UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN
SEGURIDAD DE LA INFORMACIÓN
RPC-SO-28-NO.669-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

DISEÑO DE UNA MÁQUINA VIRTUAL Y ANÁLISIS DE SUS VULNERABILIDADES CON FINES PRÁCTICOS: SERVIDOR FTP SERVIDOR DE BASE DE DATOS, INYECCIONES SQL Y DE COMANDOS

AUTORES:

CRISTIAN ANDRÉS GUACHAMIN HERNÁNDEZ
RENÉ MARCELO SIERRA MONTESINOS

DIRECTOR:

MIGUEL ARTURO ARCOS ARGUDO

CUENCA – ECUADOR
2024

Autores:

Cristian Andrés Guachamin Hernández
Ingeniero de Sistemas mención Telemática.
Candidato a Magíster en Seguridad de la Información por la Universidad Politécnica Salesiana – Sede Cuenca.
cguachamin@est.ups.edu.ec

René Marcelo Sierra Montesinos
Ingeniero de Sistemas mención Telemática.
Candidato a Magíster en Seguridad de la Información por la Universidad Politécnica Salesiana – Sede Cuenca.
rsierra@est.ups.edu.ec

Dirigido por:

Miguel Arturo Arcos Argudo
Ingeniero de Sistemas.
Doctor en Ciencias de la Computación.
marcos@ups.edu.ec

Diseño de una máquina virtual y análisis de sus vulnerabilidades con fines prácticos: servidor FTP servidor de base de datos, inyecciones SQL y de comandos

DEDICATORIA

Dedico este proyecto de tesis a aquellos que han sido mi inspiración y apoyo inquebrantable a lo largo de este arduo viaje académico.

A mis padres, cuyo amor incondicional y sacrificio han sido la base de mi educación. Gracias por siempre creer en mí y alentarme a perseguir mis sueños.

A mis profesores y mentores, quienes han compartido su sabiduría y conocimientos conmigo. Su guía y orientación han sido fundamentales en mi desarrollo académico.

A mis amigos y seres queridos, por su paciencia, comprensión y palabras de aliento cuando más los necesitaba.

A mi esposa, por su amor y apoyo constante, por ser mi refugio en los momentos de estrés y por creer en mí incluso cuando yo dudaba de mí mismo.

A mis hijos, por ser un apoyo e inspiración constante para no decaer y alcanzar mis objetivos.

Este proyecto de tesis es el resultado de mucha dedicación y esfuerzo, y se lo dedico a todos ustedes, porque cada uno ha desempeñado un papel importante en mi vida y en este logro académico. Gracias por estar a mi lado en esta travesía.

Con gratitud y cariño,

Cristian Andrés Guachamin Hernández

AGRADECIMIENTO

Quiero expresar mi sincero agradecimiento a mi familia por su amor incondicional, su aliento constante y por ser mi fuente de inspiración.

Agradezco a todas las personas que participaron en esta investigación, ya sea como participantes o colaboradores, por su tiempo y contribución.

Este proyecto de tesis no habría sido posible sin la ayuda y el apoyo de todos ustedes. Estoy profundamente agradecido y honrado por haber tenido la oportunidad de trabajar en este proyecto y de contar con personas tan maravillosas en mi vida.

Con gratitud,

Cristian Andrés Guachamin Hernández

DEDICATORIA

Dedico esta tesis de maestría con amor y gratitud, primero a mis hijos, quienes han sido mi fuente inagotable de inspiración y fortaleza. Cada logro que alcanzo es un tributo a su futuro y al ejemplo que busco establecer para ellos.

A mis amigos, quienes me han acompañado en esta travesía académica con su apoyo constante, palabras de aliento y amistad inquebrantable. Su presencia me ha recordado la importancia de la amistad en la vida.

Esta tesis es un testimonio de la importancia de mantener un equilibrio entre las responsabilidades académicas, la vida personal, cómo el amor de mis hijos y la amistad de mis queridos amigos han sido el motor que me impulsó a seguir adelante. A todos ustedes, les agradezco de corazón por su contribución a este logro.

René Marcelo Sierra Montesinos

AGRADECIMIENTO

Quisiera expresar mi sincero agradecimiento a todas las personas que han contribuido de manera significativa en la realización de esta tesis de maestría.

En primer lugar, a mis queridos hijos, quienes han sido mi mayor fuente de motivación y amor incondicional a lo largo de este desafiante camino. Este logro es, en gran parte, un tributo a ustedes y a la esperanza que depositan en mí.

A mis respetados docentes, agradezco profundamente su orientación experta, paciencia y apoyo constante a lo largo de este proceso académico. Sus conocimientos y perspectivas fueron fundamentales para dar forma a esta investigación y mi crecimiento como estudiante.

A todas las personas que participaron en la investigación, ya sea como colaboradores, entrevistados o fuentes de datos, su valiosa contribución fue esencial para el éxito de este proyecto. Sin su generosidad y disposición para compartir su conocimiento, este trabajo no habría sido posible.

René Marcelo Sierra Montesinos

Tabla de Contenido

DISEÑO DE UNA MÁQUINA VIRTUAL Y ANÁLISIS DE SUS VULNERABILIDADES CON FINES PRÁCTICOS: SERVIDOR FTP, SERVIDOR DE BASE DE DATOS, INYECCIONES SQL Y DE COMANDOS.

AUTOR(ES):

CRISTIAN ANDRES GUACHAMIN HERNANDEZ
RENÉ MARCELO SIERRA MONTESINOS

Resumen

La seguridad informática es un aspecto crítico en la gestión de sistemas y redes, especialmente en entornos que manejan datos sensibles y servicios esenciales. El presente proyecto presenta el diseño y análisis de una máquina virtual configurada con un servidor FTP con el servicio VSFTPD y un servidor de bases de datos con PostgreSQL, con el objetivo de identificar y estudiar vulnerabilidades en estos servicios.

El estudio se enfoca en analizar vulnerabilidades configuradas a propósito dentro de los archivos de configuración, manejo de nombres de usuario predecibles y contraseñas vulnerables dentro de cada uno de los servicios, demostrando que una configuración indebida puede provocar un gran riesgo tanto a la información como a los servicios en específico, además se muestra la creación de una pequeña aplicación web desarrollada en Python para poder demostrar el riesgo de inyecciones SQL y de comandos del sistema.

El proceso de diseño de la máquina virtual involucró la configuración intencional de parámetros inseguros en los archivos de configuración de cada uno de los servicios (VSFTPD, POSGRESQL), creando un entorno controlado y educativo para la simulación de ataques. Se utilizaron herramientas de escaneo como NMAP y técnicas de explotación para identificar las vulnerabilidades, permitiendo una comprensión profunda de los riesgos asociados y las posibles consecuencias de un ataque exitoso.

Los resultados del análisis revelaron múltiples vulnerabilidades críticas que podrían ser explotadas para comprometer la seguridad del sistema, destacando la importancia de una configuración segura desde la fase inicial del diseño de cualquier sistema. Entre las vulnerabilidades encontradas se pudo observar el acceso anónimo al servidor FTP, pudiendo visualizar todas las carpetas del sistema, y copiar archivos del mismo. Dentro de la base de datos se pudo acceder a la misma desde un equipo externo, crear usuarios y modificar datos dentro de la base. Las inyecciones SQL y de comandos fueron

identificadas como críticas también, permitiendo el acceso no autorizado y la visualización de datos sensibles.

Las conclusiones del estudio enfatizan la necesidad de aplicar buenas prácticas de seguridad y la implementación de medidas correctivas en la configuración de servidores FTP y de bases de datos. Además, se proponen recomendaciones específicas para mitigar las vulnerabilidades encontradas, con el fin de fortalecer la seguridad en entornos de producción.

Este trabajo contribuye al campo de la seguridad informática al ofrecer un enfoque práctico y educativo para la identificación y mitigación de vulnerabilidades, facilitando la formación de profesionales en un entorno simulado pero realista.

Abstract

Cybersecurity is a critical aspect of system and network management, especially in environments handling sensitive data and essential services. This project presents the design and analysis of a virtual machine configured with an FTP server using VSFTPD and a database server using PostgreSQL, with the objective of identifying and studying vulnerabilities in these services.

The study focuses on analyzing intentionally configured vulnerabilities within configuration files, handling of predictable usernames, and vulnerable passwords in each of the services, demonstrating that improper configuration can pose significant risks to both information and services. Additionally, it includes the development of a small web application created in Python to demonstrate the risk of SQL and system command injections.

The process of designing the virtual machine involved the intentional configuration of insecure parameters in the configuration files of each service (VSFTPD, PostgreSQL), creating a controlled and educational environment for simulating attacks. Scanning tools such as NMAP and exploitation techniques were used to identify vulnerabilities, allowing a deep understanding of associated risks and the potential consequences of a successful attack.

The analysis results revealed multiple critical vulnerabilities that could be exploited to compromise system security, highlighting the importance of secure configuration from the initial design phase of any system. Among the vulnerabilities found, anonymous access to the FTP server was observed, allowing visibility of all system folders and copying of files. Within the database, access from an external machine was possible, and users could be created and data modified. SQL and command injections were also identified as critical, enabling unauthorized access and visibility of sensitive data.

The study's conclusions emphasize the need to apply good security practices and implement corrective measures in the configuration of FTP and database servers. Additionally, specific recommendations are proposed to mitigate the identified vulnerabilities to strengthen security in production environments.

This work contributes to the field of cybersecurity by providing a practical and educational approach to identifying and mitigating vulnerabilities, facilitating the training of professionals in a simulated but realistic environment.

1. Introducción

En el mundo empresarial actual, la virtualización de servidores se destaca como una parte crucial de la modernización y adopción de nuevas tecnologías. Se utiliza en sistemas informáticos para abordar y, en muchos casos, eliminar el problema de servidores subutilizados. Esta práctica optimiza el uso de los recursos del servidor, mejora su disponibilidad, facilita la recuperación y descentraliza los servicios de administración [1].

Hoy en día la virtualización se encuentra presente en equipos de escritorio razón por la cual se ha visto un crecimiento vertiginoso en su uso e implementación con ello ha incrementado las inquietudes en cuanto a la seguridad de la información en equipos virtualizados, y cuáles serían las mejores prácticas en cuanto a su correcta configuración y mitigación de posibles vulnerabilidades [2].

En este contexto, un servidor FTP (File Transfer Protocol) juega un papel fundamental en el intercambio eficiente de archivos en entornos de red. Facilita la transferencia de datos de manera rápida y confiable entre sistemas remotos, siendo esencial en entornos empresariales, educativos y de investigación [3]. Sin embargo, su implementación conlleva riesgos, como la exposición de información a amenazas de seguridad en transmisiones no cifradas y la posibilidad de autenticación débil.

Por otro lado, tenemos al servidor de bases de datos que es esencial en entornos informáticos, facilitando el almacenamiento y gestión eficiente de datos estructurados a través de un Sistema de Gestión de Bases de Datos (DBMS). Ofrece beneficios como la organización efectiva de la información y el acceso concurrente, pero presenta riesgos potenciales, como vulnerabilidades de seguridad, accesos no permitidos y denegaciones de servicios [4].

Con esto en mente, el presente proyecto se enfoca en el diseño de una máquina virtual con el propósito de realizar un análisis minucioso de las posibles debilidades al tener una configuración o instalación incorrecta, contraria a las mejores prácticas o guías de instalación determinadas por cada fabricante o desarrollador. Para llevar a cabo esta investigación, se han seleccionado dos componentes fundamentales en un entorno virtualizado: un servidor FTP y un servidor de base de datos. Utilizando la herramienta VMWare versión 8, se levantará un servidor con el sistema operativo CentOS versión 7, donde se probarán estos servidores, además de analizar las vulnerabilidades asociadas a inyecciones SQL y de comandos.

2. Determinación del Problema

A medida que las organizaciones adoptan entornos virtualizados para mejorar la flexibilidad y escalabilidad de sus sistemas, también se intensifican los riesgos asociados a posibles amenazas de seguridad. La exposición de servicios críticos como servidores FTP y bases de datos a través de una máquina virtual plantea interrogantes sobre la capacidad de estas infraestructuras para resistir intentos malintencionados de explotación.

Las vulnerabilidades, como las inyecciones SQL y de comandos, representan amenazas potenciales que podrían comprometer la integridad, confidencialidad y disponibilidad de la información almacenada y gestionada por estos servicios. La falta de conciencia y comprensión exhaustiva de las posibles vulnerabilidades en el diseño de la máquina virtual puede resultar en sistemas propensos a ataques cibernéticos, lo que pone en riesgo la seguridad de los datos y la continuidad de las operaciones.

3. Marco teórico referencial

3.1 Máquinas Virtuales (VMs)

Las máquinas virtuales son entornos de software que imitan sistemas de hardware completos, permitiendo la ejecución de múltiples sistemas operativos y aplicaciones en un solo servidor físico. Estas VMs desempeñan un papel fundamental en la consolidación de servidores, la gestión eficiente de recursos y la implementación flexible de servicios [5].

3.2 Virtualización.

La virtualización es una herramienta que funciona como cualquier otro programa, con la diferencia de que se ejecuta sobre un entorno que no es físico, es decir, se ejecuta sobre el sistema operativo principal de una computadora o servidor. La virtualización permite instalar y hacer uso de un sistema operativo nuevo, el cual puede interactuar con los recursos que se le asignen [6].

La virtualización también ofrece ventajas en la seguridad y movilidad de los sistemas. La realización de copias de seguridad es un servicio prestado por cualquier software de virtualización que, además, al permitir copias de la máquina virtual completa, facilita su traslado a nuevas ubicaciones [7].

3.3 Hipervisores.

Los hipervisores mejor conocidos como supervisores de máquinas virtuales, son elementos esenciales en la virtualización de sistemas ya que estos permiten la creación y gestión de las máquinas virtuales (VMs) en un entorno de hardware o medio físico, facilitando la consolidación de servidores y la optimización de recursos. El hipervisor es denominado comúnmente monitor de máquina virtual (VMM) y es el encargado de validar todas las peticiones e instrucciones de los sistemas virtuales a la CPU, supervisando todas las ejecuciones que requieran cambios de privilegios [1].

3.3.1 Características de Hipervisores:

Isolación de Recursos: Los hipervisores garantizan la separación completa de recursos entre las VMs, lo que impide que una VM afecte el rendimiento o la seguridad de otras. Esta característica es esencial para la seguridad y estabilidad del entorno virtualizado [8].

Gestión de Recursos: Los hipervisores permiten la asignación y gestión flexible de recursos, como CPU, memoria y almacenamiento, a cada VM según sus necesidades. Esto facilita la optimización de recursos y mejora la eficiencia operativa [8].

Portabilidad: Las VMs creadas en un hipervisor pueden ser fácilmente transferidas entre servidores físicos compatibles. Esto brinda flexibilidad y agilidad en la administración de cargas de trabajo [8].

Snapshotting: Los hipervisores permiten la creación de instantáneas (snapshots) de VMs en un estado específico. Esto es valioso para la recuperación ante desastres y la realización de pruebas sin afectar la VM original [8].

Migración en Vivo: Algunos hipervisores admiten la migración en vivo de VMs de un servidor físico a otro sin interrupciones de servicio, lo que garantiza alta disponibilidad y continuidad operativa [8].

Aislamiento de Seguridad: Los hipervisores proporcionan un alto nivel de aislamiento de seguridad entre VMs, lo que reduce el riesgo de propagación de amenazas y malware dentro del entorno virtualizado [8].

3.3.2Tipos de Hipervisores

Existen dos tipos principales de hipervisores:

Hipervisores de Tipo 1 (Bare-Metal): Estos hipervisores se ejecutan directamente sobre el hardware físico del servidor, sin necesidad de un sistema operativo anfitrión. Son altamente eficientes y se utilizan comúnmente en entornos empresariales. Ejemplos incluyen VMware vSphere/ESXi, Microsoft Hyper-V y KVM [8].

Hipervisores de Tipo 2 (Hosted): Estos hipervisores se ejecutan sobre un sistema operativo anfitrión, que a su vez se ejecuta en el hardware físico. Son adecuados para entornos de desarrollo y pruebas, pero generalmente son menos eficientes que los de Tipo 1 debido a la capa adicional. Ejemplos incluyen Oracle VirtualBox y VMware Workstation [8].

3.4 Servidores.

Los servidores son componentes esenciales en la infraestructura de tecnología de la información (TI) de una organización [9]. Estos sistemas están diseñados para proporcionar servicios, recursos y aplicaciones a otros dispositivos o usuarios en una red. Los servidores pueden variar en función de su propósito y funcionalidad, y se clasifican en varios tipos según sus características y roles.

3.4.1 Tipos de Servidores

Servidores de Archivos o Servidor FTP (Protocolo de Transferencia de Archivos)**:** Estos servidores están diseñados para almacenar y gestionar archivos compartidos en una red. Facilitan el acceso centralizado a documentos y datos, lo que mejora la colaboración en entornos empresariales [9].

Servidores Web: Los servidores web son responsables de alojar sitios web y proporcionar contenido web a través de protocolos como HTTP. Son fundamentales para la disponibilidad y accesibilidad de sitios en línea [9].

Servidores de Correo Electrónico (Email Servers): Estos servidores gestionan el flujo de correo electrónico, almacenando y entregando mensajes electrónicos a usuarios dentro de una organización o en Internet [10].

Servidores de Bases de Datos: Los servidores de bases de datos almacenan, gestionan y recuperan datos en una base de datos. Son fundamentales para aplicaciones que requieren almacenamiento y recuperación eficiente de información [10].

Servidores de Aplicaciones: Estos servidores ejecutan aplicaciones empresariales y proporcionan servicios relacionados con la lógica empresarial, como la gestión de transacciones y la escalabilidad [11].

Servidores Proxy: Los servidores proxy actúan como intermediarios entre los clientes y otros servidores. Pueden mejorar la seguridad y el rendimiento al cachear contenido y controlar el tráfico [11].

Firewalls: Un servidor Firewall es un componente esencial de la infraestructura de red que actúa como un punto de control y filtro para el tráfico entrante y saliente. Siendo su principal función el proteger una red o sistema de amenazas de seguridad [11].

