



Virtualización

VirtualBox y Seguridad en
Máquinas Virtuales

Curso

Tecnicatura en Programación

Alumnos

Barroso, Nikolas - nhbouharriet@gmail.com

Huarcaya, Ivan. - Hf.rivan@gmail.com

Materia


Arquitectura y Sistemas Operativos

Docentes

Prof. Martín Aristiaran – Tutor Luis Miguel Tola

Fecha de entrega

05/06/2025



Contenido

1. Introducción.....	2
2. Marco Teórico	3
2.1 Virtualización de hardware.....	3
2.2 Hipervisores.....	4
2.3 Imágenes ISO y snapshots	5
2.4 Redes virtuales	6
2.5 Enfoque del trabajo	7
3. Caso Práctico	8
Caso 1 – Análisis de archivo sospechoso.....	8
Caso 2 – Desarrollo de un script en Python.....	12
4. Metodología Utilizada	15
5. Resultados Obtenidos.....	15
6. Conclusiones.....	16
7. Bibliografía	17

1. Introducción

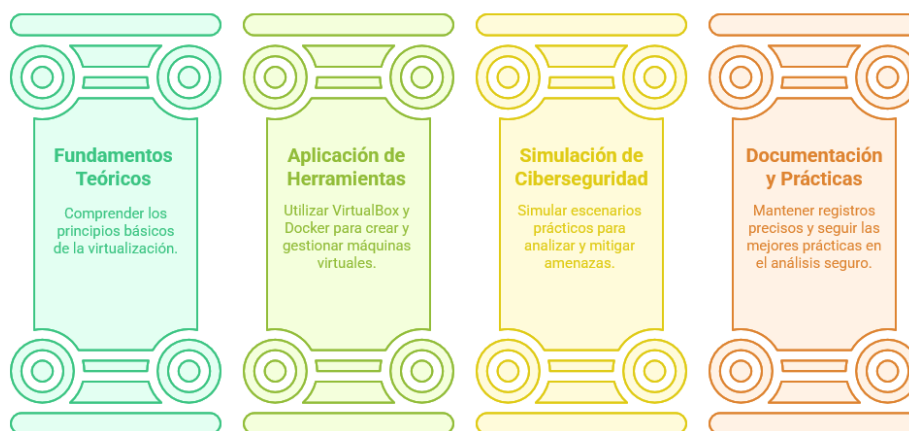
Este trabajo integrador aborda el estudio y aplicación práctica de la virtualización, las herramientas VirtualBox, las redes virtuales y la seguridad asociada a las máquinas virtuales. Se eligió este tema dada su relevancia creciente en la informática actual, especialmente en el ámbito de la ciberseguridad. El análisis de archivos maliciosos, que pueden disfrazarse de otros tipos de archivos y propagarse por plataformas de compra-venta o redes sociales, requiere entornos controlados que eviten riesgos para los sistemas principales. El trabajo se desarrolla en un marco grupal, entre dos integrantes, con un enfoque académico y técnico, buscando que el conocimiento generado sea replicable y útil para la formación como técnicos en programación. Los objetivos principales incluyen comprender los fundamentos teóricos de la virtualización, aplicar herramientas para crear y gestionar máquinas virtuales, simular casos prácticos que reflejen escenarios de ciberseguridad, y valorar la importancia de la documentación y buenas prácticas en el análisis seguro.

2. Marco Teórico

2.1 Virtualización de hardware

La virtualización de hardware es una tecnología que permite ejecutar uno o más sistemas operativos completos, denominados máquinas virtuales (VM), sobre un sistema anfitrión (host) físico. Cada máquina virtual funciona de forma aislada y emula un equipo independiente, incluyendo CPU, memoria, disco, y periféricos.

Componentes de la Virtualización



Aplicaciones de Máquinas Virtuales

Desarrollo de software en entornos virtuales

Desarrollo de software complejo en entornos virtuales avanzados.



Análisis de malware en entornos aislados

Análisis avanzado de malware en entornos seguros y aislados.



Ejecución de aplicaciones en máquinas virtuales

Ejecución básica de aplicaciones en máquinas virtuales.



Contención de amenazas en máquinas virtuales

Contención básica de amenazas en máquinas virtuales.



2.2 Hipervisores

Los hipervisores son programas que permiten la creación, gestión y ejecución de máquinas virtuales. Se clasifican en dos tipos principales:

- Tipo 1 o 'bare-metal': se instalan directamente sobre el hardware físico y gestionan los recursos para las máquinas virtuales sin un sistema operativo anfitrión.
- Tipo 2 o 'hosted': se ejecutan como una aplicación sobre un sistema operativo anfitrión, y permiten crear y administrar máquinas virtuales dentro de este entorno.

Comparación de tipos de hipervisores



Jerarquía de Virtualización de Hardware



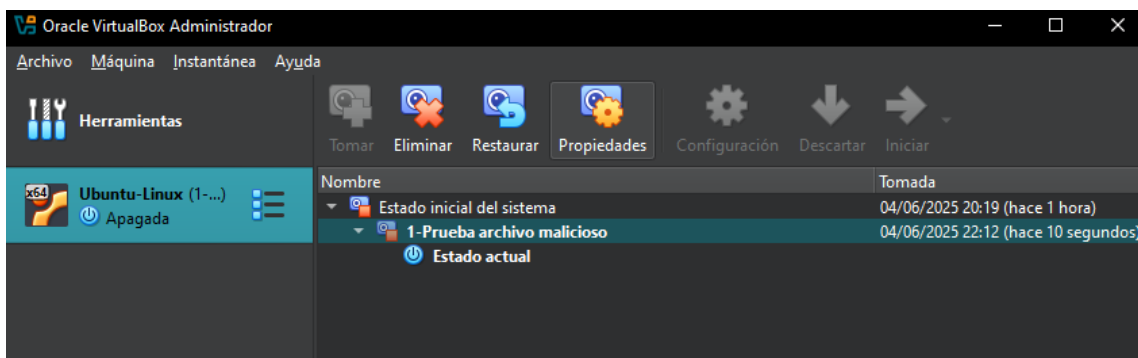
En nuestro caso utilizaremos un hipervisor del Tipo 2 llamado VirtualBox.

