Estudio y Aplicación de la Virtualización

Este trabajo explora la virtualización, VirtualBox, redes virtuales y seguridad en seguridad en máquinas virtuales. Es crucial para la ciberseguridad, permitiendo permitiendo analizar archivos maliciosos en entornos controlados. Nuestro objetivo Nuestro objetivo es comprender, aplicar y simular escenarios de ciberseguridad, ciberseguridad, valorando la documentación y buenas prácticas.

Marco Teórico: Virtualización y Hipervisores

Virtualización de Hardware

Permite ejecutar sistemas operativos (VM) aislados sobre un host físico, físico, emulando un equipo independiente con CPU, memoria y disco. disco.

Hipervisores

Programas para crear y gestionar VMs. Tipo 1 ('bare-metal') se instala directamente. Tipo 2 ('hosted') se ejecuta como aplicación sobre un SO SO anfitrión. Usaremos VirtualBox (Tipo 2).

Imágenes ISO y Snapshots

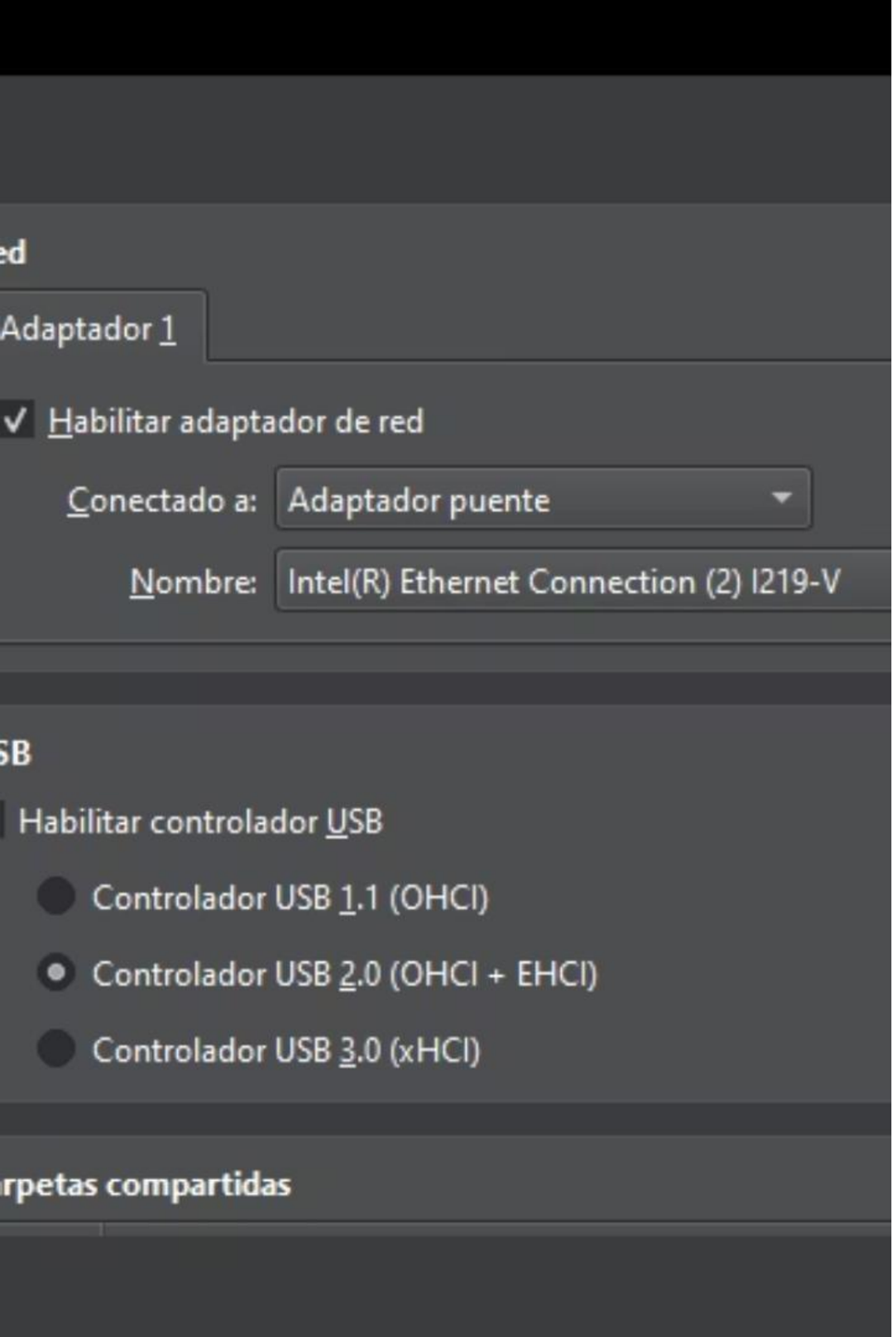
Imágenes ISO

Archivos que contienen una copia exacta de un medio óptico, usados para instalar sistemas operativos en máquinas virtuales.

Snapshots

Instantáneas del estado completo de una máquina virtual.

Permiten guardar y restaurar el sistema para pruebas, análisis o recuperación.



