo un ataque. Puede realizarse remotamente a través de Internet (red), desde la misma red local o segmento (adyacente), mediante acceso lógico o físico al sistema (local), o requiere acceso físico directo al dispositivo.
- **Complejidad del ataque (AC)**: La complejidad del ataque mide el nivel de dificultad para llevar a cabo el ataque. Puede ser baja si no requiere condiciones especiales o configuraciones inusuales, o alta si depende de factores complejos o condiciones específicas fuera del control del atacante.
- **Privilegios necesarios (PR)**: Los privilegios necesarios indican si el atacante requiere acceso previo al sistema. Puede ser ninguno si no se necesita acceso, bajo si se requiere una cuenta estándar, o alto si se necesitan privilegios administrativos.
- **Interacción del usuario (UI)**: La interacción del usuario evalúa si el ataque depende de que un usuario legítimo realice una acción que ponga en riesgo al sistema. Puede ser ninguna si el ataque ocurre sin participación del usuario, o requerida si es necesaria una acción, como abrir un archivo o hacer clic en un enlace.
- **Alcance (S)**: El alcance mide si el ataque afecta únicamente el sistema objetivo o también otros componentes. Se considera sin cambio si solo impacta el sistema atacado, o cambiado si afecta a otros sistemas relacionados.
- **Confidencialidad (C)**: La confidencialidad analiza si el ataque compromete datos sensibles. Puede ser ninguna si no hay impacto, baja si se accede a datos limitados o no crítico, y alta si se expone información sensible o crítica.
- **Integridad (I)**: La integridad mide si el ataque puede alterar información o componentes. Puede ser ninguna si no hay impacto, baja si los cambios son menores, y alta si los cambios son significativos o totales.

- **Disponibilidad (A)**: La disponibilidad evalúa si el ataque afecta el funcionamiento del sistema o servicio. Puede ser ninguna si no hay impacto, baja si reduce el rendimiento sin interrumpir el servicio, y alta si causa interrupciones severas o pérdida total de disponibilidad.

Cada uno de estos aspectos contribuye al cálculo del puntaje, proporcionando una evaluación precisa del nivel de riesgo asociado a la vulnerabilidad.

2.13. Seguridad de contraseñas: Buenas prácticas y métodos para evaluar la fortaleza de las contraseñas

La protección mediante contraseñas es una técnica de control de acceso ampliamente utilizada en diversos servicios para mantener nuestros datos seguros frente a posibles amenazas. Las contraseñas representan la primera línea de defensa, por lo que es fundamental que sean lo más robustas y seguras posible, asegurando así la protección de nuestra información [20].

En entornos donde se manejan múltiples servicios, puede parecer práctico reutilizar una misma contraseña o elegir una que sea corta y fácil de recordar. Sin embargo, esta práctica es altamente desaconsejable, ya que facilita a los ciberdelincuentes la posibilidad de descubrir las credenciales y acceder a la información. Este tipo de brechas puede tener graves consecuencias, incluyendo pérdidas económicas, problemas legales y afectaciones personales.

No existe una regla especifica que produzca contraseñas infalibles, en realidad todas las contraseñas pueden ser obtenidas, pero la idea principal es dificultar este proceso, en este caso, creando contraseñas largas y con patrones poco comunes, las cuales podrían tardar décadas en ser recuperadas. Algunas recomendaciones que se deben tener en cuenta para poder generar una contraseña segura se listan a continuación.

- **Longitud mayor a 8 caracteres**: Mientras más larga sea una contraseña, mucho más tiempo toma el poder recuperarla.
- **Uso de una combinación de letras, números y símbolos**: Las contraseñas que no incluyen palabras comunes ni tampoco patrones suelen ser muy seguras.
- **Uso de letras mayúsculas**: Esta recomendación aumenta considerablemente la seguridad, ya que se incremente el número de caracteres y por ende, aumenta el tiempo que tardaría en recuperar esta contraseña.
- **Uso de contraseñas distintas para cada cuenta**: Es altamente recomendado no tener una misma contraseña para diferentes servicios y cuentas, ya que los ciberdelincuentes solo tendrían que identificar una contraseña para tener acceso a toda nuestra información sensible de cada servicio.

Algunos ejemplos de contraseñas inseguras son:

- Abc123
- 1q2w3e
- Contraseña
- Qwerty123

Estas contraseñas son muy inseguras, tomaría desde un segundo hasta 5 días en poder descifrarlas [21].

Algunos ejemplos de contraseñas seguras son:

- Tbontbtitq31!
- BlueMoon*2024
- Sunny!Day/85

- StarFish!78

Estas contraseñas son tan seguras que tomaría de 400 a 4 billones de años poder descifrarlas utilizando una computadora básica de escritorio [21].

Capítulo 3

Estructura de OpenStack

OpenStack es fundamental en la creación de nubes privadas porque proporciona una plataforma de orquestación flexible y de código abierto, permitiendo a las organizaciones desplegar y gestionar infraestructuras en la nube sin depender de proveedores comerciales. Además, facilita el cumplimiento de ciertos estándares de privacidad internacionales y ofrece compatibilidad con tecnologías de nube pública y entornos basados en contenedores, lo que amplía su interoperabilidad y versatilidad.

3.1. Estructura de OpenStack

OpenStack requiere de dos nodos para poder funcionar, estos nodos tienen funciones específicas, por lo que su configuración suele ser diferente. Los nodos reciben el nombre de Controller y Compute.

3.1.1. Nodo Controller

Controller es el encargado de gestionar todos los módulos de OpenStack, por lo que tiene control total sobre la nube, es el nodo más importante, puesto que toda la nube depende de este nodo.

3.1.2. Nodo Compute

El nodo Compute se encarga de crear las máquinas virtuales, y de la gestión de los recursos físicos asociados a ellas. A diferencia del nodo Controller, Compute solo utiliza los módulos Nova, Neutron y Keystone. Estos módulos se comunican con los módulos del nodo Controller para así ofrecer una funcionalidad completa de la nube.

3.1.3. Endpoints

Como ya se mencionó en el capítulo anterior, OpenStack opera a través del uso de módulos, de los cuales se destacan a Nova, Neutron, Keystone, Horizon, Placement y Glance, estos realizan las funciones básicas de OpenStack y están presentes en cualquier proyecto desarrollado en esta plataforma. Cada módulo de OpenStack expone una API que puede ser usada por otros módulos para comunicarse entre ellos usando HTTP o HTTPS conocidos como enpoints, los cuales son configurados durante la instalación de OpenStack.

3.1.4. Implementaciones de seguridad en OpenStack

Usualmente, los endpoints usados para la comunicación entre módulos se encuentran de forma predeterminada en HTTP, por lo que la comunicación entre los módulos no está cifrada y esto representa un riesgo de seguridad. Para la implementación de la nube que se

va a realizar en este proyecto, los endpoints deberán ir configurados en HTTPS siguiendo la estructura que se muestra en la tabla 3.1.

Módulo	Descripción	Endpoint
Keystone	API de autenticación	https://controller:5000
Glance	API de imágenes	https://controller:9292
Nova	API de cómputo	https://controller:8774
Neutron	API de redes	https://controller:9696
Horizon	Dashboard web	https://controller:440
Placement	API de asignación de recursos	https://controller:8778

Tabla 3.1: Características de los endpoints.

Usualmente, el módulo Horizon trabaja en el puerto 443, sin embargo, en este proyecto se va a configurar en el puerto 440. Las herramientas etcd, MySQL, y RabbitMQ también deberán ser configuradas para trabajar con cifrado bajo SSL. En la tabla 3.2 se muestran las características que estas herramientas deberán tener.

Herramienta	Descripción	Puerto/Endpoint
etcd	Almacén de claves/valores distribuido utilizado para la coordinación y almacenamiento de configuración	https://controller:2379
MySQL	Base de datos relacional utilizada para almacenar información persistente de OpenStack	tcp://controller:3306
RabbitMQ	Middleware de mensajería usado para la comunicación entre servicios de OpenStack	tcp://controller:5672

Tabla 3.2: Características de las herramientas utilizadas en OpenStack.

3.2. Estructura de los servidores

3.2.1. Especificaciones de hardware y software de los servidores

Cada nodo de OpenStack debe estar montado en un servidor, por lo que es necesario contar con 2 máquinas virtuales que se encuentren en el mismo segmento de red. Las especificaciones de las máquinas virtuales se muestran en la tabla 3.3.

Especificaciones	Nodo Controller	Nodo Compute
Sistema Operativo	RHEL 9.4	RHEL 9.4
Núcleos	6	10
RAM	8192 MB	3072 MB
Almacenamiento	40GB	100GB
Tarjetas de red	1	1
IP	192.168.10.188	192.168.10.180

Tabla 3.3: Especificaciones de los nodos Controller y Compute.

Todos los servidores utilizados en este trabajo tienen instalado el servicio `SSH`. Por razones de practicidad, el número de puerto no será modificado en ningún servidor, manteniendo el puerto 22 en todos los casos.

Capítulo 4

Implementación de las medidas de seguridad

En este proyecto de tesis no se aborda el proceso de instalación de OpenStack, ya que este proceso está documentado en un trabajo de tesis previo (para más detalles consultar [14]). La plataforma utilizada corresponde a la versión Antelope, cuyo proceso de instalación es consistente entre diferentes distribuciones de Linux. En este capítulo solo se detallan los cambios que se deben hacer en la configuración de OpenStack, así como los cambios que se deben hacer en los servidores. Específicamente, este capítulo está dividido en tres secciones. En la primera sección se presenta la migración de los endpoints de HTTP a HTTPS. Posteriormente, se presenta la configuración del firewall tanto en el Controller como en el Compute. Finalmente, se presentan varios scripts para la protección de ataques por fuerza bruta.

4.1. Migración de endpoints de HTTP a HTTPS

Para migrar los endpoints a HTTPS es necesario generar certificados que serán utilizados por los módulos de OpenStack para cifrar la comunicación. Dado que los módulos de OpenStack no siguen un estándar unificado. A continuación, se presentan los métodos para la generación de certificados, detallando los módulos a los que se aplican y las configuraciones necesarias. También, se muestra el proceso para cifrar la comunicación de las herramientas etcd, MySQL y RabbitMQ. Estas configuraciones aplican para el nodo Controller. Sin embargo, algunas se repiten en el nodo Compute. A lo largo de la explicación, se indicará explícitamente cuándo la configuración aplica al nodo Compute.

Para comenzar, se instala el módulo mod_ssl en el nodo Controller. Posteriormente, se habilita el módulo mod_ssl en la configuración del servicio httpd, añadiendo la línea que se muestra en la consola 4.1 al archivo /etc/httpd/conf/httpd.conf.

Consola 4.1: Línea agregada en /etc/httpd/conf/httpd.conf.

```
...
LoadModule ssl_module modules/mod_ssl.so
...
```

El contenido completo del archivo /etc/httpd/conf/httpd.conf puede consultarse en el Apéndice A.1. Finalmente, se reinicia el servicio httpd.

Es importante mencionar que los certificados creados a lo largo de este capítulo serán autofirmados. En términos de cifrado y seguridad criptográfica, un certificado autofirmado ofrece el mismo nivel de seguridad que uno firmado por una Autoridad de Certificación (CA), ya que ambos utilizan los mismos estándares de cifrado, algoritmos y tamaños de clave, garantizando la protección del canal de comunicación. La principal diferencia entre ambos radica en el nivel de confianza. Un certificado autofirmado no cuenta con el respaldo de una

CA reconocida, lo que permite que pueda ser utilizado para suplantar identidades, ya que cualquier entidad podría generar uno similar sin restricciones. Por esta razón, los navegadores web no confían en este tipo de certificados y muestran advertencias de seguridad al intentar acceder a sitios que los utilizan. Se recomienda el uso de certificados autofirmados únicamente en entornos de prueba y desarrollo, mientras que en despliegues de producción es preferible utilizar certificados emitidos por una CA, aunque estos últimos suelen implicar un costo adicional.

4.1.1. Horizon

4.1.1.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso /etc/httpd/ssl/horizon. Para crear el certificado SSL/TLS de Horizon se debe seguir el procedimiento que se muestra en el Apéndice C.1 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.1.

Archivo	Sustituir por
ca.crt	cah.crt
ca.key	cah.key
modulo.key	horizon.key
modulo.csr	horizon.csr
modulo.crt	horizon.crt

Tabla 4.1: Nombres correspondientes a las llaves de Horizon.

4.1.1.2. Cambios en la configuración de OpenStack

A continuación se listan todos los archivos de configuración que deben ser modificados para que Horizon pueda funcionar usando HTTPS en el puerto 440 (puede ser en el puerto 443 si es que se decide trabajar sobre ese), el nombre del archivo de configuración del servicio Apache puede variar según la distribución de Linux que se esté empleando. Dentro del archivo /etc/httpd/conf.d/openstack-dashboard.conf se añade en el apartado <VirtualHost *:440>, las líneas que se muestran en la consola 4.2 .

Consola 4.2: Líneas agregadas en el archivo /etc/httpd/conf.d/openstack-dashboard.conf.

```
 ...
<VirtualHost *:440>
 ...
    SSLEngine On
    SSLCertificateFile /etc/httpd/ssl/horizon/horizon.crt
    SSLCertificateKeyFile /etc/httpd/ssl/horizon/horizon.key
    SSLCACertificateFile /etc/httpd/ssl/horizon/cah.crt
 ...
</VirtualHosty>
```

El contenido completo del archivo /etc/httpd/conf.d/openstack-dashboard.conf puede consultarse en el Apéndice A.2.

Dentro del archivo /etc/openstack-dashboard/local_settings se añaden las líneas que se muestran en la consola 4.3.

Consola 4.3: Líneas agregadas en el archivo /etc/openstack-dashboard/local_settings.

```
...
```

```
USE_SSL=True
OPENSTACK_SSL_NO_VERIFY=True
```

El contenido completo del archivo `/etc/openstack-dashboard/local_settings` puede consultarse en el Apéndice A.3. Finalmente, se reinicia el servicio `httpd`.

4.1.2. Keystone

4.1.2.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/keystone`. Para crear el certificado `SSL/TLS` de `Keystone` se debe seguir el procedimiento que se muestra en el Apéndice C.1, dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.2.

Archivo	Sustituir por
ca.crt	cak.crt
ca.key	cak.key
modulo.key	keystone.key
modulo.csr	keystone.csr
modulo.crt	keystone.crt

Tabla 4.2: Nombres correspondientes a las llaves de Keystone.

4.1.2.2. Cambios en la configuración de OpenStack

Dado que `Keystone` es el módulo de autenticación utilizado por todos los servicios, es necesario modificar los archivos de configuración de cada uno de ellos para que utilicen el nuevo `endpoint` de `Keystone` (`https://controller:5000`) en lugar de `http://controller:5000`. En la configuración del servicio `httpd` se debe indicar que `Keystone` debe usar los certificados que se generaron, para ello, se abre el archivo `/etc/httpd/conf.d/wsgi-keystone.conf` y en el apartado `<VirtualHost *:5000>` se añaden las líneas que se muestran en la consola 4.4.

Consola 4.4: Líneas agregadas en el archivo /etc/httpd/conf.d/wsgi-keystone.conf.

```
...
<VirtualHost *:5000>
    ...
    SSLEngine On
    SSLCertificateFile /etc/httpd/ssl/keystone/keystone.crt
    SSLCertificateKeyFile /etc/httpd/ssl/keystone/keystone.key
    SSLCACertificateFile /etc/httpd/ssl/keystone/cak.crt
    ...
</VirtualHost>
```

Se debe reiniciar el servicio `httpd`, con el fin de que los cambios sean aplicados. El contenido completo del archivo `/etc/httpd/conf.d/wsgi-keystone.conf` puede consultarse en el Apéndice A.4. Ahora, se debe modificar la configuración de todos los módulos que usan `Keystone`.

1. **Cambios en la configuración interna de Keystone:** No solo basta con cambiar la configuración de `Keystone` en el servicio `httpd`, es necesario reestructurar el módulo para que trabaje adecuadamente con el nuevo `endpoint`. A continuación, se listan los cambios que deben efectuarse.

- Se reescribe el comando de inicialización de `Keystone`. En la consola 4.5 se muestra el nuevo comando que indica el nuevo `endpoint` de `Keystone` usando `HTTPS`. Es importante hacer notar que el `password abc123` solo se usa como ejemplo, pero no es el usado en esta tesis.

Consola 4.5: Comando de inicialización de Keystone.

```
root@controller:#keystone-manage bootstrap --bootstrap-password abc123 \
--bootstrap-admin-url https://controller:5000/v3/ \
--bootstrap-internal-url https://controller:5000/v3/ \
--bootstrap-public-url https://controller:5000/v3/ \
--bootstrap-region-id RegionOne
```

- Se actualiza el `endpoint` en la configuración de los usuarios `admin` y `demo`, modificando el valor de la línea `export OS_AUTH_URL` en ambos entornos. El nuevo contenido de esta línea se muestra en la consola 4.6.

Consola 4.6: Línea modificada en los archivos admin_openrc y demo_openrc.

```
...
export OS\_AUTH\_URL=https://controller:5000/v3
...
```

- Dentro del archivo `/etc/keystone/keystone.conf`, se añade la etiqueta `[ssl]` junto con las líneas que se muestran en la consola 4.7.

Consola 4.7: Configuración de la etiqueta -ssl- del archivo /etc/keystone/keystone.conf.

```
...
[ssl]
enable = true
certfile = /etc/httpd/ssl/keystone/keystone.crt
keyfile = /etc/httpd/ssl/keystone/keystone.key
ca_certs = /etc/httpd/ssl/keystone/cak.crt
...
```

El contenido completo del archivo `/etc/keystone/keystone.conf` puede consultarse en el Apéndice A.5.

2. **Cambios en la configuración de Glance:** Para que `Glance` use el nuevo `endpoint` de `Keystone` solo basta con entrar al archivo de configuración `/etc/glance/glance-api.conf` y modificar los parámetros `www_authenticate_uri` y `auth_url` dentro de la etiqueta llamada `[keystone_authtoken]`, y posteriormente se modifica el parámetro `auth_url` dentro de la etiqueta `[oslo_limit]` como se muestra en la consola 4.8.

Consola 4.8: Configuración de las etiquetas -keystone_authtoken- y -oslo_limit- del archivo /etc/glance/glance-api.conf.

```
...
[keystone_authtoken]
www_authenticate_uri  = https://controller:5000
auth_url = https://controller:5000
...
[oslo_limit]
...
auth_url = https://controller:5000
...
```

Para cargar la nueva configuración de `Glance`, se reinicia el servicio `openstack-glance-api`.

El contenido completo del archivo `/etc/glance/glance-api.conf` puede consultarse en el Apéndice A.6.

3. **Cambios en la configuración de Placement:** Para que `Placement` use el nuevo `enpoint` de `Keystone` se debe modificar el archivo de configuración`/etc/placement/placement.conf`, cambiando el contenido del parámetro `auth_url` de la etiqueta `[keystone_authtoken]` por el que se muestra en la consola 4.9.

Consola 4.9: Configuración de la etiqueta -keystone_authtoken- del archivo /etc/placement/placement.conf.

```
...
[keystone_authtoken]
auth_url = https://controller:5000/v3
...
```

Para cargar la nueva configuración se reinicia el servicio de `httpd`.

El contenido completo del archivo `/etc/placement/placement.conf` puede consultarse en el Apéndice A.7.

4. **Cambios en la configuración de Nova:** Se debe acceder al archivo de configuración `/etc/nova/nova.conf`, y cambiar el contenido de los parámetros `www_authenticate_uri` y `auth_url` dentro de la etiqueta `[keystone_authtoken]`. Posteriormente, se modifica el parámetro `auth_url` dentro de la etiqueta `[placement]` como se muestra en la consola 4.10.

Consola 4.10: Configuración de las etiquetas -keystone_authtoken- y -placement- del archivo /etc/nova/nova.conf.

```
...
[keystone_authtoken]
www_authenticate_uri  = https://controller:5000
auth_url = https://controller:5000
...
[placement]
...
auth_url = https://controller:5000/v3
...
```

Finalmente, se reinician todos los servicios de `Nova`. El contenido completo del archivo `/etc/nova/nova.conf` del nodo `Controller` puede consultarse en el Apéndice A.8. Dado que `Nova` también está presente en el nodo `Compute`, es necesario editar su archivo de configuración y aplicar los mismos cambios realizados previamente.

El contenido completo del archivo `/etc/nova/nova.conf` del nodo `Compute` puede consultarse en el Apéndice A.9.

5. **Cambios en la configuración de Neutron:** Para configurar el nuevo `endpoint` de `Keystone` se debe acceder al archivo de configuración `/etc/neutron/neutron.conf` y modificar el contenido de los parámetros `www_authenticate_uri` y `auth_url` dentro de la etiqueta `[keystone_authtoken]`. Posteriormente, se modifica el parámetro `auth_url` dentro de la etiqueta `[nova]` como se muestra en la consola 4.11.

Consola 4.11: Configuración de las etiquetas -keystone_authtoken- y -nova- del archivo /etc/neutron/neutron.conf.

```
...
[keystone_authtoken]
www_authenticate_uri  = https://controller:5000
auth_url = https://controller:5000
...
[nova]
...
auth_url = https://controller:5000
...
```

Para que los cambios se apliquen se deben reiniciar todos los servicios de `Neutron` en el nodo `Controller`.

El contenido completo del archivo `/etc/neutron/neutron.conf` del nodo `Controller` puede consultarse en el Apéndice A.10. Posteriormente, dentro de la etiqueta `[keystone_authtoken]` del archivo `/etc/neutron/neutron.conf` del nodo `Compute`, se debe modificar de la misma manera como se hizo con el nodo `Controller`. Por otro lado, en el nodo `Compute` se accede al archivo de configuración `/etc/nova/nova.conf`, donde se debe modificar el contenido del parámetro `auth_url` dentro de la etiqueta `[neutron]` como se muestra en la consola 4.12.

Consola 4.12: Configuración de la etiqueta -neutron- del archivo /etc/nova/nova.conf en Compute.

```
...
[nova]
...
auth_url = https://controller:5000
...
```

Finalmente, se deben reiniciar los servicios de `Neutron` y `Nova` en el nodo `Compute`. El contenido completo del archivo `/etc/nova/nova.conf` del nodo `Compute` puede consultarse en el Apéndice A.9.

6. **Cambios en la configuración de Horizon:** Para cambiar el `endpoint` de `Keystone` se accede al archivo de configuración `/etc/openstack-dashboard/local_settings` y se modifica el contenido del parámetro `OPENSTACK_KEYSTONE_URL` como se muestra en la consola 4.32.

Consola 4.13: Cambio del parámetro OPENSTACK_KEYSTONE_URL dentro del archivo de configuración /etc/openstack-dashboard/local_settings.

```
...
OPENSTACK_KEYSTONE_URL = "https://%s:5000/identity/v3" %
...
```

Se debe reiniciar el servicio de `httpd` para aplicar este cambio.

4.1.3. Glance

4.1.3.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/glance`. Para crear el certificado `SSL/TLS` de `Glance` se debe seguir el procedimiento que se muestra en el Apéndice C.2 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.3.

Archivo	Sustituir por
modulo.key	glance.key
modulo.csr	glance.csr
modulo.crt	glance.crt
modulo.pem	glance.pem

Tabla 4.3: Nombres correspondientes a las llaves de Glance.

4.1.3.2. Cambios en la configuración de OpenStack

Para modificar la configuración interna de Glance, se accede al archivo /etc/glance/glance-api.conf (ver Apéndice A.6), y en la etiqueta [DEFAULT] se añaden las variables bind_host y public_endpoint como se muestra en la consola 4.14.

Consola 4.14: Variables agregadas en la etiqueta -DEFAULT- en /etc/glance/glance-api.conf.

```
...
bind_host=127.0.0.1
public_endpoint=https://controller:9292
...
```

Se debe reiniciar el servicio openstack-glance-api para que los cambios se apliquen.

Posteriormente, se deben eliminar todos los endpoints de Glance que fueron generados durante la instalación de OpenStack que inicialmente se encontraban con HTTP. Para ello se pueden emplear los comandos mostrados en la consola 4.15.

Consola 4.15: Eliminación de enpoints HTTP de Glance.

```
root@controller:# openstack endpoint list
root@controller:# openstack endpoint delete <ID>
```

Luego se deben generar los nuevos endpoints que utilizan HTTPS con los comandos que se muestran en la consola 4.16.

Consola 4.16: Generación de los nuevos endpoints de Glance.

```
root@controller:# openstack endpoint create --region RegionOne \
  image public https://controller:9292
root@controller:# openstack endpoint create --region RegionOne \
  image internal https://controller:9292
root@controller:# openstack endpoint create --region RegionOne \
  image admin https://controller:9292
```

Dado que no existe un acceso directo a la configuración de Glance en el servicio httpd que nos permita modificar la configuración del puerto 9292 para que trabaje en HTTPS en lugar de HTTP, se debe usar un servidor proxy inverso. Para ello se debe usar el programa Nginx, el cual está disponible en la mayoría de las distribuciones de Linux. Para añadirlo al nodo se deben instalar los paquetes nginx y nginx-mod-stream.

Una vez instalado, se debe configurar un bloque de servidor (server) dentro de la sección http de la configuración de Nginx como se muestra en la consola 4.17.

Consola 4.17: Bloque de servidor de Nginx en el archivo /etc/nginx/nginx.conf.

```
stream {
    upstream glance-api {
        server 127.0.0.1:9292;
    }
    server {
        listen 10.0.2.7:9292 ssl;
        proxy_pass glance-api;
    }
    ssl_certificate "/etc/httpd/ssl/glance/glance.pem";
    ssl_certificate_key "/etc/httpd/ssl/glance/glance.key";
}
```

Este bloque se encarga de redirigir todas las solicitudes que se hacen al nuevo endpoint https://controller:9292 hacia http://controller:9292, utilizando los certificados generados para cifrar la información y garantizar la seguridad de las conexiones hacia Glance. El contenido completo del archivo /etc/nginx/nginx.conf puede consultarse en el Apéndice A.10. Para aplicar los cambios hechos en la configuración de Nginx, se debe reiniciar el servicio nginx.

Glance es utilizado únicamente por el módulo Nova, por lo que es necesario actualizar la configuración de Nova para incluir el nuevo endpoint de Glance. Este proceso es sencillo, solo

se accede al archivo de configuración de Nova y, dentro de la sección [glance], se modifica el valor de la variable api_servers según lo indicado en la consola 4.18. Este procedimiento debe realizarse tanto en el nodo Controller como en el nodo Compute.

Consola 4.18: Etiqueta -glance- de la configuración de Nova en ambos nodos.

```
[glance]
...
api_servers = https://controller:9292
...
```

Se deben reiniciar los servicios de Nova en el nodo Controller para aplicar los cambios.

De igual manera, se debe reiniciar el servicio de Nova en el nodo Compute para aplicar los cambios.

4.1.4. Placement

4.1.4.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso /etc/httpd/ssl/placement. Para crear el certificado SSL/TLS de Placement se debe seguir el procedimiento que se muestra en el Apéndice C.1 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.4.

Archivo	Sustituir por
ca.crt	cap.crt
ca.key	cap.key
modulo.key	placement.key
modulo.csr	placement.csr
modulo.crt	placement.crt

Tabla 4.4: Nombres correspondientes a las llaves de Placement.

4.1.4.2. Cambios en la configuración de OpenStack

Para la configuración interna de Placement, se accede al archivo de configuración /etc/placement/placement.conf (ver Apéndice A.7) y se añade la variable enable_ssl_api en la etiqueta [api] como se muestra en la consola 4.19.