VirtualBox es un software de virtualización para arquitecturas x86/amd64.

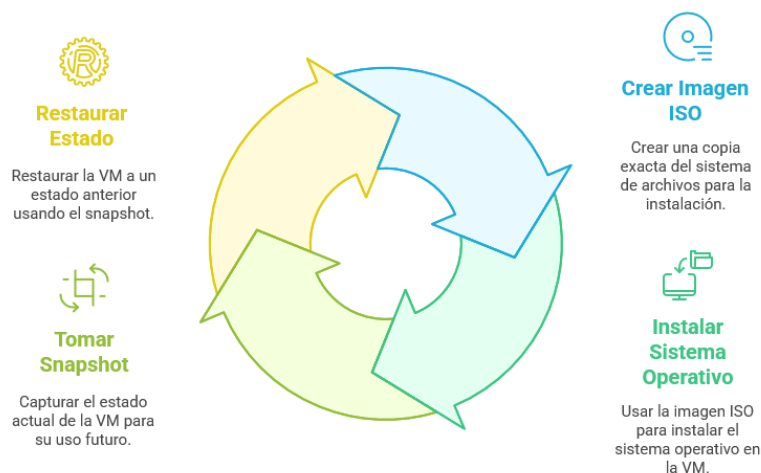
Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como «sistemas invitados», dentro de otro sistema operativo «anfitrión», cada uno con su propio ambiente virtual.

2.3 Imágenes ISO y snapshots

Las imágenes ISO son archivos que contienen una copia exacta del sistema de archivos de un medio óptico, como un CD o DVD, y se utilizan para instalar sistemas operativos en máquinas virtuales. Los snapshots son instantáneas del estado completo de una máquina virtual en un momento dado, permitiendo guardar y restaurar ese estado para facilitar pruebas, análisis o recuperación.