3.5 Software Libre.

El software libre es aquel que ofrece a los usuarios la libertad de ejecutar, modificar, copiar y distribuir el software. No se limita simplemente al aspecto económico, sino que se centra en la libertad del usuario. Los programas de software libre deben cumplir con ciertos principios, como la posibilidad de modificar el código fuente y la distribución libre, fomentando asi la colaboración y la formación de comunidades. Este enfoque se ha consolidado como una filosofía que promueve no solo el acceso sino tambien la particiación activa de la comunidad en el desarrollo y mejora del software [10].

3.6 Software de Código Abierto (Open Source)

El software de código abierto se destaca por su desarrollo colaborativo por parte de la comunidad y el acceso al código fuente, lo que permite su revisión y modificación. Los programadores pueden realizar cambios en el software y, en algunos casos, ofrecer versiones modificadas, a menudo a un costo menor que el software propietario. El movimiento de código abierto se basa en la colaboración y la descentralización para resolver problemas en diversas industrias y comunidades [10].

3.7 Vulnerabilidades.

Las vulnerabilidades en sistemas de información representan debilidades o fallos que pueden ser explotados por individuos malintencionados para comprometer la seguridad de los sistemas y la integridad de los datos [11].

3.7.1 Clases de Vulnerabilidades

Vulnerabilidades de Software: Estas vulnerabilidades están relacionadas con errores en el código de software, que pueden ser explotados para llevar a cabo ataques. Pueden incluir desbordamientos de búfer, inyecciones de código y errores de validación de entradas [11].

Vulnerabilidades de Hardware: Las vulnerabilidades de hardware se refieren a fallos o debilidades en componentes físicos de un sistema, como microprocesadores o dispositivos de almacenamiento [11].

Vulnerabilidades de Red: Estas vulnerabilidades se relacionan con debilidades en la infraestructura de red, como puertos abiertos, configuraciones de enrutadores, switches mal configurados, y protocolos desactualizados [11].

Vulnerabilidades de Configuración: Las vulnerabilidades de configuración resultan de ajustes incorrectos o inseguros en sistemas, aplicaciones o dispositivos. Esto puede incluir permisos de acceso inapropiados y configuraciones no seguras [11].

3.7.2 Tipos de Vulnerabilidades

Desbordamiento de Búfer (Buffer Overflow): Esta vulnerabilidad se produce cuando un programa permite que datos ingresados excedan el espacio asignado en la memoria, lo que puede llevar a la ejecución de código malicioso [14].

Inyección de Código (Code Injection): Implica la inserción de código malicioso en una aplicación o sistema para ejecutar comandos no autorizados [14].

Cross-Site Scripting (XSS): Los ataques de XSS permiten a un atacante inyectar scripts maliciosos en páginas web visitadas por otros usuarios [14].

Inyección SQL: En este tipo de vulnerabilidad, los atacantes insertan comandos SQL maliciosos en formularios web o aplicaciones para acceder o modificar bases de datos [14].

Autenticación Deficiente: Las vulnerabilidades de autenticación deficiente se relacionan con contraseñas débiles o la falta de autenticación adecuada, lo que permite el acceso no autorizado [14].

3.8 Hackers.

Los hackers son individuos con habilidades técnicas y conocimientos avanzados en informática que utilizan su experiencia para acceder a sistemas, redes y dispositivos de forma creativa y no necesariamente maliciosa. Los hackers se dividen en distintas categorías en función de sus intenciones y acciones [15].

3.8.1 Características de los Hackers

Conocimiento Técnico: Los hackers poseen un profundo conocimiento técnico en áreas como programación, sistemas operativos, redes y seguridad informática [15].

Creatividad: La creatividad es una característica clave de los hackers, ya que utilizan su ingenio para resolver problemas y encontrar soluciones innovadoras [15].

Ética Diversa: Los hackers pueden tener diversas éticas. Algunos trabajan en la mejora de la seguridad informática (hackers éticos), mientras que otros pueden realizar actividades ilegales (hackers maliciosos) [15].

Curiosidad: La curiosidad es una característica común entre los hackers. Tienen un deseo innato de explorar y entender cómo funcionan los sistemas [15].

3.8.2 Tipos de Hackers

Hacker Ético (White Hat Hacker): Estos hackers utilizan sus habilidades para identificar y corregir vulnerabilidades de seguridad en sistemas y redes. Trabajan dentro de la legalidad y con el permiso de los propietarios de los sistemas [15].

Hacker Malicioso (Black Hat Hacker): Los hackers maliciosos utilizan sus habilidades para actividades ilegales, como robar datos, realizar fraudes o causar daño a sistemas y redes [15].

Hacker Gris (Gray Hat Hacker): Estos hackers operan en una zona intermedia entre el bien y el mal. Pueden realizar acciones sin el permiso explícito de los propietarios de sistemas, pero sin intenciones maliciosas [15].

Script Kiddie: Los scripts kiddies son individuos con habilidades limitadas que utilizan scripts y herramientas desarrolladas por otros para llevar a cabo ataques sin comprender completamente cómo funcionan [15].

3.9 Ciberataques.

Los ciberataques son acciones maliciosas llevadas a cabo por individuos o grupos con la intención de comprometer la seguridad de sistemas informáticos, redes, dispositivos y datos [16]. Estos ataques pueden variar en complejidad y escala, y representan una amenaza constante en el entorno digital.

3.9.1 Características de los Ciberataques

Intencionalidad Maliciosa: Los ciberataques se llevan a cabo con la intención deliberada de causar daño, robar información o interrumpir operaciones. Los atacantes tienen motivos específicos para sus acciones [16].

Utilización de Vulnerabilidades: Los atacantes explotan vulnerabilidades en sistemas, aplicaciones o redes para llevar a cabo sus ataques. Estas vulnerabilidades pueden ser errores de software, configuraciones incorrectas o debilidades en la seguridad [16].

Anonimato y Encubrimiento: Los atacantes a menudo utilizan técnicas para ocultar su identidad, como el uso de servidores proxy y técnicas de enmascaramiento de direcciones IP, dificultando su rastreo [16].

Innovación Constante: Los ciberataques evolucionan con el tiempo a medida que los atacantes desarrollan nuevas técnicas y herramientas. La innovación constante es una característica distintiva de los ciberataques [16].

3.9.2 Tipos de Ciberataques

Malware: El malware (software malicioso) incluye virus, gusanos, troyanos y ransomware, diseñados para infectar sistemas y causar daño, robar datos o extorsionar a las víctimas [16].

Ataques de Ingeniería Social: Los atacantes utilizan la manipulación psicológica para engañar a las personas y obtener información confidencial, contraseñas o acceso a sistemas [16].

Ataques de Denegación de Servicio (DDoS): Estos ataques inundan un sistema o red con tráfico falso o sobrecargan recursos, lo que provoca la interrupción de servicios [16].

Phishing: Los ataques de phishing implican el envío de correos electrónicos o mensajes falsos que parecen legítimos para engañar a las víctimas y obtener información personal o financiera [16].

3.10 Software Utilizado

VSFTPD

Very Secure FTP Daemon, es un servidor FTP de código abierto para sistemas Unix y Linux, que se utiliza para facilitar la transferencia de archivos entre sistemas en una red [17].

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de propósito general que ha ganado popularidad debido a su robustez, capacidad de extensión y conformidad con los estándares SQL [18].

Nmap

Network Mapper, es una herramienta de código abierto utilizada para descubrir, mapear host y servicios en una red [19].

Hydra

Es una herramienta de software libre utilizada para realizar ataques de fuerza bruta en diversos servicios de autenticación, como servidores FTP, SSH, bases de datos, correos electrónicos, aplicaciones web, entre otros. El objetivo principal de Hydra es probar la robustez de las credenciales (usuario/contraseña) de dichos servicios, intentando diferentes combinaciones de nombres de usuario y contraseñas hasta encontrar la correcta [20].

Kali Linux

Kali Linux es una distribución de Linux basada en Debian diseñada específicamente para realizar pruebas de penetración y análisis forense digital. Desarrollada y mantenida por Offensive Security, Kali Linux proporciona una amplia gama de herramientas de seguridad y hacking ético que son utilizadas por profesionales de ciberseguridad para evaluar la seguridad de sistemas informáticos, redes y aplicaciones [21].

Python

Es un lenguaje de programación altamente utilizado por su facilidad en el aprendizaje y por su versatilidad en el desarrollo web, desarrollo de software y ciencia de datos [22].

Flask

Es un framework basado en Python que permite generar aplicaciones web de manera rápida y sencilla [23].

Pip

Es un sistema de gestión de paquetes de Python que se utiliza para instalar y administrar las bibliotecas externas. Este permite que todas sus fuentes sean fáciles de instalar y actualizar [24].

Venv

Es un módulo que permite la gestión de entornos virtuales para Python, estos entornos son ligeros y permiten estar aislados de los directorios del sistema.[25]

4. Materiales y metodología

En la siguiente sección se describe el proceso de identificación y análisis de vulnerabilidades en los servidores FTP y de base de datos, así como las inyecciones SQL y de comandos dentro de un entorno virtualizado. Esta fase se desarrolla en tres etapas fundamentales: la selección de herramientas de análisis adecuadas, la planificación detallada de la ejecución de estas herramientas y la ejecución misma del análisis en los servidores FTP y de base de datos.

4.1 Materiales a Utilizar

Hemos determinado los siguientes materiales a utilizar:

Servidor:

SO: Centos 7

CPU: Core I7

Memoria: 1 GB

Disco: 20Gb

Hipervisor: VMWare Workstation 16

De igual manera se utiliza pendrives, imágenes ISO para la implementación del servidor, al igual que un acceso a internet.
Se ha optado también por el uso de software libre compatible entre el servidor y el SO de CentOS en su versión 7.

4.2 Metodología

La metodología implementada para el análisis de vulnerabilidades en un entorno controlado se estructuró en tres fases esenciales: preparación y configuración del entorno virtual, identificación y análisis de vulnerabilidades, y análisis de resultados con propuestas de soluciones. Cada fase está diseñada para abordar aspectos específicos de la seguridad informática en los servicios desplegados, garantizando un enfoque integral y sistemático.

Fase 1: Preparación y Configuración del Entorno Virtual

En esta primera etapa, se centra en la preparación y configuración de un entorno virtualizado que permita simular un servidor FTP y un servidor de base de datos expuestos a vulnerabilidades.

- **Selección de Recursos de Hardware:** Se determinó la cantidad de memoria RAM, espacio de almacenamiento y núcleos de CPU necesarios para soportar los servicios de FTP y base de datos en la máquina virtual. Esta selección se realizó en función de las demandas de cada servicio, basándose en configuraciones típicas de servidores en producción.
- **Justificación del Software de Virtualización:** Se optó por el uso de software de virtualización como VMware debido a su amplia compatibilidad con diversos sistemas operativos y su capacidad de replicar entornos controlados. Esta plataforma permite una fácil administración y configuración del entorno necesario para implementar servidores, facilitando el acceso remoto y la conectividad a través de redes virtualizadas, aspectos críticos para simular condiciones reales de trabajo.
- **Implementación de Servidores:** Se instalaron y configuraron un servidor FTP y un servidor de base de datos PostgreSQL en el entorno virtual. Los servidores fueron configurados inicialmente con parámetros inseguros para fines de análisis, siguiendo las pautas estándar sin implementar medidas de seguridad adicionales. Este enfoque permitió evaluar el impacto de las vulnerabilidades en un entorno vulnerable por diseño.

Fase 2: Identificación y Análisis de Vulnerabilidades

En esta fase, se llevó a cabo la identificación de las vulnerabilidades presentes en los servicios implementados, enfocándose en servidores FTP, bases de datos, y aplicaciones web susceptibles a inyecciones SQL y de comandos.

- **Selección de Herramientas de Análisis:** Se seleccionaron herramientas reconocidas en el campo de la ciberseguridad que

ofrecieran un análisis detallado y exhaustivo de los servicios expuestos. Las herramientas elegidas fueron:

- **Nmap:** Herramienta utilizada para detectar servicios activos, puertos abiertos y versiones de software. Se seleccionó por su capacidad de descubrir posibles vulnerabilidades en configuraciones de red y servicios.
- **Hydra:** Utilizada para realizar ataques de fuerza bruta en los servidores FTP y PostgreSQL, con el fin de probar la robustez de las credenciales y mecanismos de autenticación. Su eficiencia y flexibilidad en pruebas de contraseñas la convierten en una elección ideal para este tipo de análisis.

La selección de estas herramientas fue basada en su reputación dentro de la comunidad de ciberseguridad y su capacidad de proporcionar un análisis amplio, cubriendo múltiples tipos de vulnerabilidades.

- **Planificación de la Ejecución:** Se desarrolló un plan de ejecución que detalla cómo y cuándo se ejecutarían las herramientas seleccionadas en el entorno virtual. La planificación incluyó procedimientos específicos para escanear los servidores y las aplicaciones web en busca de vulnerabilidades, priorizando la detección de configuraciones inseguras, vulnerabilidades de autenticación, y vulnerabilidades de inyección SQL y de comandos.
- **Ejecución del Análisis:** Se procedió a ejecutar las herramientas seleccionadas en los servidores FTP, PostgreSQL y la aplicación web desarrollada en Python. Los resultados obtenidos fueron recopilados para identificar posibles vulnerabilidades, documentando sus causas y las áreas del sistema afectadas.

Fase 3: Análisis de Resultados y Propuesta de Soluciones

En la tercera fase, se examinaron los resultados obtenidos durante el análisis de vulnerabilidades y se presentaron las soluciones propuestas para mitigar los riesgos identificados.

- **Examen de Resultados:** Los resultados de las vulnerabilidades fueron analizados minuciosamente, clasificando las vulnerabilidades por su nivel de criticidad, frecuencia de ocurrencia, y su impacto potencial en el sistema. El análisis se centró en comprender las causas subyacentes de las vulnerabilidades, especialmente aquellas que afectaban la integridad y confidencialidad de los datos.
- **Propuesta de Soluciones:** Basado en el análisis, se formularon soluciones para mitigar cada vulnerabilidad identificada. Estas soluciones incluyeron la aplicación de parches de seguridad, la configuración de políticas de seguridad más estrictas, y la implementación de mejores prácticas en la configuración de los servidores. Se enfocó en asegurar que las soluciones fueran viables y alineadas con las necesidades del entorno virtualizado.
- **Documentación y Recomendaciones:** Se documentaron detalladamente las soluciones propuestas y las recomendaciones específicas para reforzar la seguridad de los servidores y aplicaciones en el entorno virtual. Además, se generaron reportes con gráficas comparativas para visualizar la efectividad de las medidas de mitigación antes y después de su implementación.

5. Desarrollo del trabajo

En la presente sección se trata sobre el proceso de instalación y configuración de un servidor CentOS, así como en la implementación de un servidor FTP utilizando VSFTPD y un servidor de base de datos Postgresql. Sin embargo, nuestro objetivo principal no será solo establecer estos servicios, sino también identificar y destacar las posibles vulnerabilidades que podrían afectar su seguridad. Abordaremos específicamente la instalación de estos servicios desde una perspectiva de seguridad, prestando especial atención a las configuraciones que podrían dejar al sistema expuesto a riesgos como inyecciones SQL y de comandos. Este enfoque nos permitirá comprender mejor las amenazas potenciales y prepararnos para mitigarlas de manera efectiva en fases posteriores del proceso de análisis y evaluación de vulnerabilidades.

5.1 Instalación del Entorno Servidor CentOS 7

Como primer paso, se procede a realizar la instalación de una máquina virtual utilizando VMware Workstation 16 como hipervisor.

Creación de la máquina virtual: Se selecciona la opción de crear una nueva máquina virtual. En este caso, se utiliza la configuración típica, seleccionando la imagen ISO del sistema operativo y avanzando con los pasos predeterminados (Fig. 1).

Fig. 1: Creación de una nueva máquina virtual.

Configuración de la cuenta y el almacenamiento: Una vez seleccionada la imagen ISO, se configura la cuenta del usuario y la contraseña, que también será la de la cuenta root. Luego, se define la ubicación donde se guardará la máquina virtual (por defecto, en la carpeta "Virtual machines") y se asigna un tamaño de disco de 20 GB, dividido en múltiples archivos (Fig. 2).

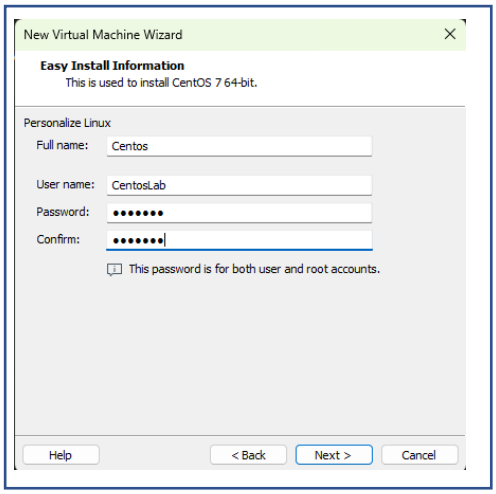

Fig. 2: Configuración y creación de cuenta y claves.

Resumen y finalización: Antes de finalizar, se presenta un resumen de la configuración, permitiendo revisar los parámetros establecidos, como el hardware asignado a la máquina virtual (Fig. 3). Finalmente, se da clic en "Finalizar" para proceder con la instalación de CentOS (Fig. 4).