Consola 4.19: Etiqueta -api- del archivo de configuración /etc/placement/placement.conf.

```
[api]
...
enable_ssl_api= true
...
```

Posteriormente, se accede al archivo /etc/httpd/conf.d/00-placement-api.conf de httpd, en el apartado <VirtualHost *:8778>, se añaden los cuatro parámetros que se muestran en la consola 4.20.

Consola 4.20: Líneas agregadas en el archivo /etc/httpd/conf.d/00-placement-api.conf.

```
...
<VirtualHost *:8778>
    ...
    SSLEngine On
    SSLCertificateFile /etc/httpd/ssl/placement/placement.crt
    SSLCertificateKeyFile /etc/httpd/ssl/placement/placement.key
    SSLCACertificateFile /etc/httpd/ssl/placement/cap.crt

    ...
</VirtualHost>
```

El contenido completo del archivo `/etc/httpd/conf.d/00-placement-api.conf` del nodo `Compute` puede consultarse en el Apéndice A.12. Se debe reiniciar el servicio de `httpd` para aplicar este cambio.

También, se debe eliminar todos los `endpoints` de `Placement` que fueron generados durante la instalación de `OpenStack` que inicialmente se encontraban en `HTTP`. Para ello se pueden emplear los comandos mostrados en la consola 4.15.

Luego, se deben generar los nuevos `endpoints` que utilizan `HTTPS` con los comandos que se muestran en la consola 4.21.

Consola 4.21: Generación de los nuevos endpoints de Placement.

```
root@controller:# openstack endpoint create --region RegionOne \
  placement public https://controller:8778
root@controller:# openstack endpoint create --region RegionOne \
  placement internal https://controller:8778
root@controller:# openstack endpoint create --region RegionOne \
  placement admin https://controller:8778
```

4.1.5. Nova

4.1.5.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/nova`. Para crear el certificado `SSL/TLS` de `Nova` se debe seguir el procedimiento que se muestra en el Apéndice C.1, dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.5.

Archivo	Sustituir por
ca.crt	cano.crt
ca.key	cano.key
modulo.key	nova.key
modulo.csr	nova.csr
modulo.crt	nova.crt

Tabla 4.5: Nombres correspondientes a las llaves de Nova.

4.1.5.2. Cambios en la configuración de OpenStack

Para la configuración de `Nova`, se debe acceder al archivo de configuración `/etc/nova/nova.conf` del nodo `Controller` (ver Apéndice A.8) y se añaden los parámetros `ssl_only` y `enabled_ssl_apis` en la etiqueta `[DEFAULT]`, de igual manera se añaden los parámetros `ssl_cert_file` y `ssl_key_file` en la etiqueta `[wsgi]` como se muestra en la consola 4.22.

Consola 4.22: Etiquetas -DEFAULT- y -wsgi- del archvio de configuración /etc/nova/nova.conf.

```
[DEFAULT]
...
ssl_only=true
enabled_ssl_apis = osapi_compute
[wsgi]
ssl_cert_file = /etc/httpd/ssl/nova/nova.crt
ssl_key_file = /etc/httpd/ssl/nova/nova.key
...
```

Posteriormente, se debe eliminar todos los `endpoints` de `Nova` que fueron generados durante la instalación de `OpenStack` que inicialmente se encontraban con `HTTP`. Para ello se pueden emplear los comandos mostrados en la consola 4.15.

Luego, se deben generar los nuevos `endpoints` que utilizan `HTTPS` con los comandos que se muestran en la consola 4.23.

Consola 4.23: Generación de los nuevos endpoints de Nova.

```
root@controller:# openstack endpoint create --region RegionOne \
  compute public https://controller:8774/v2.1
root@controller:# openstack endpoint create --region RegionOne \
  compute internal https://controller:8774/v2.1
root@controller:# openstack endpoint create --region RegionOne \
  compute admin https://controller:8774/v2.1
```

Finalmente, se deben reiniciar todos los servicios de `Nova` en el nodo `Controller` para aplicar los cambios. No es necesario aplicar ningún cambio en la configuración del nodo `Compute`.

4.1.6. Neutron

4.1.6.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/neutron`. Para crear el certificado `SSL/TLS` de `Neutron` se debe seguir el procedimiento que se muestra en el Apéndice C.1 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.6.

Archivo	Sustituir por
ca.crt	cane.crt
ca.key	cane.key
modulo.key	neutron.key
modulo.csr	neutron.csr
modulo.crt	neutron.crt

Tabla 4.6: Nombres correspondientes a las llaves de Neutron.

4.1.6.2. Cambios en la configuración de OpenStack

Para la configuración de `Neutron`, se debe acceder al archivo de configuración `/etc/neutron/neutron.conf` del nodo `Controller` (ver Apéndice A.9) y se añaden los parámetros `bind_host`, `bind_port`, `use_ssl`, `ssl_cert_file` y `ssl_key_file` en la etiqueta `[DEFAULT]` como se muestra en la consola 4.24.

Consola 4.24: Etiqueta -DEFAULT- del archvio de configuración /etc/neutron/neutron.conf.

```
[DEFAULT]
...
bind_host = 0.0.0.0
bind_port = 9696
use_ssl = True
ssl_cert_file=/etc/httpd/ssl/neutron/neutron.crt
ssl_key_file=/etc/httpd/ssl/neutron/neutron.key
...
```

Se debe eliminar todos los `endpoints` de `Neutron` que fueron generados durante la instalación de `OpenStack` que inicialmente se encontraban con `HTTP`. Para ello se pueden emplear los comandos mostrados en la consola 4.15. Posteriormente, se deben generar los nuevos `endpoints` que utilizan `HTTPS` con los comandos que se muestran en la consola 4.25.

Consola 4.25: Generación de los nuevos endpoints de Nova.

```
root@controller:# openstack endpoint create --region RegionOne \
  network public https://controller:9696
root@controller:# openstack endpoint create --region RegionOne \
  network internal https://controller:9696
root@controller:# openstack endpoint create --region RegionOne \
  network admin https://controller:9696
```

Finalmente, se deben reiniciar todos los servicios de `Neutron` en el nodo `Controller` para aplicar los cambios. No es necesario aplicar ningún cambio en la configuración del nodo `Compute`.

4.1.7. etcd

4.1.7.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/etcd`. Para crear el certificado `SSL/TLS` de `etcd` se debe seguir el procedimiento que se muestra en el Apéndice C.3 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.7.

Archivo
cae.key
cae.crt
etcd.key
etcd.cnf
etcd.csr
etcd.crt

Tabla 4.7: Archivos generados durante el proceso de configuración de etcd.

4.1.7.2. Cambios en la configuración de OpenStack

Para esta herramienta solo es necesario modificar su configuración interna para añadir los certificados, para ello, se debe acceder al archivo de configuración `/etc/etcd/etcd.conf` y eliminar las líneas existentes para luego agregar las líneas que se muestran en la consola 4.26. En esencia, se están modificando las variables que contenían los `endpoints` en `HTTP` por `HTTPS`, a demás se le está indicando a `etcd` que debe usar los certificados generados previamente.

Consola 4.26: Variables añadidas en el archivo /etc/etcd/etcd.conf.

```
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_PEER_URLS="https://10.0.2.7:2380"
ETCD_LISTEN_CLIENT_URLS="https://10.0.2.7:2379"
ETCD_NAME="controller"
#[Clustering]
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://10.0.2.7:2380"
ETCD_ADVERTISE_CLIENT_URLS="https://10.0.2.7:2379"
ETCD_INITIAL_CLUSTER="controller=https://10.0.2.7:2380"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER_STATE="new"
#[Security]
ETCD_CERT_FILE="/etc/httpd/ssl/etcd/etcd.crt"
ETCD_KEY_FILE="/etc/httpd/ssl/etcd/etcd.key"
ETCD_TRUSTED_CA_FILE="/etc/httpd/ssl/etcd/cae.crt"
ETCD_PEER_CERT_FILE="/etc/httpd/ssl/etcd/etcd.crt"
ETCD_PEER_KEY_FILE="/etc/httpd/ssl/etcd/etcd.key"
ETCD_PEER_TRUSTED_CA_FILE="/etc/httpd/ssl/etcd/cae.crt"
```

Finalmente, se debe reiniciar el servicio `etcd` para aplicar los cambios realizados.

4.1.8. MySQL

4.1.8.1. Creación de certificados

Primero se crea el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/sql`. Para crear el certificado `SSL/TLS` de `MySQL` se debe seguir el procedimiento que se muestra en el apéndice C.2 dentro del directorio que se creó previamente. Se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.8.

Archivo	Sustituir por
modulo.key	sql.key
modulo.csr	sql.csr
modulo.crt	sql.crt
modulo.pem	sql.pem

Tabla 4.8: Nombres correspondientes a las llaves de MySQL.

4.1.8.2. Cambios en la configuración de OpenStack

Primero se debe cambiar la configuración interna de `MySQL`, para ello se accede al archivo `/etc/my.cnf` y se añaden los tres parámetros dentro de la etiqueta `[mysqld]` como se muestra en la consola 4.27.

Consola 4.27: Parámetros añadidos en el archivo /etc/my.cnf.

```
[mysqld]
ssl-cert = /etc/httpd/ssl/sql/sql.crt
ssl-key = /etc/httpd/ssl/sql/sql.key
ssl-ca = /etc/httpd/ssl/sql/cam.crt
```

El contenido completo del archivo `/etc/my.cnf` puede consultarse en el Apéndice A.12.

Para aplicar los cambios realizados, se debe reiniciar el servicio `mariadb`. Dado que la mayoría de los módulos de `OpenStack` dependen de la base de datos, es necesario realizar ajustes en la configuración de múltiples módulos para garantizar su correcto funcionamiento. En todas las configuraciones de los módulos se debe modificar el parámetro `connection` que

se encuentra en la etiqueta de configuración de la base de datos. El parámetro se muestra en la consola 4.28.

Consola 4.28: Nuevo valor del parámetro connection.

```
[database]
connection =mysql+pymysql://user:passwd@controller/user?ssl_ca=/etc/httpd
    /ssl/sql/sql.pem&ssl_cert=/etc/httpd/ssl/sql/sql.pem&ssl_key=/etc/
    httpd/ssl/sql/sql.key
```

1. **Cambios en la configuración de Keystone:** Se accede al archivo de configuración `/etc/keystone/keystone.conf` (ver Apéndice A.5) y se debe modificar el valor del parámetro `connection` de la etiqueta `[database]`, esto se muestra en la consola 4.28, donde se debe sustituir los valores de `user` y `passwd` por lo que se muestra en la tabla 4.9. Es importante mencionar que la contraseña `abc123`, solo es un ejemplo.

Valor 'user'	Valor 'passwd'
keystone	abc123

Tabla 4.9: Tabla con valores de usuario y contraseña de Keystone.

Posteriormente, se reinicia el servicio `httpd`. Finalmente, se debe sincronizar la base de datos de `Keystone`.

Consola 4.29: Sincronización de la base de datos de Keystone.

```
root@controller:# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

2. **Cambios en la configuración de Glance:** Primero se debe acceder al archivo de configuración `/etc/glance/glance-api.conf` (ver Apéndice A.6) y se debe modificar el valor del parámetro `connection` de la etiqueta `[database]`, esto se muestra en la consola 4.28, donde se deben sustituir los valores de `user` y `passwd` por lo que se muestra en la tabla 4.10.

Valor 'user'	Valor 'passwd'
glance	abc123

Tabla 4.10: Tabla con valores de usuario y contraseña para Glance.

Posteriormente, se debe reiniciar el servicio `openstack-glance-api` y también se debe sincronizar la base de datos de `Glance`.

Consola 4.30: Sincronización de la base de datos de Glance.

```
root@controller:# su -s /bin/sh -c "glance-manage db_sync" glance
```

3. **Cambios en la configuración de Placement:** Primero se debe acceder al archivo de configuración `/etc/placement/placement.conf` (ver Apéndice A.7) y se debe modificar el valor del parámetro `connection` de la etiqueta `[placement_database]`, esto se muestra en la consola 4.28, donde se deben sustituir los valores de `user` y `passwd` por lo que se muestra en la tabla 4.11.

   Posteriormente, se debe reiniciar el servicio `httpd` y también se debe sincronizar la base de datos de `Placement`.

Consola 4.31: Sincronización de la base de datos de Placement.

```
root@controller:# su -s /bin/sh -c "placement-manage db sync" placement
```

Valor 'user'	Valor 'passwd'
placement	abc123

Tabla 4.11: Tabla con valores de usuario y contraseña para Placement.

4. **Cambios en la configuración de Nova:** Se accede al archivo de configuración de Nova en el nodo `Controller` (`/etc/nova/nova.conf`, ver Apéndice A.8) y se modifica el valor del parámetro `connection` de la etiqueta `[api_database]` por lo que se muestra en la consola 4.32.

Consola 4.32: Nuevo valor del parámetro connection de la etiqueta -api_database- del archivo /etc/nova/nova.conf.

```
[api_database]
connection =mysql+pymysql://nova:abc123@controller/nova_api?ssl_ca=/
    etc/httpd/ssl/sql/sql.pem&ssl_cert=/etc/httpd/ssl/sql/sql.pem&
    ssl_key=/etc/httpd/ssl/sql/sql.key
```

En ese mismo archivo de configuración, el valor de la variable `connection` de la etiqueta `[database]` debe ser cambiado por lo que se muestra en la consola 4.28, considerando los valores de `user` y `passwd` en la tabla 4.12.

Valor 'user'	Valor 'passwd'
nova	abc123

Tabla 4.12: Tabla con valores de usuario y contraseña para Nova.

Posteriormente, se debe sincronizar cada una de las bases de datos de `Nova` y reiniciar sus servicios.

Consola 4.33: Sincronización de la base de datos de Nova.

```
root@controller:# su -s /bin/sh -c "nova-manage api_db sync" nova
root@controller:# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
root@controller:# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1
    --verbose" nova
root@controller:# su -s /bin/sh -c "nova-manage db sync" nova
```

5. **Cambios en la configuración de Neutron:** Primero se debe acceder al archivo de configuración `/etc/neutron/neutron.conf` (ver Apéndice A.9) y se debe modificar el valor del parámetro `connection` de la etiqueta `[database]` por el mostrado en la consola 4.28, sustituyendo los valores de `user` y `passwd` por lo que se muestra en la tabla 4.13.

Valor 'user'	Valor 'passwd'
neutron	abc123

Tabla 4.13: Tabla con valores de usuario y contraseña para Neutron.

Posteriormente, se debe sincronizar la base de datos de `Neutron` y reiniciar sus servicios.

Consola 4.34: Sincronización de la base de datos de Neutron.

```
root@controller:# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/
    neutron.conf \
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

4.1.9. RabbitMQ

4.1.9.1. Creación de certificados

Primero se debe crear el directorio donde se van a almacenar los certificados, en este caso `/etc/httpd/ssl/rabbit`. Para crear el certificado `SSL/TLS` de `RabbitMQ` se debe seguir el procedimiento que se muestra en el apéndice C.1 dentro del directorio que se creó previamente. Además, se deben sustituir los nombres de los archivos por los mostrados en la tabla 4.14.

Archivo	Sustituir por
ca.crt	cara.crt
ca.key	cara.key
modulo.key	rabbit.key
modulo.csr	rabbit.csr
modulo.crt	rabbit.crt

Tabla 4.14: Nombres correspondientes a las llaves de RabbitMQ.

4.1.9.2. Cambios en la configuración de OpenStack

Lo único que se debe modificar es la configuración de `RabbitMQ`, para ello se accede al archivo `/etc/rabbitmq/rabbitmq.conf` y se añaden las variables que se muestran en la consola 4.35.

Consola 4.35: Parámetros agregados en el archivo /etc/rabbitmq/rabbitmq.conf.

```
listeners.ssl.default = 5671
ssl_options.cacertfile =/etc/httpd/ssl/rabbit/cara.crt
ssl_options.certfile =/etc/httpd/ssl/rabbit/rabbit.crt
ssl_options.keyfile =/etc/httpd/ssl/rabbit/rabbit.key
ssl_options.verify = verify_none
ssl_options.fail_if_no_peer_cert = false
```

Para aplicar los cambios realizados, se debe reiniciar el servicio `rabbitmq`.

4.1.10. Confianza del sistema en los certificados y verificación de los cambios realizados

Dado que se está utilizando certificados autofirmados, el sistema Linux y los servicios de `OpenStack` no confiarán en ellos de manera predeterminada, y esto puede provocar múltiples errores, como fallos de conexión o advertencias de seguridad.

Para solucionar esto, todos los certificados autofirmados deben añadirse al directorio `/etc/pki/ca-trust/source/anchors/`. Luego, es necesario ejecutar el siguiente comando para actualizar la lista de certificados de confianza del sistema.

Consola 4.36: Carga de los certificados en el sistema.

```
root@controller:# update-ca-trust extract
```

Este proceso carga los certificados en el sistema, permitiendo que tanto Linux y `OpenStack` confíen en ellos. Este paso es crucial para garantizar una comunicación segura y evitar problemas con las conexiones `HTTPS`. Este procedimiento debe hacerse tanto en el nodo `Controller` como en el nodo `Compute`, por lo que se requiere transferir los certificados creados desde el nodo `Controller` al `Compute`.

Para observar los cambios realizados se puede ingresar como administrador en la página de Horizon, y en la pestaña Acceso a la API se muestran todos los endpoints en HTTPS (ver figura 4.1).

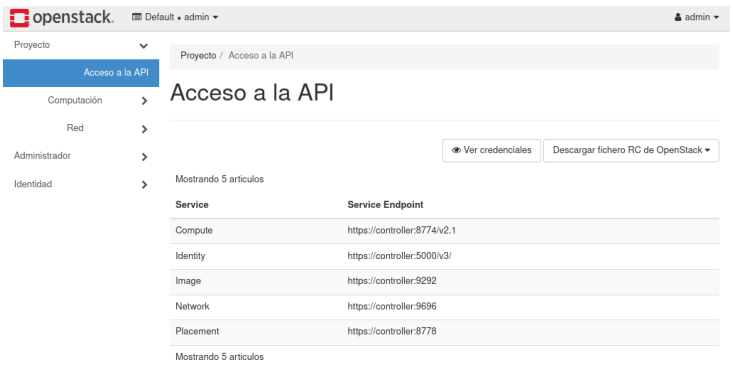

Figura 4.1: Endpoints de OpenStack vistos desde Horizon.

De igual manera se puede usar el comando mostrado en la consola 4.37.

Consola 4.37: Endpoints reconocidos por OpenStack.

```
root@controller:# openstack --insecure endpoint list
```

El comando anterior muestra todos los endpoints de OpenStack. La salida de este comando se muestra en la figura 4.2.

```
[root@controller ~]# openstack --insecure endpoint list
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------+
| ID                               | Region    | Service Name | Service Type | Enabled | Interface | URL                         |
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------+
| 001522c90afb42c8be8f3abb01fb0696 | RegionOne | neutron      | network      | True    | admin     | https://controller:9696     |
| 05308e64028f4961baf937dd32d56e9a | RegionOne | glance       | image        | True    | admin     | https://controller:9292     |
| 0fd7fd51a4564ab4a55d970199e92d4b | RegionOne | nova         | compute      | True    | internal  | https://controller:8774/v2.1|
| 275c649806284ebcb214926203f803b3 | RegionOne | keystone     | identity     | True    | admin     | https://controller:5000/v3/ |
| 4db02db0cbbb4a2792ca4ec3fb5db6e5 | RegionOne | keystone     | identity     | True    | public    | https://controller:5000/v3/ |
| 4e378680cafe4670b289eaeaab40ad29 | RegionOne | glance       | image        | True    | public    | https://controller:9292     |
| 4e60f3b263f7497c838296a23bcc4693 | RegionOne | placement    | placement    | True    | internal  | https://controller:8778     |
| 649f28656a8d4445bda02cf9a43be09e | RegionOne | neutron      | network      | True    | internal  | https://controller:9696     |
| 73a4f5a18c0640058453f95ce743d6c1 | RegionOne | glance       | image        | True    | internal  | https://controller:9292     |
| 9e2393c02d3842239891ce3e6c6262dc | RegionOne | keystone     | identity     | True    | internal  | https://controller:5000/v3/ |
| a4e0a5b68d5740e3a6fb4227d56e3440 | RegionOne | nova         | compute      | True    | admin     | https://controller:8774/v2.1|
| b70f036eab92494591970bf085d741ec | RegionOne | nova         | compute      | True    | admin     | https://controller:8774/v2.1|
| cae0135265b94cfdb35032914c197eb0 | RegionOne | neutron      | network      | True    | public    | https://controller:9696     |
| e9462b05db7d41b8914fb04830405ddc | RegionOne | nova         | compute      | True    | public    | https://controller:8774/v2.1|
| f5a5dd7874614741a67ebbc81405f58c | RegionOne | placement    | placement    | True    | admin     | https://controller:8778     |
| f94835ca5640422898c657d740188db6 | RegionOne | placement    | placement    | True    | public    | https://controller:8778     |
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------+
```

Figura 4.2: Salida del comando openstack –insecure endpoint list.

Se puede observar que todos los endpoints son reconocidos de manera adecuada por OpenStack.

4.2. Configuración del Firewall en los nodos Controller y Compute

4.2.1. Firewall para el nodo Controller

La primera etapa consiste en activar el firewall en el nodo Controller. RHEL 9.4 cuenta con la herramienta firewalld de manera predeterminada. Esta herramienta fue diseñada para sustituir a IPTABLES de las versiones anteriores de RHEL. En el nodo Controller se debe crear una zona que contenga las características mostradas en la tabla 4.15.

Se genera la zona llamada openstack dentro del firewall, ver consola 4.38.

Puerto	Dirección IP	Acción
440	Todas	Permitido (Servicio HTTP sobre HTTPS)
22	Todas	Bloqueado (SSH)
3306	Solo 192.168.10.180	Permitido (MySQL)
5000	Solo 192.168.10.180	Permitido (API de Keystone)
9696	Solo 192.168.10.180	Permitido (API de Neutron en OpenStack)
8774	Solo 192.168.10.180	Permitido (API de Nova en OpenStack)
80	Todas	Bloqueado (HTTP)
8775	Solo 192.168.10.180	Permitido (API de Nova en OpenStack)
6080	Solo 192.168.10.180	Permitido (Web Management de OpenStack)
11211	Solo 192.168.10.180	Permitido (Memcached)
5671	Solo 192.168.10.180	Permitido (RabbitMQ)
5672	Solo 192.168.10.180	Permitido (RabbitMQ)

Tabla 4.15: Puertos y reglas del firewall para el nodo Controller.

Consola 4.38: Creación de la zona -openstack- en el firewall del nodo Controller.

```
root@controller:# firewall-cmd --permanent --new-zone=openstack
root@controller:# firewall-cmd --reload
```

Posteriormente, se fija la zona `openstack` como zona predeterminada de nuestro sistema, ver consola 4.39.

Consola 4.39: Cambio de zona predeterminada del firewall.

```
root@controller:# firewall-cmd --set-default-zone=openstack
root@controller:# firewall-cmd --reload
```

Se añaden las reglas que se establecieron en la tabla 4.15 en la zona `openstack`, ver consola 4.40.

Consola 4.40: Proceso de añadir reglas a la zona -openstack- del firewall en el nodo Controller.

```
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" port port="80" protocol="tcp" reject'
root@controller:# firewall-cmd --permanent --zone=openstack --add-port=440/tcp
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" port port="22" protocol="tcp" reject'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="3306" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="5000" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="9696" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="8774" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="8775" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="6080" protocol="tcp" accept'
```

```
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="11211" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="5671" protocol="tcp" accept'
root@controller:# firewall-cmd --permanent --zone=openstack --add-rich-rule='rule family="
    ipv4" source address="192.168.10.180" port port="5672" protocol="tcp" accept'
root@controller:# firewall-cmd --reload
```

Usando el comando mostrado en la consola 4.41, se verifica que la zona exista y que tenga las reglas que se le añadieron dentro del `firewall`, como se muestra en la figura 4.3.

Consola 4.41: Comprobación de la existencia y funcionamiento de la zona -openstack- en el firewall.

```
root@controller:# sudo firewall-cmd --zone=openstack --list-all
```

```
openstack (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services:
  ports: 440/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" port port="22" protocol="tcp" reject
        rule family="ipv4" source address="192.168.10.180" port port="8775" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="6080" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="11211" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="5672" protocol="tcp" accept
        rule family="ipv4" port port="80" protocol="tcp" reject
        rule family="ipv4" source address="192.168.10.180" port port="5671" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="8774" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="5000" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="9696" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.180" port port="3306" protocol="tcp" accept
```

Figura 4.3: Salida del comando sudo firewall-cmd –zone=openstack –list-all en el nodo Controller.

Se puede consular el archivo `XML` generado por `firewalld` para esta zona en el Apéndice H.1.

4.2.2. Firewall para el nodo Compute

Para el nodo `Compute`, no es necesario tener el puerto 440 ni 3306, puesto que no hay un servidor web o servidor de base de datos en este nodo. En la tabla 4.16 se muestran los detalles del `firewall`.

En este nodo se realiza el mismo proceso de creación y establecimiento de la zona `openstack` como zona predeterminada (consolas 4.38 y 4.39). Posteriormente, se añaden todas las reglas establecidas en la tabla 4.16, ver consola 4.42.

Puerto	Dirección IP	Acción
22	Todas	Bloqueado (SSH)
5000	Solo 192.168.10.188	Permitido (API de Keystone)
9696	Solo 192.168.10.188	Permitido (API de Neutron en OpenStack)
8774	Solo 192.168.10.188	Permitido (API de Nova en OpenStack)
8775	Solo 192.168.10.188	Permitido (API de Nova en OpenStack)
6080	Solo 192.168.10.188	Permitido (Web Management de OpenStack)
11211	Solo 192.168.10.188	Permitido (Memcached)
5671	Solo 192.168.10.188	Permitido (RabbitMQ)
5672	Solo 192.168.10.188	Permitido (RabbitMQ)

Tabla 4.16: Puertos y reglas del firewall para el nodo Compute.

Consola 4.42: Proceso de añadir reglas a la zona -openstack- del firewall en el nodo Compute.

```
root@controller:# firewall-cmd --permanent --zone=compute --add-rich-rule='rule family="
    ipv4" port port="22" protocol="tcp" reject'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="5000" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="9696" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="8774" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="8775" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="6080" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="11211" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="5671" protocol="tcp" accept'
root@controller:#sudo firewall-cmd --permanent --zone=compute --add-rich-rule='rule family
    ="ipv4" source address="192.168.10.188" port port="5672" protocol="tcp" accept'
```

Mediante el comando de la consola 4.41, se verifica la existencia y correcto funcionamiento de la nueva zona del `firewall` en el nodo `Compute` (figura 4.4).

```
openstack (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="192.168.10.188" port port="5672" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="5000" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="5671" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="8775" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="6080" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="9696" protocol="tcp" accept
        rule family="ipv4" source address="192.168.10.188" port port="8774" protocol="tcp" accept
        rule family="ipv4" port port="22" protocol="tcp" reject
        rule family="ipv4" source address="192.168.10.188" port port="11211" protocol="tcp" accept
```

Figura 4.4: Salida del comando sudo firewall-cmd –zone=openstack –list-all en el nodo Compute.

Se puede consular el archivo XML generado por firewalld para esta zona en el Apéndice H.2.

4.3. Script para la detección de ataques de fuerza bruta en el nodo Controller

Dentro del nodo Controller, es crucial mantener una vigilancia constante sobre los mensajes de logs generados por la página de Horizon y por SSH (si este se encuentra habilitado). La mayoría de los ataques de fuerza bruta dirigidos a estos servicios consisten en realizar solicitudes repetidas con diversas credenciales en un corto periodo de tiempo, usualmente desde una misma dirección IP.