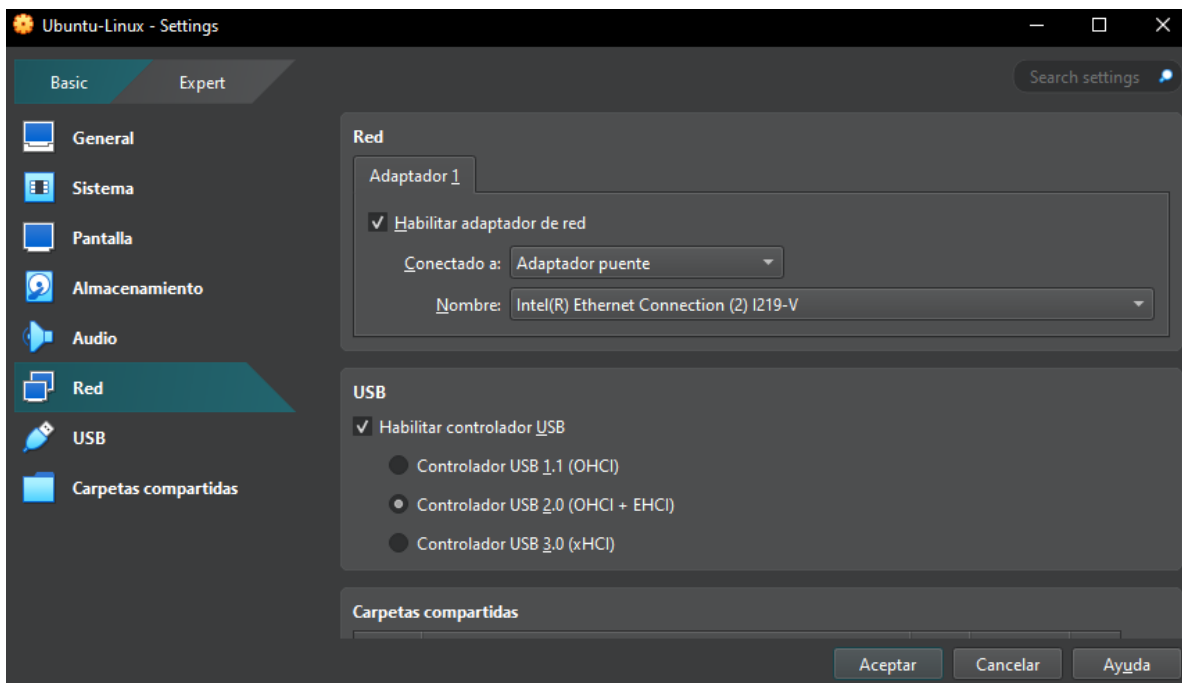
Ciclo de Gestión de Máquinas Virtuales



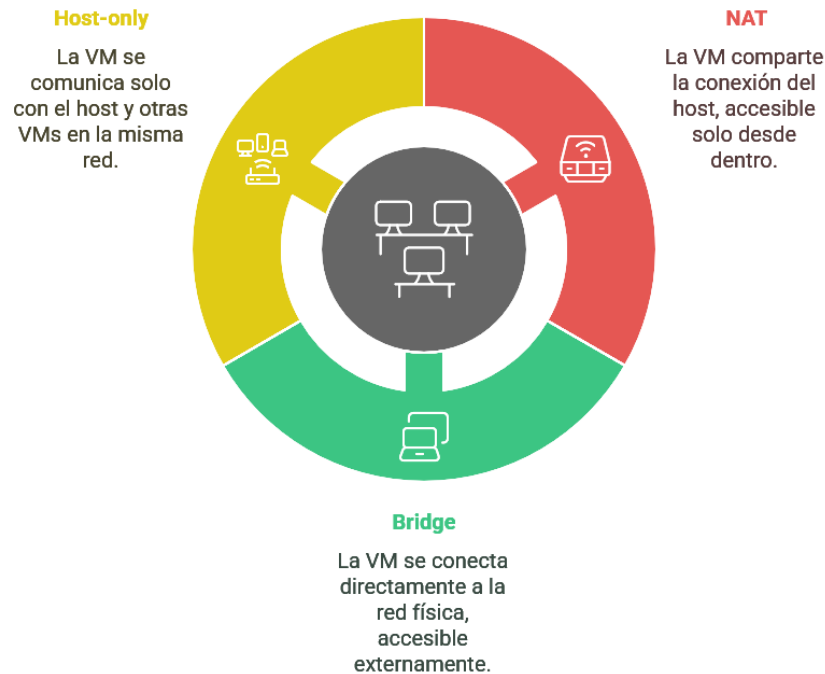
2.4 Redes virtuales

Las máquinas virtuales pueden configurarse con diferentes tipos de redes virtuales para controlar su comunicación:

- NAT (Network Address Translation): la VM comparte la conexión de red del host y puede acceder a internet, pero no es accesible desde la red externa.
- Bridge (Puentes): la VM se conecta directamente a la red física, obteniendo una dirección IP en la misma red que el host, lo que permite comunicación plena.
- Host-only: la VM solo se comunica con el host y otras máquinas virtuales configuradas en esta red, sin acceso a internet.



Tipos de Redes Virtuales



2.5 Enfoque del trabajo

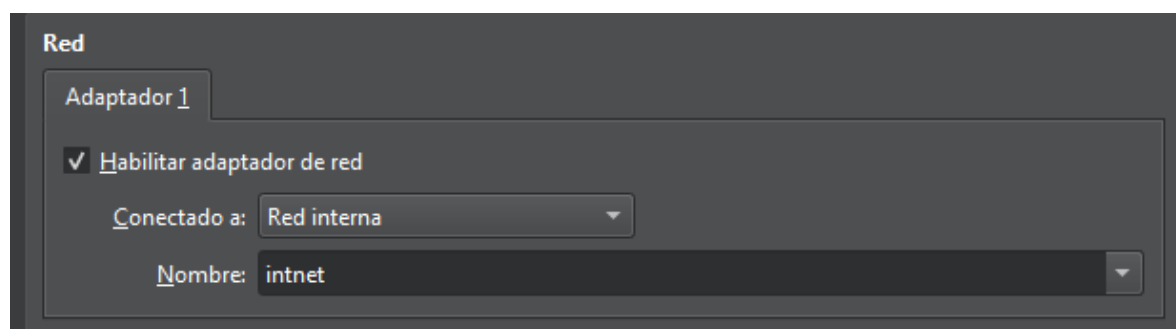
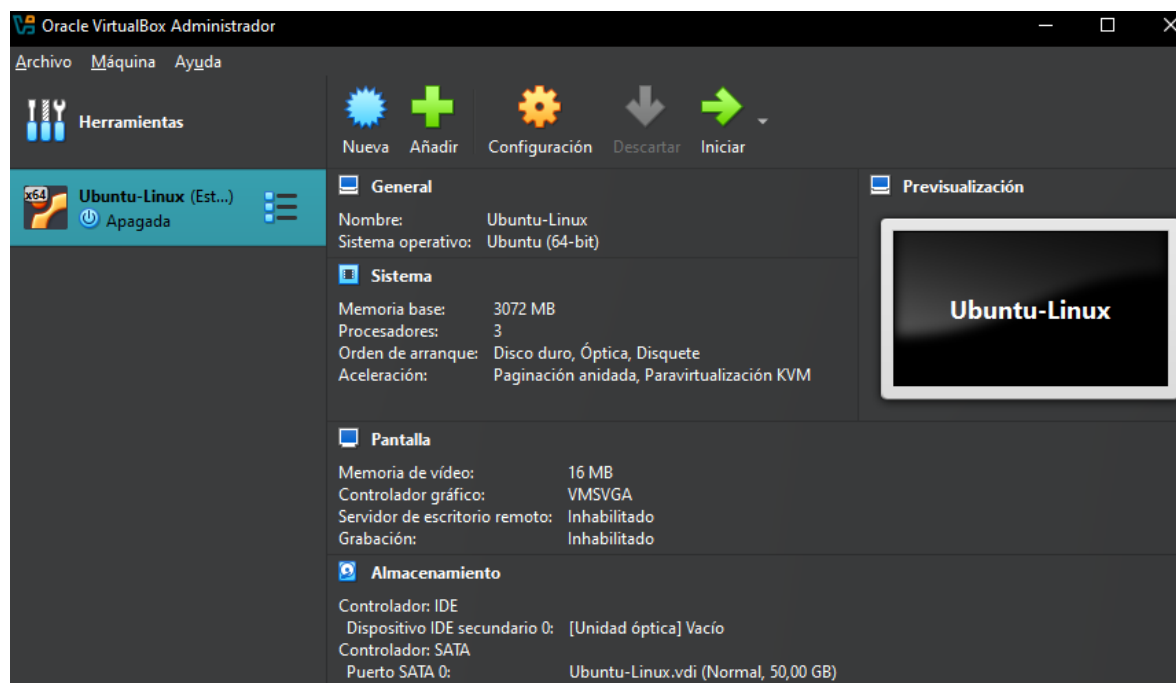
Este trabajo se enfoca específicamente en el uso de máquinas virtuales como herramientas de apoyo tanto en entornos de desarrollo como en análisis de seguridad informática. El objetivo es mostrar cómo la virtualización no solo facilita la ejecución aislada de sistemas operativos, sino que además se convierte en un recurso esencial a la hora de desarrollar tareas ,detectar, analizar o contener amenazas informáticas. Se pone especial énfasis en el uso de entornos virtuales para el análisis de archivos potencialmente maliciosos, práctica cada vez más común en el ámbito de la ciberseguridad, tanto profesional como educativo.