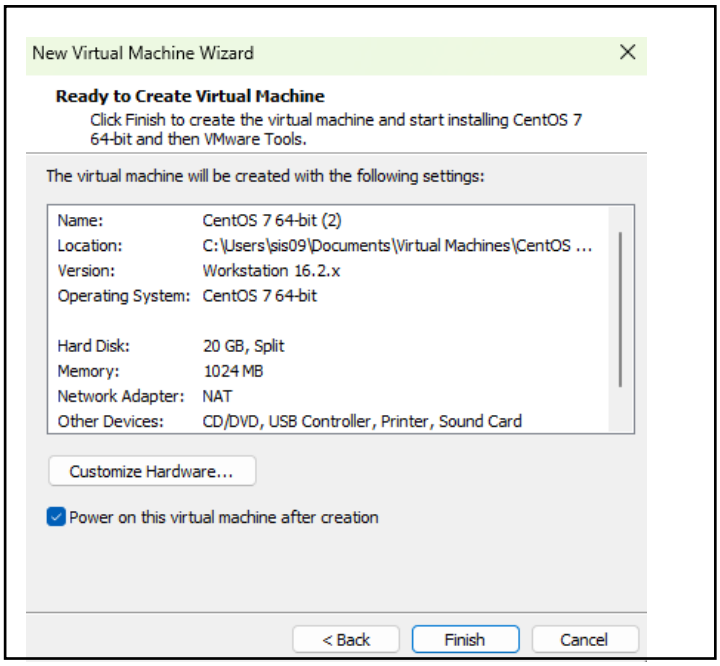

Fig. 3: Resumen y configuración de la máquina virtual.

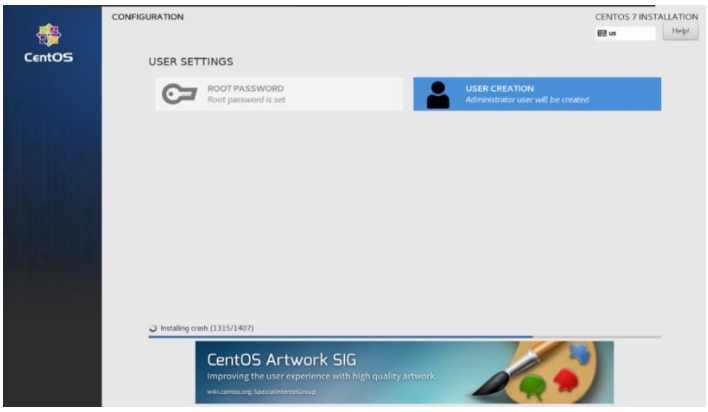

Fig. 4: Finalización de instalación.

Instalación y primer inicio de sesión: Una vez completada la instalación, se muestra la pantalla de inicio de sesión donde se utiliza la cuenta configurada previamente (Fig. 5).

Fig. 5: Inicio de sesión CentOS

5.2 Instalación Servidor FTP

Acceso al terminal y permisos de root: Después de iniciar sesión en CentOS, se abre una ventana de terminal y se ingresa el comando `su` para obtener permisos de root (Fig. 6).

Fig. 6: Ingreso terminal y acceso root.

Instalación de VSFTPD: Con el comando `yum install vsftpd`, se inicia el proceso de instalación del servidor FTP. Durante la instalación, el sistema solicitará dos veces confirmar con la tecla "Y" para proceder con la descarga e instalación de los paquetes necesarios. Una vez completada la instalación, se muestra un mensaje de confirmación (Fig. 7).

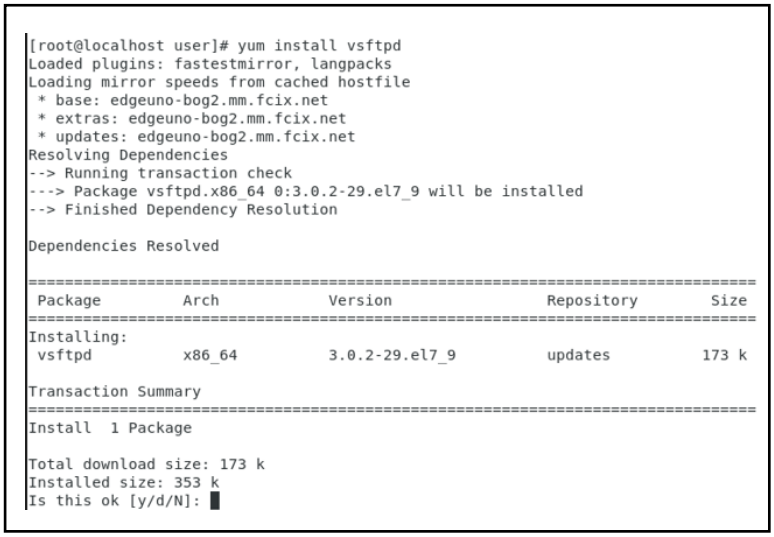

Fig. 7: Instalación de Servidor FTP.

Estado del servicio FTP: Al finalizar la instalación, se verifica el estado del servicio utilizando el comando `systemctl status vsftpd`. Inicialmente, el servicio aparecerá inactivo. Para iniciarlo y habilitarlo permanentemente, se ejecutan los comandos `systemctl start vsftpd` y `systemctl enable vsftpd` (Fig. 8).

```
[root@localhost user]# systemctl status vsftpd
● vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@localhost user]#
```

Fig. 8: Estado servicio FTP.

Configuración del firewall: Para evitar que el firewall bloquee los puertos 20 y 21 necesarios para el funcionamiento de FTP, se añaden con el siguiente comando (Fig. 9).

```bash
firewall-cmd --permanent --add-port=20-21/tcp
firewall-cmd --permanent --add-port=40000-41000/tcp
firewall-cmd --reload
```

```
[root@localhost centoslab]# firewall-cmd --permanent --add-port=20-21/tcp
success
[root@localhost centoslab]# firewall-cmd --permanent --add-port=40000-41000/tcp
success
[root@localhost centoslab]# firewall-cmd --reload
success
```

Fig. 9: Actualización de puertos en Firewall

Modificación del archivo de configuración de VSFTPD: Antes de modificar el archivo de configuración, se realiza una copia de seguridad con el comando `cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.origbackup`. Luego, se edita el archivo con el comando `vi /etc/vsftpd/vsftpd.conf` para establecer una configuración intencionalmente insegura con el fin de evaluar vulnerabilidades (Fig. 10).

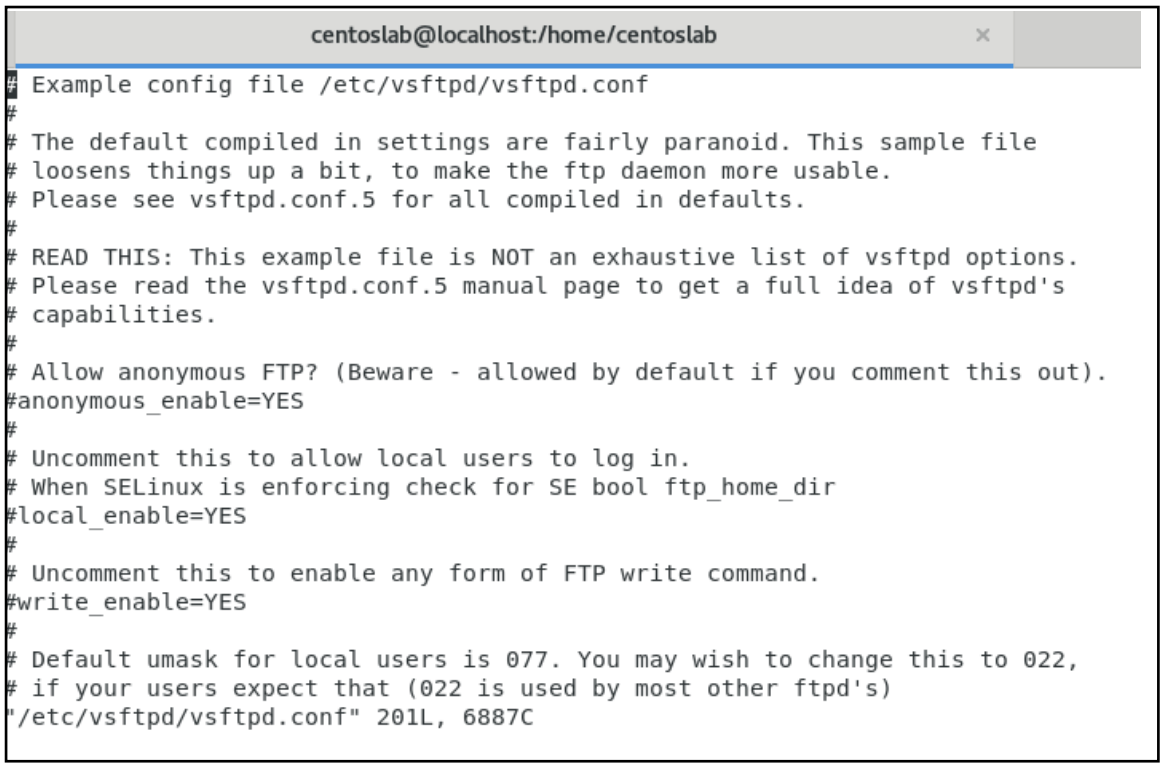

Fig. 10: Edición archivo de configuración VSFTPD

En el archivo de configuración se determinaron los siguientes parámetros para mantener una configuración insegura:

- **anonymous_enable=YES**: Permite acceso anónimo, lo que expone el servidor a cualquier usuario sin autenticación.
- **anon_root=/**: Define el directorio raíz para los usuarios anónimos. En este caso, si el root del sistema de archivos / se utiliza, se otorga un acceso potencialmente peligroso.
- **write_enable=YES**: Permite que cualquier usuario (incluyendo anónimos) pueda ejecutar comandos de escritura (como eliminar o modificar archivos).
- **anon_upload_enable=YES**: Permite que los usuarios anónimos suban archivos al servidor FTP, lo que puede derivar en ataques como la carga de archivos maliciosos.
- **anon_mkdir_write_enable=YES**: Permite a los usuarios anónimos crear directorios, lo cual puede ser explotado para alojar contenido no deseado o ejecutar código malicioso.

- **chroot_local_user=NO**: No habilitar `chroot` para usuarios locales puede permitir que estos accedan a todo el sistema de archivos, comprometiendo la seguridad del servidor.
- **local_root=/**: Al igual que con `anon_root`, establece la raíz del sistema de archivos como el directorio de inicio para usuarios locales, lo que podría permitir un acceso inadecuado a los archivos del sistema.
- **xferlog_enable=NO**: La falta de registro de transferencias dificulta el monitoreo de actividades y puede hacer que un administrador no detecte actividades sospechosas.
- **idle_session_timeout=NO** y **data_connection_timeout=NO**: La falta de límites de tiempo para las sesiones o conexiones de datos puede dejar sesiones abiertas indefinidamente.

Reinicio del servidor FTP: Guardados los cambios en el archivo de configuración, se reinicia el servicio de VSFTPD con el comando `` `systemctl restart vsftpd` `` para aplicar los ajustes (Fig. 11).

```
[root@localhost centoslab]# systemctl restart vsftpd
```

Fig. 11: Reinicio Servidor VSFTPD.

Verificación de la IP del servidor FTP: Para obtener la dirección IP del servidor, se ejecuta el comando `` `ifconfig` ``**.** Esta dirección se utilizará para realizar pruebas de conexión desde otro equipo dentro de la misma red (Fig. 12).

```
[root@localhost centoslab]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.137.132  netmask 255.255.255.0  broadcast 192.168.137.255
        inet6 fe80::f0a3:e4de:2387:647f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:2e:bc:8a  txqueuelen 1000  (Ethernet)
        RX packets 456950  bytes 652248950 (622.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 115897  bytes 7214437 (6.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig. 12: Ifconfig servidor FTP

Pruebas con FileZilla: Finalmente, se realiza una conexión al servidor FTP utilizando FileZilla para verificar el correcto funcionamiento del servicio (Fig. 13).

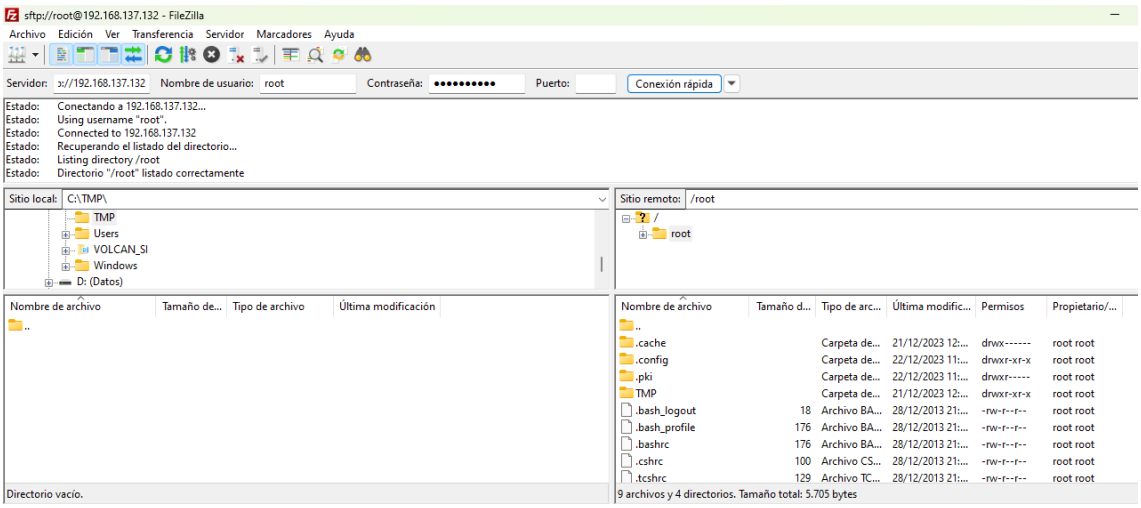

Fig. 13: Acceso mediante FileZilla.

5.3 Instalación Servidor de Base de Datos PostgreSQL

Instalación de paquetes: Para instalar PostgreSQL en CentOS 7, primero necesitamos descargar e instalar los paquetes de PostgreSQL desde el repositorio oficial. Esto se logra con el siguiente comando, el cual añade el repositorio necesario para descargar los paquetes de PostgreSQL:

```bash
yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

Esto permite que nuestro sistema obtenga las versiones más recientes de PostgreSQL desde los repositorios oficiales de PostgreSQL (Fig. 14).

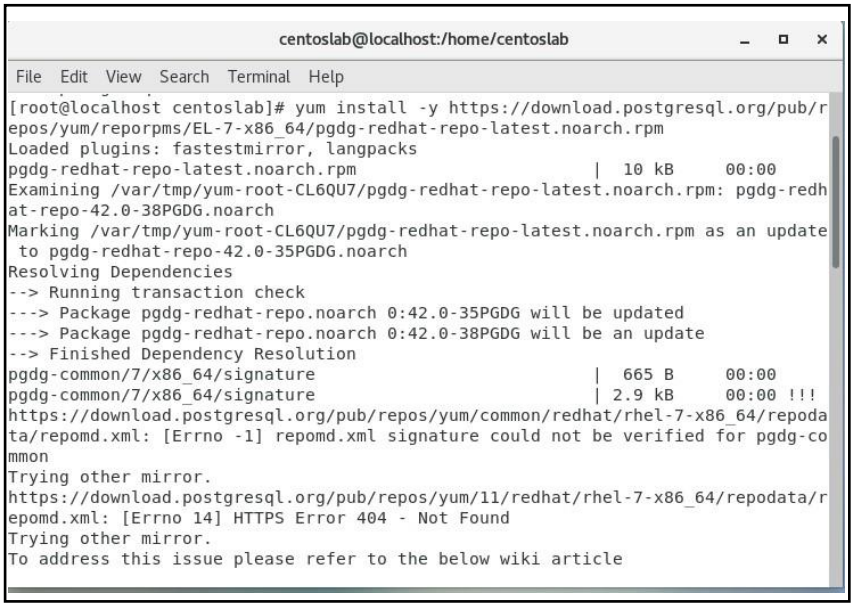

Fig. 14: Instalación paquetes PostgreSQL.

Actualización del sistema: Una vez que se haya agregado el repositorio de PostgreSQL, es importante actualizar los paquetes de software en el sistema. Esto se hace ejecutando el siguiente comando, que asegurará que todos los paquetes estén actualizados a sus versiones más recientes:

```bash
sudo yum update -y
```

La actualización de paquetes garantiza que el sistema funcione de manera eficiente y que las nuevas versiones de PostgreSQL se instalen sin conflictos con versiones anteriores de dependencias (Fig. 15).

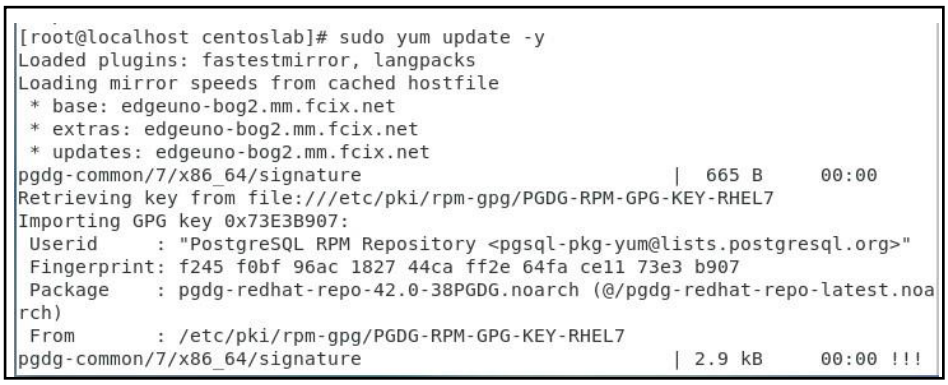

Fig. 15: Actualización de paquetes PostgreSQL.

Instalación de PostgreSQL: Con los paquetes actualizados, ahora podemos instalar PostgreSQL en el sistema. El siguiente comando instala PostgreSQL 13 junto con sus dependencias:

```bash
sudo yum -y install postgresql13-server
```

Este comando instalará tanto el servidor de bases de datos PostgreSQL como las utilidades adicionales necesarias para su funcionamiento, lo que permitirá al sistema ejecutar y administrar bases de datos (Fig. 16).