Dado que no es viable monitorizar manualmente los logs de estos servicios de forma continua, se puede optar por desarrollar un script utilizando Python 3 y Bash que haga revisiones periódicas de estos logs en busca de actividad sospechosa. En caso de detectarla, el script podría bloquear la dirección IP del atacante para evitar que continúe accediendo al servicio que está atacando.

El proceso para crear el script descrito anteriormente se divide en dos pasos. Primero, la generación de un lector de archivos de logs y, posteriormente, en Bash, la implementación de las acciones necesarias para bloquear una dirección IP si el programa en Python así lo indica. A continuación se muestran estos procesos para monitorizar Horizon y SSH.

4.3.1. Script para la protección de Horizon

4.3.1.1. Programa en Python para el análisis de logs en Horizon

Cuando ocurre un intento fallido de inicio de sesión en la plataforma de Horizon, se suele generar el mensaje log que se muestra en la consola 4.43.

Consola 4.43: Mensajes generados por Horizon cuando hay un intento fallido de inicio de sesión.

```
[Thu Dec 01 18:17:55.023109 2024] [wsgi:error] [pid 2633:tid 2806] [remote
    192.168.10.191:49896] INFO openstack_auth.forms Login failed for user "a"
     using domain "a", remote address 192.168.10.191.
```

El mensaje se almacena en el archivo /var/log/httpd/error_log. A partir de esta estructura es posible generar un programa en Python que lea este archivo y si encuentra por

lo menos 5 mensajes como los mostrados en la consola 4.43 en un intervalo menor a 15 minutos, entonces el programa registra la dirección IP como atacante guardándola en el archivo /var/log/dropIP. Es importante mencionar que se asignó un periodo de 15 minutos como un tiempo máximo en lo que una persona podría equivocarse 5 veces a lo más. Sin embargo, este valor puede cambiar bajo un estudio más completo en el tiempo. No se detallará la estructura del código utilizado en este programa. Sin embargo, el código completo, junto con una breve documentación, se puede consultar en el Apéndice D.1.

4.3.1.2. Script en Bash para implementar medidas frente a un ataque

El programa desarrollado en el punto anterior deberá integrarse dentro de un script en Bash. Este script ejecutará el programa de Python y, en caso de que este último retorne una dirección IP (indicando que el servidor está siendo atacado), el script tomará dicha dirección IP y añadirá una nueva regla al firewall. Esta regla bloqueará el acceso al puerto 440 (asignado a Horizon según la configuración realizada en este trabajo) para la dirección IP identificada. El código completo, junto con una breve documentación, se puede consultar en el Apéndice D.2.

4.3.2. Script para la protección de SSH

4.3.2.1. Programa en Python para el análisis de logs en SSH

Cuando ocurre un intento fallido de inicio de sesión por medio de SSH, se suele generar el mensaje log que se muestra en la consola 4.44.

Consola 4.44: Mensajes generados por SSH cuando hay un intento de inicio de sesión.

```
Dec  7 20:11:33 server1 sshd[4728]: Failed password for student from 10.0.2.5
    port 39782 ssh2
Dec  7 20:11:34 server1 sshd[4728]: Failed password for student from 10.0.2.5
    port 39782 ssh2
Dec  7 20:11:34 server1 sshd[4728]: Connection closed by authenticating user
   student 10.0.2.5 port 39782 [preauth]
```

El proceso es similar al que se realizó en Horizon, el programa de Python se va a encargar de leer el archivo /var/log/secure, en búsqueda de patrones de ataques, si encuentra por lo menos 5 intentos de inicio de sesión fallidos en un intervalo menor a 15 minutos entonces va a registrar esa dirección IP en /var/log/dropIPSSH, y la enviará al script en Bash para que sea bloqueada. El código completo, junto con una breve documentación, se puede consultar en el Apéndice D.3.

4.3.2.2. Script en Bash para implementar medidas frente a un ataque

El programa desarrollado en el punto anterior deberá integrarse dentro de un script en Bash. Similar al funcionamiento del script de Horizon, este programa generará una regla en el firewall que bloqueará el acceso al puerto 22 (SSH) para la dirección IP identificada como atacante. El código completo, junto con una breve documentación, se puede consultar en el Apéndice D.4.

Los scripts para ambos casos se pueden programar para que se ejecuten cada 15 minutos dentro del servidor mientras está activo usando crontab. Se debe ejecutar el comando mostrado en la consola 4.45.

Consola 4.45: Creación de tarea en crontab.

```
root@controller:# crontab -e
```

Esto abrirá un editor de textos donde se debe colocar las líneas mostradas en la consola 4.46, sustituyendo los puntos suspensivos por la ruta de los scripts.

Consola 4.46: Tareas de crontab.

```
*/15 * * * *  /../../scriptHorizon.sh
*/15 * * * *  /../../scriptSSH.sh
```

Capítulo 5

Creación de imágenes ISO seguras

En OpenStack, el usuario admin sube a la nube las imágenes ISO de los sistemas operativos disponibles para que los usuarios puedan seleccionar aquella que mejor se adapte a sus necesidades (módulo Glance). Por ejemplo, en entornos empresariales, es común requerir servicios específicos como servidores web, bases de datos, servidor de correo, copias de seguridad y servidores DNS, cada uno corriendo en máquinas virtuales independientes.

Es posible personalizar el contenido de una imagen ISO para incluir los paquetes y dependencias necesarias, de manera que se instalen automáticamente al crear máquinas virtuales. Además, estas imágenes pueden ser configuradas con medidas de seguridad adicionales, especialmente en distribuciones de Linux que, por defecto, carecen de configuraciones de seguridad robustas, como Ubuntu. Esto permite incorporar funcionalidades preinstaladas similares a las que ofrecen distribuciones más enfocadas en seguridad empresarial, como Red Hat.

Para crear estas imágenes personalizadas, se utiliza la herramienta Cubic, que facilita la personalización de imágenes ISO.

Cubic no es compatible con RHEL 9, por lo que en esta tesis, los clientes serán máquinas virtuales basadas en Ubuntu 20.04.6 Desktop. El proceso de instalación de Cubic puede consultarse en el Apéndice F.4.

Para crear una imagen personalizada se selecciona el directorio donde Cubic pueda hacer el proceso de creación de la imagen (figura 5.1). Después, se presiona el botón Next.

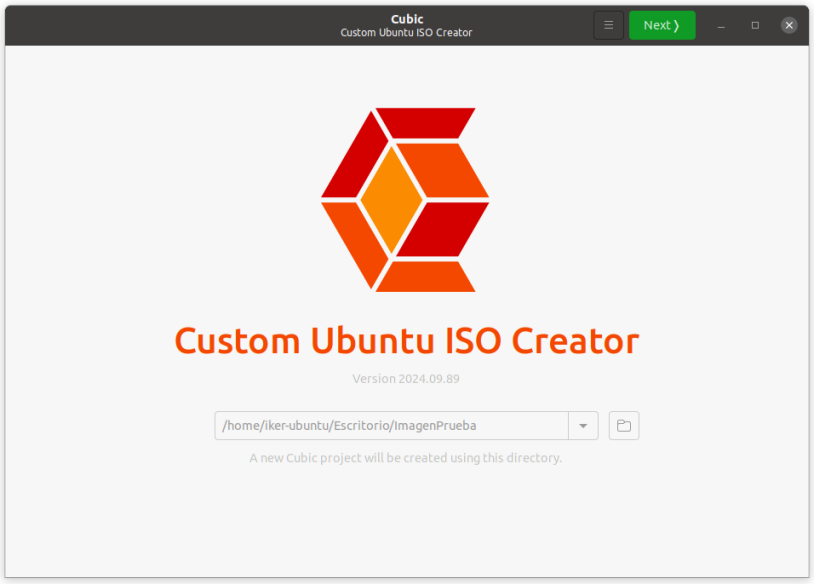

Figura 5.1: Pantalla inicial de Cubic.

Posteriormente, se selecciona la imagen que se va a usar como base, una vez seleccionada, se despliega la información que `Cubic` recupera de ella (figura 5.2). No es necesario realizar modificaciones en esta sección, ya que el programa analiza automáticamente la imagen y completa los campos requeridos.

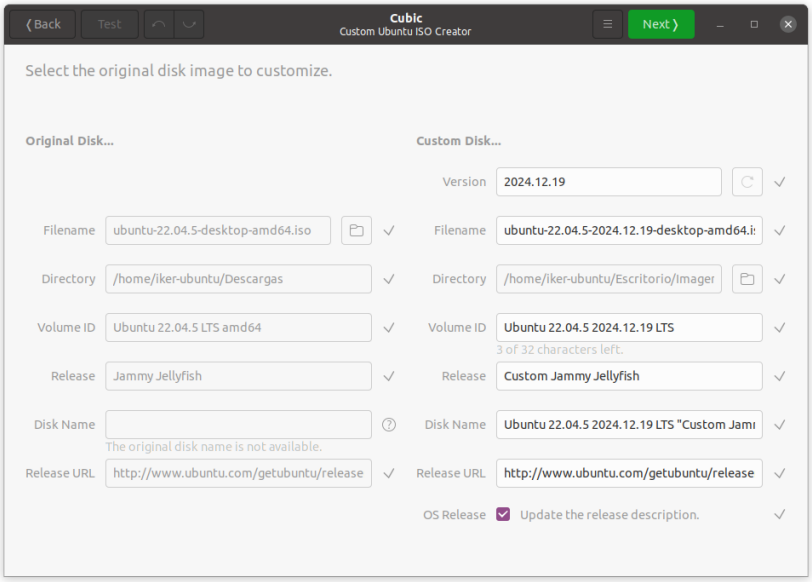

Figura 5.2: Pantalla de configuración inicial de Cubic.

Después, se accede a una consola que emula un entorno basado en la imagen cargada (figura 5.3). En esta consola, se puede instalar y configurar nuevas dependencias.

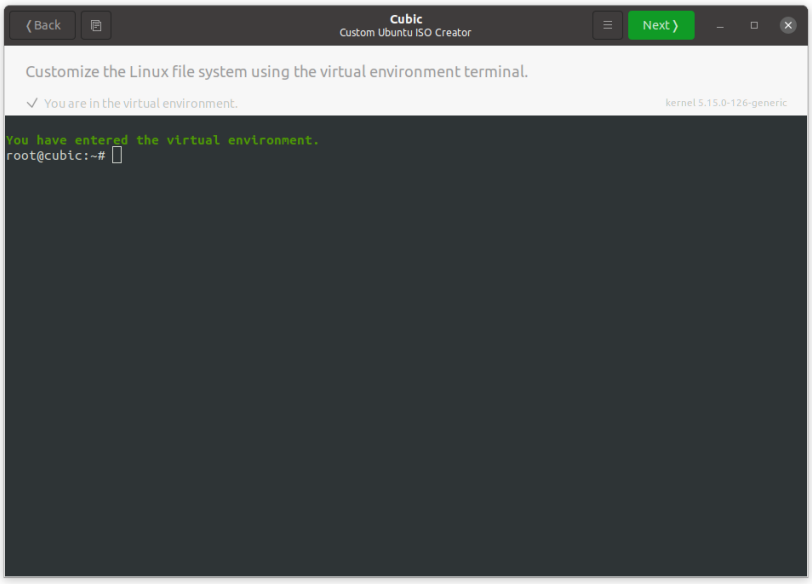

Figura 5.3: Pantalla de la consola de Cubic.

A continuación, se detalla el proceso de instalación y configuración que debe aplicarse a todas las imágenes, con el objetivo de mejorar su seguridad y garantizar un entorno más protegido 2.11.

1. **Actualización de paquetes:** Uno de los inconvenientes comunes al crear una máquina virtual con una distribución de Linux, es el tiempo que lleva actualizar todos los paquetes al estado más reciente. Para resolver este problema, se realiza una actualización previa de los paquetes, asegurando que estén al día. Esto no solo optimiza el tiempo de despliegue, sino que también mejora la experiencia de los usuarios que utilicen las imágenes. Para hacer esto, basta con colocar el comando mostrado en la consola 5.1.

Consola 5.1: Actualización de paquetes de Ubuntu.

```
root@cubic:# sudo apt upgrade
```

De igual manera, se instala el paquete `unattended-upgrades`, y se programa para que haga actualizaciones automáticas cada que lo requiera con el comando que se muestra en la consola 5.2.

Consola 5.2: Programación de actualizaciones automáticas.

```
root@cubic:# sudo apt install unattended-upgrades
root@cubic:# sudo dpkg-reconfigure unattended-upgrades
```

2. **Limitaciones en el uso de sudo:** Para evitar que cualquier usuario pueda usar sudo para cambiar la contraseña del `root`, se añade la linea que se muestran en la consola 5.3 al archivo de configuración `/etc/sudoers`.

Consola 5.3: Modificación al grupo -sudo- en /etc/sudoers.

```
%sudo ALL=(ALL:ALL) ALL,!/usr/bin/passwd root,!/bin/su, !/bin/bash,
    !/bin/sh, !/usr/bin/sudo -i
```

Se crea el archivo `/var/log/sudo.log` y se configuran los permisos correspondientes. Al propietario (`root`) y al grupo propietario (`root`) se les otorgan permisos de lectura y escritura, mientras que a cualquier otro usuario se les niegan todos los permisos. Dentro de esa misma configuración, se añade la ruta para realizar la auditoria del uso de `sudo`.

Consola 5.4: Modificación para añadir auditoria en /etc/sudoers.

```
Defaults logfile="/var/log/sudo.log"
```

3. **Limitaciones en SSH:** Para poder implementar las mejoras en la seguridad de `SSH`, primero se debe instalar el paquete `openssh-server`. Posteriormente, se accede al archivo de configuración del servidor `SSH` (`/etc/ssh/sshd_config`) y en el parámetro `PermitRootLogin`, se coloca el valor `no`, como se muestra en la consola 5.5. Esto evita que se acepte cualquier intento de conexión como usuario `root`. Posteriormente, dentro del mismo archivo de configuración de `SSH` se configura un límite de tres intentos para ingresar credenciales correctas. Además, se restringe el período de autenticación a 60 segundos y se establece un máximo de dos sesiones activas simultáneas, estas reglas son establecidas por las últimas 3 líneas de la consola 5.5. Es importante mencionar que el puerto en el que opera inicialmente el servicio `SSH` no será modificado, ya que la elección de un nuevo puerto queda a discreción del usuario de la imagen ISO.

Consola 5.5: Modificación en la configuración /etc/ssh/sshd_config.

```
PermitRootLogin no
MaxAuthTries 3
LoginGraceTime 60
MaxSessions 2
```

4. **Instalación del firewall:** Se debe instalar el paquete `ufw`. Las reglas que se van a añadir dependerán del servicio que se ofrezca y se mencionarán más adelante.

5. **Creación de contraseñas seguras:** Se debe instalar el paquete `libpam-pwquality`. Posteriormente, se accede al archivo de configuración `/etc/pam.d/common-password` y se agrega la regla mostrada en la consola 5.6 (se debe eliminar cualquier otra regla de contraseña existente dentro del archivo). Esta regla permite hasta tres intentos para ingresar una contraseña que no cumpla con los requisitos. Además, establece que la contraseña debe tener una longitud mínima de 8 caracteres, incluir al menos un dígito, contener al menos una letra mayúscula, un carácter especial y al menos una letra minúscula.

Consola 5.6: Modificación en la configuración /etc/pam.d/common-password.

```
password requisite pam_pwquality.so retry=3 minlen=8 dcredit=-1
    ucredit=-1 ocredit=-1 lcredit=-1
```

Una vez completado este proceso, es posible seguir con la instalación o modificación de las configuraciones necesarias para implementar los servicios. A continuación, se describen los procedimientos específicos para finalizar la creación de cada una de las imágenes correspondientes a los diferentes servicios.

5.1. Imagen 1: Servidor Web

Para configurar un servidor Web básico, es necesario instalar el paquete `apache2`, así como `mariadb-server` para gestionar bases de datos, y `openssl` en caso de que se necesiten generar certificados. Finalmente, se configura el `firewall` básico, para permitir solo los puertos `HTTPS` (443/`TCP`), `SSH` (22/`TCP`) y el puerto de `mariadb` (3306/`TCP`).

Consola 5.7: Configuración del firewall para servidor web.

```
root@cubic:# sudo ufw allow 443/tcp
root@cubic:# sudo ufw allow 22/tcp
root@cubic:# sudo ufw allow 3306/tcp
root@cubic:# sudo ufw enable
root@cubic:# sudo ufw reload
```

5.2. Imagen 2: Servidor de correo

Para configurar un servidor de correo básico, es necesario instalar los paquetes `postfix`, `openssl`, `libsasl2-modules` y `dovecot-imapd dovecot-pop3d`. Finalmente, se configura el `firewall` básico, para permitir solo los puertos de `SMTP` (25/`TCP`), `SMTP` con `STARTTLS` (587/`TCP`), `SMTP` con `SSL/TLS` (465/`TCP`), el puerto de `mariadb` (3306/`TCP`) y el puerto de `SSH` (22/`TCP`).

Consola 5.8: Configuración del firewall para servidor de correo.

```
root@cubic:# sudo ufw allow 25/tcp
root@cubic:# sudo ufw allow 587/tcp
root@cubic:# sudo ufw allow 465/tcp
root@cubic:# sudo ufw allow 3306/tcp
root@cubic:# sudo ufw allow 22/tcp
root@cubic:# sudo ufw enable
root@cubic:# sudo ufw reload
```

5.3. Imagen 3: DNS

Para configurar un servidor `DNS`, es necesario instalar los paquetes `bind9`, `bind9utils`, `bind9-doc`. Finalmente, se configura el `firewall` básico, para permitir solo los puertos de `DNS` (53/`UDP`), `DNS` sobre `TLS` (853/`TCP`) y el puerto de `SSH` (22/`TCP`).

Consola 5.9: Configuración del firewall para el DNS.

```
root@cubic:# sudo ufw allow 53/udp
root@cubic:# sudo ufw allow 853/tcp
root@cubic:# sudo ufw allow 22/tcp
root@cubic:# sudo ufw enable
root@cubic:# sudo ufw reload
```

5.4. Imagen 4: Servidor de base de datos

Para configurar un servidor con una base de datos, es necesario instalar los paquetes `mariadb-server`, `postgresql` y `postgresql-contrib`. Finalmente, se configura el `firewall` básico, para permitir solo el puerto de `mariadb` (3306/`TCP`), el puerto de `postgresql` (5432/`TCP`), el puerto de `SSH` (22/`TCP`).

Consola 5.10: Configuración del firewall para servidor con la base de datos.

```
root@cubic:# sudo ufw allow 3306/tcp
root@cubic:# sudo ufw allow 5432/tcp
root@cubic:# sudo ufw allow 22/tcp
root@cubic:# sudo ufw enable
root@cubic:# sudo ufw reload
```

5.5. Imagen 5: Servidor de respaldos

Para configurar un servidor de respaldos, es necesario instalar los paquetes `samba` y `nfs-kernel-server`. Finalmente, se configura el `firewall` básico, para permitir solo el puerto de `samba` (445/`TCP`), el puerto de `NFS` (2049/`TCP`), el puerto de `SSH` (22/`TCP`).

Consola 5.11: Configuración del firewall para servidor de respaldo.

```
root@cubic:# sudo ufw allow 445/tcp
root@cubic:# sudo ufw allow 2049/tcp
root@cubic:# sudo ufw allow 22/tcp
root@cubic:# sudo ufw enable
root@cubic:# sudo ufw reload
```

Una vez instalado todos los paquetes, en la interfaz de `Cubic` se presiona el botón `Next`. A continuación, `Cubic` solicitará si se desea añadir o eliminar paquetes existentes (figura 5.4). Por defecto, `Cubic` elimina algunos paquetes innecesarios de `Ubuntu`, por lo que generalmente no es necesario realizar cambios en este paso. Simplemente se debe presionar nuevamente el botón `Next` para continuar.

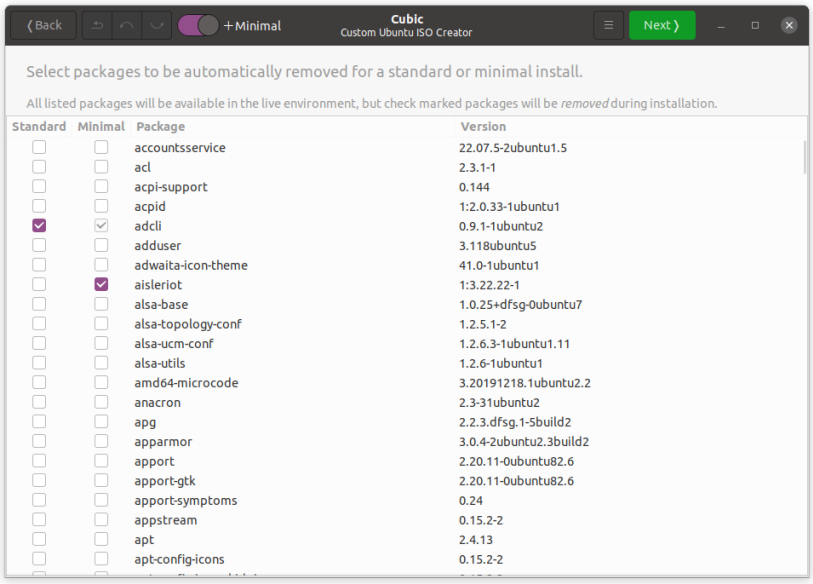

Figura 5.4: Pantalla de la consola de Cubic.

Posteriormente, se le debe indicar a `Cubic` el tipo de compresión que será usado en la imagen que se va a generar (figura 5.5), se recomienda usar la opción por defecto que da el programa (`gzip`).

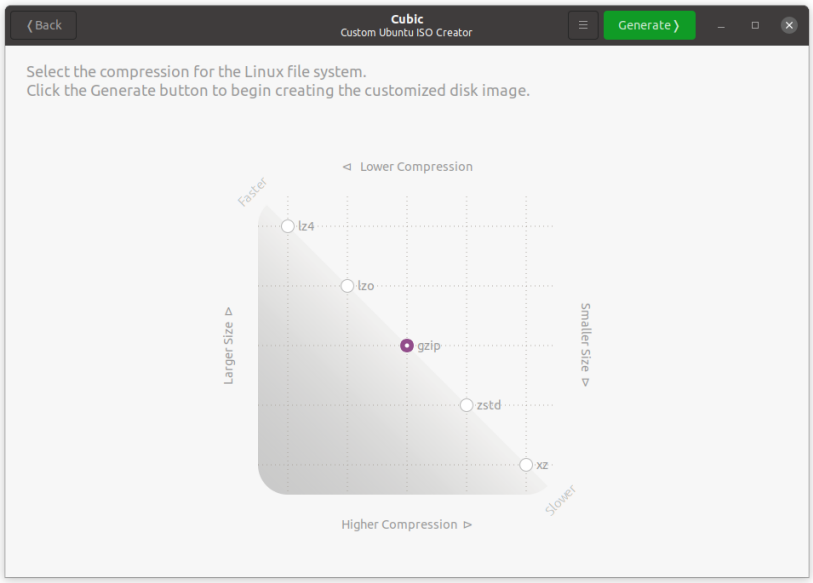

Figura 5.5: Pantalla de selección del tipo de compresión en Cubic.

Como último paso, `Cubic` va a generar la imagen tomando en consideración todos los ajustes que se le dieron, y verifica que no existan errores (figura 5.6), este proceso suele tardar varios minutos.

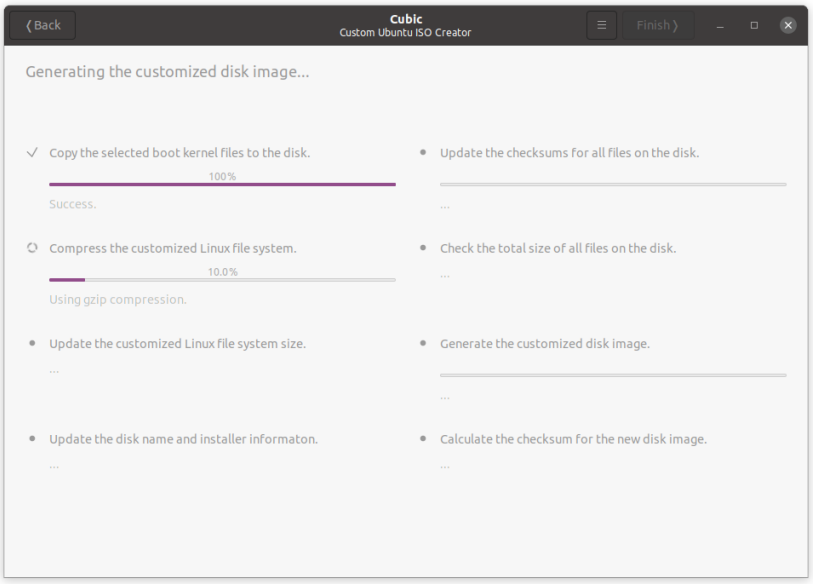

Figura 5.6: Pantalla del proceso de generación de la imagen en Cubic.

Al terminar el proceso anterior la imagen está lista, se debe seleccionar la casilla que nos indica que se borrarán todos los archivos generados a excepción de la imagen y documentación, esto con el fin de que no se desperdicie el almacenamiento de nuestro disco duro (figura 5.7).

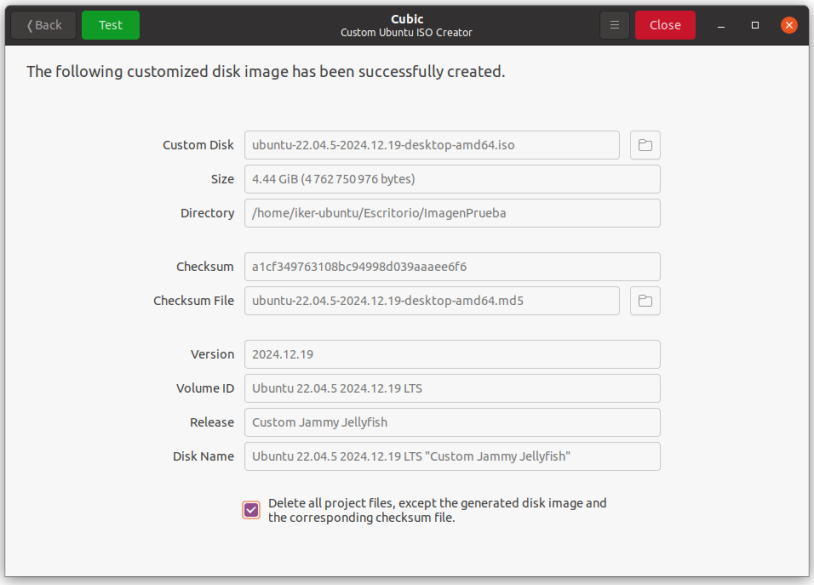

Figura 5.7: Pantalla final del proceso de creación de una imagen en Cubic.

En este punto se han generado las imágenes mostradas en la tabla 5.1.

Nombre de la Imagen	Propósito
ubuntu-22.04.5-2024.10.15-DBServer.iso	Servidor de base de datos
ubuntu-22.04.5-2024.10.15-SambaServer.iso	Servidor Samba
ubuntu-22.04.5-2024.10.15-DNSServer.iso	Servidor DNS
ubuntu-22.04.5-2024.10.15-WebServer.iso	Servidor Web
ubuntu-22.04.5-2024.10.15-MailServer.iso	Servidor de correo electrónico

Tabla 5.1: Nombres correspondientes a las imágenes de Ubuntu con sus propósitos específicos.

Ahora, se deben transferir estas imágenes al nodo `Controller`, una vez que estén ahí, se deben subir para que estén disponibles en la nube, esto se hace mediante los comandos de la consola 5.12.