3. Caso Práctico

Se presentan dos casos prácticos que aplican los conceptos y herramientas analizados en este trabajo.

Caso 1 – Análisis de archivo sospechoso

Se utilizó VirtualBox para crear una máquina virtual con Ubuntu Desktop, configurada con una red completamente aislada mediante el modo “Red Interna” para evitar cualquier fuga de datos hacia internet.

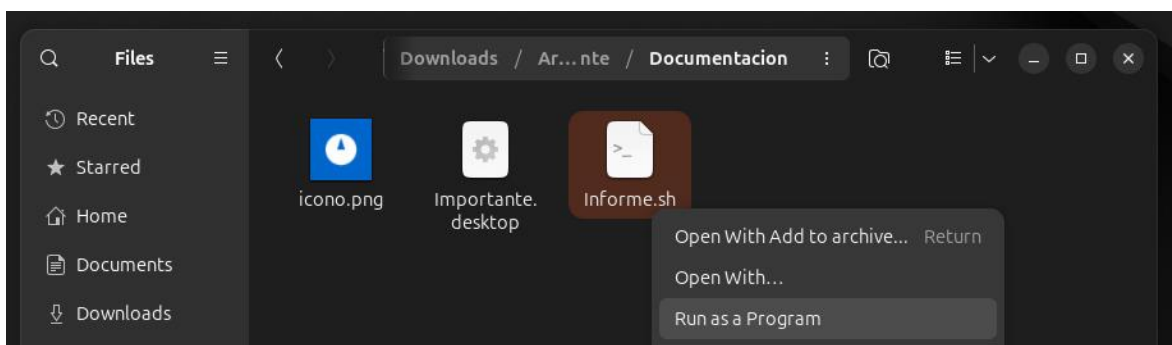
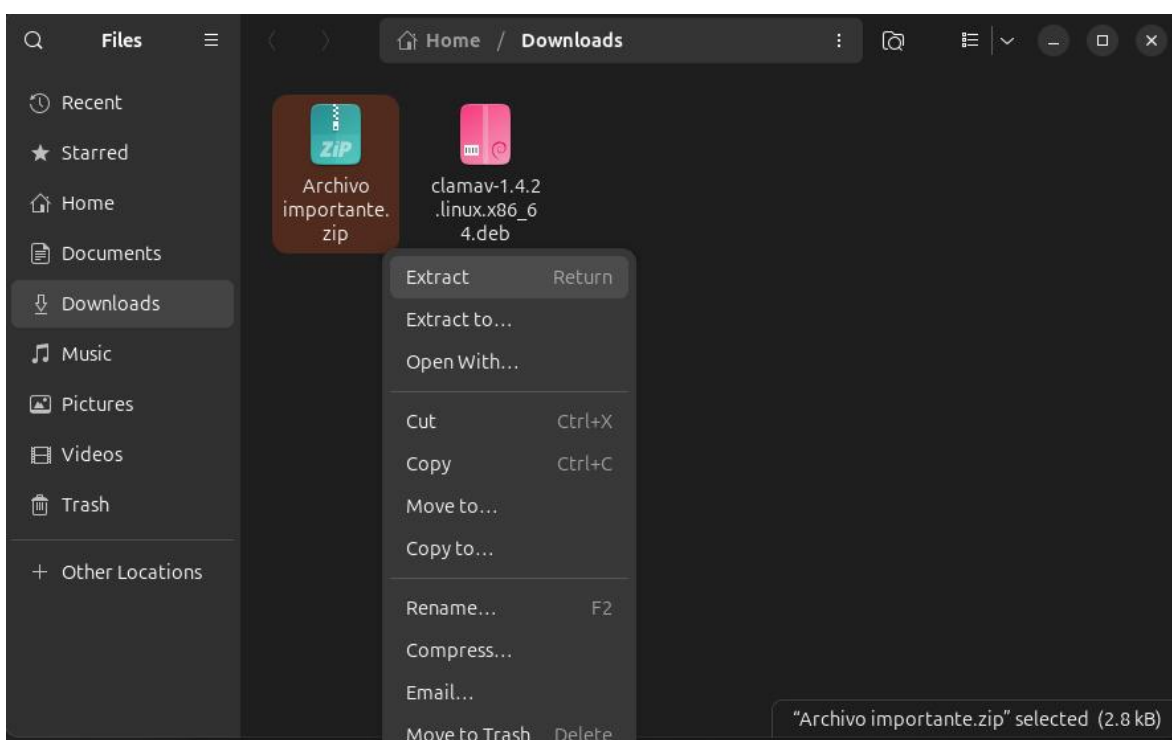


Previamente, se tomó una instantánea (snapshot) del estado inicial del sistema.