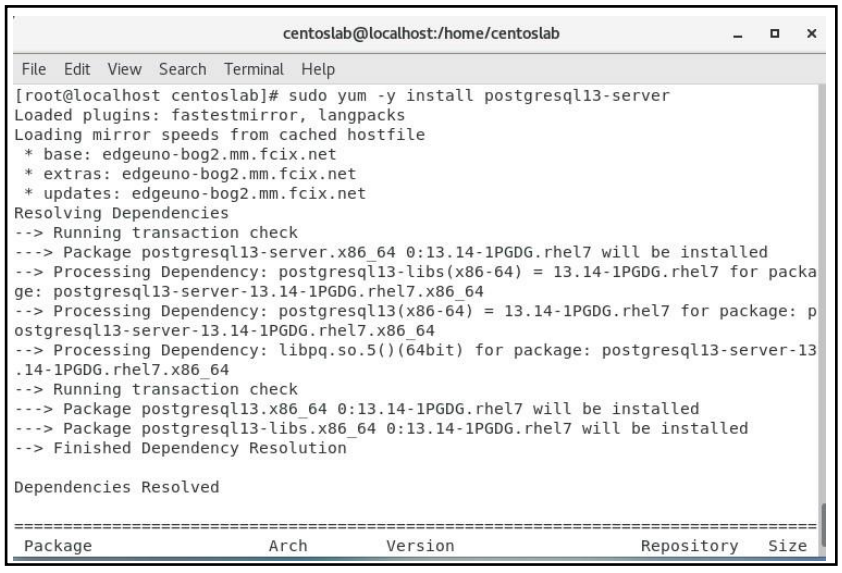

Fig. 16: Instalación de PostgreSQL.

Inicialización de la base de datos: Después de la instalación, debemos inicializar la base de datos para que PostgreSQL pueda comenzar a gestionar datos. Esto se hace ejecutando el comando de inicialización de PostgreSQL:

```bash
sudo /usr/pgsql-13/bin/postgresql-13-setup initdb
```

Este proceso crea las estructuras internas necesarias en el servidor para que pueda comenzar a manejar bases de datos, como los directorios y archivos de configuración predeterminados (Fig. 17).

```
Complete!
[root@localhost centoslab]# sudo /usr/pgsql-13/bin/postgresql-13-setup initdb
Initializing database ... OK
```

Fig. 17: Inicializando Base de Datos PostgreSQL.

Habilitación e inicio del servicio PostgreSQL: Para asegurar que PostgreSQL se inicie automáticamente después de cada reinicio del sistema, habilitamos el servicio con el siguiente comando:

```bash
sudo systemctl enable postgresql-13
```

Este comando hace que el servicio se ejecute automáticamente al arrancar el sistema. Luego, iniciamos el servicio para que el servidor comience a funcionar inmediatamente:

```bash
sudo systemctl start postgresql-13
```

Iniciar el servicio permite que las aplicaciones comiencen a conectarse y utilizar la base de datos recién configurada (Figs. 18 y 19).

```
[root@localhost centoslab]# sudo systemctl enable postgresql-13
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-13.ser
vice to /usr/lib/systemd/system/postgresql-13.service.
```

Fig. 18: Habilitación del Servicio PostgreSQL.

```
[root@localhost centoslab]# sudo systemctl start postgresql-13
[root@localhost centoslab]#
```

Fig. 19: Inicio del Servicio PostgreSQL

Verificación del estado del servicio: Para verificar que PostgreSQL esté funcionando correctamente y que no haya errores, utilizamos el siguiente comando que muestra el estado actual del servicio:

```bash
sudo systemctl status postgresql-13
```

Este comando nos proporciona información sobre si el servicio está activo o detenido, además de mostrar cualquier error o advertencia que pueda requerir atención (Fig. 20).

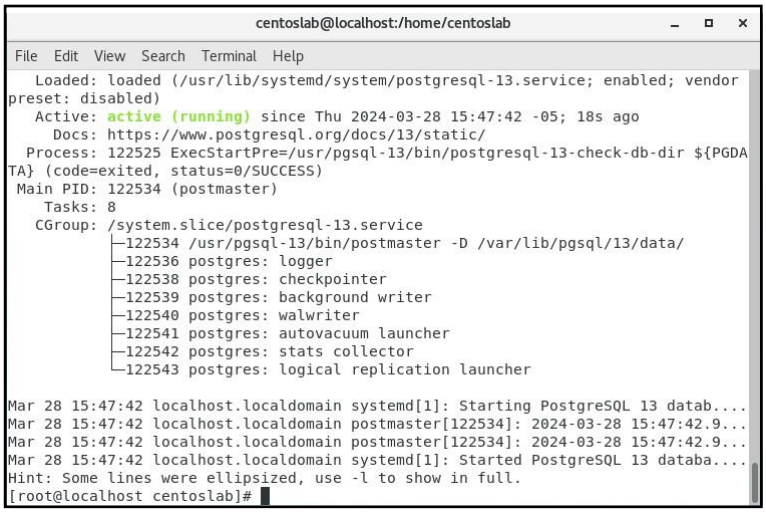

Fig. 20: Estatus del servicio PostgreSQL.

Respaldo de archivos de configuración: Antes de realizar cualquier cambio en los archivos de configuración de PostgreSQL, es recomendable realizar copias de seguridad para poder restaurar los archivos originales si es necesario. Ejecutamos los siguientes comandos para crear copias de los archivos `pg_hba.conf` y `postgresql.conf`:

```bash
cp /var/lib/pgsql/13/data/pg_hba.conf /var/lib/pgsql/13/data/pg_hba.conf.copiaori
cp /var/lib/pgsql/13/data/postgresql.conf /var/lib/pgsql/13/data/postgresql.conf.copiaori
```

Estos archivos de configuración controlan aspectos cruciales del acceso y la operación de PostgreSQL, por lo que tener un respaldo es una práctica esencial (Figs. 21 y 22).

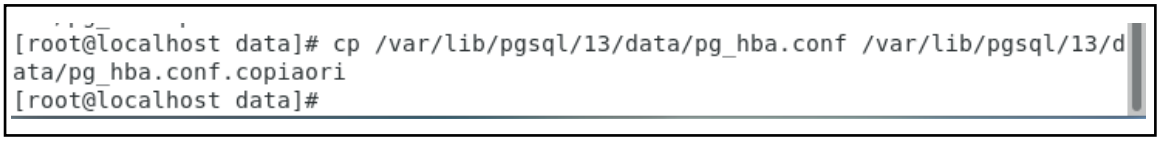

Fig. 21: Copia del archivo de configuración pg_hba.conf.

Fig. 22: Copia del archivo de configuración postgresql.conf.

Edición del archivo `pg_hba.conf`: El archivo `pg_hba.conf` controla quién puede acceder al servidor PostgreSQL y desde qué direcciones IP. Para permitir conexiones remotas desde una red específica, agregamos la siguiente línea en el archivo:

```bash
host    all     all     192.168.137.132/24    md5
```

Esto permite a cualquier usuario (indicando `all`) conectarse a cualquier base de datos desde la subred 192.168.137.132/24 utilizando autenticación MD5. Utilizamos el editor `vi` para realizar esta modificación (Fig. 23):

```bash
vi /var/lib/pgsql/13/data/pg_hba.conf
```

Fig. 23: Comando vi ph_hba.conf

Edición del archivo `postgresql.conf`: El archivo `postgresql.conf` es el principal archivo de configuración del servidor. Para permitir que PostgreSQL escuche conexiones en todas las interfaces de red, editamos la línea `listen_addresses`, reemplazando `localhost` por `*`:

```bash
vi /var/lib/pgsql/13/data/postgresql.conf
```

```
listen_addresses = '*'
```

Este cambio permitirá que el servidor acepte conexiones de cualquier dirección IP, en lugar de solo conexiones locales (Fig. 24).

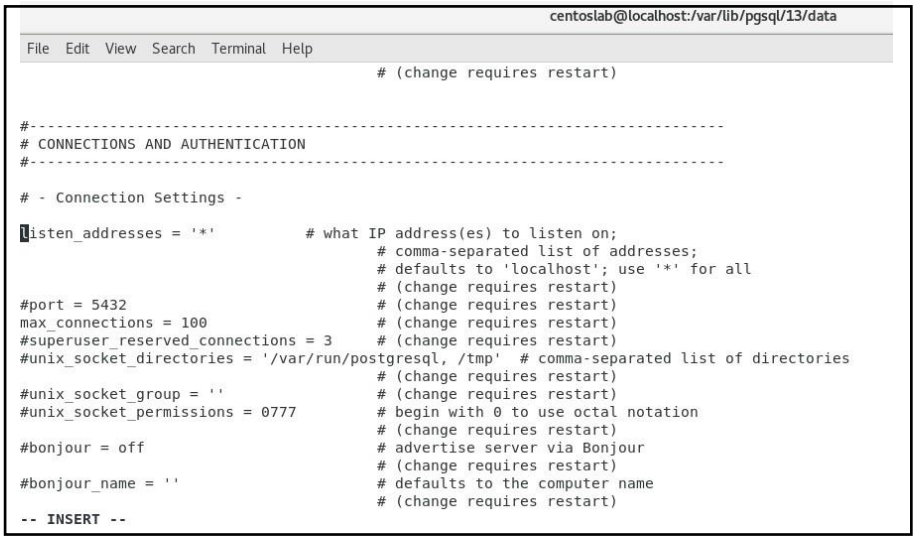

Fig. 24: Modifica archivo posgresql.conf.

Reinicio del servicio: Después de realizar cambios en los archivos de configuración, es necesario reiniciar PostgreSQL para aplicar los nuevos ajustes. Esto se hace ejecutando el siguiente comando:

```bash
sudo systemctl restart postgresql-13
sudo systemctl status postgresql-13
```

El reinicio asegura que el servidor PostgreSQL reconozca y aplique los nuevos valores configurados. Verificamos nuevamente que el servicio esté funcionando correctamente. (Figs. 25 y 26).

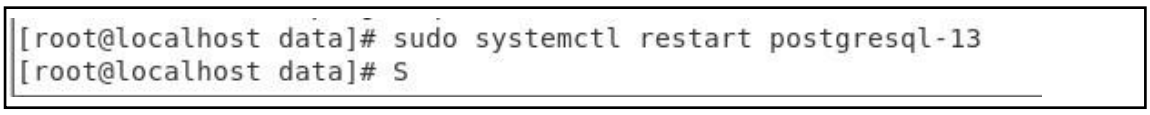

Fig. 25: Reinicio servicio postgresql.

Fig. 26: Estado servicio postgresql.

Cambio de contraseña del usuario `postgres`: Se procede a cambiar la contraseña del usuario administrador `postgres`. Para ello, ingresamos a la consola de PostgreSQL con el siguiente comando:

```bash
sudo -u postgres -i psql
```

Una vez dentro de la consola, cambiamos la contraseña ejecutando el comando:

```bash
\password
```

Este proceso asegura que la cuenta de administración esté protegida con una contraseña segura (Figs. 37 y 38).

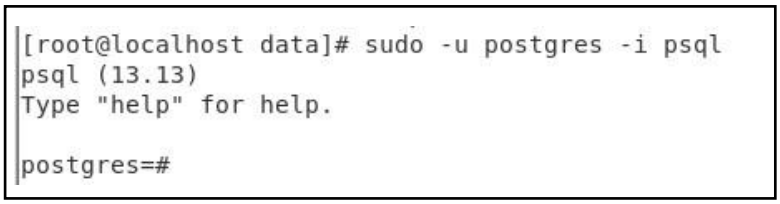

Fig. 27: Ingreso consola postgresql.

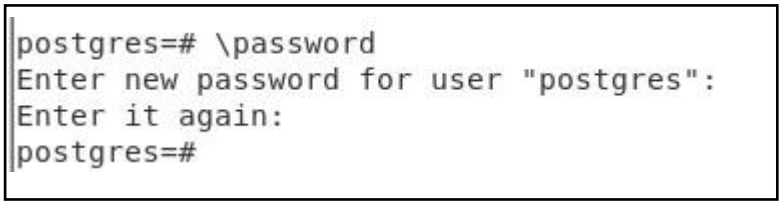

Fig. 28: Cambio contraseña usuario postgres

Modificación del Firewall: Para permitir el acceso remoto a PostgreSQL, necesitamos abrir el puerto 5432 en el firewall, ya que este es el puerto por defecto que utiliza PostgreSQL para aceptar conexiones. Esto se hace con los siguientes comandos:

```bash
firewall-cmd --zone=public --add-port=5432/tcp --permanent
firewall-cmd reload
```

El primer comando abre el puerto 5432 en la zona pública del firewall y la segunda recarga la configuración del firewall para aplicar los cambios (Figs. 39 y 40).

```
[root@localhost data]# firewall-cmd --zone=public --add-port=5432/tcp --permanent
success
[root@localhost data]#
```

Fig. 29: Configuración Firewall

```
[root@localhost data]# firewall-cmd --reload
success
[root@localhost data]#
```

Fig. 30: Reload Firewall

Instalación de pgAdmin: Finalmente, para gestionar de manera remota la base de datos PostgreSQL, utilizamos una herramienta gráfica llamada pgAdmin. Descargamos e instalamos pgAdmin desde [la página oficial](https://www.pgadmin.org/download/) seleccionando la versión adecuada para el sistema operativo en uso. Una vez instalada, configuramos una conexión al servidor PostgreSQL con los datos de acceso correspondientes, lo que nos permitirá gestionar las bases de datos de forma más sencilla y visual (Figs. 31-32).

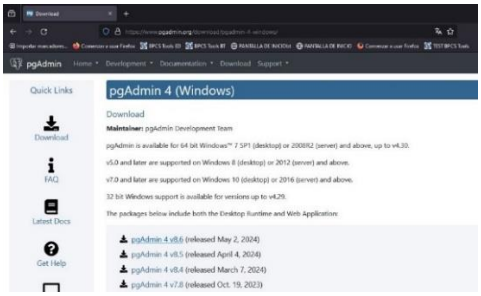

Fig. 31: Versión a descargar PgAdmin.

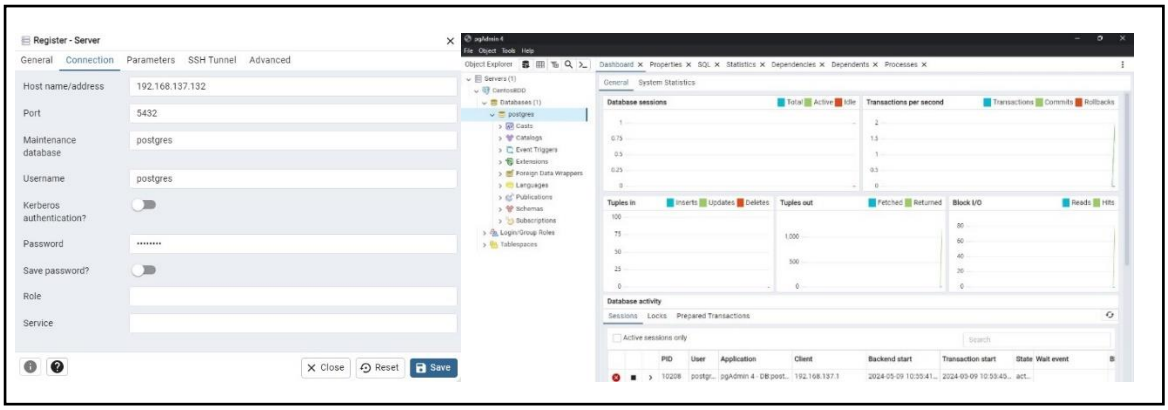

Fig. 32: Finalización de configuración pgAdmin.

5.4 Instalación y desarrollo de una aplicación web

La aplicación web desarrollada en Python tiene como objetivo simular un entorno vulnerable a ataques comunes como inyección SQL y ejecución de comandos del sistema. Este tipo de vulnerabilidades son representativas de los riesgos a los que pueden estar expuestas las aplicaciones web mal configuradas o mal diseñadas. La aplicación fue diseñada intencionalmente para demostrar los efectos de estos ataques y cómo pueden comprometer la seguridad de los datos y del sistema operativo del servidor.

5.4.1 Instalación del Entorno de Desarrollo

Para el desarrollo de nuestra aplicación, es necesario instalar algunas herramientas que permitirán su correcto funcionamiento. A continuación, se detallan los pasos para llevar a cabo esta instalación y configuración.

Actualización del sistema: Como primer paso, realizamos una actualización del sistema ejecutando el siguiente comando en la terminal.

```bash
sudo yum update -y
```

Este comando se asegura de que todos los paquetes y dependencias en el sistema estén actualizados antes de instalar nuevas aplicaciones (Fig. 33).

```
[root@localhost centoslab]# sudo yum update -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: edgeuno-bog2.mm.fcix.net
 * extras: edgeuno-bog2.mm.fcix.net
 * updates: edgeuno-bog2.mm.fcix.net
pgdg-common/7/x86_64/signature
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG
Importing GPG key 0x442DF0F8:
 Userid     : "PostgreSQL RPM Building Project <pgsql-pkg-yum@postgresql.org>"
 Fingerprint: 68c9 e2b9 1a37 d136 fe74 d176 1f16 d2e1 442d f0f8
 Package    : pgdg-redhat-repo-42.0-35PGDG.noarch (@/pgdg-redhat-repo-latest.noarch)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG
pgdg-common/7/x86_64/signature
```

Fig. 33: Actualización del Sistema.

Instalación de Python: Después de la actualización, instalamos Python, que será el lenguaje de programación utilizado para desarrollar nuestra aplicación web.

```bash
sudo yum install -y python3 python3-pip
```

Python 3 y `pip3` (el gestor de paquetes de Python) son esenciales para poder instalar y ejecutar librerías en nuestro entorno (Fig. 34).

```
[root@localhost centoslab]# sudo yum install -y python3 python3-pip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: edgeuno-bog2.mm.fcix.net
 * extras: edgeuno-bog2.mm.fcix.net
 * updates: edgeuno-bog2.mm.fcix.net
pgdg-common/7/x86_64/signature
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG
Importing GPG key 0x442DF0F8:
 Userid     : "PostgreSQL RPM Building Project <pgsql-pkg-yum@postgresql.org>"
 Fingerprint: 68c9 e2b9 1a37 d136 fe74 d176 1f16 d2e1 442d f0f8
 Package    : pgdg-redhat-repo-42.0-35PGDG.noarch (@/pgdg-redhat-repo-latest.noarch)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG
pgdg-common/7/x86 64/signature
```

Fig. 34: Instalación Python.