Consola 5.12: Comando para cargar las imagenes a OpenStack.

```
root@controller:# glance --insecure image-create --name "Servidor de base de datos" --file
    ubuntu-22.04.5-2024.10.15-DBServer.iso --disk-format qcow2 --container-format bare   --
    visibility=public
root@controller:# glance --insecure image-create --name "Servidor de respaldo" --file
    ubuntu-22.04.5-2024.10.15-SambaServer.iso --disk-format qcow2 --container-format bare
      --visibility=public
root@controller:# glance --insecure image-create --name "Servidor DNS" --file ubuntu
    -22.04.5-2024.10.15-DNSServer.iso   --disk-format qcow2 --container-format bare   --
    visibility=public
root@controller:# glance --insecure image-create --name "Servidor WEB" --file ubuntu
    -22.04.5-2024.10.15-WebServer.iso  --disk-format qcow2 --container-format bare   --
    visibility=public
root@controller:# glance --insecure image-create --name "Servidor de Correo" --file ubuntu
    -22.04.5-2024.10.15-MailServer.iso  --disk-format qcow2 --container-format bare   --
    visibility=public
```

Se puede comprobar que las imágenes se encuentran disponibles desde la interfaz de `Horizon` (figura 5.8).

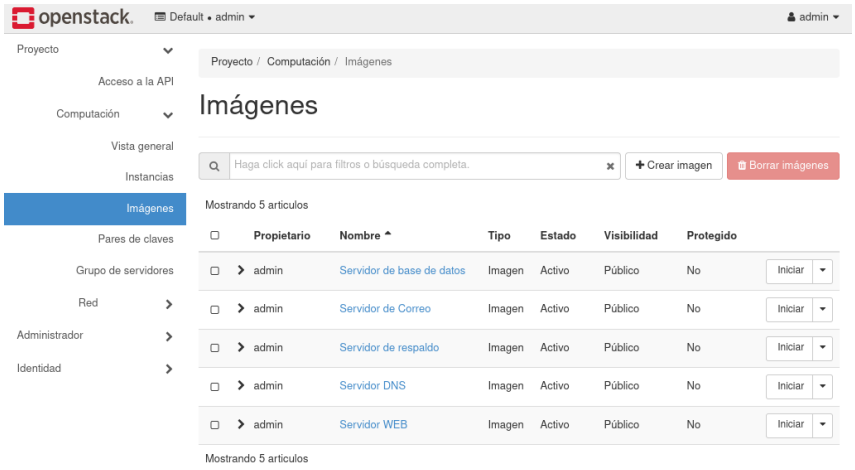

Figura 5.8: Verificación de la carga de las imágenes a la nube.

Capítulo 6

Pruebas de seguridad

En este capítulo, se presenta el análisis de seguridad mediante la ejecución de ataques controlados. En primer lugar, se evalúa un servidor `OpenStack` sin ninguna medida de seguridad, excepto el tráfico cifrado en el puerto 443 (HTTPS). Posteriormente, se realizan los mismos ataques sobre el servidor propuesto en esta tesis, al que se le han aplicado las medidas de seguridad presentadas en los capítulos anteriores. Esto con el objetivo de comparar los resultados y evaluar la robustez de dichas implementaciones. El patrón de ataque que se va a utilizar se basa en los puntos listados en la sección 2.12. El nodo `Controller` será el blanco de todas las pruebas.

Los atacantes son máquinas virtuales creadas en el mismo servidor donde se aloja el nodo `Controller`, sin embargo, se encuentran en segmentos de red distintos. Las características de estas máquinas se muestran en la tabla 6.1.

Especificaciones	**Hacker 1**	**Hacker 2**	**Hacker 3**
Sistema Operativo	Ubuntu 20.04.6 Desktop	Ubuntu 20.04.6 Desktop	Ubuntu 20.04.6 Desktop
Núcleos	4	4	4
RAM	4048 MB	4048 MB	4048 MB
Almacenamiento	50 GB	50 GB	50 GB
Tarjetas de red	2	2	2
IP del atacante	10.0.20.2	172.10.20.2	10.0.50.1

Tabla 6.1: Especificaciones técnicas de los atacantes.

A continuación se listan las máquinas virtuales que se van a usar a lo largo de este capítulo.

1. **Nodo Controller inseguro**
2. **Nodo Controller seguro**
3. **Hacker 1**
4. **Hacker 2**
5. **Hacker 3**

Para poder realizar las pruebas, se crea un entorno seguro que evita posibles conflictos legales al realizar los ataques. Este entorno permite colocar las máquinas virtuales en segmentos de red diferentes y que estas puedan comunicarse entre sí sin necesidad de requerir acceso a Internet. En la figura 6.1 se muestra el diagrama general de la comunicación entre las máquinas.

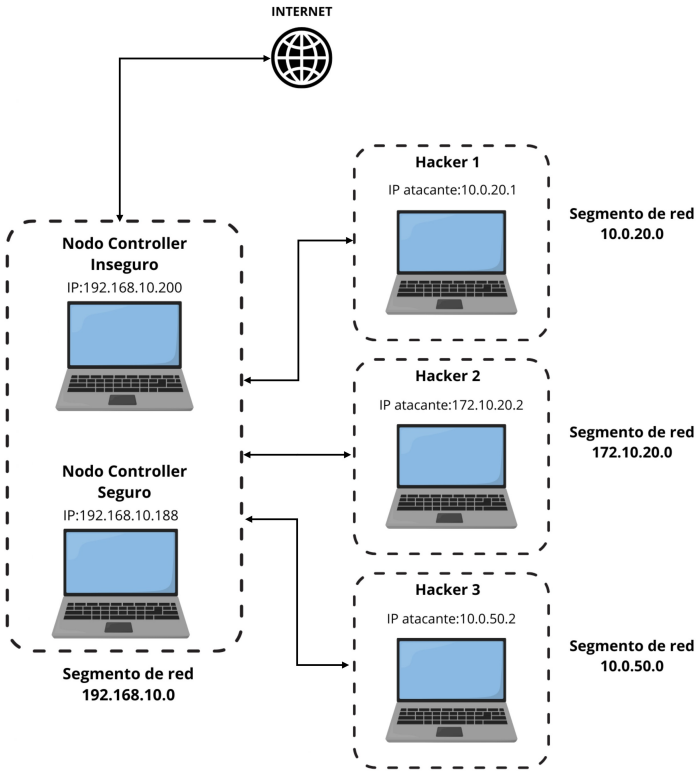

Figura 6.1: Diagrama general de la comunicación entre atacantes y OpenStack.

Para obtener detalles de cómo fueron creadas las máquinas virtuales y el proceso realizado para generar el entorno mostrado en la figura 6.1, se puede consultar el Apéndice E.

6.1. Servidor sin implementaciones de seguridad

Los detalles del servidor `OpenStack` inseguro se pueden encontrar en la tabla 6.2.

Especificaciones	Nodo Controller
Sistema Operativo	Ubuntu Server
Núcleos	6
RAM	8192 MB
Almacenamiento	200GB
Tarjetas de red	1
IP	192.168.10.200

Tabla 6.2: Especificaciones técnicas del nodo Controller del servidor inseguro.

Dado que este servidor no tiene políticas de seguridad, el ataque se realizará al usuario `root`. Sin embargo, se cambiará la contraseña mediante diferentes patrones con el fin de obtener una contraseña con seguridad baja, media y alta como se mencionó en la sección 2.13. La longitud de estas contraseñas se fijó a 4 caracteres, dado que es un ataque de prueba y no se cuenta con tiempo suficiente para romper contraseñas de más de 5 caracteres. Las características de cada contraseña se muestran en la tabla 6.3.

Tipo de contraseña	Características
Insegura	Tiene solo letras minúsculas
Medianamente segura	Tiene letras minúsculas y números
Segura	Tiene letras minúsculas, números y los caracteres especiales `@`, `#`, `$`, `%`, `&` y `*`

Tabla 6.3: Contraseñas de los usuarios del nodo Controller inseguro.

En la tabla 6.4 se muestran las contraseñas que corresponden a cada usuario.

Usuario	Contraseña	Nivel de seguridad
root	jshd	Baja
root	j22i	Media
root	5z8@	Alta

Tabla 6.4: Contraseñas de los usuarios del nodo Controller inseguro.

6.1.1. Identificación de puntos débiles

6.1.1.1. Escaneo de puertos

Para identificar puntos débiles, se usa la máquina virtual `Hacker 1`, en ella se instala `Nmap`. El proceso de instalación de `Nmap` se puede consular en el Apéndice F.1. `Hacker 1` ejecuta el comando mostrado en la consola 6.1, el cual realiza un escaneo intensivo de puertos de una dirección IP dada, identificando servicios y versiones, además de recopilar información avanzada sobre el sistema operativo, todo esto con la máxima velocidad posible.

Consola 6.1: Escaneo de puertos a nodo Controller inseguro.

```
root@hacker1:# nmap -sV -A -p- 192.168.10.200 -T5
```

Para saber más detalles sobre los parámetros del comando anterior, se puede consultar el Apéndice B.1. La salida de este comando se muestra en la consola 6.2.

Consola 6.2: Resultado del análisis de puertos usando Nmap en nodo Controller inseguro.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2024-11-04 12:48 CST
Stats: 0:00:04 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:01:40 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 70.59% done; ETC: 12:51 (0:00:36 remaining)
Stats: 0:02:15 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 94.12% done; ETC: 12:51 (0:00:08 remaining)
Nmap scan report for 192.168.10.200
Host is up (0.00015s latency).
Not shown: 65518 closed ports
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux;
    protocol 2.0)
80/tcp    open  http        Apache httpd 2.4.52
443/tcp   open  ssl/http    Apache httpd 2.4.52 ((Ubuntu))
2379/tcp  open  http        Golang net/http server (Go-IPFS json-rpc or
    InfluxDB API)
2380/tcp  open  http        Golang net/http server (Go-IPFS json-rpc or
    InfluxDB API)
3306/tcp  open  mysql       MySQL 5.5.5-10.6.12-MariaDB-0ubuntu0.22.04.1
4369/tcp  open  epmd        Erlang Port Mapper Daemon
5000/tcp  open  http        Apache httpd 2.4.52 ((Ubuntu))
```

```
5672/tcp  open  amqp        RabbitMQ 3.9.13 (0-9)
6082/tcp  open  ssl/p25cai?
8774/tcp  open  unknown
8775/tcp  open  unknown
8778/tcp  open  http        Apache httpd 2.4.52 ((Ubuntu))
9292/tcp  open  http        BaseHTTPServer 0.6 (Python 3.10.12)
9696/tcp  open  unknown
11211/tcp open  memcached   Memcached 1.6.14 (uptime 1450 seconds)
25672/tcp open  unknown
```

Se puede observar que existen múltiples puertos abiertos, la mayoría de ellos empleados por los módulos y herramientas de `Openstack`. Esto indica que hay múltiples puntos de acceso y servicios que pueden ser objetivos de ataque. Más adelante nos enfocaremos en el puerto 22 (que corresponde al servicio `SSH`).

6.1.1.2. Búsqueda de rutas ocultas de la página web de Horizon

El punto de acceso a `Openstack` es a través de `Horizon`, por lo que es importante encontrar puntos débiles. Para ello, se realiza una búsqueda de posibles rutas ocultas que tenga el sitio web de `Horizon`. Con la máquina `Hacker 2`, la cual tiene instalado `Gobuster` se emplea el comando mostrado en la consola 6.3 con la lista de nombres (`common.txt` de `SecLists`). El proceso de instalación de `Gobuster` y las listas de subdirectorios se puede consular en el Apéndice F.2.

Consola 6.3: Búsqueda de rutas ocultas a Horizon del nodo Controller inseguro.

```
root@hacker2:# gobuster dir -u https://192.168.10.200/horizon -w /path/to/SecLists/
    Discovery/Web-Content/common.txt
```

Para saber más detalles sobre los parámetros del comando anterior, se puede consultar el Apéndice B.2. La salida de este comando se muestra en la consola 6.4. Se puede ver en esta consola que existen múltiples rutas a las que se puede acceder usando la dirección `https://192.168.10.200/horizon/`. Más adelante se muestra que contienen estos directorios.

Consola 6.4: Resultado de la búsqueda de subdirectorios ocultos en la página de Horizon del nodo Controller inseguro.

```
===============================================================
Gobuster v2.0.1              OJ Reeves (@TheColonial)
===============================================================
[+] Mode         : dir
[+] Url/Domain   : https://192.168.10.200/horizon/
[+] Threads      : 10
[+] Wordlist     : /root/SecLists-master/Discovery/Web-Content/common.txt
[+] Status codes : 200,204,301,302,307,403
[+] Timeout      : 10s
===============================================================
===============================================================
/admin (Status: 301)
/header (Status: 301)
/home (Status: 301)
/identity (Status: 301)
/project (Status: 301)
/settings (Status: 301)
/static (Status: 301)
===============================================================
===============================================================
```

6.1.2. Ataque a puntos débiles

6.1.2.1. Fuerza bruta para SSH

Para este ataque, en el Hacker 3, se emplea un script hecho en Bash y Python que crea contraseñas con las características mencionadas en la tabla 6.3. Estas características entran como argumento a la herramienta Hydra, el cual se ejecuta hasta que logre acceder con alguna de las contraseñas generadas. Cuando esto suceda, el script guarda la contraseña y el tiempo que le tomó. Si se requiere conocer el proceso de instalación de Hydra y sus parámetros, se pueden consultar los Apéndices F.2 y B.3, respectivamente.

Este ataque está enfocado en el puerto 22 (que corresponde al servicio SSH), sin embargo, se puede hacer en otro puerto. El primer objetivo del ataque será el usuario root cuya contraseña está compuesta únicamente por letras minúsculas. Para este propósito, se emplea el script que se puede consultar en el Apéndice G.1, cuya salida se muestra en la consola 6.5.

Consola 6.5: Resultado del ataque al usuario root con contraseña insegura por medio de SSH.

```
Contrasena encontrada: jshd
Numero total de intentos: 58293
Tiempo total: 209136 segundos
```

En la consola se puede ver que el script encontró la contraseña de root, y le tomó $58,293$ intentos, aproximadamente dos días y medio.

Ahora, para el usuario root considerando una contraseñas de seguridad media, se utiliza el script descrito en el Apéndice G.2. Los resultados de este ataque se presentan en la consola 6.6.

Consola 6.6: Resultado del ataque al usuario root con contraseña medianamente segura por medio de SSH.

```
Contrasena encontrada: j22i
Numero total de intentos: 68356
Tiempo total: 249517 segundos
```

El script encontró la contraseña de root, y le tomó $68,356$ intentos, casi tres días. Finalmente, para el caso del usuario root considerando contraseñas seguras, se utiliza el script detallado en el Apéndice G.3. A pesar de que el script se ejecutó durante más de tres semanas, no se logró obtener la contraseña. Cabe resaltar que se interrumpió el script por cuestiones de tiempo. Sin embargo, si se deja un mayor tiempo, encontrará la contraseña.

6.1.2.2. Rutas ocultas de la página web de Horizon

Al explorar las rutas ocultas identificadas por Gobuster, no se encontró información comprometedora. Las rutas /admin, /home, /identity, /project y /settings redirigen a la pantalla de inicio de sesión de Horizon (figura 6.2).

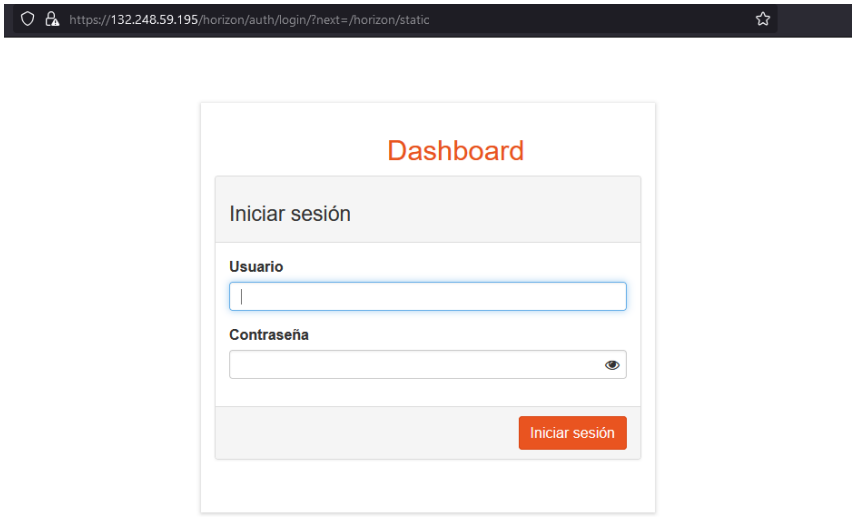

Figura 6.2: Página obtenida con las rutas /admin, /home, /identity, /project y /settings.

En cuanto a `/header`, solo muestra una página en blanco y `/static` muestra una página que indica que no se tienen permisos para poder acceder a ella (figura 6.3).

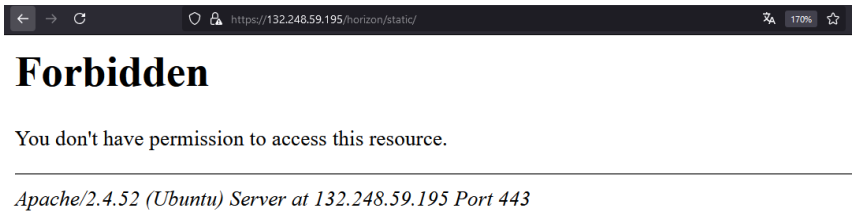

Figura 6.3: Página obtenida de la ruta /static.

Con estas pruebas, `Horizon` no ha proporcionado información que comprometa su funcionamiento, por lo que no es posible atacarlo por medio de rutas ocultas. De igual manera no es posible realizar un ataque de fuerza bruta en la plataforma, ya que cuenta con la técnica `CSRF` (consultar la Sección 2.8.1) que detecta el uso de herramientas de ataque como `Hydra` y bloquea sus intentos de acceso.

6.1.2.3. Captura y análisis de la información enviada a través de los endpoints de OpenStack

El escaneo de puertos realizado mediante el comando de la consola 6.1 expuso los módulos de `OpenStack`. Con esta información, ahora se puede entablar una conexión a ellos escribiendo el `endpoint` en el navegador web. El tráfico en los `endpoints` se puede observar con la herramienta `Wireshark` del `Hacker 2`. A continuación se muestra la comunicación en cada `endpoint` de `OpenStack`.

- **Horizon**: El acceso a la página se realiza a través de `https://192.168.10.200/horizon`, se puede observar en la figura 6.4 la captura de `Wireshark`. En esta figura se puede ver que la información relevante de la página web es transmitida por medio del protocolo `TLS` versión 1.3, lo que implica que los datos están cifrados y no se puede saber el contenido. Es importante recordar que el servidor `OpenStack` inseguro solo cifra el tráfico en el puerto 443.

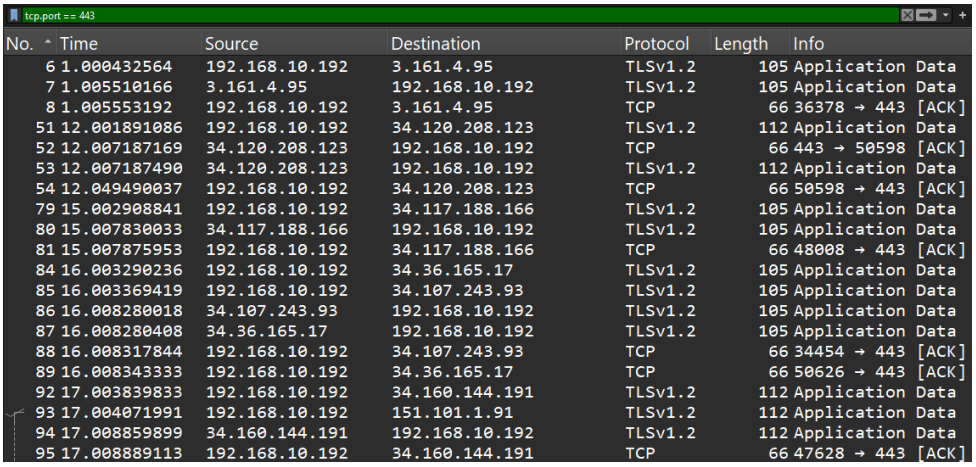

Figura 6.4: Análisis de los paquetes que trasmite y recibe Horizon usando Wireshark.

- **Neutron**: El `endpoint` se puede acceder mediante `http://192.168.10.200:9696`. En la figura 6.5 se puede ver el tráfico que capta `Wireshark`. Es importante hacer notar que el tráfico del puerto 9696 no está cifrado, por lo que es posible visualizar su contenido.

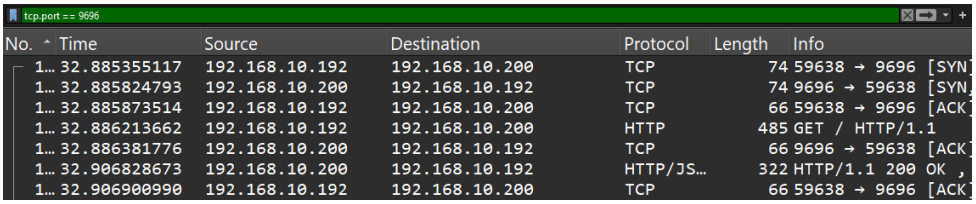

Figura 6.5: Análisis de los paquetes que trasmite y recibe Neutron usando Wireshark.

- **Keystone**: El `endpoint` se puede acceder mediante `http://192.168.10.200:5000`. En la figura 6.6 se pueden observar los paquetes que captura `Wireshark`. Es importante hacer notar que hay paquetes con el protocolo `HTTP`, por lo que la comunicación hacia `Keystone` no está cifrada.

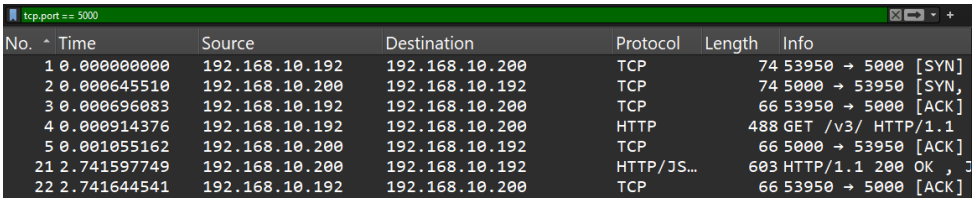

Figura 6.6: Análisis de los paquetes que trasmite y recibe Keystone usando Wireshark.

- **Placement**: El `endpoint` se puede acceder mediante `http://192.168.10.200:8778`. La figura 6.7 muestra los paquetes que captura `Wireshark`. Es importante notar que hay paquetes con el protocolo `HTTP`, por lo que la comunicación hacia `Placement` no está cifrada.

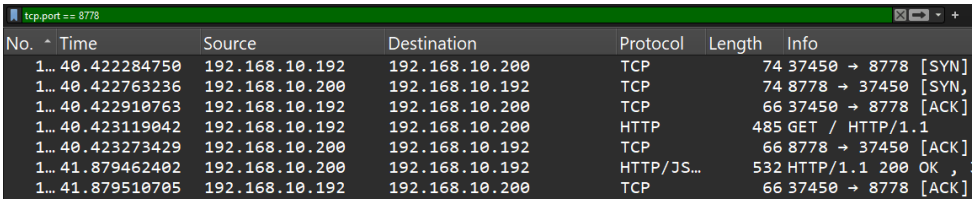

Figura 6.7: Análisis de los paquetes que trasmite y recibe Placement usando Wireshark.

- **Glance**: El `endpoint` se puede acceder mediante `http://192.168.10.200:9292`. La figura 6.8 muestra los paquetes que captura `Wireshark`. Es importante notar que hay paquetes con el protocolo `HTTP`, por lo que la comunicación hacia `Glance` no está cifrada.

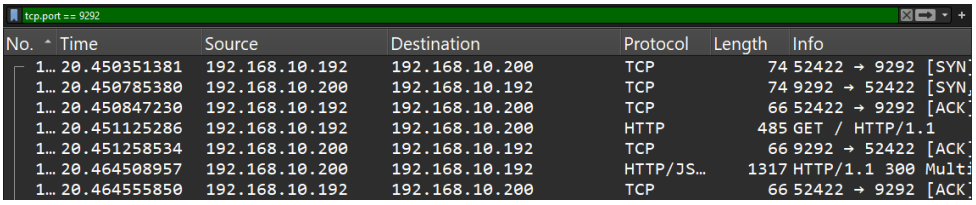

Figura 6.8: Análisis de los paquetes que trasmite y recibe Glance usando Wireshark.

- **Nova**: El `endpoint` se puede acceder mediante `http://192.168.10.200:8774`. La figura 6.9 muestra los paquetes que captura `Wireshark`. Es importante notar que hay paquetes con el protocolo `HTTP`, por lo que la comunicación hacia `Nova` no está cifrada.

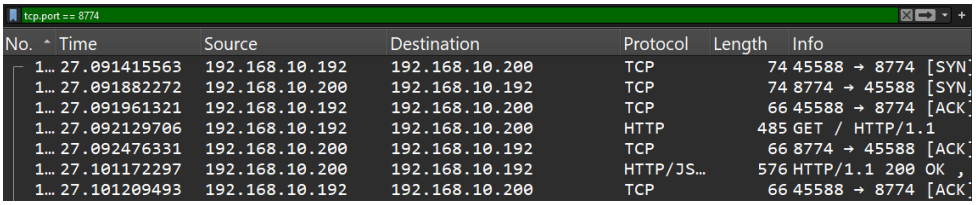

Figura 6.9: Análisis de los paquetes que trasmite y recibe Nova usando Wireshark.

Se puede observar que las comunicaciones entre los `endpoint` no están cifradas y por tanto, el contenido de los paquetes puede ser visualizado por cualquier atacante dentro y fuera del nodo.

El nodo `Controller` mostró múltiples puntos débiles, entre ellos el puerto 22 de `SSH`. Al realizar ataques de fuerza bruta fue posible acceder como usuario `root` en dos ocasiones, ya que las contraseñas que se usaron no eran lo suficientemente seguras. En cuanto a las rutas ocultas, ninguna de ellas representa un peligro de seguridad o una puerta de entrada para realizar un ataque u obtener información sensible. Sin embargo, la comunicación entre los `endpoints` es un foco de vulnerabilidad al no estar cifrada, ya que entre los módulos de `OpenStack` se transmite información de recursos disponibles, máquinas virtuales, segmentos de red, entre mucha más información.

6.2. Servidor con implementaciones de seguridad

Para esta prueba, los detalles técnicos del servidor se pueden encontrar en la tabla 3.3. En este nodo también se ataca el usuario `root` con tres niveles de seguridad en contraseñas. Esta configuración se muestra en la tabla 6.3. Se sigue la misma ruta de ataque que se realizó en el nodo `Controller` inseguro.

6.2.1. Identificación de puntos débiles

6.2.1.1. Escaneo de puertos

Se realizar el escaneo de puertos usando el comando mostrado en la consola 6.1, solo que en esta ocasión se emplea la dirección IP del nodo `Controller` seguro (192.168.10.188). El resultado de este escaneo se muestra en la consola 6.7.

Consola 6.7: Resultado del análisis de puertos usando Nmap en nodo Controller seguro.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2024-11-22 12:38 CST
Stats: 0:00:57 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth
    Scan
SYN Stealth Scan Timing: About 66.43% done; ETC: 12:40 (0:00:28 remaining)
Stats: 0:00:59 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth
    Scan
SYN Stealth Scan Timing: About 69.88% done; ETC: 12:40 (0:00:25 remaining)
Stats: 0:00:59 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth
    Scan
SYN Stealth Scan Timing: About 70.28% done; ETC: 12:40 (0:00:25 remaining)
Nmap scan report for 192.168.10.188
Host is up (0.00041s latency).
Not shown: 65534 filtered ports
PORT    STATE SERVICE  VERSION
440/tcp open  ssl/http Apache httpd 2.4.57 ((Red Hat Enterprise Linux)
    OpenSSL/3.0.7 mod_wsgi/4.7.1 Python/3.9)
MAC Address: 52:54:00:AC:3E:3D (QEMU virtual NIC)
Service detection performed. Please report any incorrect results at https://
    nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 103.44 seconds
```

Se puede observar que solo el puerto 440 está expuesto, el cual nos permite ingresar a la página web de `Horizon`.