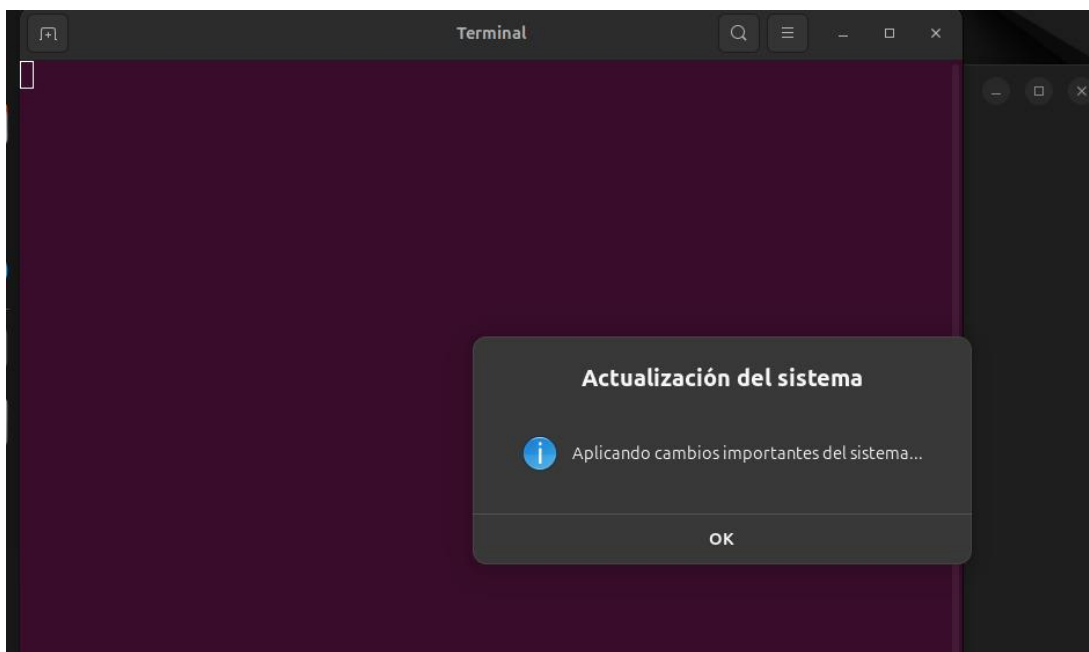
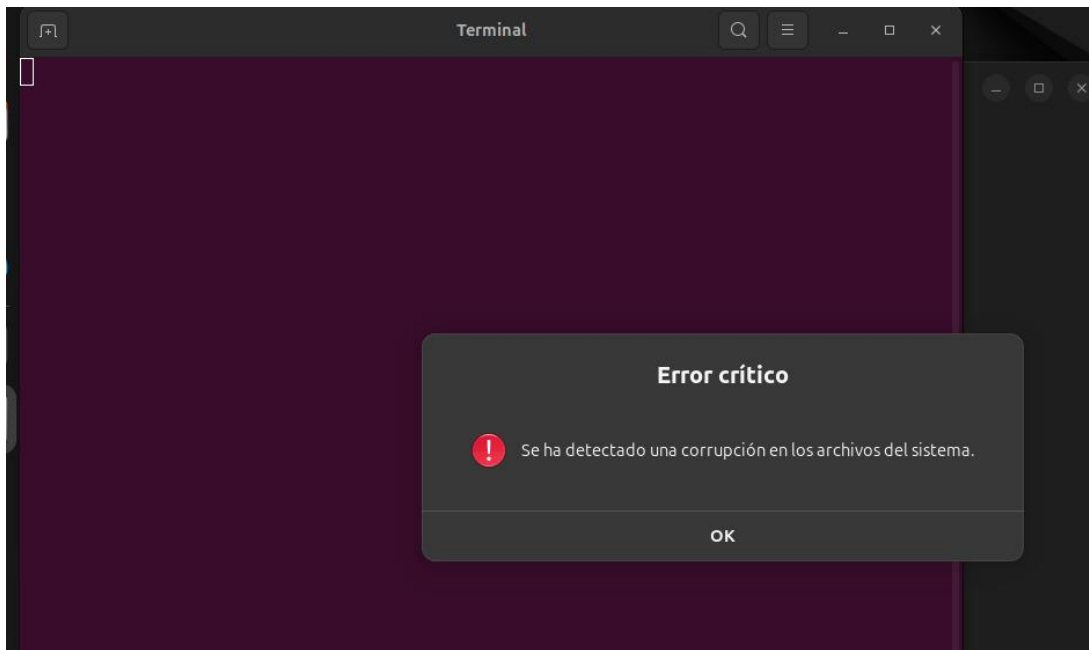
Nombre	Tomada
Estado inicial del sistema	04/06/2025 20:19 (hace 2 horas)
1-Prueba archivo malicioso	04/06/2025 22:12 (hace 7 minutos)
Estado actual (modificado)	