Instalación del entorno virtual: Para aislar la aplicación y evitar conflictos con otras aplicaciones, instalamos un entorno virtual con el siguiente comando:

```bash
sudo pip3 install virtualenv
```

Esto nos permitirá ejecutar la aplicación en un entorno separado y controlado (Fig. 35).

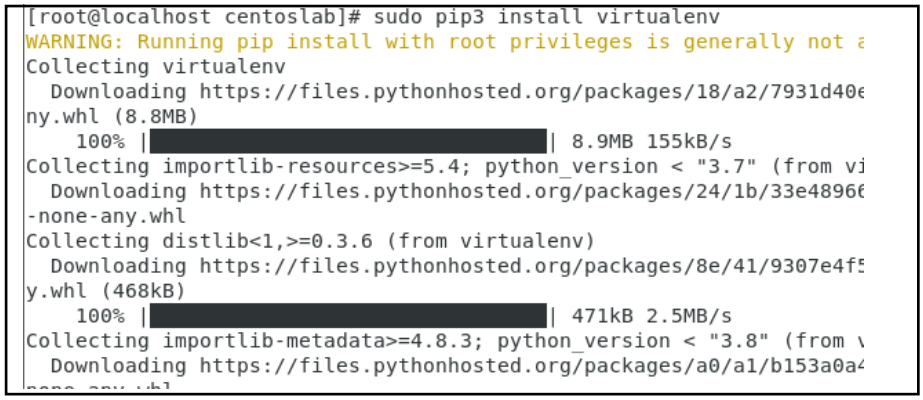

Fig. 35: Instalación entorno virtual.

Creación del entorno virtual: Creamos un directorio para nuestra aplicación web y configuramos el entorno virtual dentro de él:

```bash
mkdir flask_app
cd flask_app
python3 -m venv myprojectenv
source myprojectenv/bin/activate
```

El entorno virtual `myprojectenv` estará aislado del resto del sistema y contendrá todas las dependencias necesarias para nuestra aplicación (Fig. 36).

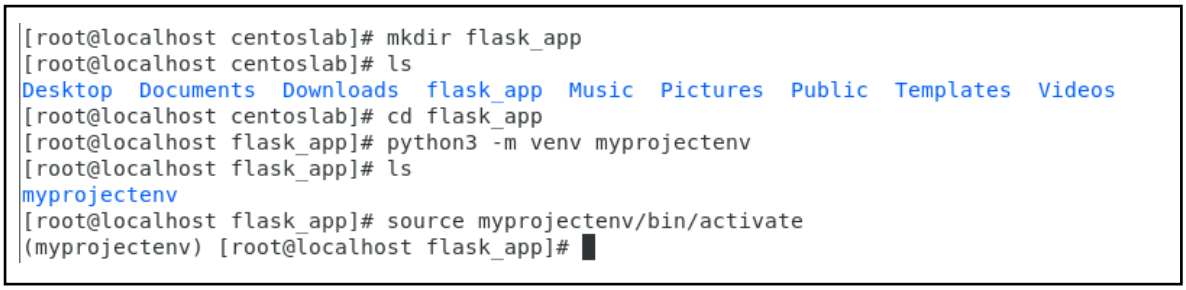

Fig. 36: Creación de directorios.

Actualización de `pip`: Con el entorno virtual activo, actualizamos `pip` a su versión más reciente para poder gestionar las dependencias de manera eficiente:

```bash
pip install --upgrade pip
```

Esto nos asegura que todas las futuras instalaciones se manejen con la última versión de `pip` (Fig. 37).

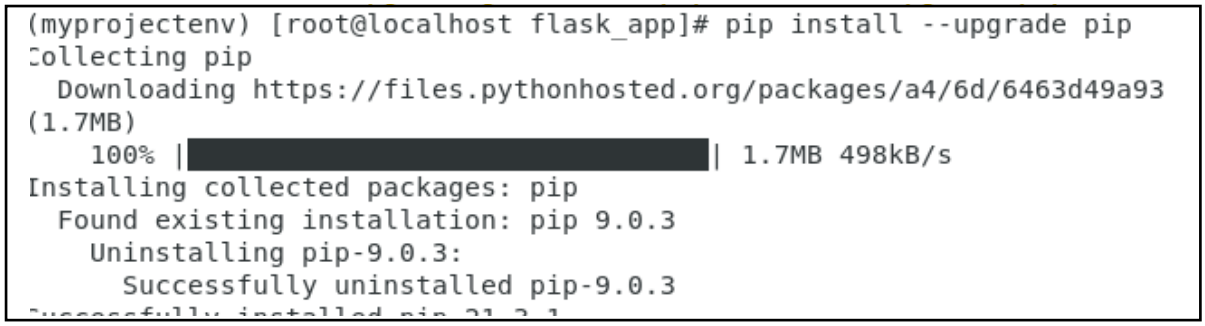

Fig. 37: Instalación de pip.

Instalación de Flask y dependencias: Instalamos Flask, el framework con el que desarrollaremos la aplicación web, y la librería `psycopg2-binary`, que nos permitirá conectarnos a la base de datos PostgreSQL.

```bash
pip install Flask psycopg2-binary
```

Flask será la base de nuestra aplicación, mientras que `psycopg2-binary` se encargará de la conexión a PostgreSQL (Fig. 38).

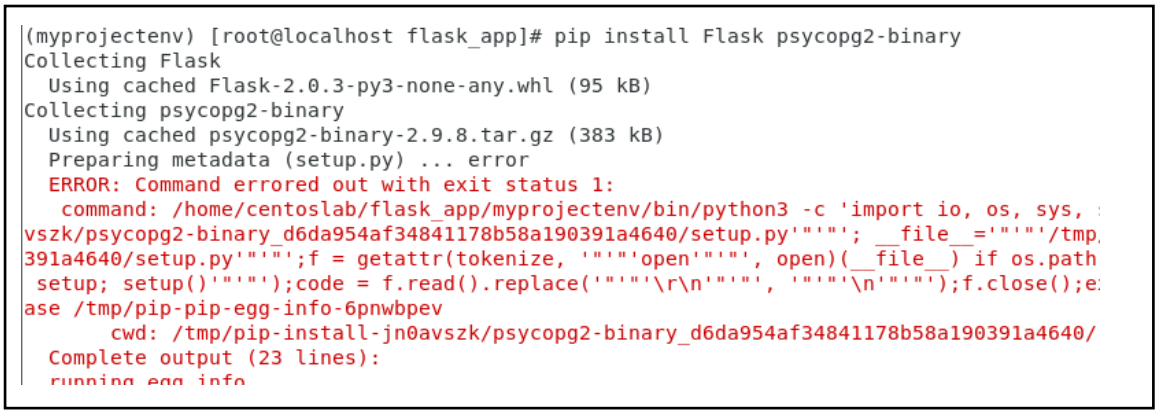

Fig. 38: Instalación Framework Flask.

Configuración de la base de datos: Con PostgreSQL ya instalado y configurado, creamos una base de datos y un usuario para la aplicación:

```bash
sudo -u postgres psql
CREATE DATABASE test_db;
CREATE USER test WITH ENCRYPTED PASSWORD '1234';
GRANT ALL PRIVILEGES ON DATABASE test_db TO test;
```

Esto crea una base de datos llamada `test_db` y un usuario `test` con todos los privilegios necesarios para gestionar los datos (Figs. 39 y 40).

```
[root@localhost centoslab]# sudo -u postgres psql
could not change directory to "/home/centoslab": Permission denied
psql (13.13)
Type "help" for help.
```

Fig. 39: Ingreso PostgreSQL Shell.

```
postgres=# CREATE DATABASE test_db;
CREATE DATABASE
postgres=# CREATE USER test WITH ENCRYPTED PASSWORD '1234';
CREATE ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE test_db TO test;
GRANT
```

Fig. 40: Creación de base de datos.

Ingreso a la base de datos con el nuevo usuario: Nos conectamos a la base de datos utilizando el usuario que acabamos de crear.

```bash
PGPASSWORD="1234" psql -U test -d test_db -h 192.168.137.132
```

Este comando permite que el usuario `test` acceda remotamente a la base de datos `test_db` (Fig. 41).

```
[root@localhost centoslab]# PGPASSWORD="1234" psql -U test -d test_db -h 192.168.137.132
psql (13.13)
Type "help" for help.

test_db=>
```

Fig. 41: Ingreso Shell PostgreSQL.

Creación de tablas: Creamos una tabla `users` dentro de la base de datos para almacenar los datos de usuario.

```bash
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL
);
```

Insertamos un usuario inicial en la tabla:

```bash
INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

Estos comandos crean una tabla básica para almacenar usuarios y una cuenta de administrador inicial (Fig. 42).

```
test_db=> CREATE TABLE users (
test_db(>     id SERIAL PRIMARY KEY,
test_db(>     username VARCHAR(50) UNIQUE NOT NULL,
test_db(>     password VARCHAR(50) NOT NULL
test_db(> );
CREATE TABLE
test_db=>
test_db=> INSERT INTO users (username, password) VALUES ('admin', 'admin123');
INSERT 0 1
test_db=> \q
```

Fig. 42: Creación de Tablas.

Ajustes en el Firewall: Para permitir el acceso a la aplicación en la red, abrimos el puerto 5000 en el firewall:

```bash
sudo firewall-cmd --permanent --add-port=5000/tcp
sudo firewall-cmd --reload
```

Esto permite que otros dispositivos en la red accedan a la aplicación web (Figs. 43 y 44).

```
[root@localhost flask_app]# sudo firewall-cmd --permanent --add-port=5000/tcp
```

Fig. 43: Permisos puerto Firewall.

```
[root@localhost flask_app]# sudo firewall-cmd --reload
```

Fig. 44: Recarga de configuración Firewall

5.4.2 Desarrollo de la aplicación en Python

La aplicación está construida utilizando Flask, un microframework en Python, y PostgreSQL como base de datos. Se desarrolla una pequeña aplicación que gestiona el inicio de sesión de usuarios y la inserción de comentarios, con el propósito de ser vulnerable a inyecciones SQL y a la ejecución de comandos del sistema.

Creación del archivo `app.py`: A continuación, creamos el archivo principal de nuestra aplicación llamado `app.py` con el siguiente código:

```bash
touch app.py
vi app.py
```

Dentro de `app.py`, introducimos el siguiente código para configurar una aplicación vulnerable a inyecciones SQL y de comandos del sistema:

```python
from flask import Flask, request
import psycopg2, os
app = Flask(__name__)
conn = psycopg2.connect(
    host="192.168.137.132",
    database="test_db",
    user="test",
    password="1234"
)
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        query = f"SELECT * FROM users WHERE username
= '{username}' AND password = '{password}'"
        cur = conn.cursor()
        cur.execute(query)
        user = cur.fetchone()
        cur.close()
        if user:
            return 'Logged in successfully'
        return 'Invalid credentials'
    return '''
        <form method="post">
```

```
                Username:        <input        type="text"
name="username"><br>
                Password:      <input      type="password"
name="password"><br>
                <input type="submit" value="Login">
            </form>
        '''
    @app.route('/comments', methods=['POST'])
def comments():
    comment = request.form['comment']
    return os.popen(comment).read()

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Este archivo contiene la lógica de la aplicación web y puntos de vulnerabilidad para propósitos educativos (Fig. 45).

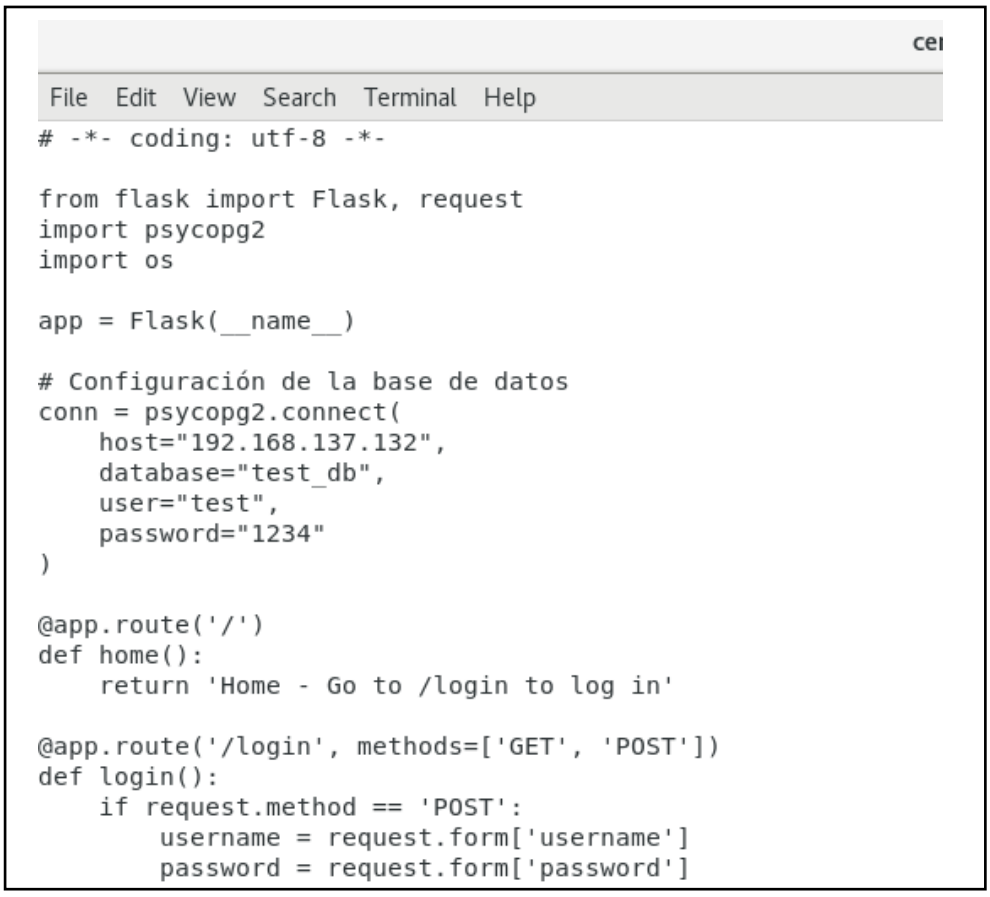

Fig. 45: Guardar datos app.py

5.4.3 Descripción de Vulnerabilidades en Aplicación Web

Punto Vulnerable de Inyección SQL

La vulnerabilidad de inyección SQL se introduce intencionalmente en la función de inicio de sesión. La consulta SQL se construye concatenando directamente los inputs del usuario sin ningún tipo de sanitización, lo cual permite a un atacante inyectar código malicioso.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        query = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
        cur = conn.cursor()
        cur.execute(query)
        user = cur.fetchone()
        cur.close()
        if user:
            return 'Logged in successfully'
        return 'Invalid credentials'
    return '''
        <form method="post">
            Username: <input type="text" name="username"><br>
            Password: <input type="password" name="password"><br>
            <input type="submit" value="Login">
        </form>
    '''
```

Punto Vulnerable de Inyección de Comandos

Otra vulnerabilidad introducida de manera intencional es la ejecución de comandos del sistema a través de una solicitud POST. El código permite que cualquier input del usuario sea ejecutado directamente por el sistema operativo.

```python
@app.route('/comments', methods=['POST'])
def comments():
    comment = request.form['comment']
    return os.popen(comment).read()
```

5.4.4 Ejecución Aplicación Web

Ejecución de la aplicación: Regresamos al directorio de la aplicación y ejecutamos el entorno virtual.

```bash
source myprojectenv/bin/activate
python app.py
```

Con esto, la aplicación Flask comenzará a ejecutarse en el puerto 5000, accesible desde cualquier navegador (Fig. 46).