6.2.1.2. Búsqueda de rutas ocultas de la página web de Horizon

Para este caso, desde `Hacker 2`, se usa el comando de `Gobuster` mostrado en la consola 6.3 cambiando la dirección IP del nodo `Controller` seguro. La salida de este comando se muestra en la consola 6.8.

Consola 6.8: Resultado de la búsqueda de subdirectorios ocultos en la página de Horizon del nodo Controller seguro.

```
===============================================================
Gobuster v2.0.1              OJ Reeves (@TheColonial)
===============================================================
[+] Mode         : dir
[+] Url/Domain   : https://192.168.10.188:440/
[+] Threads      : 10
[+] Wordlist     : /root/SecLists-master/Discovery/Web-Content/common.txt
[+] Status codes : 200,204,301,302,307,403
[+] Timeout      : 10s
===============================================================
/admin (Status: 301)
/cgi-bin/ (Status: 403)
/header (Status: 301)
/home (Status: 301)
/project (Status: 301)
/settings (Status: 301)
/static (Status: 301)
===============================================================
```

Se pueden observar las mismas rutas que se obtuvieron en el nodo `Controller` inseguro.

6.2.2. Ataque a puntos débiles

6.2.2.1. Fuerza bruta para SSH

Dado que no se encontraron puertos expuestos, no es posible realizar ataques de fuerza bruta como se hizo en el caso del nodo `Controller` inseguro. Suponiendo que por razones extraordinarias se requiere tener el puerto 22 abierto para su uso como servidor `SSH`, se puede intentar un ataque de fuerza bruta, sin embargo, en este caso se tendría que generar nombres de usuarios y contraseñas, ya que este servidor no permite el acceso del usuario `root`. Esto complica mucho más un ataque. Sin embargo, suponiendo que se tiene abierto el puerto 22, se pueden usar los `scripts` usados en el nodo `Controller` inseguro, con usuarios distintos. Al ejecutar el `script`, se produce un error que genera una excepción tras varios intentos fallidos. Al intentar acceder manualmente por `SSH`, se muestra el mensaje de error indicado en la consola 6.9.

Consola 6.9: Resultado de atacar repetidamente al nodo Controller seguro.

```
'root@hacker:#' ssh usuario@192.168.10.188
ssh:connect to host 192.168.10.188 port 22: Connection refused
```

Esto quiere decir que la conexión `SSH` fue bloqueada por el servidor destino, y ya no es posible acceder a él. Por otro lado, si se intenta un ataque de fuerza bruta (introduciendo credenciales directamente a la página web de `Horizon`), el resultado es similar, después de varios intentos fallidos queda bloqueado, como se muestra en la figura 6.10.

Figura 6.10: Acceso bloqueado a la página Web de Horizon desde Hacker 3.

6.2.2.2. Rutas ocultas de la página web de Horizon

En cuanto a las rutas ocultas encontradas de la página web, se tiene que `/admin`, `/cgi-bin`, `/home`, `/project` y `/settings` se dirigen a la página de acceso de `Horizon` que se muestra en la figura 6.11.

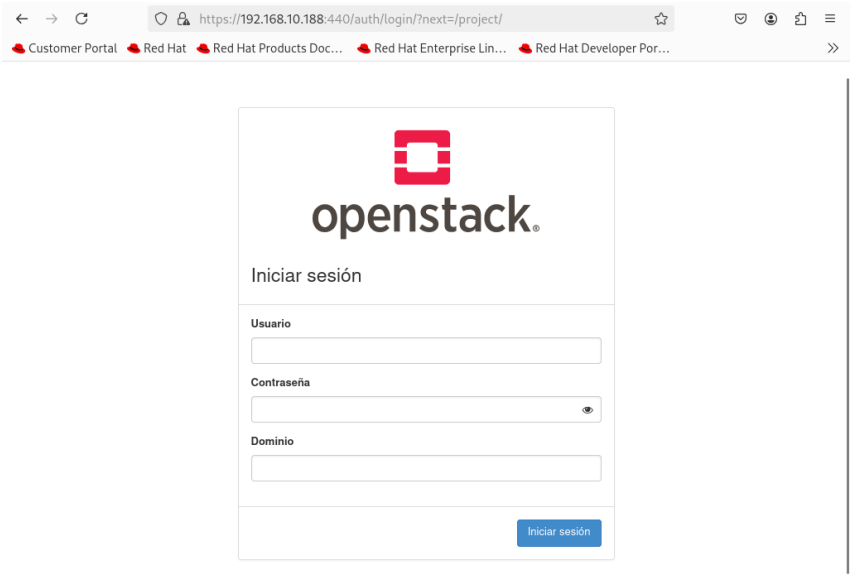

Figura 6.11: Página obtenida con las rutas /admin, /home, /cgi-bin, /project y /settings.

La ruta `/header` muestra una página en blanco y la ruta `/static` tiene el acceso restringido, como se observa el la figura 6.12.

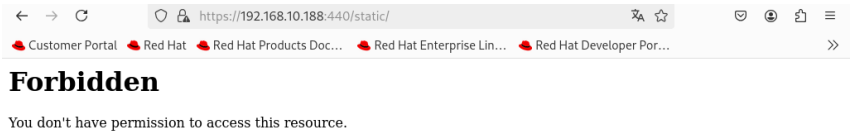

Figura 6.12: Página obtenida con la ruta /static.

6.2.2.3. Captura y análisis de la información enviada a través de los endpoints de OpenStack

Debido a la implementación del `firewall` (ver Sección 4.2.1) en el nodo `Controller`, no existe comunicación con los `endpoints`, y por tanto no es posible observar el tráfico entre ellos de forma externa, sin embargo, dentro del nodo `Controller` puede instalarse `Wireshark` y analizar los paquetes que se envían al nodo `Compute` seguro, el cual es el único nodo que puede comunicarse con los `endpoints`. A continuación se muestra la comunicación para cada `endpoint` en `OpenStack`.

- **Horizon**: El `endpoint` se puede acceder mediante `https://192.168.10.188/`. La figura 6.13 muestra el tráfico por `Wireshark`. Se puede notar que la información de la página web se transmite utilizando el protocolo `TLS` versión 1.3. Esto significa que los datos están cifrados, lo que impide conocer su contenido.

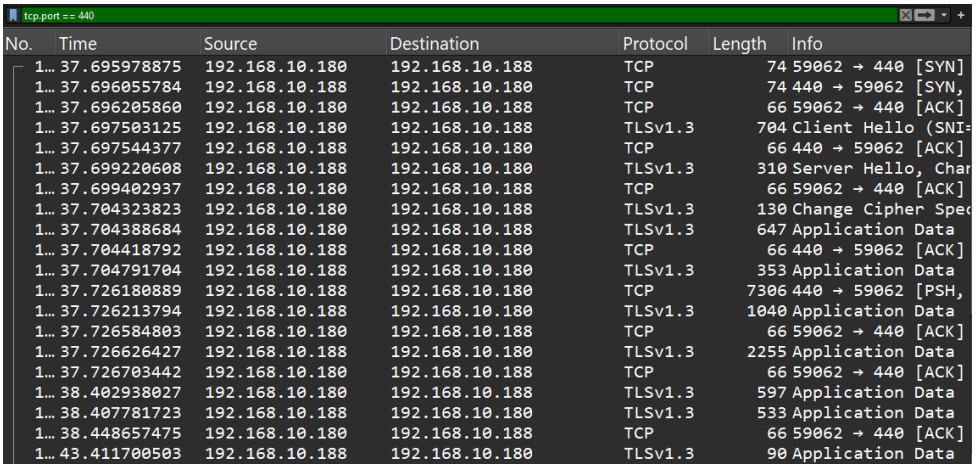

Figura 6.13: Análisis de los paquetes que trasmite y recibe Horizon usando Wireshark.

- **Neutron**: El `endpoint` se puede acceder mediante `https://192.168.10.188:9696`. La figura 6.14 muestra el tráfico detectado por `Wireshark`. Se puede observar que hay paquetes con el protocolo `TLS` versión 1.2, lo que quiere decir que están cifrados.

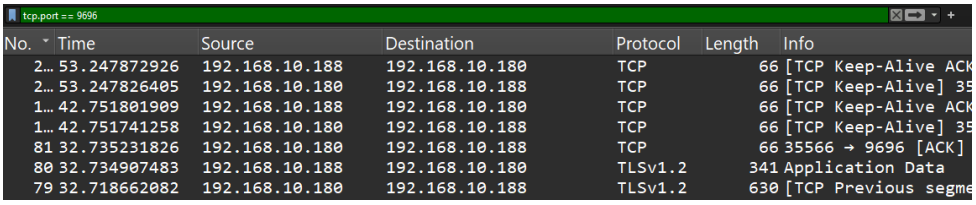

Figura 6.14: Análisis de los paquetes que trasmite y recibe Neutron usando Wireshark.

- **Keystone**: El `endpoint` se puede acceder mediante `https://192.168.10.188:5000`. La figura 6.15 muestra el tráfico captado por `Wireshark`.

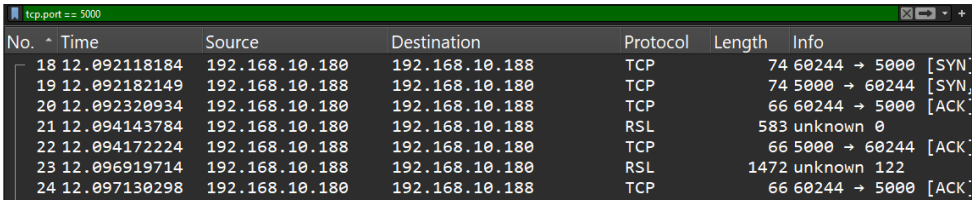

Figura 6.15: Análisis de los paquetes que trasmite y recibe Keystone usando Wireshark.

En esta figura se observa que los paquetes viajan en el protocolo `RSL`, el cual no es el protocolo deseado. Esto se debe a que `Wireshark` no puede interpretar los paquetes cifrados como `HTTPS`. Sin embargo, mediante una solicitud hacia el `endpoint` usando el comando mostrado en la consola 6.10, cuya salida del comando se muestra en la consola 6.11, se puede observar que la conexión se realiza usando `TLS` versión 1.3, por lo que la información se encuentra cifrada.

Consola 6.10: Solicitud al endpoint https://192.168.10.188:5000/v3.

```
root@controller:# curl -kv https://192.168.10.188:5000/v3
```

Consola 6.11: Salida del comando curl -kv https://192.168.10.188:5000/v3.

```
    *   Trying 192.168.10.188:5000...
* Connected to 192.168.10.188 (192.168.10.188) port 5000 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /etc/pki/tls/certs/ca-bundle.crt
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS header, Finished (20):
* TLSv1.2 (IN), TLS header, Unknown (23):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.2 (IN), TLS header, Unknown (23):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS header, Unknown (23):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.2 (IN), TLS header, Unknown (23):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.2 (OUT), TLS header, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS header, Unknown (23):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use http/1.1
* Server certificate:
*  subject: C=MX; ST=CDMX; L=CDMX; O=IKCORP; OU=IKCORP; CN=controller
*  start date: Oct  5 05:43:53 2024 GMT
*  expire date: Oct  3 05:43:53 2034 GMT
*  issuer: C=MX; ST=CDMX; L=CDMX; O=IKCORP; OU=IKCORP; CN=controller
*  SSL certificate verify result: self-signed certificate (18),
   continuing anyway.
* TLSv1.2 (OUT), TLS header, Unknown (23):
> GET / HTTP/1.1
> Host: 192.168.10.188:5000
> User-Agent: curl/7.76.1
> Accept: */*
```

- **Placement**: El `endpoint` se puede acceder mediante `https://192.168.10.188:8778`. La figura 6.16 muestra el tráfico capturado por `Wireshark`. Se puede observar que hay paquetes con el protocolo `TLS` versión 1.3, por lo que están cifrados.

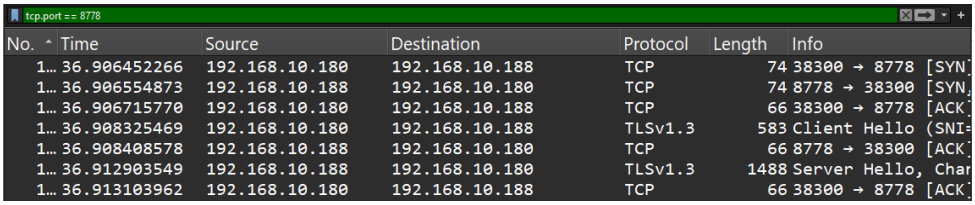

Figura 6.16: Análisis de los paquetes que trasmite y recibe Placement usando Wireshark.

- **Glance**: El `endpoint` se puede acceder mediante `https://192.168.10.200:9292`. La figura 6.17 muestra el tráfico capturado por `Wireshark`. Se puede observar que hay paquetes con el protocolo `TLS` versión 1.3, por lo que se encuentran cifrados.

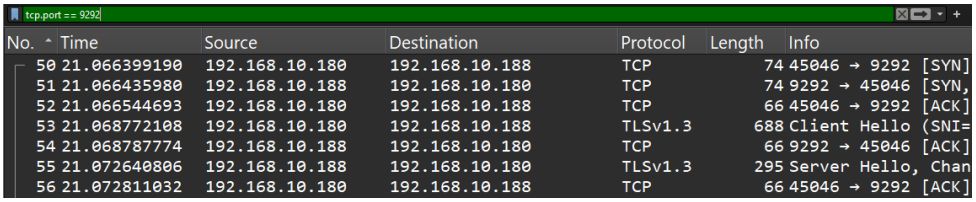

Figura 6.17: Análisis de los paquetes que trasmite y recibe Glance usando Wireshark.

- **Nova**: El `endpoint` se puede acceder mediante `https://192.168.10.188:8774`. La figura 6.18 muestra el tráfico capturado por `Wireshark`. Se puede observar que hay paquetes con el protocolo `TLS` versión 1.2, por lo que estos también se encuentran cifrados.

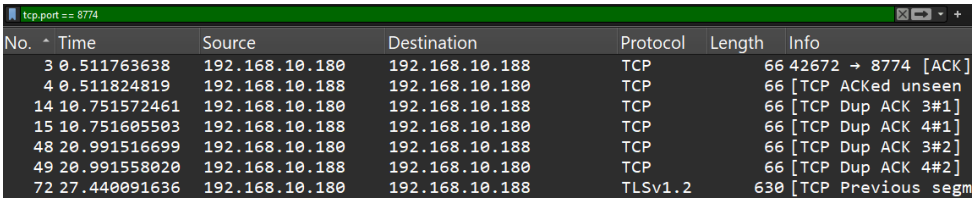

Figura 6.18: Análisis de los paquetes que trasmite y recibe Nova usando Wireshark.

6.3. Análisis y comparación de los resultados obtenidos

A continuación, se comparan los resultados obtenidos en el nodo `Controller` seguro e inseguro, y se identifican las diferencias junto con la capacidad de protección ante ataques.

6.3.1. Puertos expuestos

El nodo `Controller` inseguro tiene una mayor cantidad de puertos expuestos a diferencia del nodo `Controller` seguro. En la tabla 6.5. se muestran las diferencias entre los puertos de cada nodo, tomando únicamente los que son usados por `Openstack`.

Puerto	Módulo	Estado en el nodo Controller inseguro	Estado en el nodo Controller seguro
443-440	Horizon	Expuesto	Expuesto
5000	Keystone	Expuesto	Bloqueado
9292	Glance	Expuesto	Bloqueado
8774	Nova	Expuesto	Bloqueado
9696	Neutron	Expuesto	Bloqueado
8778	Placement	Expuesto	Bloqueado

Tabla 6.5: Tabla de estados de los puertos de ambos nodos Controller.

Es lógico que el puerto 440 esté accesible para cualquier cliente, ya que representa el medio principal para interactuar con la nube. Si este puerto se bloqueara, la funcionalidad de la nube perdería su propósito. En cambio, el resto de los puertos deben permanecer ocultos, ya que, podrían ser utilizados para explorar vulnerabilidades o llevar a cabo ataques más sofisticados. En situaciones excepcionales donde sea necesario mantener abierto el puerto 22, se pueden seguir las recomendaciones descritas en la Sección 2.11.

A través de la plataforma de cálculo de `CVSS V3.0` se determinó el grado de vulnerabilidad en el nodo `Controller`, tanto en su versión segura como insegura. A continuación se explica cómo se eligieron los valores:

- **Vector de ataque (AV)**: Dado que el ataque se efectuó vía Internet, se eligió la opción `Network`.
- **Complejidad del ataque (AC)**: Puesto que la herramienta `Nmap` no requiere ninguna credencial de acceso o una metodología compleja para poder escanear los puertos, se eligió la opción `Low`.
- **Privilegios requeridos (RP)**: Dado que no se requieren privilegios dentro del nodo atacado para escanear los puertos, se eligió la opción `None`.
- **Interacción con el usuario (UI)**: Dado que el ataque no requiere que el administrador de los nodos realice una acción específica para que `Nmap` detecte los puertos, se eligió la opción `None`.
- **Alcance (S)**: Dado que solo se usó `Nmap` para un escaneo de puertos, el alcance de este ataque es limitado y su uso no afecta directamente a otros vectores de ataque ni a otros sistemas, por lo que se eligió la opción `Unchanged`.
- **Integridad (I)**: La herramienta `Nmap` solo se usó para escanear puertos y no se cambió ninguna configuración dentro del nodo atacado, por lo que se eligió la opción `None`.
- **Disponibilidad (A)**: La herramienta `Nmap` no afectó el rendimiento ni el funcionamiento de los nodos atacados, por lo que se eligió la opción `None`.

Con respecto a la **Confidencialidad (C)**:

- `Nodo Controller inseguro`: Dado que este nodo expuso todos los puertos de los módulos y herramientas utilizados por `OpenStack`, se tuvo un impacto directo en la privacidad del servidor y de la nube al abrir posibles rutas de ataque. Por esta razón se eligió la opción `High`.
- `Nodo Controller seguro`: Dado que este nodo solo expuso el puerto necesario para el funcionamiento de la nube, la privacidad de esta no se vio afectada y por ello se eligió la opción `None`.

El nodo `Controller` inseguro mostró una vulnerabilidad con una puntuación de 7.5 (ver figura 6.19) lo que corresponde a un nivel alto en la escala.

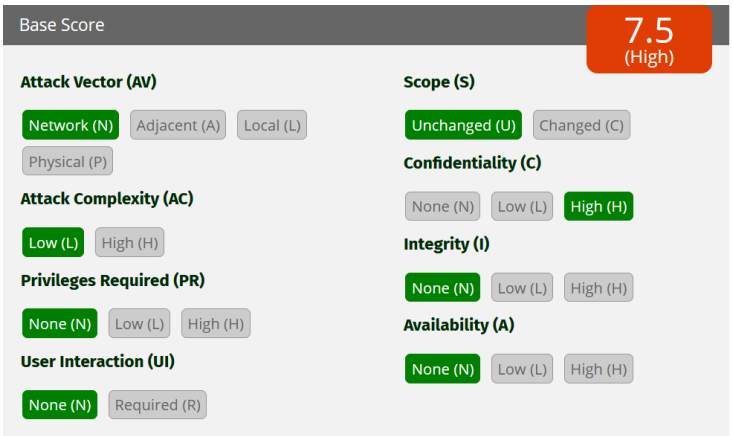

Figura 6.19: Puntaje CVSS V3.0 del escaneo de puertos en el nodo Controller inseguro.

En cambio, el análisis de puertos realizado al nodo `Controller` seguro mostró una vulnerabilidad con puntuación de 0 (ver figura 6.20), lo que corresponde al nivel más bajo de la escala.

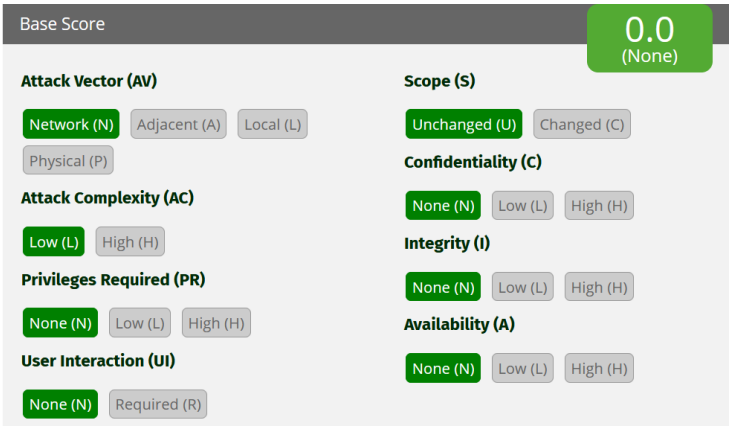

Figura 6.20: Puntaje CVSS V3.0 del escaneo de puertos en el nodo Controller seguro.

6.3.2. Ataques por fuerza bruta

El ataque por fuerza bruta realizado al servicio `SSH` en el nodo `Controller` inseguro fue exitoso en los casos en que se utilizaron contraseñas débiles y moderadamente seguras. En el caso del ataque al usuario con la contraseña segura, es altamente probable que el ataque hubiera tenido éxito si el `script` se hubiera ejecutado durante más tiempo.

Dado que se usaron contraseñas de cuatro caracteres de longitud, es posible determinar el número total de combinaciones posibles y, con ello, calcular el tiempo máximo que un atacante necesitaría para descubrir la contraseña del nodo. Esta información se encuentra en la tabla 6.6.

Tipo de contraseña	Número de intentos que le tomó al script descifrarla	Tiempo en segundos que tomó al script descifrarla	Número máximo de intentos que le tomaría al script descifrarla	Tiempo máximo en segundos que le tomaría al script descifrarla
Insegura	58,293	209,136	456,976	1,639,478
Medianamente segura	68,356	249,517	1,679,616	6,025,906
Segura	–	–	3,111,696	11,163,735

Tabla 6.6: Información del ataque al nodo Controller inseguro.

El ataque realizado en el nodo `Controller` inseguro obtuvo un resultado altamente favorable para el atacante, ya que el `script` descifró rápidamente las contraseñas de `root`, incluso en comparación con el tiempo máximo que habría requerido en el peor de los escenarios. Sin embargo, este no es un resultado positivo desde el punto de vista de la seguridad del servidor, ya que evidencia que el sistema es vulnerable y un atacante puede acceder.

Para determinar el grado de vulnerabilidad del servicio `SSH` se consideraron los siguientes valores para ambos nodos:

- **Vector de ataque (AV)**: Dado que el ataque provino de Internet usando la dirección IP del nodo atacado, se eligió la opción `Network`.
- **Complejidad del ataque (AC)**: Dado que no fue necesario realizar una acción compleja que vulnere al servicio `SSH` usando la herramienta `Hydra`, se eligió la opción `Low`.
- **Privilegios requeridos (RP)**: Dado que no se requieren privilegios dentro del nodo atacado con la herramienta `Hydra`, se eligió la opción `None`.

- **Interacción con el usuario (UI)**: Dado que el ataque no requiere que el administrador del nodo realice una acción específica para que la herramienta `Hydra` pueda utilizarse, se eligió la opción `None`.

Con respecto al **Alcance (S)**:

- `Nodo Controller inseguro`: Se logró acceder como `root`, lo que le permite vulnerar cualquier otro servicio dentro del nodo, por lo que se eligió la opción `Changed`.

- `Nodo Controller seguro`: Dado que el puerto 22 está bloqueado, este ataque no tuvo ningún alcance en él, por lo que se eligió la opción `Unchanged`.

Con respecto a la **Confidencialidad (C)**, **Integridad (I)** y **Disponibilidad (A)**:

- `Nodo Controller inseguro`: Dado que se logró acceder al nodo como `root`, la privacidad del nodo se ve severamente afectada, ya que se tiene acceso total a cualquier archivo con información personal o de configuración. Además, al tener privilegios de `root`, se puede modificar cualquier configuración del sistema, dañando gravemente su integridad. También, se tiene la capacidad de apagar el nodo, lo que compromete totalmente la disponibilidad del sistema. Por estos motivos, el valor elegido para los tres parámetros fue `High`.

- `Nodo Controller seguro`: Dado que no se pudo acceder al nodo a través de `SSH`, la confidencialidad, integridad y disponibilidad del nodo permanecieron intactas, por lo que el valor elegido para los tres parámetros fue `None`.

Para el servicio `SSH` dentro del nodo `Controller` inseguro se obtuvo una puntuación de 10 (ver figura 6.21), por lo que es considerado un punto vulnerable crítico en el servidor.

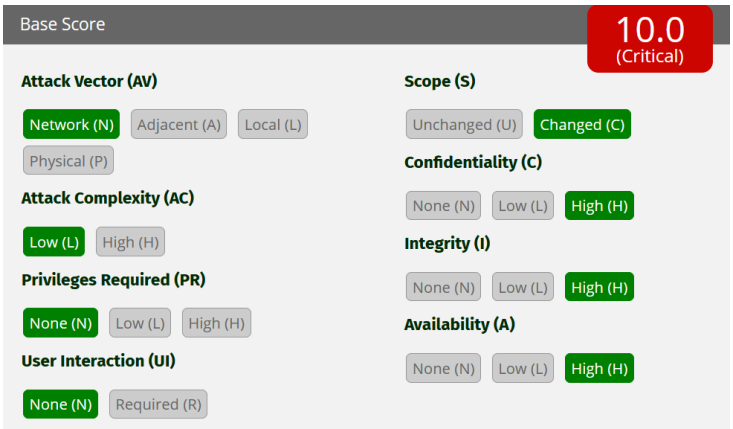

Figura 6.21: Puntaje CVSS V3.0 del servicio SSH en el nodo Controller inseguro.

Para el caso donde el `firewall` del nodo `Controller` seguro bloquea el puerto 22, se tiene una puntuación de 0 (ver figura 6.22), por lo que la vulnerabilidad del servicio `SSH` es nula.

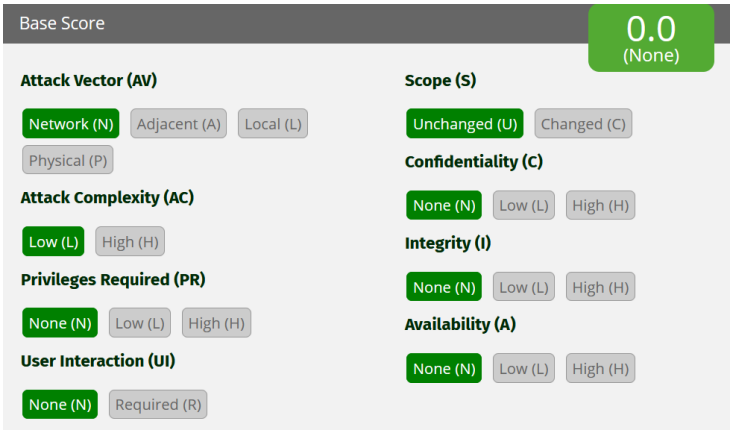

Figura 6.22: Puntaje CVSS V3.0 del servicio SSH en el nodo Controller seguro con bloqueo al puerto 22.