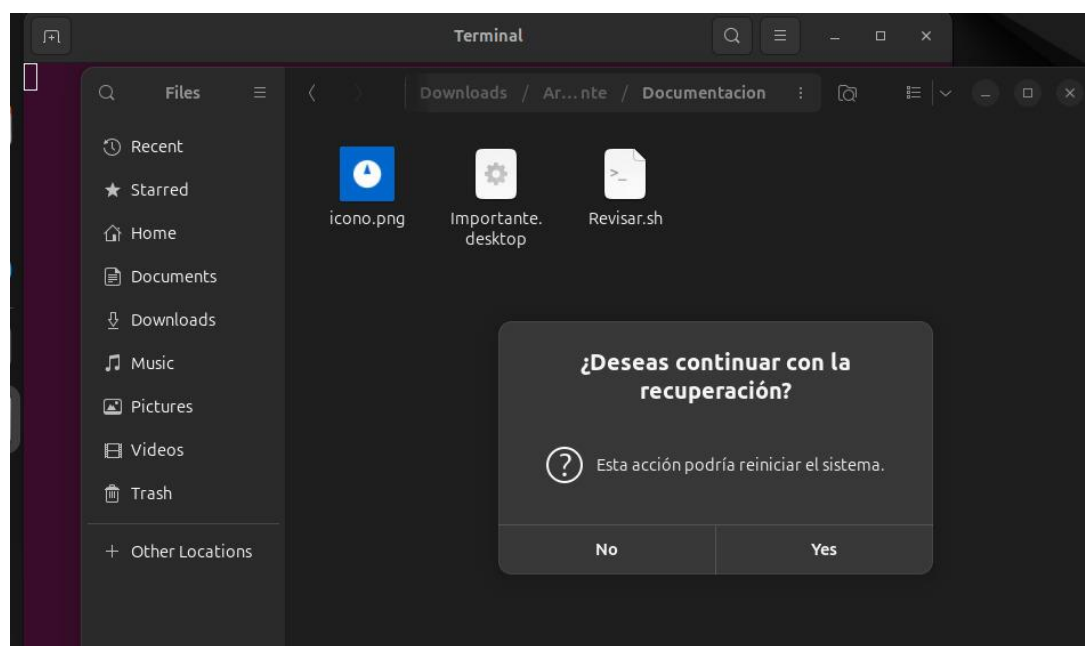
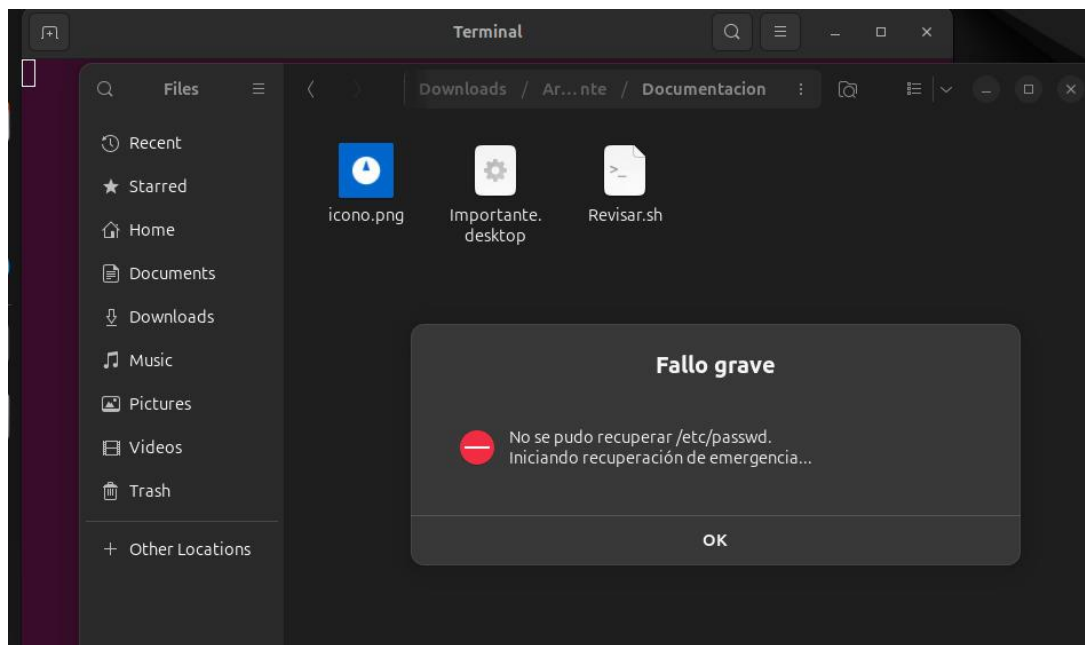
Luego, se descargó un archivo comprimido sospechoso, simulado como proveniente de una correo por las redes sociales o plataformas de compra-venta.

Este archivo fue descomprimido y ejecutado.



A continuación se muestra el accionar del archivo malicioso.