```
(myprojectenv) [root@localhost flask_app]# python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://192.168.137.132:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 145-375-783
192.168.137.132 - - [10/Jun/2024 12:39:49] "GET / HTTP/1.1" 200 -
192.168.137.132 - - [10/Jun/2024 12:39:50] "GET /favicon.ico HTTP/1.1" 404 -
192.168.137.1 - - [10/Jun/2024 12:42:51] "GET /login HTTP/1.1" 200 -
192.168.137.1 - - [10/Jun/2024 12:42:51] "GET /favicon.ico HTTP/1.1" 404 -
192.168.137.1 - - [10/Jun/2024 12:57:50] "POST /login HTTP/1.1" 200 -
192.168.137.1 - - [10/Jun/2024 12:57:56] "GET /login HTTP/1.1" 200 -

Message from syslogd@localhost at Jun 12 07:19:54 ...
 kernel:NMI watchdog: BUG: soft lockup - CPU#0 stuck for 40s! [kworker/0:3:70684]
^C(myprojectenv) [root@localhost flask_app]# source myprojectenv/bin/activate
(myprojectenv) [root@localhost flask_app]# python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
```

Fig. 46: Ejecución aplicación app.py

Prueba de la aplicación: Para probar la aplicación, abrimos un navegador e ingresamos la siguiente dirección-

```
http://192.168.137.132:5000/
```

Aquí se podrá visualizar la interfaz de la aplicación e interactuar con las vulnerabilidades de inyección SQL y de comandos (Fig. 47).

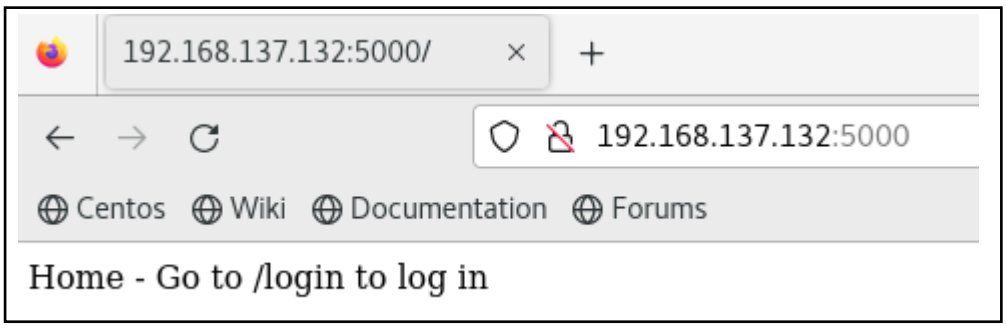

Fig. 47: Visualización Aplicación Navegador.

6. Análisis de Vulnerabilidades

En esta parte del documento vamos a revisar las vulnerabilidades en el servidor FTP configurado con VSFTPD, el servidor de base de datos configurado con PostgreSQL y la aplicación web desarrollada en Python.

6.1 Vulnerabilidades Servidor FTP

Ahora desde un equipo con Kali vamos a buscar vulnerabilidades del servidor FTP.

Verificamos conectividad desde el servidor Kali mediante un comando ping (Fig. 48).

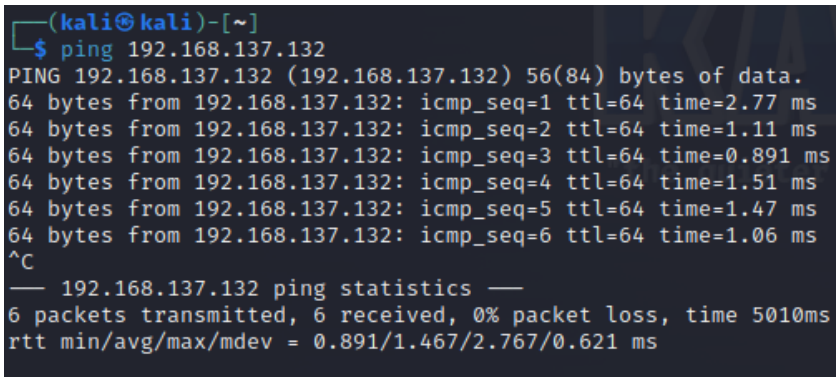

Fig. 48: Verificar conectividad con servidor FTP.

Ejecutamos el comando Nmap que es una herramienta dentro de Kali Linux que nos permite descubrir dispositivos dentro de una red, dentro del comando se utiliza las siguientes opciones:

-sC: Detecta vulnerabilidades

-sV: Detecta versiones de los servicios y los puertos abiertos y en ejecución

-Pn: Detecta todos los hosts disponibles en la red.

"sudo nmap -sC -sV -Pn 192.168.137.132" (Fig.49)

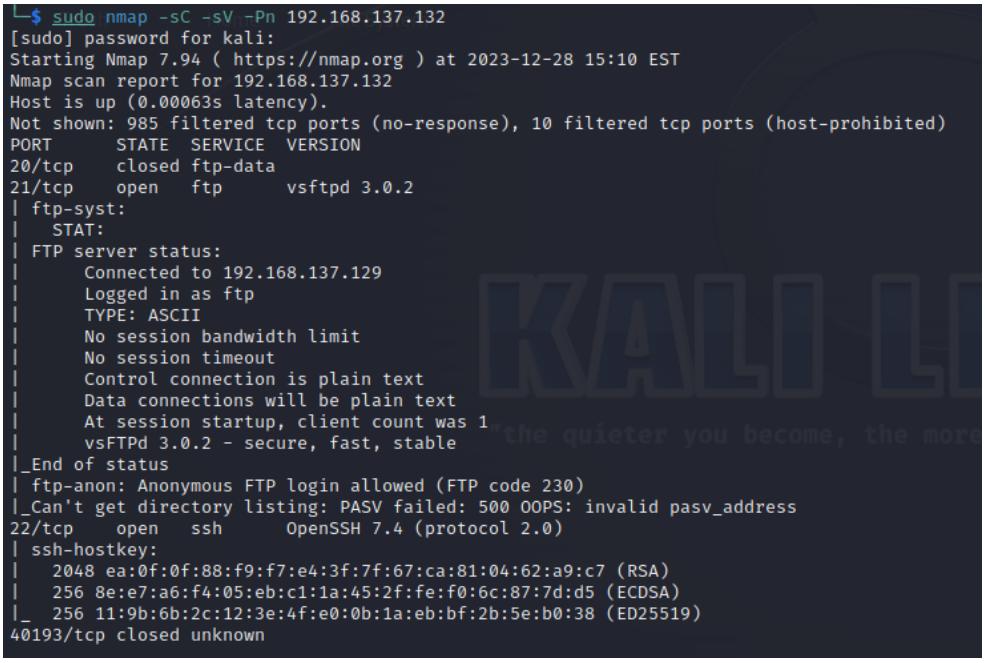

Fig. 49: Comando Nmap ejecutado a servidor FTP.

Los resultados del comando ejecutado presentan:

Puerto 21 - Servicio FTP (vsftpd 3.0.2):

- El puerto 21 está abierto, indicando que el servicio FTP (vsftpd 3.0.2) está en ejecución en el sistema.
- La versión del servidor FTP es vsftpd 3.0.2.

Información del Servicio FTP:

- El escaneo ha revelado información sobre la configuración del servidor FTP.
- La conexión FTP muestra que se permite el acceso anónimo (`ftp-anon: Anonymous FTP login allowed`), lo cual es visible en la respuesta `FTP code 230`.

Puerto 22 - SSH (OpenSSH 7.4):

- El puerto 22 está abierto, indicando que el servicio SSH (OpenSSH 7.4) también está en ejecución en el sistema.

Con el comando FTP y la dirección IP vamos a conectarnos al servidor (Fig.50):

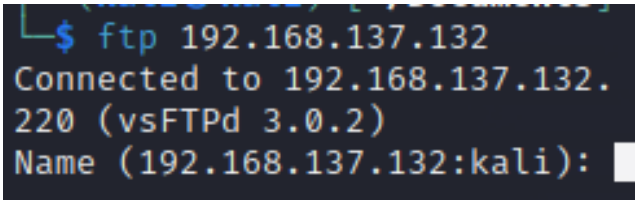

Fig. 50: Comando Ftp.

En el resultado del comando Nmap se pudo visualizar que el acceso con usuarios anónimos se encuentra habilitado al servidor FTP por esta razón vamos a ingresar con el usuario **"anonymous"** (Fig.51).

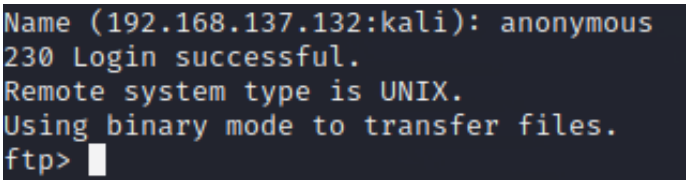

Fig. 51: Acceso servidor Ftp.

Vemos que el acceso al servidor es un éxito, a continuación, ejecutamos el comando "`ls`" para verificar que tenemos acceso a todo el directorio raíz según la configuración en el archivo vsftpd.conf (Fig. 52).

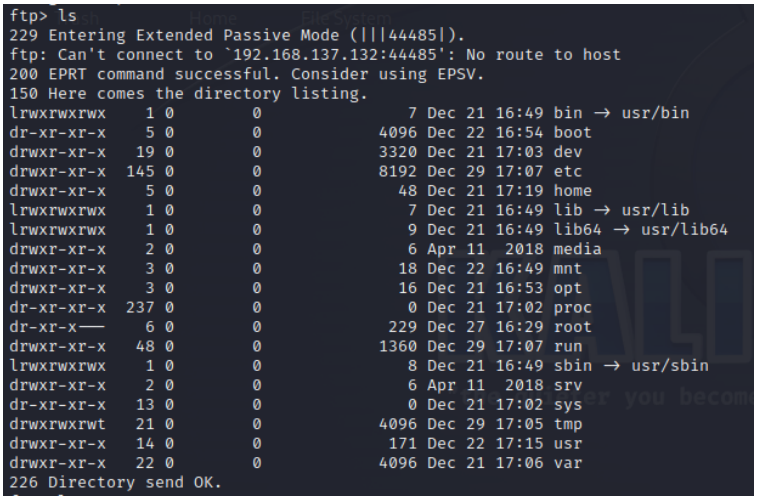

Fig. 52: Listado de acceso directorios FTP.

Ingresamos al directorio etc con el comando `cd etc` y ejecutamos un `ls` para poder visualizar los directorios y archivos contenidos en el mismo (Fig. 53).

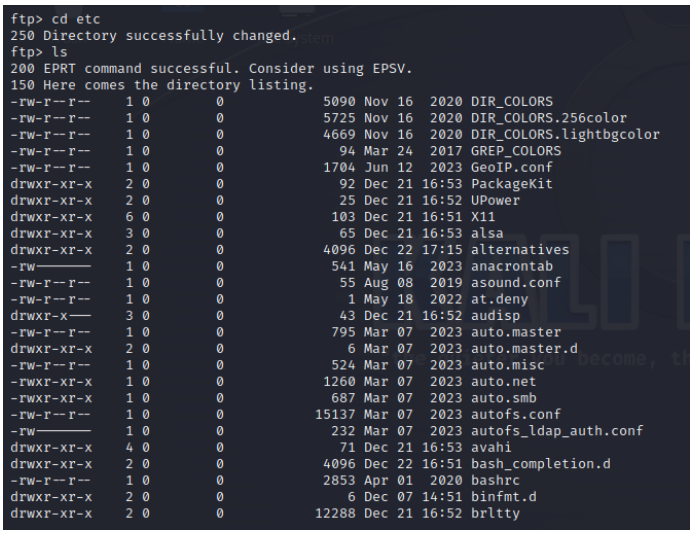

Fig. 53: Acceso a directorio etc.

Nuestro objetivo es obtener información sobre los usuarios así que dentro del directorio etc vamos a ejecutar el comando `get` para obtener el archivo **passwd** y copiarlo hacia nuestro equipo Kali (Fig. 54).

Fig. 54: Comando get para obtener archivo.

Desde un terminal diferente vamos a visualizar el contenido del archivo **passwd** ejecutando un comando **vi** para abrir el archivo.

Ahora se puede observar los usuarios se encuentran configurados en la máquina objetivo (Fig. 55).

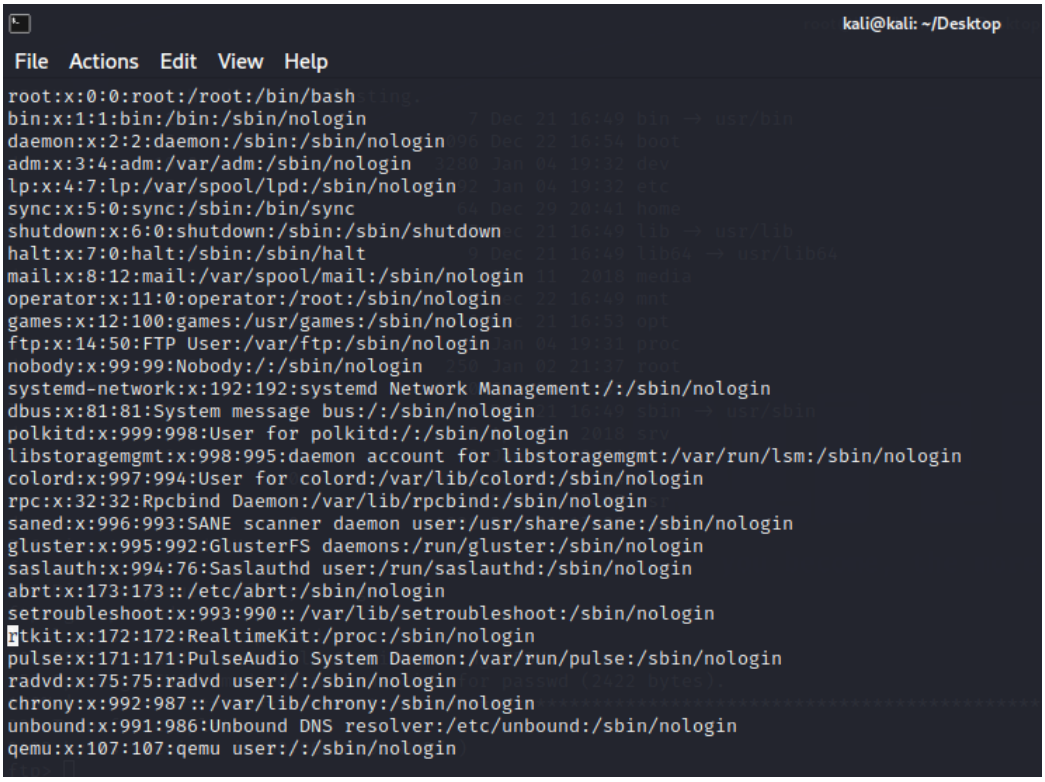

Fig. 55: Archivo passwd de maquina objetivo.

A continuación, realizaremos un ataque de fuerza bruta utilizando Hydra, el usuario objetivo será “root” (Fig. 56).

```
“ hydra 192.168.137.132 ftp -l root -P
/usr/share/wordlists/rockyou.txt -s 21”
```

hydra: Es una herramienta de fuerza bruta para probar contraseñas en varios servicios.

192.168.137.132: La dirección IP del objetivo

ftp: Indica que el protocolo que se va a atacar es **FTP** (File Transfer Protocol).

-l root: Especifica que el nombre de usuario que se va a usar en el ataque es root. El -l (con "L" minúscula) se usa para definir un único nombre de usuario.

-P /usr/share/wordlists/rockyou.txt: Especifica la lista de contraseñas que se usará en el ataque de fuerza bruta. En este caso, es el archivo rockyou.txt, que es un diccionario de contraseñas comunes. El -P (con "P" mayúscula) se usa para indicar una lista de contraseñas.

-s 21: Especifica el puerto del servicio **FTP** que está activo en el servidor. En este caso, es el puerto **21**, que es el puerto predeterminado para **FTP**.

```
┌──(kali㉿kali)-[/usr/share]
└─$ hydra 192.168.137.132 ftp -l root -P /usr/share/wordlists/rockyou.txt -s 21
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret
these *** ignore laws and ethics anyway).
```

Fig. 56: Ataque Fuerza bruta Hydra.

El resultado de nuestro ataque de fuerza bruta muestra que la contraseña del usuario root es “password” (Fig.57).

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-01 05:46:25
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344400 login tries (l:1/p:14344400), ~896525 tries per task
[DATA] attacking ftp://192.168.137.132:21/
[21][ftp] host: 192.168.137.132   login: root   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-01 05:46:32
```

Fig. 57: Resultado Hydra.

Con las credenciales del usuario root, ingresaremos al servidor FTP y comprobaremos que las credenciales sean correctas (Fig. 58).

```
┌──(kali㉿kali)-[/usr/share]
└─$ ftp 192.168.137.132
Connected to 192.168.137.132.
220 (vsFTPd 3.0.2)
Name (192.168.137.132:kali): root
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Fig. 58 Acceso root servidor FTP.

6.2 Vulnerabilidades en el Servidor de Base de Datos.

Como primer paso probamos la conectividad con el servidor CentOS desde Kali (Fig. 59).

```
ping 192.168.137.132
```

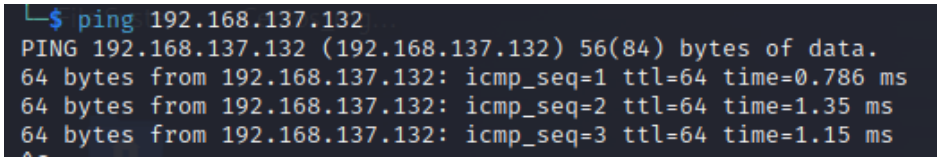

Fig. 59: Conectividad Servidor CentOS.

Procedemos a ejecutar el siguiente comando:

```
nmap -sV -p 5432 --script=pgsql-brute 192.168.137.132
```

nmap: Es la herramienta de escaneo de red utilizada.

-sV: Esta opción le dice a Nmap que detecte los servicios en ejecución y las versiones de software en los puertos abiertos.

-p 5432: Especifica que Nmap debe escanear el puerto 5432. Este es el puerto por defecto donde PostgreSQL escucha las conexiones.

--script=pgsql-brute: Indica a Nmap que use el script pgsql-brute durante el escaneo. Este script intenta realizar un ataque de fuerza bruta para adivinar las credenciales de acceso a PostgreSQL.

192.168.137.132: Es la dirección IP del objetivo que se va a escanear (Fig. 60).

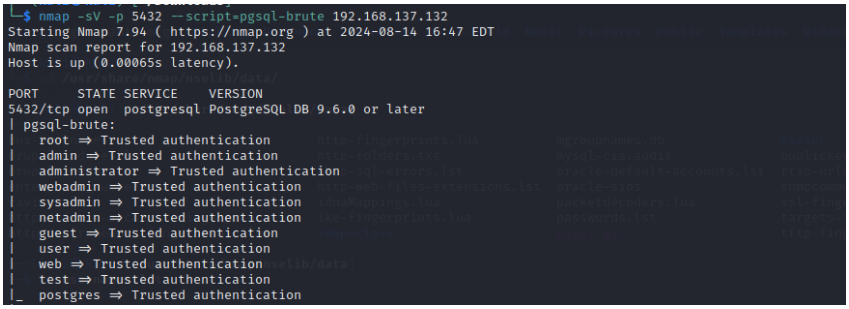

Fig. 60: Resultado comando Nmap.

- El resultado de Nmap nos presenta que el servicio se encuentra activo.
- El puerto 5432 esta abierto y ejecutando el servicio de PostgreSQL la versión detectada es la 9.6.0 o posterior.
- El apartado pgsql-brute pudo identificar varios usuarios comunes habilitados con la configuración “Trusted authentication”, permitiendo el acceso sin contraseña.

Conexión a PostgreSQL Trusted Authentication.

Se realiza la conexión con la base de datos tratando de validar los usuarios encontrados (Fig. 61).

Fig.61: Prueba de acceso PSQL.

Ingresamos a psql con el usuario postgres sin ingresar contraseña.

Y procedemos a crear un usuario con el perfil superuser, obteniendo una escalada de privilegios (Fig. 62).

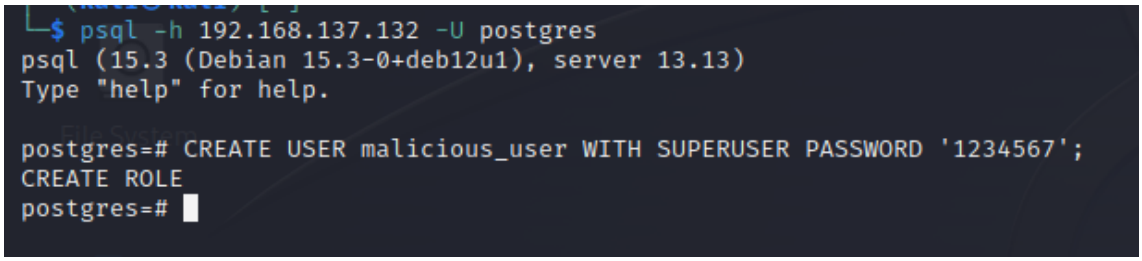

Fig. 62: Creación de Usuario PostgreSQL.

6.3 Vulnerabilidades Inyección SQL

Desde nuestro equipo Kali Linux abrimos el navegador y digitamos la dirección (Fig. 63): `https//192.168.137.132:5000`

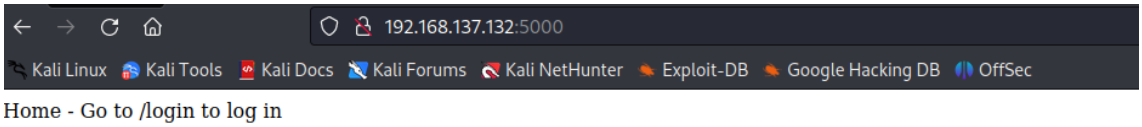

Fig. 63: Ingreso App Web.

Con la aplicación levantada ingresamos al apartado login y procedemos a realizar pruebas de vulnerabilidades de Inyeccion SQL (Fig. 64).

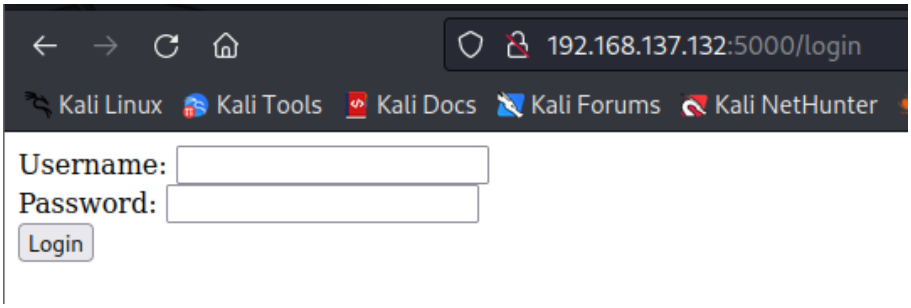

Fig. 64: Login App Web.

Si ingresamos los valores de Usuario y contraseña incorrectos la aplicación nos muestra el siguiente error (Fig. 65).

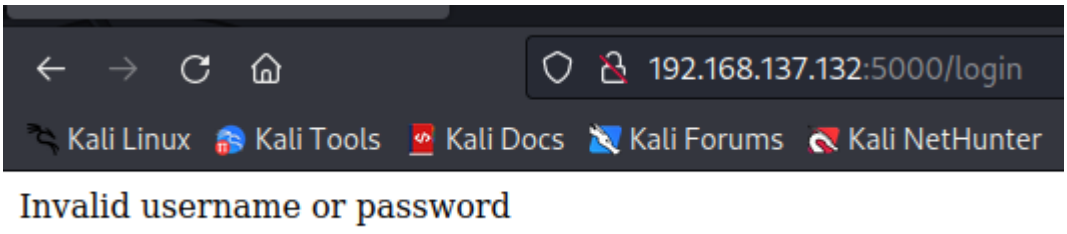

Fig. 65: Datos erróneos Login.

Procedemos a ingresar los siguientes parámetros en el campo Username: `admin' --` y en el campo Password ingresamos cualquier valor.

Este intento debería iniciar sesión con éxito porque el comentario `--` ignora el resto de la consulta SQL (Fig. 66).

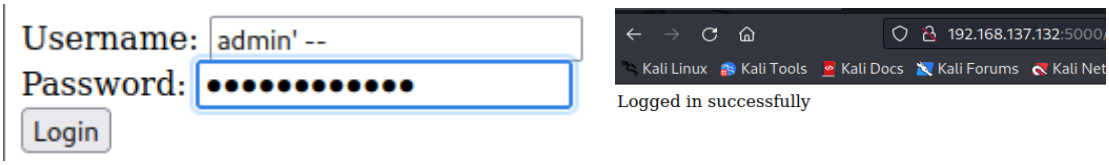

Fig. 66: Inyección SQL Ejemplo 1.

En el segundo ejemplo vamos a ejecutar el siguiente parámetro en el campo Username: `admin' OR '1'='1` y en el campo Password ingresamos cualquier valor.

Este intento debería iniciar sesión debido a la condición OR permite que la consulta siempre sea válida (Fig. 67).

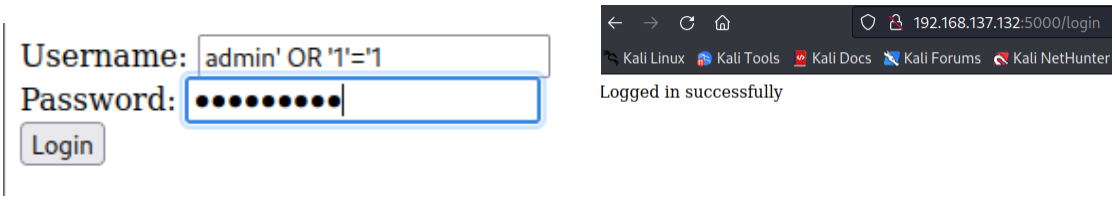

Fig. 67: Inyección SQL Ejemplo 2.

6.4 Vulnerabilidades Inyección de Comandos

Para las pruebas de inyección de comandos vamos a ingresar a la dirección: `http:192.168.137.132:5000/comments`

Como primer ejemplo de inyección de comandos vamos a realizar la prueba ingresando el comando `ls` (Fig. 68).

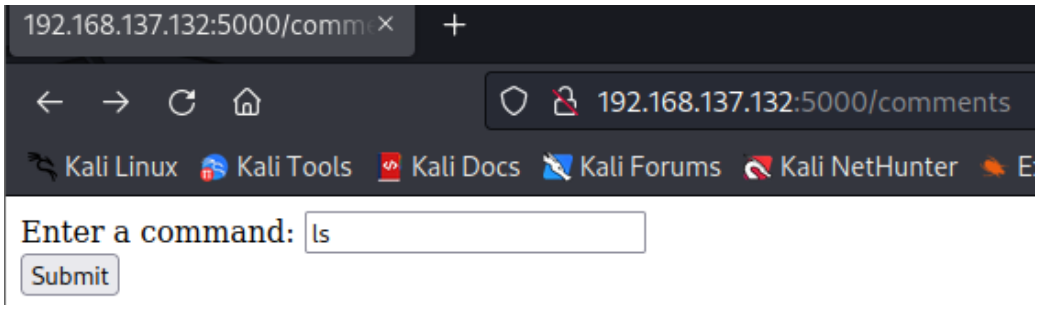

Fig. 68: Inyección de comandos Ejemplo 1.

El resultado del comando nos debe presentar los archivos presentes en el directorio del servidor (Fig. 69).

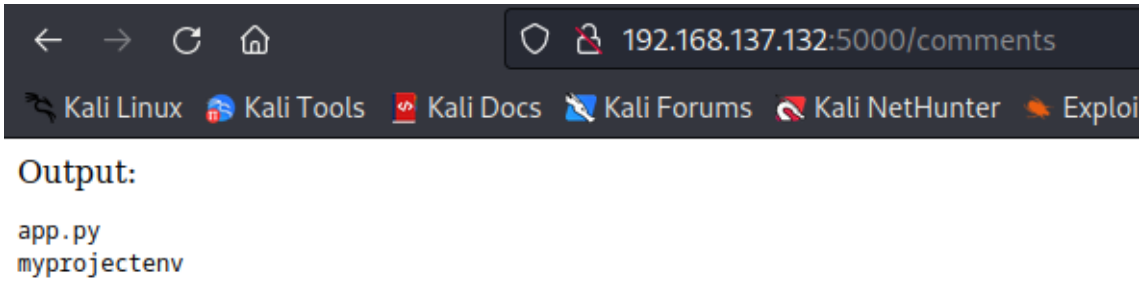

Fig. 69: Resultado Inyección de comandos Ejemplo 1.

Para el segundo ejemplo vamos a utilizar el comando: `uname -a` (Fig. 70).

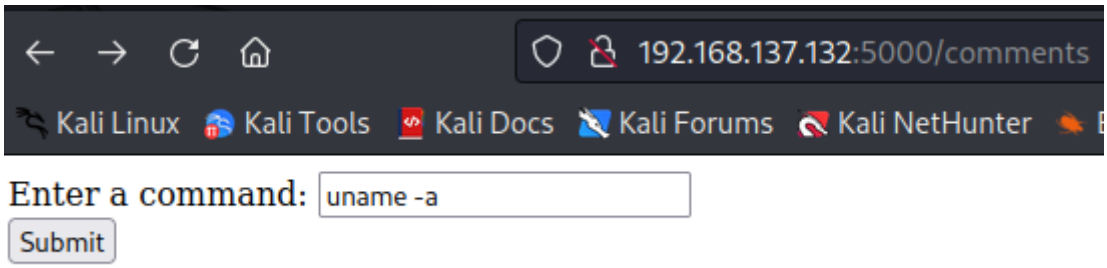

Fig. 70: Inyección de comandos Ejemplo 2.

Este comando permite conocer información sobre el sistema donde esta alojada la aplicación (Fig. 71).

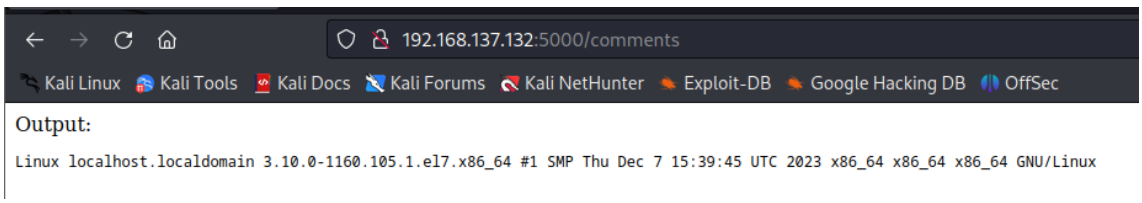

Fig. 71: Resultado Inyección de comandos Ejemplo 2.

6.5 Resumen de Vulnerabilidades encontradas

La siguiente tabla resume las vulnerabilidades identificadas en los servidores FTP, base de datos y la aplicación web. El análisis resalta que estos sistemas están altamente expuestos a ataques graves debido a configuraciones inseguras. Es crucial corregir estas vulnerabilidades de manera inmediata, ya que comprometen seriamente la seguridad de la infraestructura evaluada.

Categoría	Vulnerabilidad Identificada	Descripción	Nivel de criticidad	Frecuencia de ocurrencia	Impacto potencial	Resultado del escaneo
Servidor FTP (vsftpd)	Acceso anónimo habilitado	Permite iniciar sesión sin autenticación, exponiendo el sistema.	**Alta**	**Alta**	Acceso no autorizado a información sensible.	ftp-anon: Anonymous FTP login allowed
Servidor FTP (vsftpd)	Acceso sin restricciones al sistema de archivos	Permite listar y copiar archivos críticos, como /etc/passwd.	**Alta**	**Alta**	Robo de datos sensibles y manipulación del sistema.	Acceso exitoso a directorios como /etc/
Servidor FTP (vsftpd)	Vulnerabilidad de fuerza bruta (Hydra)	Contraseña root débil permitiendo acceso al servidor.	**Alta**	**Media**	Acceso completo al servidor y potencial escalada de privilegios.	Fuerza bruta exitosa con Hydra, contraseña descubierta
Base de Datos (PostgreSQL)	Autenticación confiable habilitada	Permite acceso sin contraseña a usuarios comunes, facilitando la escalada.	**Alta**	**Media**	Escalada de privilegios y posible compromiso de toda la base de datos.	Acceso sin contraseña con Trusted Authentication
Base de Datos (PostgreSQL)	Creación de usuario superuser sin restricciones	Facilita la escalada de privilegios en la base de datos.	**Alta**	**Media**	Control total sobre la base de datos, creación de usuarios maliciosos.	Usuario superuser creado sin restricciones
Aplicación Web	Inyección SQL (admin' --)	Permite eludir la autenticación en el login con inyección SQL.	**Alta**	**Alta**	Acceso no autorizado, eludir autenticación de usuarios.	Login exitoso con inyección SQL
Aplicación Web	Inyección SQL (admin' OR '1'='1)	Permite iniciar sesión sin credenciales válidas utilizando una consulta siempre verdadera.	**Alta**	**Alta**	Acceso no autorizado, comprometiendo la autenticación.	Login exitoso usando inyección SQL
Aplicación Web	Inyección de comandos (ls)	Permite la ejecución de comandos del sistema desde la aplicación web.	**Alta**	**Alta**	Exposición de archivos del sistema, robo o manipulación de datos.	Listado de archivos del servidor (directorio visible)
Aplicación Web	Inyección de comandos (uname -a)	Permite obtener información sobre el sistema operativo del servidor.	**Media**	**Media**	Exposición de información del sistema operativo, facilitando ataques futuros.	Información del sistema operativo expuesta (uname -a)

Tabla. 1: Resumen de Vulnerabilidades Encontradas.

7. Mitigación de Vulnerabilidades

7.1 Mitigación de Vulnerabilidades Servidor FTP

A continuación, realizaremos cambios en el archivo de configuración de VSFTPD en donde se especificaron parámetros inseguros que permitían el acceso y vulneraban la seguridad del servidor.

Desactivación del Acceso Anónimo:

El acceso anónimo al servidor FTP debe ser deshabilitado para prevenir que usuarios no autenticados realicen conexiones al servidor. Esto se logra comentando o eliminando la línea `anonymous_enable=YES` y asegurando que `anonymous_enable=NO` esté configurado. Al desactivar el acceso anónimo, se protege el servidor de posibles abusos y se restringe el acceso solo a usuarios autenticados.

Aislamiento de Directorios para Usuarios Locales (Chroot):

Para evitar que los usuarios locales accedan a directorios fuera de sus áreas de trabajo asignadas, se debe habilitar el aislamiento de directorios (`chroot`). Esto se configura cambiando `chroot_local_user=NO` a `chroot_local_user=YES`, lo que asegura que cada usuario esté restringido a su propio directorio, reduciendo así el riesgo de acceso no autorizado a archivos críticos del sistema.

Deshabilitar Permisos de Escritura para Usuarios Anónimos:

Permitir que usuarios anónimos carguen archivos o creen directorios es una práctica insegura que puede llevar a la introducción de contenido malicioso en el servidor. Para mitigar este riesgo, las configuraciones `anon_upload_enable=YES` y `anon_mkdir_write_enable=YES` deben ser cambiadas a `NO`, deshabilitando así cualquier posibilidad de escritura por parte de usuarios anónimos.

Implementación de Límites de Conexiones y Sesiones:

Configurar límites en el número de conexiones y sesiones ayuda a prevenir ataques de denegación de servicio (DoS) y fuerza bruta. Se deben establecer valores como `max_per_ip=5`, `max_clients=10`, `idle_session_timeout=300`, y

`max_login_fails=3` para limitar la cantidad de recursos que un atacante puede consumir y restringir los intentos fallidos de autenticación.

Habilitación de Registros de Transferencias y Accesos:

Para poder monitorear y auditar la actividad en el servidor FTP, es fundamental habilitar el registro de transferencias y accesos. Esto se logra configurando `xferlog_enable=YES` y `log_ftp_protocol=YES`, asegurando que todas las acciones realizadas en el servidor sean registradas y almacenadas en el archivo de log especificado. Esto permite detectar y responder a actividades sospechosas de manera oportuna.

7.2 Mitigación de Vulnerabilidades Servidor POSTGRES

A continuación, realizaremos cambios en el archivo de configuración de POSTGRESQL en donde se especificaron parámetros inseguros que permitían el acceso y vulneraban la seguridad del servidor.

Conexiones al servidor de base:

Para mitigar esta vulnerabilidad, se debe restringir el acceso a direcciones IP específicas y de confianza, limitando así quién puede conectarse al servidor. Además, se recomienda actualizar el método de autenticación a `md5` o `scram-sha-256`, que ofrece un nivel de seguridad al proteger mejor las credenciales de usuario.

```
host    all     all    192.168.137.0/24        md5
```

Limitación de escucha por direcciones:

La configuración para escuchar en todas las direcciones (`listen_addresses = '*'`) expone el servidor a posibles ataques desde cualquier origen. Para mitigar este riesgo, es recomendable modificar la configuración para que el servidor solo escuche en las direcciones IP necesarias para su operación, restringiendo así el acceso solo a las fuentes autorizadas y reduciendo la superficie de ataque.

```
listen_addresses = 'localhost, 192.168.137.0'
```

Autenticación Fortalecida:

La autenticación también debe ser fortalecida, especialmente para el usuario `postgres`. Utilizar contraseñas débiles o por defecto constituye una vulnerabilidad grave que podría ser explotada fácilmente. Por ello, es esencial cambiar la contraseña del usuario `postgres` a una más compleja, que incluya una combinación de caracteres especiales, números y mayúsculas. Este cambio dificultará los intentos de acceso no autorizado.