Suponiendo un escenario donde el puerto 22 está abierto, y un atacante llegara a entrar al sistema por un descuido de un usuario. El valor del parámetro **Interacción con el usuario (UI)** cambiaría a `Required`. En este supuesto, la **Confidencialidad (C)** e **Integridad (I)** cambiarían de valor a `Low` ya que aún si logran acceder al servidor por `SSH`, este sería un usuario común, y no tendría privilegios de `root` debido a las configuraciones iniciales del servicio `SSH`, reduciendo el impacto que podría tener este ataque al servidor. Más aún, a pesar de que el atacante pudiera acceder y modificar archivos, este no podría interactuar directamente con configuraciones críticas del servidor, ni de `OpenStack`. Para este escenario, se obtiene una puntuación de 5.4 (ver figura 6.23), la cual se considera una vulnerabilidad media en la escala.

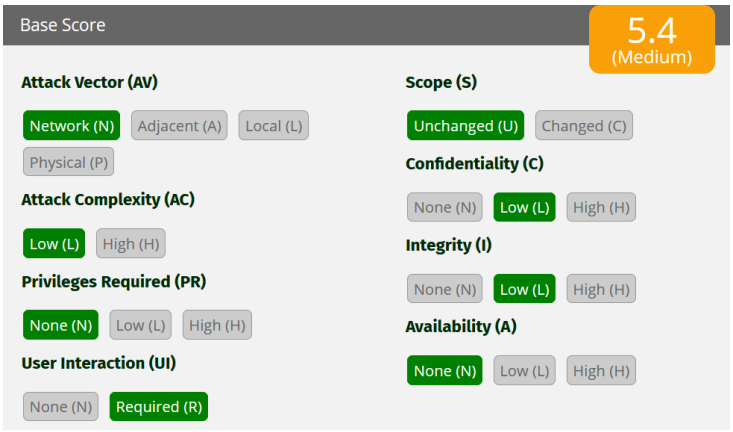

Figura 6.23: Puntaje CVSS V3.0 del servicio SSH en el nodo Controller seguro sin bloqueo al puerto 22.

6.3.3. Seguridad en la interfaz web y comunicaciones de los módulos de OpenStack

En cuanto a la página web `Horizon`, esta demostró ser una interfaz muy segura, ya que no expone rutas ocultas con información sensible, y es capaz de detectar ataques por fuerza bruta realizados con herramientas como `Hydra`, bloqueando al atacante de manera efectiva.

Para determinar el grado de vulnerabilidad en `Horizon`, se consideraron los siguientes valores para ambos nodos:

- **Vector de ataque (AV)**: Dado que el ataque provino de Internet usando la dirección IP del nodo atacado, se eligió la opción `Network`.

- **Complejidad del ataque (AC)**: Dado que no fue necesario realizar una acción compleja que vulnere al servicio `Horizon` usando `Gobuster`, se eligió la opción `Low`.
- **Privilegios requeridos (RP)**: Dado que no se requieren privilegios dentro del nodo atacado para ejecutar `Gobuster`, se eligió la opción `None`.
- **Interacción con el usuario (UI)**: Dado que el ataque no requiere que el administrador del nodo realice una acción específica en `Horizon` para que la herramienta `Gobuster` pueda realizar la búsqueda de rutas, se eligió la opción `None`.
- **Alcance (S)**: Dado que solo se usó `Gobuster` para la búsqueda de rutas ocultas en `Horizon`, el alcance de este ataque es limitado y su uso no afectó directamente a otros servicios, por lo que se eligió la opción `Unchanged`.
- **Confidencialidad (C)**: A pesar de que se encontraron rutas ocultas, ninguna de ellas expuso datos sensibles sobre la plataforma ni sobre los usuarios, por lo que se eligió la opción `None`.
- **Integridad (I)**: La herramienta `Gobuster` solo permite realizar búsqueda de rutas, por lo que no hay manera de que esta búsqueda pudiese cambiar o modificar parámetros o configuraciones de `Horizon`, por lo que se eligió la opción `None`.
- **Disponibilidad (A)**: La herramienta `Gobuster` no afectó el rendimiento ni el funcionamiento de los nodos, por lo que se eligió la opción `None`.

La puntuación `CVSS V3.0` obtenida para el servicio de `Horizon` en el nodo `Controller` inseguro y seguro, a partir de las pruebas realizadas es de `0` (ver figura 6.24), lo que corresponde al nivel más bajo de la escala.

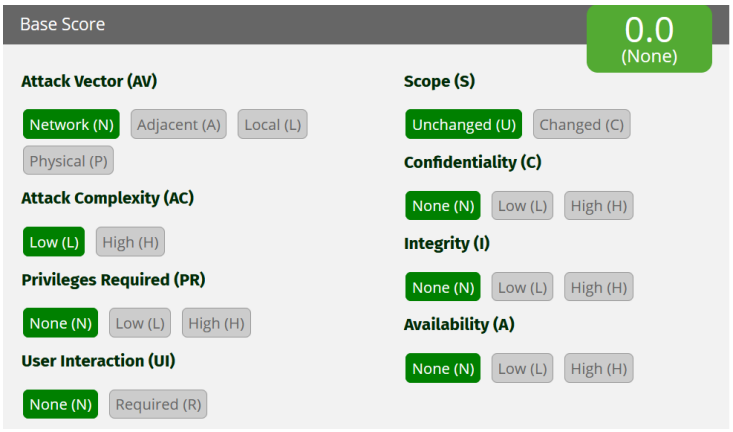

Figura 6.24: Puntaje CVSS V3.0 del servicio Horizon en el nodo Controller inseguro.

Finalmente, la implementación de los `endpoints` con `HTTPS` resultó ser una medida altamente beneficiosa, ya que garantiza que la información intercambiada entre los módulos esté cifrada y sea inaccesible para clientes externos/internos no autorizados. Esto es especialmente importante, ya que en el caso del nodo `Controller` inseguro, cualquier persona dentro o fuera del nodo puede capturar paquetes y acceder a información sensible del servidor. Esto ocurre porque los `endpoints` utilizados en ese nodo operan bajo `HTTP`, un protocolo que no ofrece cifrado, dejando expuesta la comunicación.

Para determinar el grado de vulnerabilidad de los `endpoints` de `OpenStack` se consideraron los siguientes valores para ambos nodos:

- **Complejidad del ataque (AC)**: Dado que no fue necesario realizar una acción compleja para que `Wireshark` pudiera escanear los paquetes enviados a los módulos, se eligió la opción `Low`.

- **Interacción con el usuario (UI)**: Dado que el ataque no requirió que el administrador de los nodos realizara una acción específica en los `endpoints` para que la herramienta `Wireshark` pudiera capturar y analizar paquetes, se eligió la opción `None`.
- **Alcance (S)**: Dado que `Wireshark` solo puede usarse para capturar y analizar paquetes, el alcance de este ataque es muy bajo por lo que se eligió la opción `Unchanged`.
- **Integridad (I)**: La herramienta `Wireshark` no tiene la capacidad de modificar ni retransmitir los paquetes que captura, por ello la integridad de estos no se ve afectada, por lo que se eligió la opción `None`.
- **Disponibilidad (A)**: La herramienta `Wireskarh` está diseñada únicamente para escuchar y no puede afectar la disponibilidad del servidor, por lo que se eligió la opción `None`.

Con respecto al **Vector de ataque (AV)**:

- `Nodo Controller inseguro`: Dado que la captura y análisis de paquetes se realizó de forma externa a través de Internet, se eligió la opción `Network`.
- `Nodo Controller seguro`: Dado que la única forma de comunicarse y observar los `endpoints` en este nodo es estando dentro del servidor como un usuario, se eligió la opción `Local`.

Con respecto a los **Privilegios requeridos (RP)**:

- `Nodo Controller inseguro`: Dado que no se requiere de ningún privilegio para poder usar `Wireshark` desde afuera del servidor para capturar y analizar paquetes, se eligió la opción `None`.
- `Nodo Controller seguro`: Si el ataque es de forma interna, el usuario del servidor requiere tener ciertos privilegios para usar `Wireshark`, sin embargo, no se requiere ser `root` para esto, por ello se escogió la opción `Low`.

Con respecto a la **Confidencialidad (C)**:

- `Nodo Controller inseguro`: Dado que la información que se capturó usando `Wireshark` no está cifrada, es posible observar todo el contenido de los paquetes en texto plano, por lo que la confidencialidad de los datos se ve totalmente comprometida, por ello se eligió la opción `High`.
- `Nodo Controller seguro`: A pesar de que fue posible capturar paquetes enviados usando `Wireshark`, la información obtenida de ellos está cifrada, por ende, la privacidad de la comunicación de los módulos no se vio comprometida, y por ello se eligió la opción `None`.

Para el caso de los `endpoints`, la puntuación `CVSS V3.0` obtenida del nodo `Controller` inseguro es de 7.5 (ver figura 6.25), lo que indica un nivel de vulnerabilidad alto en la escala.

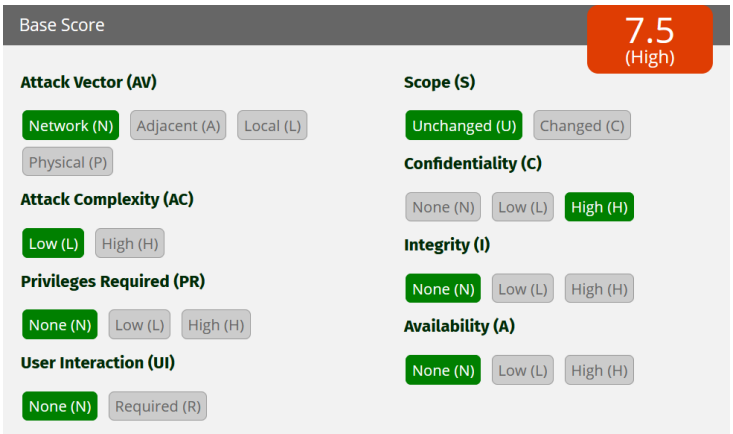

Figura 6.25: Puntaje CVSS V3.0 de los endpoints del nodo Controller inseguro.

Por otro lado, para los `endpoints` del nodo `Controller` seguro obtuvieron una puntuación de 0 (vef figura 6.26), lo que se considera un nivel nulo de vulnerabilidad.

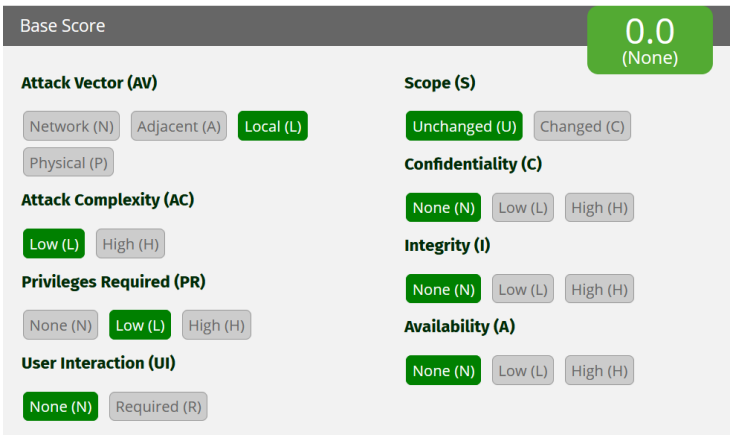

Figura 6.26: Puntaje CVSS V3.0 de los endpoints del nodo Controller seguro.

Capítulo 7

Conclusiones

En este capítulo se presentan las conclusiones generales y trabajos futuros.

7.1. Conclusiones generales

Esta tesis tuvo como objetivo principal realizar un análisis de seguridad en una nube privada. Las soluciones propuestas en el Capítulo 3 se llevaron a cabo en el Capítulo 4 y, posteriormente fueron evaluadas en el Capítulo 6. Como resultado, se determinó que las mejoras implementadas son funcionales y efectivas, logrando un incremento en la seguridad de la nube privada. Se puede afirmar que las implementaciones de seguridad fueron correctamente añadidas y probadas, ya que los servicios de `OpenStack` operan de manera normal; sin embargo, ahora todas las comunicaciones están cifradas.

La nube privada propuesta en esta tesis representa una opción accesible para su implementación, ya que los programas utilizados son de código abierto y los programas creados son compatibles con la mayoría de las distribuciones Linux.

Las mejoras de seguridad implementadas en este trabajo ofrecen un servicio de nube privada, significativamente más seguro que en su configuración inicial. Esto representa una ventaja importante para empresas o instituciones de educación superior que utilicen o planeen utilizar una nube privada orquestada por `OpenStack`. Además de los beneficios inherentes de contar con una nube privada, podrán tener la certeza de que la información almacenada está más protegida.

Durante las pruebas de seguridad realizadas, se concluyó que las mejoras implementadas tuvieron un impacto significativo y positivo en la seguridad de la nube privada y de los servidores donde está alojada. Esto se evidenció en el escaneo inicial de puertos realizado con `Nmap`, donde el servidor propuesto en esta tesis mostró únicamente el puerto de `Horizon`, en contraste con la nube sin medidas de seguridad, que exponía puertos altamente vulnerables, como el puerto 22.

Al exponer únicamente el puerto 440, la nube privada disminuye la posibilidad de un ataque. En caso de que el puerto 22 (`SSH`) tuviese que estar abierto, los `scripts` implementados para la detección de ataques de fuerza bruta logran mitigar este tipo de ataques de forma efectiva. Las configuraciones predeterminadas de `RHEL 9` para el servicio `SSH`, junto con las herramientas de protección como `CSRF` integradas en `Horizon`, también contribuyen significativamente al fortalecimiento de la seguridad de los servicios.

Es importante mencionar que, a pesar de que los `scripts` desarrollados para detectar y detener ataques de fuerza bruta son muy útiles, la mejor manera de aumentar la seguridad del sistema es utilizando contraseñas seguras. Las contraseñas deben tener al menos 8 caracteres e incluir letras mayúsculas, minúsculas, números y caracteres especiales. Este nivel de complejidad genera más de 6.1 billones de combinaciones posibles, lo que dificulta considerablemente su descifrado y prolonga el tiempo requerido por los atacantes para comprometerlas, fortaleciendo así la seguridad del sistema.

Asimismo, se puede decir con certeza que las comunicaciones entre los módulos de `OpenStack`

son significativamente más seguras que en su configuración inicial. Esto se debe a la implementación de certificados SSL/TLS en cada endpoint en OpenStack, lo que permite cifrar la información transmitida y, a su vez, reduce el riesgo de exponer datos sensibles. El uso de certificados autofirmados representó un desafío para lograr el cifrado de los endpoints en OpenStack, ya que estos certificados suelen ser considerados inseguros. Sin embargo, en una implementación de producción, se debe optar por certificados de una autoridad certificadora reconocida, lo que elimina este inconveniente y garantiza un nivel de seguridad más alto y confiable.

Otra mejora implementada, y no menos importante, fue la creación de imágenes seguras en Ubuntu dentro de OpenStack para el uso de los clientes. Estas máquinas virtuales pueden responder de manera más efectiva ante ataques que inicialmente habrían puesto en riesgo al cliente. Esto garantiza una protección adicional al asegurar que las máquinas virtuales estén preparadas para mitigar posibles amenazas desde el principio.

Con base en las puntuaciones CVSS V3.0 calculadas para los nodos Controller inseguro y seguro se observó una reducción significativa en las vulnerabilidades analizadas. Los puertos del servidor, que inicialmente presentaban un puntaje de 7.5 (alto), disminuyeron a 0 (nulo). El servicio SSH, con un puntaje inicial de 10 (crítico), se redujo a 0 (nulo) cuando el puerto 22 fue bloqueado por el firewall, y a 5.4 (medio) en caso de que el puerto 22 permaneciera abierto. Por último, el servicio de Horizon demostró tener un valor de 0 (nulo) para ambos servidores analizados.

La plataforma OpenStack proporciona medidas básicas de seguridad en su configuración inicial, a pesar de ello, aún existen áreas susceptibles de mejora, como el cifrado de los endpoints que se realizó en el presente trabajo. Las mejoras más significativas se centraron en el servidor donde se aloja la nube, ya que este suele ser el objetivo principal de los atacantes y presenta un mayor número de vulnerabilidades en comparación con la plataforma OpenStack. No obstante, las medidas implementadas garantizan que el servidor no quede expuesto ni en riesgo, proporcionando una capa adicional de protección frente a posibles amenazas.

7.2. Trabajo futuro

Las implementaciones realizadas en el presente trabajo tenían como objetivo específico evitar la exposición de datos sensibles y prevenir los ataques de fuerza bruta a la página web de Horizon. Sin embargo, estas no son las únicas amenazas cibernéticas a las que nuestra nube o servidor están expuestos. Por ejemplo, es posible proteger la nube contra un ataque de denegación de servicio (DDoS), donde se utiliza una red de bots (zombis) para enviar tráfico desde múltiples puntos con el fin de sobrecargar el servidor.

Además, también se puede implementar la herramienta SELinux disponible en RHEL 9, la cual ayudaría al servidor a restringir el acceso a archivos, servicios y comandos, evitando que las aplicaciones maliciosas o los procesos con privilegios limitados puedan realizar acciones indebidas. SELinux también puede prevenir intentos de escalamiento de privilegios al restringir las capacidades de los procesos y las aplicaciones, asegurando que solo los servicios autorizados puedan realizar tareas administrativas o modificar configuraciones críticas. Esta medida de seguridad es crucial para proteger la infraestructura de la nube, manteniéndola a salvo de amenazas y ataques más sofisticados.

Finalmente, se podrían analizar los impactos que las implementaciones realizadas tuvieron en el desempeño de la nube. Por ejemplo, se podría realizar un análisis de latencia para determinar si la migración de los endpoints de HTTP a HTTPS provocó un aumento considerable en el tiempo de comunicación entre ellos.

Apéndice A

Archivos de configuración de Openstack

Este apéndice contiene todos los archivos de configuración de los módulos de OpenStack que fueron modificados.

A.1. Nodo Controller

Para el nodo Controller se tienen los archivos de configuración que se muestran a continuación.

A.1.1. /etc/httpd/conf/httpd.conf

En la consola A.1 se muestra el archivo de configuración del servicio de httpd.

```
ServerRoot "/etc/httpd"
Listen 80
Listen 440
Include conf.modules.d/*.conf
LoadModule ssl_module modules/mod_ssl.so
User apache
Group apache
ServerAdmin root@localhost
ServerName controller
<Directory />
    AllowOverride none
    Require all granted
</Directory>
<Directory /usr/bin>
    Require all granted
</Directory>
DocumentRoot "/var/www/html"
<Directory "/var/www">
    AllowOverride None
    Require all granted
</Directory>
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
<Files ".ht*">
    Require all granted
</Files>
ErrorLog "logs/error_log"
```

```
LogLevel warn
<IfModule log_config_module>
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    <IfModule logio_module>
        LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \
        \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>
    CustomLog "logs/access_log" combined
</IfModule>
<IfModule alias_module>
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
</IfModule>
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>
<IfModule mime_module>
    TypesConfig /etc/mime.types
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
    AddType text/html .shtml
    AddOutputFilter INCLUDES .shtml
</IfModule>
AddDefaultCharset UTF-8
<IfModule mime_magic_module>
    MIMEMagicFile conf/magic
</IfModule>
EnableSendfile on
IncludeOptional conf.d/*.conf
```

Consola A.1: Archivo /etc/httpd/conf/httpd.conf.

A.1.2. /etc/httpd/conf.d/openstack-dashboard.conf

En la consola A.2 se muestra el archivo de configuración del dashboard de `Horizon`.

```
WSGISocketPrefix run/wsgi
<VirtualHost *:440>
  SSLEngine On
  SSLCertificateFile /etc/httpd/ssl/horizon/horizon.crt
  SSLCertificateKeyFile /etc/httpd/ssl/horizon/horizon.key
  SSLCACertificateFile /etc/httpd/ssl/horizon/cah.crt
  WSGIDaemonProcess dashboard
  WSGIProcessGroup dashboard
  WSGIApplicationGroup %{GLOBAL}
  WSGIScriptAlias / /usr/share/openstack-dashboard/openstack_dashboard/wsgi.py
  Alias /static /usr/share/openstack-dashboard/static
  <Directory /usr/share/openstack-dashboard/openstack_dashboard>
    Options All
    AllowOverride All
    Require all granted
  </Directory>
  <Directory /usr/share/openstack-dashboard/static>
    Require all granted
  </Directory>
</VirtualHost>
```

Consola A.2: Archivo /etc/httpd/conf.d/openstack-dashboard.conf.

A.1.3. /etc/openstack-dashboard/local_settings

En la consola A.3 se muestra el archivo de configuración del `dashboard` de `Horizon`.

```
import os
from django.utils.translation import gettext_lazy as _
```

```
from openstack_dashboard.settings import HORIZON_CONFIG
DEBUG = False
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 3,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': False,
    'enable_quotas': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_fip_topology_check': False,
}
ALLOWED_HOSTS = ['*']
USE_SSL=True
OPENSTACK_SSL_NO_VERIFY=True
LOCAL_PATH = '/tmp'
SECRET_KEY='b6bea8c4213093214049'
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '192.168.10.188:11211',
    },
}
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
OPENSTACK_HOST = "controller"
OPENSTACK_KEYSTONE_URL = "https://%s:5000/identity/v3" % OPENSTACK_HOST
TIME_ZONE = "UTC"
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'console': {
            'format': '%(levelname)s %(name)s %(message)s'
        },
        'operation': {
            'format': '%(message)s'
        },
    },
    'handlers': {
        'null': {
            'level': 'DEBUG',
            'class': 'logging.NullHandler',
        },
        'console': {
            'level': 'DEBUG' if DEBUG else 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'console',
        },
        'operation': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'operation',
        },
    },
    'loggers': {
        'horizon': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'horizon.operation_log': {
            'handlers': ['operation'],
            'level': 'INFO',
            'propagate': False,
        },
```

```
74          'openstack_dashboard': {
75              'handlers': ['console'],
76              'level': 'DEBUG',
77              'propagate': False,
78          },
79          'novaclient': {
80              'handlers': ['console'],
81              'level': 'DEBUG',
82              'propagate': False,
83          },
84          'cinderclient': {
85              'handlers': ['console'],
86              'level': 'DEBUG',
87              'propagate': False,
88          },
89          'keystoneauth': {
90              'handlers': ['console'],
91              'level': 'DEBUG',
92              'propagate': False,
93          },
94          'keystoneclient': {
95              'handlers': ['console'],
96              'level': 'DEBUG',
97              'propagate': False,
98          },
99          'glanceclient': {
100             'handlers': ['console'],
101             'level': 'DEBUG',
102             'propagate': False,
103         },
104         'neutronclient': {
105             'handlers': ['console'],
106             'level': 'DEBUG',
107             'propagate': False,
108         },
109         'swiftclient': {
110             'handlers': ['console'],
111             'level': 'DEBUG',
112             'propagate': False,
113         },
114         'oslo_policy': {
115             'handlers': ['console'],
116             'level': 'DEBUG',
117             'propagate': False,
118         },
119         'openstack_auth': {
120             'handlers': ['console'],
121             'level': 'DEBUG',
122             'propagate': False,
123         },
124         'django': {
125             'handlers': ['console'],
126             'level': 'DEBUG',
127             'propagate': False,
128         },
129         'django.template': {
130             'handlers': ['console'],
131             'level': 'INFO',
132             'propagate': False,
133         },
134         'django.db.backends': {
135             'handlers': ['null'],
136             'propagate': False,
137         },
138         'requests': {
139             'handlers': ['null'],
140             'propagate': False,
141         },
142         'urllib3': {
143             'handlers': ['null'],
144             'propagate': False,
```

```
        },
        'chardet.charsetprober': {
            'handlers': ['null'],
            'propagate': False,
        },
        'iso8601': {
            'handlers': ['null'],
            'propagate': False,
        },
        'scss': {
            'handlers': ['null'],
            'propagate': False,
        },
    },
}
SECURITY_GROUP_RULES = {
    'all_tcp': {
        'name': _('All TCP'),
        'ip_protocol': 'tcp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_udp': {
        'name': _('All UDP'),
        'ip_protocol': 'udp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_icmp': {
        'name': _('All ICMP'),
        'ip_protocol': 'icmp',
        'from_port': '-1',
        'to_port': '-1',
    },
    'ssh': {
        'name': 'SSH',
        'ip_protocol': 'tcp',
        'from_port': '22',
        'to_port': '22',
    },
    'smtp': {
        'name': 'SMTP',
        'ip_protocol': 'tcp',
        'from_port': '25',
        'to_port': '25',
    },
    'dns': {
        'name': 'DNS',
        'ip_protocol': 'tcp',
        'from_port': '53',
        'to_port': '53',
    },
    'http': {
        'name': 'HTTP',
        'ip_protocol': 'tcp',
        'from_port': '80',
        'to_port': '80',
    },
    'pop3': {
        'name': 'POP3',
        'ip_protocol': 'tcp',
        'from_port': '110',
        'to_port': '110',
    },
    'imap': {
        'name': 'IMAP',
        'ip_protocol': 'tcp',
        'from_port': '143',
        'to_port': '143',
    },
    'ldap': {
```

```
        'name': 'LDAP',
        'ip_protocol': 'tcp',
        'from_port': '389',
        'to_port': '389',
    },
    'https': {
        'name': 'HTTPS',
        'ip_protocol': 'tcp',
        'from_port': '440',
        'to_port': '440',
    },
    'smtps': {
        'name': 'SMTPS',
        'ip_protocol': 'tcp',
        'from_port': '465',
        'to_port': '465',
    },
    'imaps': {
        'name': 'IMAPS',
        'ip_protocol': 'tcp',
        'from_port': '993',
        'to_port': '993',
    },
    'pop3s': {
        'name': 'POP3S',
        'ip_protocol': 'tcp',
        'from_port': '995',
        'to_port': '995',
    },
    'ms_sql': {
        'name': 'MS SQL',
        'ip_protocol': 'tcp',
        'from_port': '1433',
        'to_port': '1433',
    },
    'mysql': {
        'name': 'MYSQL',
        'ip_protocol': 'tcp',
        'from_port': '3306',
        'to_port': '3306',
    },
    'rdp': {
        'name': 'RDP',
        'ip_protocol': 'tcp',
        'from_port': '3389',
        'to_port': '3389',
    },
}
```

Consola A.3: Archivo /etc/openstack-dashboard/local_settings.

A.1.4. /etc/httpd/conf.d/wsgi-keystone.conf

En la consola A.4 se muestra el archivo de configuración de `WSGI` en el servicio `httpd`.

```
Listen 5000
<VirtualHost *:5000>
    SSLEngine On
    SSLCertificateFile /etc/httpd/ssl/keystone/keystone.crt
    SSLCertificateKeyFile /etc/httpd/ssl/keystone/keystone.key
    SSLCACertificateFile /etc/httpd/ssl/keystone/cak.crt
    WSGIDaemonProcess keystone-public processes=5 threads=1 \
    user=keystone group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    LimitRequestBody 114688
    <IfVersion >= 2.4>
```

```
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    ErrorLog /var/log/httpd/keystone.log
    CustomLog /var/log/httpd/keystone_access.log combined
    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>
Alias /identity /usr/bin/keystone-wsgi-public
<Location /identity>
    SetHandler wsgi-script
    Options +ExecCGI
    WSGIProcessGroup keystone-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
</Location>
```

Consola A.4: Archivo /etc/httpd/conf.d/wsgi-keystone.conf.