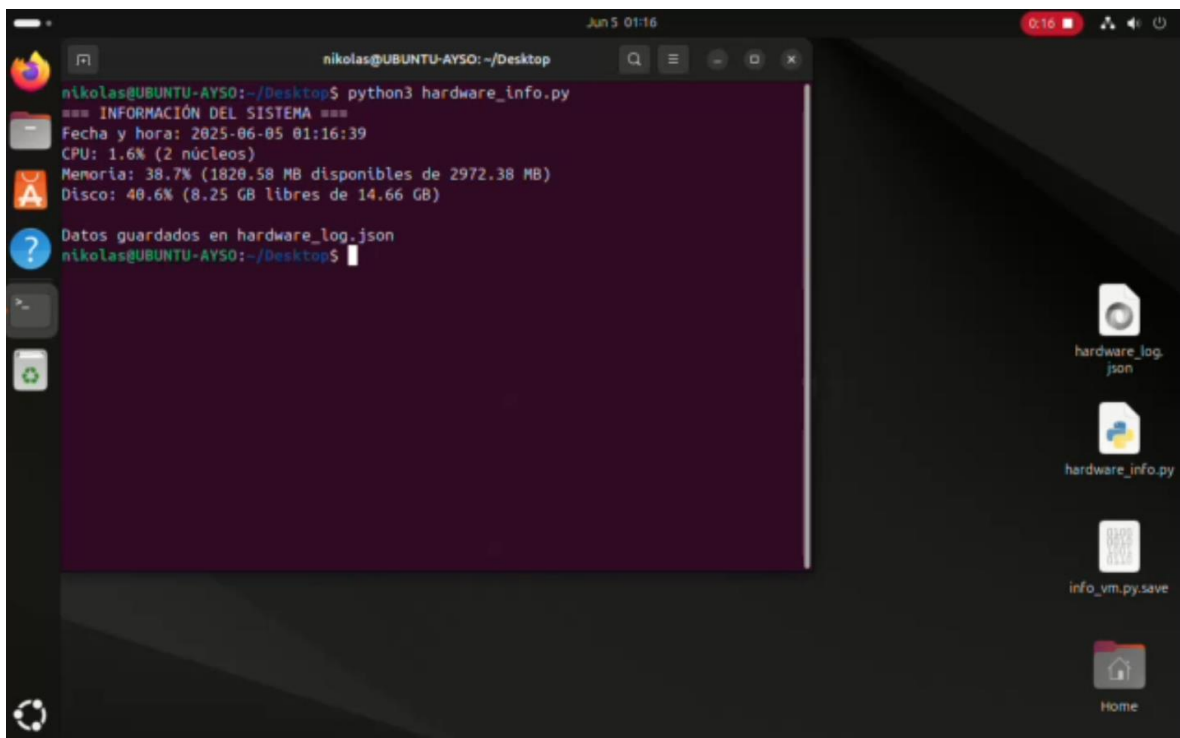
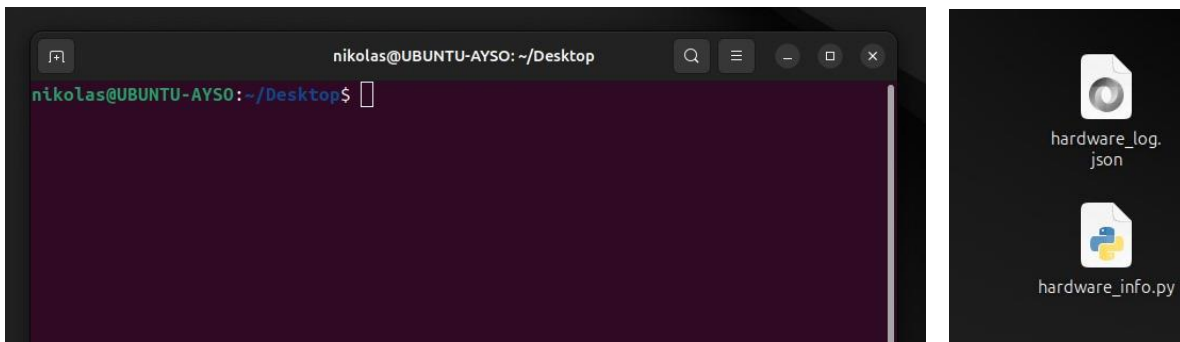




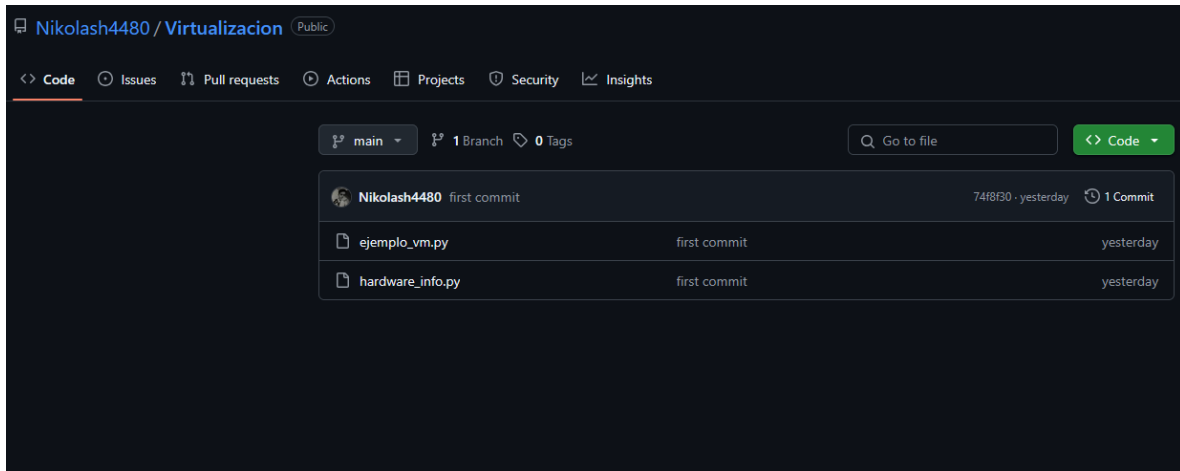
Caso 2 – Desarrollo de un script en Python

En una segunda máquina virtual Ubuntu, se implementó un sistema de monitoreo de hardware mediante un script desarrollado en Python que utiliza la librería psutil para obtener métricas en tiempo real del sistema virtualizado.