Tipos de Redes Virtuales

NAT

La VM comparte la conexión del host, con acceso a internet, pero no es accesible externamente.

Bridge

La VM se conecta a la red física, física, obteniendo una IP propia propia para comunicación plena.

Host-only

La VM solo se comunica con el host y otras VMs en la misma red, sin sin acceso a internet.

Enfoque del Trabajo

Máquinas Virtuales

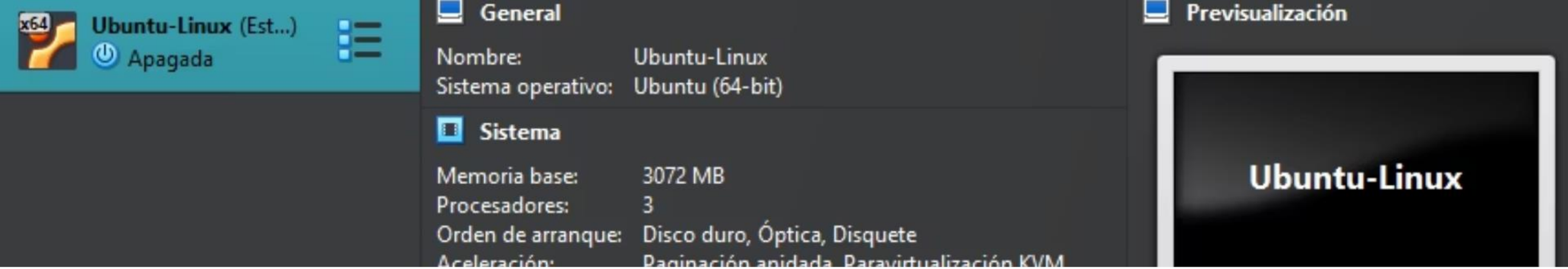
Herramientas de apoyo en desarrollo y análisis de seguridad seguridad informática.

Seguridad Informática

Recurso esencial para detectar, analizar y contener amenazas. amenazas.

Análisis de Malware

Énfasis en entornos virtuales para archivos potencialmente maliciosos.



Caso Práctico 1: Análisis de Archivo Sospechoso

VM Aislada

Ubuntu Desktop en VirtualBox con red interna aislada.

1

Snapshot Inicial

Instantánea del estado del sistema antes del análisis.

análisis.

2

Descarga y Ejecución

Archivo comprimido sospechoso descargado y ejecutado.

ejecutado.

3

Accionar Malicioso

Demostración del comportamiento del archivo malicioso.

m

4

Caso Práctico 2: Script de Monitoreo en Python

Desarrollo del Script

Script Python con psutil para monitorear hardware en VM Ubuntu.

Recopilación de Datos

Obtiene uso de CPU, RAM, núcleos y almacenamiento en tiempo real.

Control de Versiones

Código subido a GitHub y clonado en la VM para flujo colaborativo.

colaborativo.

Reporte Detallado

Muestra el estado de recursos y genera un archivo JSON.

```python
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()
```

Metodología Utilizada

Investigación

Documentación oficial y tutoriales en en video.

VirtualBox

Plataforma principal para la virtualización.

Capturas de Pantalla

Documentación de cada paso en sesiones de terminal.

Buenas Prácticas

Snapshots y redes virtuales aisladas para para seguridad.

Resultados Obtenidos

1

Ambientes Seguros

Configuración de entornos virtuales estables y seguros.

2

Funcionalidad Demostrada

Análisis de malware y desarrollo de scripts en VMs.