A.1.5. /etc/keystone/keystone.conf

En la consola A.5 se muestra el archivo de configuración de `Keystone`.

```
[DEFAULT]
[application_credential]
[assignment]
[auth]
[cache]
[catalog]
[cors]
[credential]
[database]
connection = mysql+pymysql://keystone:abc123@controller/keystone? \
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key
[domain_config]
[endpoint_filter]
[endpoint_policy]
[eventlet_server]
[federation]
[fernet_receipts]
[fernet_tokens]
[healthcheck]
[identity]
[identity_mapping]
[jwt_tokens]
[ldap]
[memcache]
[oauth1]
[oauth2]
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[policy]
[profiler]
[receipt]
[resource]
[revoke]
[role]
```

```
[saml]
[security_compliance]
[shadow_users]
[ssl]
certfile = /etc/httpd/ssl/keystone/keystone.crt
keyfile = /etc/httpd/ssl/keystone/keystone.key
ca_certs = /etc/httpd/ssl/keystone/cak.crt
[token]
provider = fernet
[tokenless_auth]
[totp]
[trust]
[unified_limit]
[wsgi]
```

Consola A.5: Archivo /etc/keystone/keystone.conf.

A.1.6. /etc/glance/glance-api.conf

En la consola A.6 se muestra el archivo de configuración de `Glance`.

```
[DEFAULT]
bind_host=127.0.0.1
public_endpoint=https://controller:9292
[barbican]
[barbican_service_user]
[cinder]
[cors]
[database]
connection = mysql+pymysql://glance:abc123@controller/glance? \
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key
[file]
[glance.store.http.store]
[glance.store.rbd.store]
[glance.store.s3.store]
[glance.store.swift.store]
[glance.store.vmware_datastore.store]
[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
[healthcheck]
[image_format]
[key_manager]
[keystone_authtoken]
www_authenticate_uri  = https://controller:5000
auth_url = https://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = abc123
insecure=true
[os_brick]
[oslo_concurrency]
[oslo_limit]
auth_url = https://controller:5000
auth_type = password
user_domain_id = default
username = glance
system_scope = all
password = abc123
endpoint_id = 340be3625e9b4239a6415d034e98aace
region_name = RegionOne
insecure=true
```

```
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[paste_deploy]
flavor = keystone
[profiler]
[store_type_location_strategy]
[task]
[taskflow_executor]
[vault]
[wsgi]
```

Consola A.6: Archivo /etc/glance/glance-api.conf.

A.1.7. /etc/placement/placement.conf

En la consola A.7 se muestra el archivo de configuración de `Placement`.

```
[DEFAULT]
[api]
auth_strategy = keystone
enable_ssl_api = true
[cors]
[keystone_authtoken]
auth_url = https://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = abc123
[oslo_middleware]
[oslo_policy]
[placement]
[placement_database]
connection = mysql+pymysql://placement:abc123@controller/placement? \
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key

[profiler]
```

Consola A.7: Archivo /etc/placement/placement.conf.

A.1.8. /etc/nova/nova.conf

En la consola A.8 se muestra el archivo de configuración de `Nova`.

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
transport_url = rabbit://openstack:abc123@controller:5672/
my_ip = 192.168.10.10
ssl_only=true
enabled_ssl_apis = osapi_compute
[api]
auth_strategy = keystone
[api_database]
connection = mysql+pymysql://nova:abc123@controller/nova_api? \
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key
[barbican]
```

```
[barbican_service_user]
[cache]
[cinder]
[compute]
[conductor]
[console]
[consoleauth]
[cors]
[cyborg]
[database]
connection = mysql+pymysql://nova:abc123@controller/nova? \
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key
[devices]
[ephemeral_storage_encryption]
[filter_scheduler]
[glance]
api_servers = https://controller:9292
[guestfs]
[healthcheck]
[hyperv]
[image_cache]
[ironic]
[key_manager]
[keystone]
[keystone_authtoken]
www_authenticate_uri = https://controller:5000/
auth_url = https://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = abc123
insecure=true
[libvirt]
virt_type = kvm
[metrics]
[mks]
[neutron]
auth_url = https://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = abc123
service_metadata_proxy = true
metadata_proxy_shared_secret = abc123
[notifications]
[os_vif_linux_bridge]
[os_vif_ovs]
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[pci]
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
```

```
auth_url = https://controller:5000/v3
username = placement
password = abc123
insecure=true
[privsep]
[profiler]
[quota]
[rdp]
[remote_debug]
[scheduler]
[serial_console]
[service_user]
send_service_user_token = true
auth_url = https://controller/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = abc123
[spice]
[upgrade_levels]
[vault]
[vendordata_dynamic_auth]
[vmware]
[vnc]
enabled = true
server_listen = $my_ip
server_proxyclient_address = $my_ip
[workarounds]
[wsgi]
ssl_cert_file = /etc/httpd/ssl/nova/nova.crt
ssl_key_file = /etc/httpd/ssl/nova/nova.key
[zvm]
```

Consola A.8: Archivo /etc/nova/nova.conf.

A.1.9. /etc/neutron/neutron.conf

En la consola A.9 se muestra el archivo de configuración de `Neutron`.

```
[DEFAULT]
core_plugin = ml2
service_plugins = router
transport_url = rabbit://openstack:abc123@controller
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
bind_host = 0.0.0.0
bind_port = 9696
use_ssl = True
ssl_cert_file=/etc/httpd/ssl/neutron/neutron.crt
ssl_key_file=/etc/httpd/ssl/neutron/neutron.key
service_plugins = firewall_v2
[service_providers]
[service_providers]
service_provider = FIREWALL_V2:fwaas_db:neutron_fwaas.services.firewall\
.service_drivers.agents.agents.FirewallAgentDriver:default
[fwaas]
agent_version = v2
driver = neutron_fwaas.services.firewall.service_drivers.agents.drivers.linux.\
    iptables_fwaas_v2.IptablesFwaasDriver
enabled = True
[agent]
[cache]
[cors]
[database]
connection = mysql+pymysql://neutron:abc123@controller/neutron? \
```

```
    ssl_ca=/etc/httpd/ssl/sql/sql.pem& \
    ssl_cert=/etc/httpd/ssl/sql/sql.pem& \
    ssl_key=/etc/httpd/ssl/sql/sql.key
[designate]
[experimental]
[healthcheck]
[ironic]
[keystone_authtoken]
www_authenticate_uri = https://controller:5000
auth_url = https://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = abc123
[nova]
auth_url = https://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = abc123
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[placement]
[privsep]
[profiler]
[quotas]
[ssl]
```

Consola A.9: Archivo /etc/neutron/neutron.conf.

A.1.10. /etc/nginx/nginx.conf

En la consola A.10 se muestra el archivo de configuración de `nginx`.

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
include /usr/share/nginx/modules/*.conf;
events {
    worker_connections 1024;
}
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;
    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;
    include /etc/nginx/conf.d/*.conf;
}
```

```
stream {
    upstream glance-api {
        server 127.0.0.1:9292;
    }
    server {
        listen 192.168.10.188:9292 ssl;
        proxy_pass glance-api;
        ssl_certificate "/etc/httpd/ssl/glance/glance.pem";
        ssl_certificate_key "/etc/httpd/ssl/glance/glance.key";
        #ssl_trusted_certificate "/etc/httpd/ssl/glance/glance.pem";
        ssl_protocols TLSv1.2 TLSv1.3;
    }
}
```

Consola A.10: Archivo /etc/nginx/nginx.conf.

A.1.11. /etc/httpd/conf.d/00-placement-api.conf

En la consola A.11 se muestra el archivo de configuración de `Placement` en el servicio `httpd`.

```
Listen 8778
<VirtualHost *:8778>
  WSGIProcessGroup placement-api
  WSGIApplicationGroup %{GLOBAL}
  WSGIPassAuthorization On
  WSGIDaemonProcess placement-api processes=3 threads=1 user=placement group=placement
  WSGIScriptAlias / /usr/bin/placement-api
  <IfVersion >= 2.4>
    ErrorLogFormat "%M"
  </IfVersion>
  ErrorLog /var/log/placement/placement-api.log
  SSLEngine On
  SSLCertificateFile /etc/httpd/ssl/placement/placement.crt
  SSLCertificateKeyFile /etc/httpd/ssl/placement/placement.key
  SSLCACertificateFile /etc/httpd/ssl/placement/cap.crt
  <Directory /var/www/cgi-bin/placement-api>
        Require all granted
  </Directory>
</VirtualHost>
Alias /placement-api /usr/bin/placement-api
<Location /placement-api>
  SetHandler wsgi-script
  Options +ExecCGI
  WSGIProcessGroup placement-api
  WSGIApplicationGroup %{GLOBAL}
  WSGIPassAuthorization On
</Location>
```

Consola A.11: Archivo /etc/httpd/conf.d/00-placement-api.conf.

A.1.12. /etc/my.cnf

En la consola A.12 se muestra el archivo de configuración de `MySQL`.

```
[mysqld]
ssl-cert = /etc/httpd/ssl/sql/sql.pem
ssl-key = /etc/httpd/ssl/sql/sql.key
ssl-ca = /etc/httpd/ssl/sql/sql.pem
[client-server]
!includedir /etc/my.cnf.d
```

Consola A.12: Archivo /etc/my.cnf.

A.1.13. /etc/rabbitmq/rabbitmq.conf

En la consola A.13 se muestra el archivo de configuración de RabbitMQ.

```
listeners.ssl.default = 5671
ssl_options.cacertfile =/etc/httpd/ssl/rabbit/cara.crt
ssl_options.certfile =/etc/httpd/ssl/rabbit/rabbit.crt
ssl_options.keyfile =/etc/httpd/ssl/rabbit/rabbit.key
ssl_options.verify = verify_none
ssl_options.fail_if_no_peer_cert = false
listeners.tcp.default = 5672
```

Consola A.13: Archivo /etc/rabbitmq/rabbitmq.conf.

A.2. Nodo Compute

Para el nodo Compute se tienen los archivos de configuración que se muestran a continuación.

A.2.1. /etc/nova/nova.conf

En la consola A.14 se muestra el archivo de configuración de MySQL.

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
transport_url = rabbit://openstack:abc123@controller
my_ip = 192.168.10.180
compute_driver = libvirt.LibvirtDriver
[api]
auth_strategy = keystone
[api_database]
[barbican]
[barbican_service_user]
[cache]
[cinder]
[compute]
[conductor]
[console]
[consoleauth]
[cors]
[cyborg]
[database]
[devices]
[ephemeral_storage_encryption]
[filter_scheduler]
[glance]
api_servers = https://controller:9292
[guestfs]
[healthcheck]
[hyperv]
[image_cache]
[ironic]
[key_manager]
[keystone]
[keystone_authtoken]
www_authenticate_uri = https://controller:5000/
auth_url = https://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = abc123
[libvirt]
virt_type = kvm
[metrics]
```

```
[mks]
[neutron]
auth_url = https://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = abc123
[notifications]
[os_vif_linux_bridge]
[os_vif_ovs]
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[pci]
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = https://controller:5000/v3
username = placement
password = abc123
insecure=true
[privsep]
[profiler]
[quota]
[rdp]
[remote_debug]
[scheduler]
[serial_console]
[service_user]
send_service_user_token = true
auth_url = https://controller/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = abc123
[spice]
[upgrade_levels]
[vault]
[vendordata_dynamic_auth]
[vmware]
[vnc]
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
[workarounds]
[wsgi]
[zvm]
```

Consola A.14: Archivo /etc/nova/nova.conf.

A.2.2. /etc/neutron/neutron.conf

En la consola A.15 se muestra el archivo de configuración de `Neutron`.

```
[DEFAULT]
transport_url = rabbit://openstack:abc123@controller
[agent]
[cache]
[cors]
[database]
[designate]
[experimental]
[healthcheck]
[ironic]
[keystone_authtoken]
www_authenticate_uri = https://controller:5000
auth_url = https://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = abc123
[nova]
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[placement]
[privsep]
[profiler]
[quotas]
[ssl]
```

Consola A.15: Archivo /etc/neutron/neutron.conf.

Apéndice B

Parámetros de las herramientas de análisis/ataque

Este apéndice contiene todos los parámetros usados por las herramientas de análisis/ataque.

B.1. Nmap

La tabla B.1 presenta parámetros para escanear redes con `nmap`.

Opción	Descripción
`-sS`	Realiza un escaneo SYN (semi-abierto). Es rápido y discreto.
`-sV`	Detecta versiones de servicios en los puertos abiertos.
`-p`	Especifica los puertos a escanear.
`-p-`	Escanea todos los puertos TCP (1 al 65535).
`-A`	Habilita detección de SO, versiones, scripts y traceroute.
`-T[0-5]`	Ajusta la velocidad/agresividad del escaneo (0: lento, 5: muy rápido).
`-Pn`	Desactiva la detección de host (asume que todos los hosts están activos).
`-oN`	Guarda los resultados en formato legible para humanos.
`-oX`	Guarda los resultados en formato XML.
`-script`	Ejecuta scripts NSE (Nmap Scripting Engine) para detección avanzada.

Tabla B.1: Opciones principales de configuración en Nmap.

Algunos ejemplos de como usar estos parámetros para realizar un análisis de una red se listan a continuación.

1. **Escaneo de puertos TCP y UDP con detección de versiones y scripts de vulnerabilidad.** Este comando combina el escaneo de puertos `TCP` y `UDP`, detecta las versiones de los servicios en esos puertos, y ejecuta `scripts` para la detección de vulnerabilidades.

Consola B.1: Escaneo de puertos TCP y UDP con detección de versiones y scripts de vulnerabilidad.

```
nmap -sS -sU -sV --script vuln -p 22,80,443,53 192.168.10.188
```

2. **Escaneo agresivo con detección de sistema operativo, traceroute y script de vulnerabilidad.** Este comando realiza un escaneo agresivo con la opción `-T5`, detecta el sistema operativo, realiza un `traceroute` y usa `scripts` para la detección de vulnerabilidades.

Consola B.2: Escaneo agresivo con detección de sistema operativo, traceroute y script de vulnerabilidad.

```
nmap -A -T5 --script vuln -p 22,80,443 192.168.10.188
```

3. **Escaneo de puertos abiertos con agresividad moderada y sin detección de host.** Este comando escanea los puertos abiertos especificados (22, 80, 443), realiza el escaneo con agresividad moderada `-T4`, y desactiva la detección de host con `-Pn`.

Consola B.3: Escaneo de puertos abiertos con agresividad moderada y sin detección de host.

```
nmap -p 22,80,443 -T4 -Pn 192.168.10.188
```

4. **Escaneo completo con detección de SO, versiones y resultados en formato legible.** Este comando realiza un escaneo completo con la detección del sistema operativo y las versiones de los servicios, y guarda los resultados en un formato legible para humanos.

Consola B.4: Escaneo completo con detección de SO, versiones y resultados en formato legible.

```
nmap -A -sV -oN resultado.txt 192.168.10.188
```

5. **Escaneo de puertos MySQL.** Este comando de `Nmap` realiza un escaneo en el puerto 3306 (puerto predeterminado de `MySQL`) de un servidor remoto. Además de verificar la disponibilidad del puerto, utiliza la opción `--script mysql-*` para ejecutar `scripts` específicos de `Nmap` relacionados con `MySQL`, lo que permite obtener información detallada sobre el servicio, como la versión de `MySQL`, configuraciones, y posibles vulnerabilidades en el servicio.

Consola B.5: Escaneo de puertos MySQL.

```
nmap -p 3306 --script mysql-* 192.168.10.200
```

B.2. Gobuster

`Gobuster` presenta parámetros muy útiles para poder realizar un análisis detallado de un sitio web, los más importantes se muestran en la tabla B.2.

Opción	Descripción
`-u`	Especifica la URL base para el escaneo, por ejemplo, `http://example.com`.
`-w`	Define el archivo de diccionario a utilizar para la búsqueda de directorios y archivos. En este caso, `/path/to/SecLists/Discovery/Web-Content/common.txt`.
`-t`	Especifica el número de hilos concurrentes a usar durante el escaneo. Un número más alto puede hacer el escaneo más rápido, pero también puede generar más tráfico.
`-x`	Permite establecer una extensión de archivo específica para buscar durante el escaneo, como `.php` o `.html`.
`-l`	Limita la cantidad de resultados a mostrar, útil cuando no deseas ver demasiados resultados.
`-r`	Permite realizar un escaneo recursivo dentro de los directorios encontrados.
`-s`	Especifica el tipo de búsqueda que debe realizarse: `dir` para directorios, o `dns` para subdominios. En este caso, `dir` se usa para buscar directorios.
`-v`	Muestra información detallada del escaneo (modo verboso).
`-exclude`	Excluye ciertos directorios o archivos durante el escaneo. Esto puede ser útil si conoces que ciertos directorios no contienen información relevante.

Tabla B.2: Opciones principales de configuración en Gobuster.

Algunos ejemplos de como usar estos parámetros para realizar un análisis de una página web se listan a continuación.

1. **Escaneo de directorios comunes usando Gobuster.** Este comando ejecuta un escaneo de directorios comunes en el sitio web especificado, utilizando el diccionario de palabras comunes. Esto puede ayudar a encontrar directorios o rutas sensibles en el servidor.

Consola B.6: Escaneo de directorios comunes usando Gobuster.

```
gobuster dir -u http://example.com -w /path/to/SecLists/Discovery/Web-Content/
    common.txt
```

2. **Escaneo de directorios con un diccionario personalizado.** Este comando realiza un escaneo de directorios utilizando un diccionario específico que puede incluir rutas más complejas o relevantes para un análisis más detallado.

Consola B.7: Escaneo de directorios con un diccionario personalizado.

```
gobuster dir -u http://example.com -w /path/to/custom-dictionary.txt
```

3. **Escaneo de subdominios con Gobuster.** Este comando realiza un escaneo de subdominios para encontrar subdominios activos bajo el dominio especificado. Utiliza el modo `dns` y el diccionario adecuado para este tipo de análisis.

Consola B.8: Escaneo de subdominios con Gobuster.

```
gobuster dns -d example.com -w /path/to/subdomain-dictionary.txt
```

4. **Escaneo con mayor concurrencia.** Este comando aumenta la concurrencia de las solicitudes, lo que permite realizar un escaneo más rápido de los directorios del sitio web objetivo.

Consola B.9: Escaneo con mayor concurrencia.

```
gobuster dir -u http://example.com -w /path/to/SecLists/Discovery/Web-Content/
    common.txt -t 50
```

5. **Escaneo con ocultación de errores.** Este comando realiza un escaneo de directorios y oculta los códigos de estado que no sean relevantes para el análisis, como 404 (no encontrado), mostrando solo los más importantes, como 200 (OK) o 403 (Forbidden).

Consola B.10: Escaneo con ocultación de errores.

```
gobuster dir -u http://example.com -w /path/to/SecLists/Discovery/Web-Content/
    common.txt -s "200,403"
```

B.3. Hydra

Hydra presenta parámetros muy poderosos para poder realizar un ataque tanto a sitios web como a SSH, los más importantes se muestran en la tabla B.3.

Opción	Descripción
-l	Especifica el nombre de usuario a utilizar en el ataque de fuerza bruta. En este caso, $usuario.
-p	Especifica la contraseña o el archivo de contraseñas que se utilizarán para el ataque. En este caso, "$contrasena".
-t	Define el número de hilos (conexiones paralelas) a usar durante el ataque. Un número mayor acelera el proceso, pero también puede aumentar la carga en el servidor objetivo.
-l	Define el nombre de usuario, por ejemplo -l admin o -l user.
-p	Define la contraseña, o el archivo de contraseñas a usar. Por ejemplo -p password o -p /path/to/wordlist.txt.
-s	Especifica el puerto a usar en el servicio objetivo. Por defecto, Hydra intenta con el puerto estándar del servicio (por ejemplo, SSH en el puerto 22).
ssh://	Define el protocolo o servicio objetivo. En este caso, ssh://$ip_ssh indica que se está realizando un ataque sobre el servicio SSH en la dirección IP especificada.
-v	Modo verboso. Muestra información detallada durante el ataque, útil para ver el progreso y los intentos.
-t	Especifica el número de tareas (hilos) a usar simultáneamente. Un número alto de hilos puede acelerar el proceso de ataque, pero también aumentar la carga en el servidor.
2>&1	Redirige la salida estándar y los errores para que se muestren juntos, permitiendo ver el progreso del ataque.

Tabla B.3: Opciones principales de configuración en Hydra.

1. **Escaneo SSH con un único usuario y una contraseña.** Este comando realiza un ataque de fuerza bruta sobre un servidor SSH utilizando un usuario específico y una contraseña. En este caso, se están utilizando hilos de ejecución con un número bajo para evitar sobrecargar el servidor objetivo.

Consola B.11: Escaneo SSH con un único usuario y una contraseña.

```
hydra -l usuario -p contrasena ssh://192.168.10.188 -t 1 2>&1
```

2. **Ataque de fuerza bruta contra un servidor SSH con un archivo de contraseñas.** Este comando realiza un ataque de fuerza bruta sobre un servidor SSH, probando múltiples contraseñas desde un archivo de diccionario. El ataque se realiza con 16 hilos de ejecución para acelerar el proceso.

Consola B.12: Ataque de fuerza bruta contra un servidor SSH con un archivo de contraseñas.

```
hydra -l usuario -P /home/usuario/wordlist.txt ssh://192.168.10.188 -t 16 2>&1
```

3. **Escaneo FTP con un archivo de usuarios y contraseñas.** Este comando realiza un ataque de fuerza bruta sobre un servidor FTP, utilizando un archivo que contiene múltiples usuarios y contraseñas. Se especifican 10 hilos para equilibrar la rapidez y la carga sobre el servidor.

Consola B.13: Escaneo FTP con un archivo de usuarios y contraseñas.

```
hydra -L /home/usuario/userlist.txt -P /home/usuario/wordlist.txt ftp://
    192.168.10.188 -t 10 2>&1
```

4. **Ataque a un servicio HTTP con un archivo de usuarios y contraseñas.** Este comando realiza un ataque de fuerza bruta sobre un servidor web que requiere autenticación básica HTTP. Se utilizan múltiples hilos para acelerar el proceso de prueba de combinaciones de usuario y contraseña.

Consola B.14: Ataque a un servicio HTTP con un archivo de usuarios y contraseñas.

```
hydra -L /home/usuario/userlist.txt -P /home/usuario/wordlist.txt http-get://
    192.168.10.188 -t 10 2>&1
```

5. **Escaneo de servicios SSH con múltiples usuarios y contraseñas.** Este comando realiza un ataque de fuerza bruta sobre un servidor SSH, probando varias combinaciones de usuarios y contraseñas, utilizando 32 hilos para un escaneo más rápido.

Consola B.15: Escaneo de servicios SSH con múltiples usuarios y contraseñas.

```
hydra -L /home/usuario/userlist.txt -P /home/usuario/passwordlist.txt ssh://
    192.168.10.188 -t 32 2>&1
```

Apéndice C

Proceso de creación de certificados SSL/TLS

Este apéndice contiene los 3 métodos para generar las llaves SSL/TLS empleadas por los módulos y herramientas de OpenStack.

C.1. Proceso de creación de certificados SSL con una CA interna.

1. Se genera un certificado autofirmado usando OpenSSL, el certificado se va a almacenar en ca.crt.

Consola C.1: Generacion del certificado autofirmado del módulo.

```
root@controller:# openssl req -new -x509 -days 3650 -keyout cah.key -out cah.crt -
    nodes -subj "/C=MX/ST=CDMX/L=CDMX/O=IKCORP/OU=IKCORP/CN=controller"
```

2. Se genera una nueva clave privada utilizando el algoritmo RSA que se va a almacenar en modulo.key.

Consola C.2: Generación de clave privada para el módulo.

```
root@controller:# openssl genpkey -algorithm RSA -out modulo.key
```

3. Utilizando la clave privada modulo.key se genera una solicitud de firma de certificado que se va a almacenar en modulo.csr

Consola C.3: Generación de solicitud de firma de certificado para el módulo.

```
root@controller:# openssl req -new -key modulo.key -out modulo.csr -subj "/C=MX/ST=
    CDMX/L=CDMX/O=IKCORP/OU=IKCORP/CN=controller"
```

4. Se firma la solicitud de firma de certificado modulo.csr con el certificado de autoridad certificadora almacenado en ca.crt generado en el punto 1. El resultado es un certificado X.509 válido que se almacena en modulo.crt.

Consola C.4: Firma de la solicitud de firma del certificado para el módulo.

```
root@controller:# openssl x509 -req -days 3650 -in modulo.csr -CA ca.crt -CAkey ca.key
    -CAcreateserial -out modulo.crt
```

C.2. Proceso de creación de certificados sin autoridad certificadora

1. Se genera una clave privada RSA de 2048 bits y se guarda en el archivo `modulo.key`.

Consola C.5: Generacion de clave privada del módulo.

```
root@controller:# openssl genrsa -out modulo.key 2048
```

2. Se utiliza la clave privada `modulo.key` para generar una solicitud de firma de certificado, que se guarda en el archivo `modulo.csr`.

Consola C.6: Generación de solicitud de firma de certificado para el módulo.

```
root@controller:# openssl req -new -key modulo.key -sha256 -out modulo.csr
```

3. Se usa la solicitud de firma de certificado `modulo.csr` para construir un certificado digital autofirmado que se almacena en el archivo `modulo.crt` y se firma con la clave privada `modulo.key`.

Consola C.7: Generación de certificado digital autofirmado para el módulo.

```
root@controller:# openssl x509 -req -days 365 -in modulo.csr -signkey modulo.key -
    sha256 -out modulo.crt
```

4. Se convierte el certificado `modulo.crt` al formato `PEM` (Privacy Enhanced Mail) y se guarda como `modulo.pem`.

Consola C.8: Cambio de formato para el certificado para el módulo.

```
root@controller:# openssl x509 -in modulo.crt -out modulo.pem -outform PEM
```

C.3. Procedimiento de generación de certificados con seguridad avanzada y clave privada cifrada

Este proceso es exclusivo de la herramienta `etcd`.

1. Se genera una clave privada `RSA` utilizando el algoritmo `AES-256` para su cifrado, y se guarda en el archivo `cae.key`.

Consola C.9: Generación de clave privada protegida con AES-256 para la autoridad certificadora.

```
root@controller:# openssl genpkey -algorithm RSA -out cae.key -aes256
```

2. Utilizando la clave privada `cae.key`, se genera un certificado autofirmado en formato `X.509` que se guarda en el archivo `cae.crt`.

Consola C.10: Generación de certificado autofirmado para la autoridad certificadora.

```
root@controller:# openssl req -x509 -new -nodes -key cae.key -sha256 -days 3650 -
    out cae.crt
```

3. Se genera una nueva clave privada `RSA` no cifrada que se guarda en el archivo `etcd.key`. Esta clave se utilizará para firmar solicitudes de certificado en servicios como `etcd`.

Consola C.11: Generación de clave privada para etcd.

```
root@controller:# openssl genpkey -algorithm RSA -out etcd.key
```

4. `etcd` requiere certificados más rigurosos que incluyan extensiones avanzadas como`keyUsage`, `extendedKeyUsage` y `subjectAltName`, ya que estas garantizan un nivel superior de seguridad y cumplimiento con estándares modernos de `TLS`. Por ello, se utiliza un archivo de configuración detallado para personalizar los certificados, a diferencia de los certificados básicos generados con comandos directos, que son adecuados únicamente para configuraciones simples o pruebas. El contenido del archivo de configuración del certificado (`etcd.cnf`) se muestra a continuación:

Consola C.12: Archivo etcd.cnf.

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
CN = etcd-server
[v3_req]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
subjectAltName = @alt_names
[alt_names]
DNS.1 = controller
IP.1 = 192.168.10.188
```

5. Se genera una solicitud de firma de certificado utilizando la clave privada `etcd.key` y el archivo de configuración `etcd.cnf`. La solicitud de firma de certificado contiene toda la información necesaria para que la `CA` genere el certificado `etcd`.