El script recopila información crítica como porcentaje de uso de CPU, cantidad de núcleos disponibles, consumo de memoria RAM (total, disponible y porcentaje utilizado), y estadísticas de almacenamiento en disco (espacio total, usado, libre y porcentaje ocupado).



El desarrollo se realizó utilizando metodologías de control de versiones, subiendo el código a un repositorio remoto (GitHub) y posteriormente clonándolo en la máquina virtual, simulando un flujo de trabajo colaborativo real.



Una vez ejecutado, el script muestra un reporte detallado del estado actual de los recursos del sistema y genera un archivo JSON con los datos para posterior análisis o integración con herramientas de monitoreo externas.

```
nikolas@UBUNTU-AYSO: ~/Desktop
nikolas@UBUNTU-AYSO:~/Desktop$ python3 hardware_info.py
=== INFORMACIÓN DEL SISTEMA ===
Fecha y hora: 2025-06-05 02:15:25
CPU: 0.0% (2 núcleos)
Memoria: 40.3% (1775.14 MB disponibles de 2972.38 MB)
Disco: 41.0% (8.19 GB libres de 14.66 GB)

Datos guardados en hardware_log.json
nikolas@UBUNTU-AYSO:~/Desktop$
```

```

import psutil
import json
from datetime import datetime

def get_hardware_info():
    """
    Devuelve el uso de CPU, cantidad de núcleos,
    memoria, y uso de disco
    """
    cpu_percent = psutil.cpu_percent(interval=1)
    cpu_count = psutil.cpu_count(logical=True)
    virtual_mem = psutil.virtual_memory()
    disk = psutil.disk_usage('/')
    return {
        "cpu_percent": cpu_percent,
        "cpu_count": cpu_count,
        "memory_total_mb": round(virtual_mem.total / (1024 ** 2), 2),
        "memory_available_mb": round(virtual_mem.available / (1024 ** 2),
2),
        "memory_used_percent": virtual_mem.percent,
        "disk_total_gb": round(disk.total / (1024 ** 3), 2),
        "disk_used_gb": round(disk.used / (1024 ** 3), 2),
        "disk_free_gb": round(disk.free / (1024 ** 3), 2),
        "disk_used_percent": disk.percent,
        "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    }

def main():
    """Ejecuta y muestra la información del hardware"""
    try:
        info = get_hardware_info()
        print("=== INFORMACIÓN DEL SISTEMA ===")
        print(f"Fecha y hora: {info['timestamp']}")
        print(f"CPU: {info['cpu_percent']}% ({info['cpu_count']} núcleos)")
        print(f"Memoria: {info['memory_used_percent']}% ({info[
'memory_available_mb']} MB disponibles de {info['memory_total_mb']} MB)"
        )
        print(f"Disco: {info['disk_used_percent']}% ({info['disk_free_gb']}
GB libres de {info['disk_total_gb']} GB)")

        # Opcional: guardar en JSON
        with open('hardware_log.json', 'w') as f:
            json.dump(info, f, indent=2)
        print("\nDatos guardados en hardware_log.json")

    except Exception as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    main()

```

Esta implementación demuestra cómo las máquinas virtuales pueden servir como plataformas de monitoreo independientes, permitiendo supervisar el rendimiento de sistemas sin impactar el host principal. Es especialmente útil para administradores de sistemas que necesitan evaluar el comportamiento de aplicaciones en entornos controlados antes de su despliegue en producción.

4. Metodología Utilizada

La metodología del trabajo incluyó investigación previa en documentación oficial y videos tutoriales, diseño y prueba de entornos virtuales, y desarrollo de scripts para automatizar configuraciones.

Se utilizó VirtualBox como plataforma principal para la virtualización

Para mejorar la trazabilidad, se tomaron capturas de pantalla de las sesiones de terminal, documentando cada paso.

Se aplicaron buenas prácticas de seguridad, incluyendo snapshots y configuración de redes virtuales aisladas para análisis seguro.

5. Resultados Obtenidos

Se logró configurar ambientes virtuales estables y seguros para pruebas y análisis, aplicando aislamiento de red para minimizar riesgos.

Los casos prácticos demostraron la funcionalidad de las máquinas virtuales para analizar archivos potencialmente maliciosos y para el desarrollo de scripts seguros.

Se identificaron dificultades iniciales en la configuración de red y en el compartir archivos entre el host y la máquina virtual, que fueron resueltas con guías y prueba sistemática.

El uso de snapshots facilitó la recuperación rápida ante errores o comportamientos inesperados.

6. Conclusiones

Se aprendió que la virtualización es una herramienta indispensable para la seguridad informática y el desarrollo de software, permitiendo entornos controlados y replicables. El análisis de malware en máquinas virtuales requiere configuraciones cuidadosas, como redes aisladas y documentación detallada, para evitar riesgos.

Además, se destacó la importancia de la responsabilidad ética al manipular software malicioso, trabajando siempre en ambientes controlados.

Futuras extensiones del trabajo podrían incluir la incorporación de herramientas avanzadas de análisis y automatización más completa del entorno.

7. Bibliografía

Oracle.(s.f.). VirtualBox Documentation.

<https://www.virtualbox.org/manual/>

Ubuntu. (s.f.). Ubuntu Documentation.

<https://help.ubuntu.com/>

Python Software Foundation. (s.f.). Python 3 Documentation.

<https://docs.python.org/3/>

YouTube. (2022). Cómo analizar archivos maliciosos en máquinas virtuales.

<https://www.youtube.com/watch?v=N5Wllb8MoL8>

YouTube. (2023). VirtualBox: Análisis seguro de malware .

<https://www.youtube.com/watch?v=MchGwIX1avk>