3

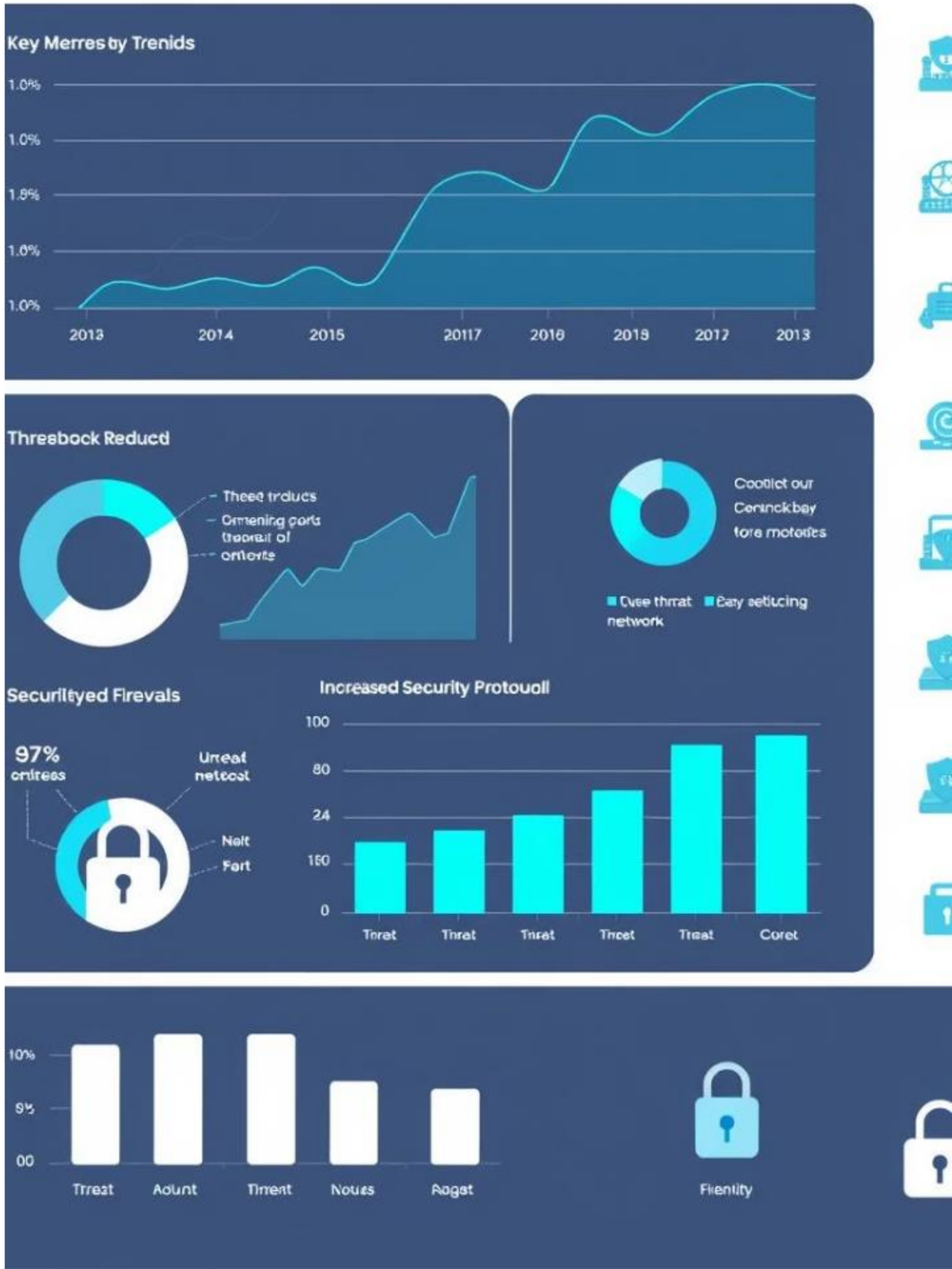
Dificultades Resueltas

Problemas de red y compartición de archivos superados.

4

Recuperación Rápida

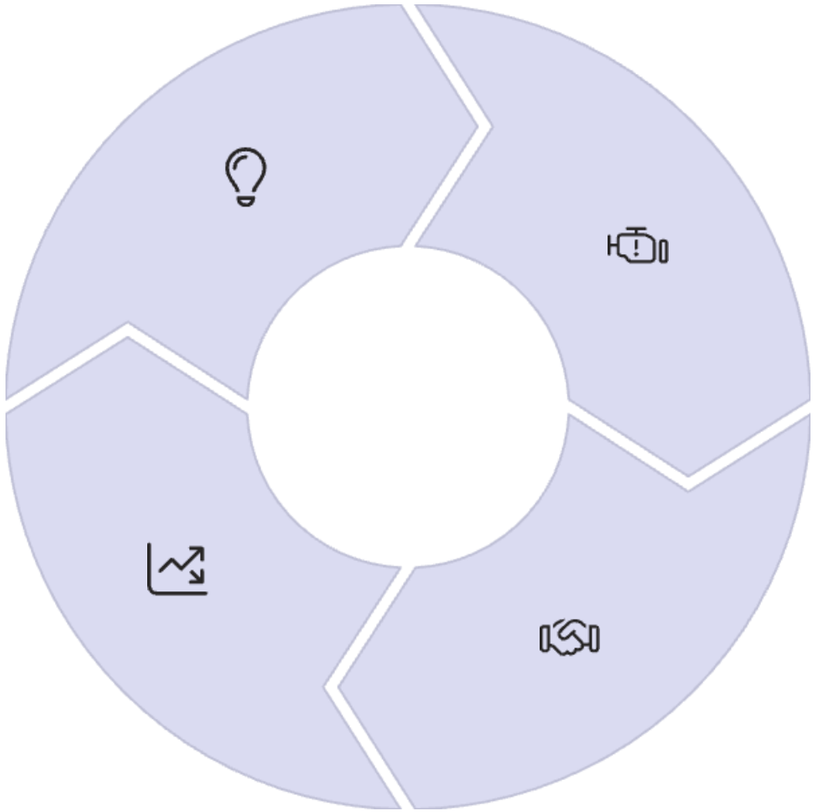
Uso de snapshots para restaurar estados fácilmente.



Conclusiones y Futuro

Virtualización Indispensable

Herramienta clave para seguridad y desarrollo de software.



Análisis de Malware

Requiere configuraciones cuidadosas y documentación detallada.

Extensiones Futuras

Incorporar herramientas avanzadas y automatización completa.

Responsabilidad Ética

Manipular software malicioso siempre en siempre en ambientes controlados.