Consola C.13: Generación de solicitud de firma de certificado para etcd.

```
root@controller:# openssl req -new -key etcd.key -out etcd.csr -config etcd.cnf
```

6. La solicitud de firma de certificado (`etcd.csr`) se firma con el certificado de la autoridad certificadora (`cae.crt`) y la clave privada de la `CA` (`cae.key`), utilizando las extensiones definidas en `etcd.cnf`. El resultado es un certificado `X.509` válido que se guarda en el archivo `etcd.crt`.

Consola C.14: Firma de la solicitud de certificado para etcd.

```
root@controller:# openssl x509 -req -in etcd.csr -CA cae.crt -CAkey cae.key -
    CAcreateserial -out etcd.crt -days 3650 -sha256 -extfile etcd.cnf -extensions
    v3_req
```

Apéndice D

Scripts de protección contra ataques de fuerza bruta

A continuación se muestra cada uno de los `scripts` y programas que se usan para proteger al servidor de ataques de fuerza bruta.

D.1. Programa en Python para la identificación de ataques realizados a Horizon

En la consola D.1 se muestra el programa en `Python` para la identificación de ataques realizados a `Horizon`.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


import re
from datetime import datetime, timedelta
import sys
import os

LOG_FILE = "/var/log/httpd/error_log"
DROP_IP_FILE = "/var/log/dropIp"
MAX_ATTEMPTS = 5
TIME_LIMIT = 15  # En minutos

def parse_log():
    failed_attempts = {}
    with open(LOG_FILE, "r") as log:
        for line in log.readlines():

            match = re.search(
```

```
                r'\[(.*?)\] .*?remote (\d+\.\d+\.\d+\.\d+):\d+.*?Login failed',
                line
            )
            if match:
                timestamp_str = match.group(1)  # Capturar timestamp
                ip_address = match.group(2)    # Capturar IP

                # Convertir el timestamp al objeto datetime
                try:
                    timestamp = datetime.strptime(timestamp_str, "%a %b %d %H:%M:%S.%f %Y")
                except ValueError:
                    continue

                if ip_address not in failed_attempts:
                    failed_attempts[ip_address] = []
                failed_attempts[ip_address].append(timestamp)

    return failed_attempts

def detect_brute_force(failed_attempts):
    for ip, timestamps in failed_attempts.items():
        # Ignorar IPs ya registradas en el archivo /var/log/dropIp
        if ip_already_logged(ip):
            continue

        timestamps.sort()
        for i in range(len(timestamps) - MAX_ATTEMPTS + 1):
            if timestamps[i+MAX_ATTEMPTS-1]-timestamps[i] <= timedelta(minutes=TIME_LIMIT):
            log_ip(ip)  # Registrar la IP en /var/log/dropIp
            print(ip)  # Imprimir la IP ofensiva
        return ip

    return None

def ip_already_logged(ip):
    if not os.path.exists(DROP_IP_FILE):
        return False
    with open(DROP_IP_FILE, "r") as file:
        return ip in file.read()

def log_ip(ip):
    """Registrar una nueva IP en /var/log/dropIp."""
    with open(DROP_IP_FILE, "a") as file:
        file.write(f"{ip}\n")

def main():
    failed_attempts = parse_log()
    ip = detect_brute_force(failed_attempts)
    if ip:
        sys.exit(ip)
    else:
        sys.exit(0)

if __name__ == "__main__":
    main()
```

Consola D.1: Programa de Python para la protección de Horizon.

D.2. Script en Bash para responder a ataques de fuerza bruta a Horizon

En la consola D.2 se muestra el `script` en `Bash` para actuar ante un ataque de fuerza bruta a `Horizon`.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
```

```
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


#!/bin/bash

PYTHON_SCRIPT="Hprotec.py"
EMERGENCY_LOG="/var/log/emergenciesH.log"
HORIZON_LOG="/var/log/httpd/error_log"
FIREWALL_ZONE="openstack"

# Ejecutar el script Python
IP=$(python3 "$PYTHON_SCRIPT")

# Verifica si se detectó actividad sospechosa
if [ $? -ne 0 ]; then
    echo "$(date) - Intentos fallidos detectados desde IP: $IP" >> "$EMERGENCY_LOG"

    # Bloquear la IP en el firewall para el puerto SSH
    firewall-cmd --zone="$FIREWALL_ZONE" \
    --add-rich-rule="rule family=ipv4 source address=$IP \
    port port=440 protocol=tcp reject" --permanent

    firewall-cmd --reload
    echo "$(date) - IP $IP bloqueada en el firewall (puerto HTTPS modificado)." \
    >> "$EMERGENCY_LOG"

    # Registrar las últimas 20 líneas relevantes del log de Horizon
    echo "$(date) - Últimas 20 líneas del log de Horizon:" \
    >> "$EMERGENCY_LOG"

    grep "$IP" "$HORIZON_LOG" | tail -n 20 >> "$EMERGENCY_LOG"
else
    echo "$(date) - No se detectaron intentos fallidos." >> "$EMERGENCY_LOG"
fi
```

Consola D.2: Script en Bash para la protección de Horizon.

D.3. Programa en Python para la identificación de ataques realizados a SSH

En la consola D.3 se muestra el programa en `Python` para la identificación de ataques realizados a `SSH`.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
```

```
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


import re
from datetime import datetime, timedelta

# Ruta del archivo de log y lista de IPs bloqueadas
log_file = "/var/log/secure"
drop_ip_file = "/var/log/dropIPSSH"

# Expresión regular para capturar intentos fallidos y conexiones cerradas
log_pattern = (
    r"(\w+\s+\d+\s+\d+:\d+:\d+)\s+\S+\s+sshd\[\d+\]:\s+"
    r"(Failed password|Connection closed by authenticating user) .* from ([\d\.]+) port"
)


# Leer el archivo de log
def analyze_ssh_log(log_file, drop_ip_file):
    ip_attempts = {}
    blocked_ips = []

    # Leer drop IPs existentes
    try:
        with open(drop_ip_file, "r") as drop_file:
            existing_ips = set(drop_file.read().splitlines())
    except FileNotFoundError:
        existing_ips = set()

    with open(log_file, "r") as file:
        for line in file:
            match = re.search(log_pattern, line)
            if match:
                timestamp_str, _, ip = match.groups()
                # Convertir el timestamp a un objeto datetime
                timestamp = datetime.strptime(timestamp_str, "%b %d %H:%M:%S")

                # Registrar el intento
                if ip not in ip_attempts:
                    ip_attempts[ip] = []
                ip_attempts[ip].append(timestamp)

    # Verificar si hay al menos 5 intentos en un intervalo de 15 minutos
    for ip, timestamps in ip_attempts.items():
        timestamps.sort()
        for i in range(len(timestamps) - 4):  # Revisar ventanas de 5 intentos
            if (timestamps[i + 4] - timestamps[i]) <= timedelta(minutes=15):
                if ip not in existing_ips:
                    blocked_ips.append(ip)
                    # Guardar la IP en dropIPSSH
                    with open(drop_ip_file, "a") as drop_file:
                        drop_file.write(ip + "\n")
                break

    # Devolver las IPs bloqueadas
    return blocked_ips

# Ejecutar la función y devolver solo las IPs
if __name__ == "__main__":
    suspicious_ips = analyze_ssh_log(log_file, drop_ip_file)
    if suspicious_ips:
        for ip in suspicious_ips:
            print(ip)
```

Consola D.3: Programa de Python para la protección de SSH.

D.4. Script en Bash para responder a ataques de fuerza bruta a SSH

En la consola D.4 se muestra el `script` en `Bash` para actuar ante un ataque de fuerza bruta a `SSH`.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


#!/bin/bash

PYTHON_SCRIPT="s.py"
EMERGENCY_LOG="/var/log/emergenciesSSH.log"
HORIZON_LOG="/var/log/secure"
FIREWALL_ZONE="openstack"

# Ejecutar el script Python
IP=$(python3 "$PYTHON_SCRIPT")

# Verifica si se detectó una IP sospechosa
if [ $? -eq 0 ] && [ -n "$IP" ]; then
    # Recuperar la IP de la salida de Python script
    echo "$(date) - Intentos fallidos detectados desde IP: $IP" >> "$EMERGENCY_LOG"

        # Bloquear la IP en el firewall para el puerto SSH
    firewall-cmd --zone="$FIREWALL_ZONE" \
    --add-rich-rule="rule family=ipv4 source address=$IP port port=22 protocol=tcp reject" \
    --permanent
    firewall-cmd --reload
    echo "$(date) - IP $IP bloqueada en el firewall (puerto SSH)." >> "$EMERGENCY_LOG"

        # Registrar las últimas 20 líneas relevantes del log de Horizon
    echo "$(date) - Últimas 20 líneas del log de Horizon:" >> "$EMERGENCY_LOG"
    grep "$IP" "$HORIZON_LOG" | tail -n 20 >> "$EMERGENCY_LOG"

else
    echo "$(date) - No se detectó una IP sospechosa." >> "$EMERGENCY_LOG"
fi
```

Consola D.4: Script en Bash para la protección de SSH.

Apéndice E

Configuración de red en el Host y proceso de creación de MV (atacantes)

Primero se crea el entorno de red mostrado en la figura 6.1, para ello se debe acceder al directorio de configuración de red del `host` (`/etc/network/interfaces.d`) y crear un archivo (el nombre no importa) y añadir cada puente (`brH2` para `Hacker 1`, `brH3` para `Hacker 2` y `brH4` para `Hacker 3`) y su respectiva configuración como se muestra en la consola E.1.

```
auto brH2
iface brH2 inet static
    address 10.0.20.1
    netmask 255.255.255.0
    bridge_ports none
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
auto brH3
iface brH3 inet static
    address 172.10.20.1
    netmask 255.255.255.0
    bridge_ports none
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
auto brH4
iface brH4 inet static
    address 10.0.50.1
    netmask 255.255.255.0
    bridge_ports none
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
```

Consola E.1: Archivo /etc/network/interfaces.d/IkerBr.

Se debe reiniciar el servicio `networking`. Posteriormente se crean las reglas de `IPTABLES` que permitan la comunicación de las 5 maquinas involucradas.

Consola E.2: Creacion de reglas en IPTABLES.

```
root@controller:# iptables -A FORWARD -i brH2 -o br0 -j ACCEPT
root@controller:# iptables -A FORWARD -i br0 -o brH2 -j ACCEPT
root@controller:# iptables -t nat -A POSTROUTING -s 10.0.20.2 -d 192.168.10.200 -o br0 -j
    MASQUERADE
root@controller:# iptables -t nat -A POSTROUTING -s 10.0.20.2 -d 192.168.10.188 -o br0 -j
    MASQUERADE
root@controller:# iptables -A FORWARD -i brH3 -o br0 -j ACCEPT
root@controller:# iptables -A FORWARD -i br0 -o brH3 -j ACCEPT
```

```
root@controller:# iptables -t nat -A POSTROUTING -s 172.10.20.2 -d 192.168.10.200 -o br0 -j
    MASQUERADE
root@controller:# iptables -t nat -A POSTROUTING -s 172.10.20.2 -d 192.168.10.188 -o br0 -j
    MASQUERADE
root@controller:# iptables -A FORWARD -i brH4 -o br0 -j ACCEPT
root@controller:# iptables -A FORWARD -i br0 -o brH4 -j ACCEPT
root@controller:# iptables -t nat -A POSTROUTING -s 10.0.50.2 -d 192.168.10.200 -o br0 -j
    MASQUERADE
root@controller:# iptables -t nat -A POSTROUTING -s 10.0.50.2 -d 192.168.10.188 -o br0 -j
    MASQUERADE
```

Una vez generadas las reglas, se procede a crear cada una de las maquinas virtuales. Para crear cada maquina virtual se utiliza el comando mostrado en la consola E.3.

Consola E.3: Creacion de maquinas virtuales con KVM/qemu.

```
root@controller:# virt-install --connect qemu:///system --name openstack --memory 4096 --
    vcpus 4 --location /var/lib/libvirt/images/ubuntu-20.04.6-live-server-amd64.iso --
    network bridge=brH2-4 model=virtio,--network bridge=br0 model=virtio --graphics=none --
    disk path=/dev/___ --console pty,target_type=serial --extra-args console=ttyS0,115200n8
     serial
```

Dentro de cada MV debemos configurar la dirección IP que se indica en la tabla 6.1, y esta configuración debe quedar como se muestra en las 3 figuras que se muestran a continuación.

Figura E.1: Configuración de red de Hacker 1.

Figura E.2: Configuración de red de Hacker 2.

Figura E.3: Configuración de red de Hacker 3.

Apéndice F

Instalación de Nmap, Gubuster, Hydra y Cubic

A continuación se muestran los comandos necesarios para realizar una correcta instalación de `Nmap`, `Gobuster`, `Hydra` y `Cubic` en `Ubuntu 20.04`.

F.1. Instalación de Nmap

Para instalar `Nmap` se utiliza el comando mostrado en la consola F.1.

Consola F.1: Instalación de Nmap.

```
root@hacker:# apt install nmap
```

F.2. Instalación de Gobuster

Para instalar `Gobuster` se utiliza el comando mostrado en la consola F.2.

Consola F.2: Instalación de Gobuster.

```
root@hacker:# sudo apt install -y golang
root@hacker:# export GOPATH=$HOME/go
root@hacker:# export PATH=$PATH:$GOPATH/bin
root@hacker:# go install github.com/OJ/gobuster/v3@latest
```

De igual manera, se deben descargar las listas de subdirectorios posibles, como se muestra en la consola F.3.

Consola F.3: Descarga de archivos complementarios para Gobuster.

```
root@hacker:# wget -c https://github.com/danielmiessler/SecLists/archive/master.zip -O
    SecList.zip \
  && unzip SecList.zip \
  && rm -f SecList.zip
```

F.3. Instalación de Hydra

Para instalar `Hydra` se utiliza el comando mostrado en la consola F.4.

Consola F.4: Instalación de Hydra.

```
root@hacker:# apt install hydra
```

F.4. Instalación de Cubic

Para instalar `Cubic` se utilizan los comandos mostrados en la consola F.5.

Consola F.5: Instalación de Cubic en Ubuntu 20.04.6.

```
root@ubuntu:# sudo apt-add-repository universe
root@ubuntu:# sudo apt-add-repository ppa:cubic-wizard/release
root@ubuntu:# sudo apt update
root@ubuntu:# sudo apt install --no-install-recommends cubic
```

Apéndice G

Scripts utilizados para los ataques de fuerza bruta

A continuación se muestran los `scripts` en `Bash` que fueron utilizados para realizar los ataques de fuerza bruta para cada tipo de contraseña (insegura, medianamente segura, insegura).

G.1. Script para el ataque de fuerza bruta para un usuario con una contraseña insegura.

A continuación se muestra el `script` usado para realizar el ataque de fuerza bruta al usuario `root`, cuya contraseña tenía una longitud de `4` caracteres y contenía solo letras minúsculas.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#

#!/bin/bash

# Configuración de la IP y usuario de SSH
usuario="root"
ip_ssh="192.168.10.200"

# Variables para el contador y temporizador
contador=0
start_time=$(date +%s)

# Comando para generar contraseñas y pasarlas a Hydra,
#con pausa de 1 segundo entre cada intento
python3 mediumpass.py | while read -r contrasena; do
    # Incrementar el contador de intentos
    contador=$((contador + 1))
```

```
35
36      # Ejecutar Hydra con la contraseña actual y mostrar la salida en consola
37      #echo "Probando contraseña #$contador: $contrasena"
38      resultado=$(hydra -l $usuario -p "$contrasena" ssh://$ip_ssh -t 1 2>&1)
39
40      # Mostrar el resultado de Hydra en la consola
41      #echo "$resultado"
42      if (( contador % 500 == 0 )); then
43          echo "Sigo probando contraseñas... Intento #$contador"
44          echo "Probando contraseña #$contador: $contrasena"
45
46      fi
47
48      # Verificar si el resultado contiene el mensaje de éxito de Hydra
49      if echo "$resultado" | grep -q "login: $usuario   password: $contrasena"; then
50          end_time=$(date +%s)
51          elapsed_time=$((end_time - start_time))
52
53          # Guardar los resultados en un archivo
54          echo "Contraseña encontrada: $contrasena" > resultados2.txt
55          echo "Número total de intentos: $contador" >> resultados2.txt
56          echo "Tiempo total: $elapsed_time segundos" >> resultados2.txt
57
58          echo " Contrase ña encontrada y guardada en 'resultados.txt'!"
59          break
60      fi
61
62      # Esperar 1 segundo entre intentos para no sobrecargar la CPU
63      #sleep 1
64  done || true
```

Consola G.1: Script para el ataque de fuerza bruta al usuario root con contraseña insegura.

A continuación se muestra el programa escrito en `Python 3` que genera las contraseñas aleatorias para el `script` con las características mencionadas anteriormente.

```
1   # Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
2   #
3   # Este programa es software libre: puedes redistribuirlo y/o modificarlo
4   # bajo los términos de la Licencia Pública General de GNU publicada por
5   # la Free Software Foundation, ya sea la versión 3 de la Licencia,
6   # o (a tu elección) cualquier versión posterior.
7   #
8   # Este programa se distribuye con la esperanza de que sea útil,
9   # pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
10  # de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
11  # Consulta la Licencia Pública General de GNU para más detalles.
12  #
13  # Deberías haber recibido una copia de la Licencia Pública General de GNU
14  # junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
15  #
16  # Nota: Si utilizas este software o partes del mismo, se requiere atribución.
17  # Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
18  #
19
20  import random
21  import string
22  import sys
23
24  # Generador de contraseñas
25  while True:
26      # Genera una contraseña aleatoria de longitud 4
27      contraseña = ''.join(random.choices(string.ascii_lowercase, k=4))
28
29      try:
30          print(contraseña)
31      except BrokenPipeError:
32          # Finalizar el script de Python si la tubería se cierra
33          sys.exit(0)
```

Consola G.2: Generador de contraseñas inseguras.

G.2. Script para el ataque de fuerza bruta para el usuario con una contraseña medianamente segura.

A continuación se muestra el `script` usado para realizar el ataque de fuerza bruta al usuario `root`, cuya contraseña tenía una longitud de 4 caracteres y contenía solo letras minúsculas y números.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


#!/bin/bash

# Configuración de la IP y usuario de SSH
usuario="root"
ip_ssh="192.168.10.200"

# Variables para el contador y temporizador
contador=0
start_time=$(date +%s)

# Comando para generar contraseñas y pasarlas a Hydra
#, con pausa de 1 segundo entre cada intento
python3 hardpass.py | while read -r contrasena; do
    # Incrementar el contador de intentos
    contador=$((contador + 1))
    # Ejecutar Hydra con la contraseña actual y mostrar la salida en consola
    #echo "Probando contraseña #$contador: $contrasena"
    resultado=$(hydra -l $usuario -p "$contrasena" ssh://$ip_ssh -t 1 2>&1)
    # Mostrar el resultado de Hydra en la consola
    #echo "$resultado"
    if (( contador % 500 == 0 )); then
        echo "Sigo probando contraseñas... Intento #$contador"
        echo "Probando contraseña #$contador: $contrasena"

    fi
```

Consola G.3: Script para el ataque de fuerza bruta al usuario root con contraseña medianamente segura.

A continuación se muestra el programa escrito en `Python 3` que genera las contraseñas aleatorias para el script con las características mencionadas anteriormente.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
```

```
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


import random
import string
import sys

# Generador de contraseñas
while True:
    # Genera una contraseña aleatoria de longitud 4 con letras minúsculas y números
    caracteres = string.ascii_lowercase + string.digits
    contraseña = ''.join(random.choices(caracteres, k=4))

    try:
        print(contraseña)
    except BrokenPipeError:
        # Finalizar el script de Python si la tubería se cierra
        sys.exit(0)
```

Consola G.4: Generador de contraseñas medianamente seguras.

G.3. Script para el ataque de fuerza bruta para el usuario con una contraseña segura.

A continuación se muestra el `script` usado para realizar el ataque de fuerza bruta al usuario cuya contraseña tenía una longitud de 4 caracteres y contenía letras minúsculas, números y los caracteres especiales `@`, `#`, `$`, `%`, `&` y `*`.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


#!/bin/bash

# Configuración de la IP y usuario de SSH
usuario="root"
ip_ssh="192.168.10.200"

# Variables para el contador y temporizador
contador=0
start_time=$(date +%s)

# Comando para generar contraseñas y pasarlas a Hydra,
#con pausa de 1 segundo entre cada intento
```

```
python3 hardpass.py | while read -r contrasena; do
    # Incrementar el contador de intentos
    contador=$((contador + 1))

    # Ejecutar Hydra con la contraseña actual y mostrar la salida en consola
    echo "Probando contraseña #$contador: $contrasena"
    resultado=$(hydra -l $usuario -p "$contrasena" ssh://$ip_ssh -t 1 2>&1)

    # Mostrar el resultado de Hydra en la consola
    echo "$resultado"

    # Verificar si el resultado contiene el mensaje de éxito de Hydra
    if echo "$resultado" | grep -q "login: $usuario   password: $contrasena"; then
        end_time=$(date +%s)
        elapsed_time=$((end_time - start_time))
```

Consola G.5: Script para el ataque de fuerza bruta al usuario root con contraseña segura.

A continuación se muestra el programa escrito en `Python 3` que genera las contraseñas aleatorias para el `script` con las características mencionadas anteriormente.

```
# Copyright (C) 2025 UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
#
# Este programa es software libre: puedes redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General de GNU publicada por
# la Free Software Foundation, ya sea la versión 3 de la Licencia,
# o (a tu elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil,
# pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita
# de COMERCIABILIDAD o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
# Consulta la Licencia Pública General de GNU para más detalles.
#
# Deberías haber recibido una copia de la Licencia Pública General de GNU
# junto con este programa. Si no, consulta <https://www.gnu.org/licenses/>.
#
# Nota: Si utilizas este software o partes del mismo, se requiere atribución.
# Por favor, cita este trabajo mencionando a IKER ALBERTO CEDILLO MARTINEZ
#


import random
import string
import sys

# Generador de contraseñas
while True:
    # Caracteres permitidos: letras minúsculas, dígitos y caracteres especiales
    caracteres = string.ascii_lowercase + string.digits + "@#$%&*"
    # Genera una contraseña aleatoria de longitud 4
    contraseña = ''.join(random.choices(caracteres, k=4))

    try:
        print(contraseña)
    except BrokenPipeError:
        # Finalizar el script de Python si la tubería se cierra
        sys.exit(0)
```

Consola G.6: Generador de contraseñas seguras.

Apéndice H

Códigos formato XML de los firewalls creados con firewalld en los nodos Controller y Compute

Para el nodo `Controller` tenemos el archivo mostrado en la consola H.1.

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <rule family="ipv4">
    <port port="22" protocol="tcp"/>
    <reject/>
  </rule>
  <port port="440" protocol="tcp"/>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="3306" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="5000" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="9696" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="8774" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="8775" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="6080" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="11211" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
```

```
    <port port="5671" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.180"/>
    <port port="5672" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <port port="80" protocol="tcp"/>
    <reject/>
  </rule>
</zone>
```

Consola H.1: Archivo /etc/firewalld/zones/openstack.xml generado por firewalld en el nodo Controller.

Para el nodo `Compute` tenemos el archivo mostrado en la consola H.2.

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <rule family="ipv4">
    <port port="22" protocol="tcp"/>
    <reject/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="5000" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="9696" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="8774" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="8775" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="6080" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="11211" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="5671" protocol="tcp"/>
    <accept/>
  </rule>
  <rule family="ipv4">
    <source address="192.168.10.188"/>
    <port port="5672" protocol="tcp"/>
    <accept/>
  </rule>
</zone>
```

Consola H.2: Archivo /etc/firewalld/zones/openstack.xml generado por firewalld en el nodo Compute.

Bibliografía

[1] TheirStack. (s. f.). *Empresas que usan OpenStack*.https://theirstack.com/es/technology/openstack

[2] J. Hurwitz, M. Kaufman, F. G. Halper. (2015). *Cloud Services for Dummies – IBM Limited Edition*. IBM.

[3] Google Cloud. (s. f.). *PaaS vs IaaS vs SaaS*. https://cloud.google.com/learn/paas-vs-iaas-vs-saas?hl=es

[4] P. Iranzo Gómez, P. Ibáñez Requena, M. Pérez Colino, S. McCarty. (2022). *Red Hat Enterprise Linux 9 Administration: A comprehensive Linux system administration guide for RHCSA certification exam candidates*. Packt Publishing.

[5] O. Khedher. (2024). *Mastering OpenStack: Implement the latest techniques for designing and deploying an operational, production-ready private cloud*. Packt Publishing.

[6] Amazon Web Services. (s. f.). *What is an SSL certificate?*. https://aws.amazon.com/es/what-is/ssl-certificate/

[7] G. Fahrnberger. (2024). *Pattern- and Similarity-Based Realtime Risk Monitoring of SSH Brute Force Attacks with Bloom Filters*. Proceedings of the XXth Conference of Open Innovations Association FRUCT. Directory of Open Access Journals.

[8] TecnetOne (21 de noviembre del 2023). *Ataques de Reconocimiento: Descifrando las Técnicas*. https://blog.tecnetone.com/ataques-de-reconocimiento-descifrando-las-t%C3%A9cnicas

[9] McAfee (s. f.). *Ataque de interceptación: causas, riesgos y cómo evitarlo*. https://www.mcafee.com/learn/es/ataque-de-interceptacion/

[10] Z. Al-Khazaali, A. Al-Ghabban, H. Al-Musawi, A. Sabah, N. Al Mahdi. (2025). *Characteristics of Port Scan Traffic: A Case Study Using Nmap*. Journal of Engineering and Sustainable Development.

[11] R. S. Hashim, A. R. Enad, M. A. Al-Khafagi, N. K. Abdalhameed. (2023). *The facilities of detection by using a tool of Wireshark*. Indonesian Journal of Electrical Engineering and Computer Science.

[12] J. O. Condal Fontanet. (2023). *Analysis of Web Applications Penetration Testing and its Realization*. Master's thesis, Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, Barcelona.

[13] C. Daskalou. (2024). *Penetration Testing Methodology*. Postgraduate thesis, Faculty of Sciences, School of Informatics, Aristotle University of Thessaloniki.

[14] Django. (s. f.). *Cross Site Request Forgery protection*. https://docs.djangoproject.com/en/5.1/ref/csrf/

[15] Van Vugt, S. (2022). *Red Hat RHCSA 9 Cert Guide EX200*. Pearson Education.

[16] Ubunlog. (27 de noviembre del 2024). *Cubic: cómo crear una ISO personalizada de Ubuntu*. https://ubunlog.com/cubic-iso-personalizada-ubuntu/

[17] N. Palma Pérez. (2020). *Eficiencia de los servidores web Apache 2 y Nginx: un estudio de caso*. Serie Científica De La Universidad De Las Ciencias Informáticas.

[18] J. Walden. (2020). *The impact of a major security event on an open source project: The case of OpenSSL*. In *Proceedings of the 17th International Conference on Mining Software Repositories*. Recuperado de https://www.example.com

[19] IBM (3 de enero del 2025). *Cómo proteger su servidor SSH*. https://www.ibm.com/docs/es/aspera-fasp-proxy/1.4?topic=appendices-securing-your-ssh-server

[20] Microsoft. (s. f.). *¿Qué es la protección con contraseña?*. https://www.microsoft.com/es-mx/security/business/security-101/what-is-password-protection

[21] security.org (s. f.). *How Secure Is My Password?*. https://www.security.org/how-secure-is-my-password/

[22] Irving Yohanan Peña Núñez. (2024). *Implementación de una red de área amplia bajo el paradigma de redes virtuales*. Facultad de Ingeniería, México.

[23] FIRST (s. f.). *Common Vulnerability Scoring System Version 3.0 Calculator*. https://www.first.org/cvss/calculator/3.0