```
\password postgres
```

Al ejecutar este comando, se solicitará una nueva contraseña que debe cumplir con los estándares de seguridad recomendados, como la inclusión de caracteres especiales, números y letras mayúsculas y minúsculas.

Conexiones de Red:

La seguridad de las conexiones de red es otro aspecto crucial. En la configuración inicial, el puerto 5432 está abierto en el firewall sin restricciones adicionales, y las conexiones no están cifradas, lo que podría permitir que datos sensibles sean interceptados. Para remediar esto, se deben establecer reglas de firewall más estrictas que permitan únicamente conexiones desde direcciones IP de confianza.

```
firewall-cmd --zone=trusted --add-port=5432/tcp --permanent
firewall-cmd –reload
```

7.3 Mitigación Inyección SQL y de Comandos.

7.3.1 Mitigación Inyección SQL

En pasos anteriores pudimos observar que la aplicación desarrollada en Python era vulnerable a los ataques de inyección SQL y de comandos, se realizaron varias pruebas en las cuales se pudo constatar las vulnerabilidades. Ahora veremos las mejores prácticas en las cuales mitigaremos las vulnerabilidades encontradas.

En el código vulnerable las variables "username" y "password" se insertan directamente en la cadena SQL mediante el método format. Dando paso a que se permita realizar ataques mediante la inyección SQL.

```
username = request.form['username']
password = request.form['password']

query = "SELECT * FROM users WHERE username = '{}' AND
password = '{}'".format(username, password)
cur.execute(query)
```

Para prevenir la inyección SQL, se deben usar parámetros de consultas conocidas también como consultas preparadas, evitando que el contenido de los campos de entrada sea interpretado como código SQL, por esta razón se utiliza dentro de la consulta marcadores de posición ('%s').

```
query = "SELECT * FROM users WHERE username = %s AND password
= %s"
```

Y a través del método 'execute' se envían los valores username y password.

```
cur.execute(query, (username, password))
```

Al usar los marcadores de posición, los datos de entrada del usuario nunca se interpretan como comandos SQL, evitando que alguien pueda inyectar código malicioso.

A continuación, adjuntamos el código con los cambios realizados.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        try:
            cur = conn.cursor()
            # Uso de parámetros de consulta para prevenir inyección SQL
            query = "SELECT * FROM users WHERE username = %s AND password = %s"
            cur.execute(query, (username, password))
            user = cur.fetchone()
            cur.close()

            if user:
                return 'Logged in successfully'
            else:
                return 'Invalid username or password'
        except psycopg2.Error:
            conn.rollback()
            return 'Invalid username or password'
```

7.3.2 Mitigación Inyección de comandos

Para la mitigación de esta vulnerabilidad debemos evitar la ejecución de comandos del sistema directamente en las entradas del usuario.
El siguiente código, permite la ejecución de los comandos de sistema en la entrada de comentarios:

```
output = os.popen(comment).read()
```

Una manera segura de manejar los comentarios es almacenarlos en una base de datos. Por esta razón procedemos a crear una tabla para poder almacenar los comentarios.

```
CREATE TABLE comments (
    id SERIAL PRIMARY KEY,
    comment TEXT NOT NULL
);
```

Luego procedemos a realizar los cambios en el código donde la línea output se elimina por completo, en cambio se inserta las siguientes líneas donde se realizan las inserciones de los comentarios.
Dentro de la consulta se envían parámetros (`%s`) para asegurar que el comentario del usuario se inserte en la base de datos de manera segura. Esto evita que el contenido del comentario sea tratado como código ejecutable.

```
cur = conn.cursor()
query = "INSERT INTO comments (comment) VALUES (%s)"
cur.execute(query, (comment,))
conn.commit()
cur.close()
```

Con la siguiente parte del código se establece un manejo de errores, estableciendo un rollback al presentarse cualquier error y con ello mantener la integridad de la base de datos.

```
except Exception as e:
    conn.rollback()
    return 'An error occurred: {}'.format(e)
```

Siguiendo todos los pasos anteriores se mitiga en gran manera las vulnerabilidades de inyección SQL y de comandos, manteniendo la integridad de la aplicación web.

Adjuntamos el código junto con los cambios realizados:

```
“””
# -*- coding: utf-8 -*-
from flask import Flask, request
import psycopg2

app = Flask(__name__)

# Configuración de la base de datos
conn = psycopg2.connect(
    host="192.168.137.132",
    database="test_db",
    user="test",
    password="1234"
)
@app.route('/')
def home():
    return 'Home - Go to /login to log in'
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        try:
```

```
            cur = conn.cursor()
            # Uso de parámetros de consulta para prevenir
inyección SQL
            query = "SELECT * FROM users WHERE username =
%s AND password = %s"
            cur.execute(query, (username, password))
            user = cur.fetchone()
            cur.close()

            if user:
                return 'Logged in successfully'
            else:
                return 'Invalid username or password'
        except psycopg2.Error:
            conn.rollback()
            return 'Invalid username or password'

    return '''
        <form method="post">
            Username: <input type="text"
name="username"><br>
            Password: <input type="password"
name="password"><br>
            <input type="submit" value="Login">
        </form>
    '''

@app.route('/comments', methods=['GET', 'POST'])
def comments():
    if request.method == 'POST':
        comment = request.form['comment']

        try:
            cur = conn.cursor()
```

```
            query = "INSERT INTO comments (comment) VALUES
(%s)"
            cur.execute(query, (comment,))
            conn.commit()
            cur.close()
            return 'Comment added successfully'
        except Exception as e:
            conn.rollback()
            return 'An error occurred: {}'.format(e)

    return '''
        <form method="post">
            Enter a comment: <input type="text"
name="comment"><br>
            <input type="submit" value="Submit">
        </form>
    '''

if __name__ == '__main__':
    # Ejecutar la aplicación Flask en todas las interfaces
de red
    app.run(host='192.168.137.132', port=5000, debug=True)

"""
```

7.4 Resumen de Vulnerabilidades mitigadas

La tabla resume las vulnerabilidades mitigadas en los servidores FTP, base de datos PostgreSQL y la aplicación web. Se indican las vulnerabilidades corregidas, su nivel de criticidad, impacto potencial y el estado después de la mitigación. La conclusión principal es que las medidas aplicadas han reducido significativamente los riesgos, reforzando la seguridad de toda la infraestructura.

Categoría	Vulnerabilidad Identificada	Descripción	Nivel de criticidad	Frecuencia de ocurrencia	Impacto potencial	Resultado del escaneo	Estado después de la mitigación
Servidor FTP (vsftpd)	Acceso anónimo habilitado	Permitía iniciar sesión sin autenticación, exponiendo el sistema.	Baja	Baja	Bajo	ftp-anon: Anonymous FTP login disabled	**Mitigada**: Acceso anónimo deshabilitado
Servidor FTP (vsftpd)	Acceso sin restricciones al sistema de archivos	Permitía listar y copiar archivos críticos, como /etc/passwd.	Baja	Baja	Bajo	Acceso restringido a directorios locales	**Mitigada**: Usuarios confinados a sus directorios (chroot)
Servidor FTP (vsftpd)	Vulnerabilidad de fuerza bruta (Hydra)	Contraseña root débil permitiendo acceso al servidor.	Baja	Baja	Bajo	Fuerza bruta fallida, contraseñas robustas implementadas	**Mitigada**: Contraseñas robustas implementadas
Base de Datos (PostgreSQL)	Autenticación confiable habilitada	Permitía acceso sin contraseña a usuarios comunes, facilitando la escalada.	Baja	Baja	Bajo	Acceso sin contraseña deshabilitado	**Mitigada**: Autenticación md5 configurada
Base de Datos (PostgreSQL)	Creación de usuario superuser sin restricciones	Facilitaba la escalada de privilegios en la base de datos.	Baja	Baja	Bajo	Creación de superusuarios controlada	**Mitigada**: Privilegios restringidos y contraseñas fuertes
Aplicación Web	Inyección SQL (admin' --)	Permitía eludir la autenticación en el login con inyección SQL.	Baja	Baja	Bajo	Consultas parametrizadas en uso	**Mitigada**: Consultas parametrizadas implementadas
Aplicación Web	Inyección SQL (admin' OR '1'='1)	Permitía iniciar sesión sin credenciales válidas utilizando una consulta siempre verdadera.	Baja	Baja	Bajo	Login fallido, consultas preparadas en uso	**Mitigada**: Uso de consultas preparadas
Aplicación Web	Inyección de comandos (ls)	Permitía la ejecución de comandos del sistema desde la aplicación web.	Baja	Baja	Bajo	Comando no ejecutado, entradas sanitizadas	**Mitigada**: Eliminación de ejecución directa de comandos
Aplicación Web	Inyección de comandos (uname -a)	Permitía obtener información sobre el sistema operativo del servidor.	Baja	Baja	Bajo	Información del sistema no expuesta	**Mitigada**: Manejo seguro de entradas

Tabla. 2: Resumen Vulnerabilidades Mitigadas

7.5 Grafico comparativo de vulnerabilidades

El gráfico comparativo muestra la relación entre las vulnerabilidades detectadas y mitigadas en tres componentes del sistema: el servidor FTP, el servidor PostgreSQL y la aplicación web. Cada barra azul representa el número de vulnerabilidades detectadas inicialmente, mientras que las barras verdes indican las vulnerabilidades que fueron mitigadas. En este caso, las tres categorías (FTP, PostgreSQL, y la aplicación web) muestran que todas las vulnerabilidades detectadas han sido mitigadas, destacando un manejo efectivo de las medidas de seguridad implementadas. No hay vulnerabilidades que permanezcan sin mitigar, lo cual es representado por la ausencia de barras rojas (Fig. 72).

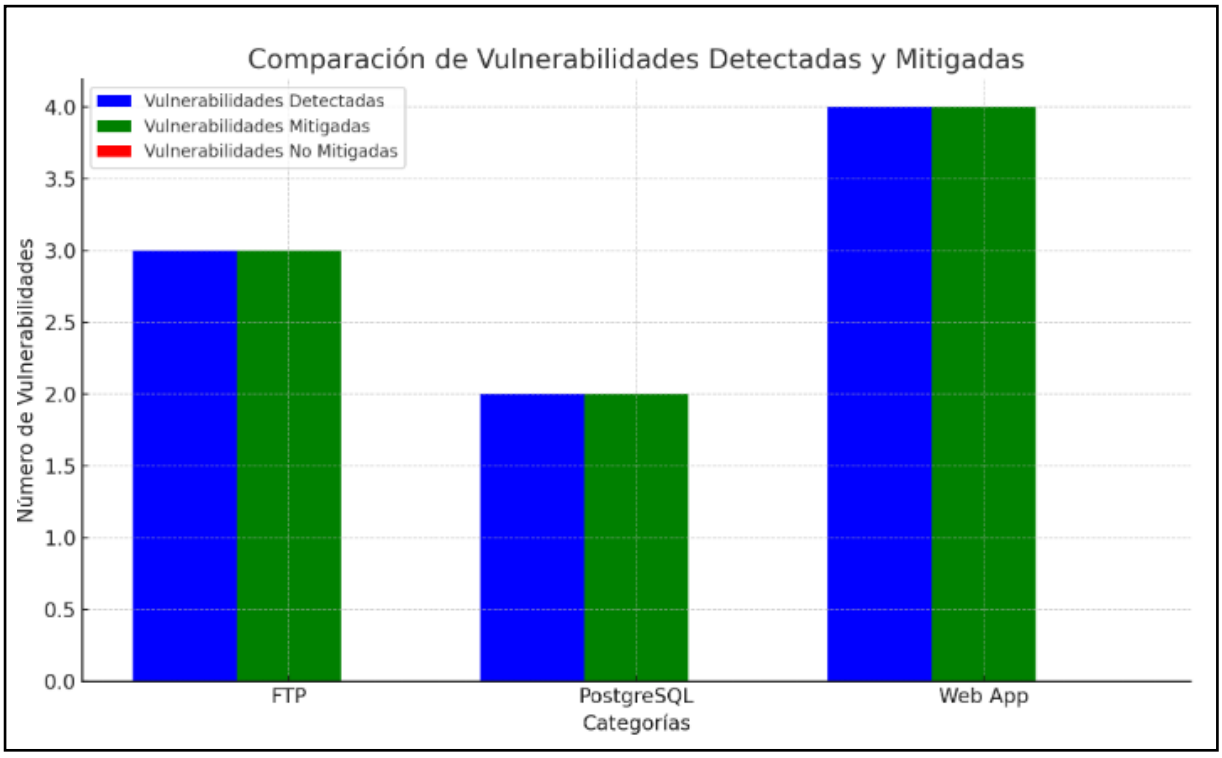

Fig. 72 Vulnerabilidades detectadas y mitigadas.

8. Conclusiones

- La investigación ha demostrado que las configuraciones incorrectas o contrarias a las mejores prácticas pueden exponer severamente los sistemas virtualizados a vulnerabilidades de seguridad. En el entorno probado con VMware y CentOS, se observó que configuraciones erróneas en los servidores FTP y de bases de datos pueden facilitar ataques de inyección SQL y de comandos, evidenciando que una mala configuración puede ser tan crítica como vulnerabilidades intrínsecas del software.
- La comparación entre configuraciones correctas y mal configuradas revela que las debilidades en las instalaciones incorrectas aumentan significativamente el riesgo de explotación. Los servidores FTP y las bases de datos mal configuradas en el entorno de pruebas fueron más susceptibles a ataques, indicando que las brechas de seguridad son amplificadas por errores en la configuración, subrayando la importancia de seguir las guías de instalación recomendadas.
- El uso de VMware ha demostrado ser una herramienta eficaz para llevar a cabo pruebas de seguridad y análisis de vulnerabilidades en un entorno controlado. La capacidad de replicar escenarios de configuración incorrecta y evaluar sus consecuencias permite una comprensión detallada de cómo las malas prácticas afectan la seguridad de los sistemas. Esto resalta la utilidad de entornos virtualizados para la formación y evaluación de medidas de seguridad.
- Los hallazgos sugieren que, para mitigar las vulnerabilidades asociadas con configuraciones incorrectas, es crucial seguir las mejores prácticas y directrices establecidas por los fabricantes. Implementar controles rigurosos de configuración y realizar auditorías periódicas pueden reducir significativamente el riesgo de explotación. Las recomendaciones incluyen la capacitación continua en configuraciones seguras y la implementación de herramientas de seguridad automatizadas para detectar y corregir errores de configuración en entornos virtualizados.

9. Referencias

[1] Doña, J., García, J. E., López, J., Pascual, F., & Pascual, R. VIRTUALIZACIÓN DE SERVIDORES – UNA SOLUCIÓN DE FUTURO. (pág. 11), (2010). Campus Universitario de Teatino,s Málaga. España.

[2] Gómez, J., & Villar, E. INTRODUCCIÓN A LA VIRTUALIZACIÓN (pág. 20-23), 2018.

[3] NAVARRO CHOLANCO, Jorge Anibal. *Implementación de un proxy en plataforma linux para el control de transferencia de archivos con FTP, E-mail y Firewall para el laboratorio de software*, (pág. 56), 2009.

[4] SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. Entity relationship model. *Database System Concepts, sixth ed., McGraw-Hill*, (pág. 36-87), 2010.

[5] Oracle, "INTRODUCTION TO VIRTUAL BOX," https://www.virtualbox.org/manual/ch01.html.

[6] Jazmín López Sánchez, responsable de desarrollo de contenidos Revista Seguridad" EMULACION DE HONEYPODS", (pág. 21), 2016, UNAM-CERT

[7] Martín Diego, Marrero Mónica, Urbano Julián, Barra Eduardo, Moreiro Jose-Antonio, "VIRTUALIZACIÓN, UNA SOLUCIÓN PARA LA EFICIENCIA SEGURIDAD Y ADMINISTRACIÓN DE INTRANETS" (pág. 2-8). 2011. DOI: 10.3145

[8] VMware, "WHAT IS A HYPERVISOR?," https://www.vmware.com/topics/glossary/content/hypervisor.

[9] Christopher Negus, THE LINUX BIBLE (10th Edition) (pág. 439-486). 2020.

[10] Marchionni, E. A. ADMINISTRADOR DE SERVIDORES (Vol. 210) (pág. 21-59). 2011, USERSHOP.

[11] David Clinton, LINUX IN ACTION (Vol. 1) (pág. 68-174). 2018, MANNING.

[12] Adell, Jordi, and Y. Bernabé. "SOFTWARE LIBRE EN EDUCACIÓN *Tecnología educativa. Madrid: McGraw-Hill* 2007. (pág. 9-173).
hg

[13] López Sempere, Alejandro."IDENTIFICACIÓN Y DETECCIÓN DE VULNERABILIDADES". Diss. Universitat Politècnica de València, (pág. 8-19) 2022.

[14] Cabezas Herrera Stalin Omar, “ANÁLISIS DE VULNERABILIDADES DE SERVIDORES VIRTUALIZADOS DE UNA RED EMPRESARIAL MEDIANTE LA UTILIZACIÓN DE HERRAMIENTAS DE SOFTWARE LIBRE” (pág. 10-85), 2022.

[15] Campus Ciberseguridad, “TIPOS DE HACKERS”.
https://www.campusciberseguridad.com/blog/item/133-tipos-de-hackers

[16] Rochina Rochina, C. G. DISEÑO Y EVALUACION DE UNA METODOLOGÍA PARA REDUCIR LOS CIBERATAQUES ORIGINADOS A TRAVES DE CORREO ELECTRONICO MEDIANTE LA APLICACIÓN DE FILTROS Y REGLAS SOBRE UN GATEWAY, (pag. 11), 2011.

[17] VSFTPD Instalación de servidor FTP.
https://ubunlog.com/vsftpd-instalar-un-servidor-ftp-ubuntu/

[18] POSTGRESQL Base de datos.
https://www.postgresql.org/

[19] NMAP, herramienta de escaneo.
https://nmap.org/

[20] Hydra, herramienta para ataques de fuerza bruta.
https://keepcoding.io/blog/hydra-en-ciberseguridad/

[21] Kali Linux Features
https://www.kali.org/

[22] Python
https://www.python.org/

[23] Flask
https://flask.palletsprojects.com/en/3.0.x/

[24] Pip
https://www.w3schools.com/python/python_pip.asp

[25] Venv
https://docs.python.org/3/library/venv.html