Tarea 1: sistemas operativos

1st Miguel Angel Jimenez Morales - 2358571
2nd Dikersson Alexis Cañon Vanegas - 2366486
Estudiantes de Ingeniería de Telecomunicaciones
Facultad de Ingeniería
Universidad Santo Tomás de Aquino
Bogotá, Colombia

***Abstract*—En esta tarea se instalan unas maquinas virtuales a través del hipervisor y se conectan entre si**

***Index Terms*—Qemu, Maquina virtual, cron, ubuntu, Windows, Centos, Scienticlinus, hipervisor.**

I. Introducción

En el siguiente documento se desarrolla el procedimiento que se siguió para desarrollar la tarea.

II. Objetivo general

Aprender a manejar maquinas virtuales, comandos cron, crontab y nmap.

III. Objetivos específicos

- Instalar las siguientes maquinas virtuales en QEMU: Windows Centos Scientific Linux
- Utilizar nmap para mapear la red interna del pc de manera local cuando las máquinas virtuales estén encendidas y de manera global para visualizar que ve el pc de manera global.
- Utilizar Cron y Crontab, teniendo presente lo visto en la clase generar tres tareas diferentes automáticas.
- Subir los scripts y todo el desarrollo explicado de manera descriptiva a un repositorio de GitHub, la carpeta se debe llamar ”Tarea 1”.

IV. Marco Teórico

V. Virtualización con QEMU

La virtualización es una tecnología que permite ejecutar múltiples sistemas operativos en una misma máquina física mediante software especializado. QEMU (Quick Emulator) es un emulador y virtualizador de código abierto que permite la creación y gestión de máquinas virtuales. Cuando se combina con KVM (Kernel-based Virtual Machine), QEMU puede ofrecer virtualización acelerada por hardware en sistemas basados en Linux, proporcionando un rendimiento cercano al nativo.

QEMU soporta una amplia variedad de arquitecturas, lo que lo convierte en una herramienta versátil para pruebas y despliegue de sistemas operativos como Windows, CentOS y Scientific Linux. Su capacidad para emular hardware permite la instalación de diferentes sistemas operativos sin necesidad de modificar la máquina física.

VI. Sistemas Operativos en QEMU

- **Windows**: Uno de los sistemas operativos más utilizados en el mundo, diseñado principalmente para entornos de escritorio y servidores. Para su instalación en QEMU, se requiere una imagen ISO y controladores adicionales, como los VirtIO, para optimizar el rendimiento.
- **CentOS**: Distribución basada en Red Hat Enterprise Linux (RHEL), enfocada en estabilidad y soporte empresarial. Es ampliamente usada en servidores y entornos de desarrollo.
- **Scientific Linux**: Basada en RHEL, está diseñada para satisfacer las necesidades científicas y académicas, proporcionando estabilidad y compatibilidad con software científico.

VII. Mapeo de Red con Nmap

Nmap (Network Mapper) es una herramienta de código abierto utilizada para exploración de redes y auditoría de seguridad. Permite descubrir dispositivos conectados a una red, identificar servicios y detectar posibles vulnerabilidades. En este caso, se usará para:

- **Mapeo de red local**: Identificar las máquinas virtuales activas en la red interna del sistema anfitrión.
- **Mapeo global**: Analizar cómo se percibe la conexión del sistema en internet y detectar puertos abiertos expuestos.

VIII. Automatización con Cron y Crontab

Cron es un servicio de Unix/Linux que permite programar la ejecución de tareas de manera periódica. Crontab es el archivo de configuración donde se definen las tareas programadas. Estas tareas pueden incluir:

- **Ejecución de scripts de mantenimiento** (ejemplo: limpieza de archivos temporales).
- **Respaldo automático de datos** (ejemplo: copia de seguridad de archivos críticos).
- **Monitoreo de procesos y recursos** (ejemplo: envío de reportes del uso del sistema).

IX. Control de Versiones con GitHub

GitHub es una plataforma para la gestión de código fuente basada en Git. Permite almacenar, gestionar y colaborar en proyectos de desarrollo de software. En esta actividad, se utilizará para almacenar los scripts y la documentación en un repositorio, organizado en una carpeta denominada `Tarea 1`.

X. Overleaf y Documentación en Ámbito Académico

Overleaf es una plataforma en línea para la creación y edición de documentos en ámbito académico utilizando LaTeX. LaTeX es un sistema de composición de textos ampliamente utilizado en publicaciones científicas y académicas, debido a su capacidad para manejar ecuaciones matemáticas, referencias y formateo avanzado de documentos.

En esta actividad, se utilizará Overleaf para documentar el proceso de instalación y configuración de las máquinas virtuales, el uso de Nmap, la automatización con Cron y la gestión del repositorio en GitHub.

XI. Desarrollo de la tarea

A continuación se mostrará paso por paso el desarrollo de la tarea 1, atendiendo a los objetivos planteados. Como primera instancia tenemos la instalación de maquinas virtuales. La cual se hace con lo siguientes pasos: Se digitó en el bash de linux:

sudo apt install qemu-kvm virt-manager virtinst libvirt-clients bridge-utils libvirt-daemon-system

Esto instalara las dependencias necesarias para nuestro entorno virtual.

sudo systemctl enable –now libvirtd sudo systemctl start libvirtd sudo systemctl status libvirtd

Los comandos mostrados están relacionados con la configuración y gestión de libvirt, un conjunto de herramientas para administrar máquinas virtuales en Linux. Primero, sudo systemctl enable –now libvirtd habilita y arranca el servicio libvirtd de inmediato, asegurando que se inicie automáticamente con el sistema. Luego, sudo systemctl start libvirtd inicia el servicio manualmente (aunque esto es redundante si ya se ejecutó el primer comando), y sudo systemctl status libvirtd verifica su estado. Los siguientes comandos, sudo usermod -aG kvm $USER sudo usermod -aG libvirt $USER, agregan al usuario actual a los grupos kvm y libvirt, otorgándole permisos para gestionar máquinas virtuales sin necesidad de privilegios de superusuario. Para que los cambios surtan efecto, es necesario cerrar sesión o reiniciar el sistema.

Finalmente para abrir los entornos virtuales se ejecuta virt manager.

sudo virt-manager.

A continuación se muestra esta configuración en el bash:

```
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo systemctl enable --now libvirtd
[sudo] contraseña para miguelitron:
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo systemctl start libvirtd
miguelitron@miguelitron-Nitro-AN515-58:~$ dudo systemctl status libvirtd~
Orden «dudo» no encontrada. Quizá quiso decir:
```

Fig. 1. Se habilita la administración para maquinas virtuales

```
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo usermod -aG kvm $USER
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo usermod -aG libvirt $USER
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo virt-manager
```

Fig. 2. Se inicializa la maquina virtual

Se mostrara una ventana similar a esta:

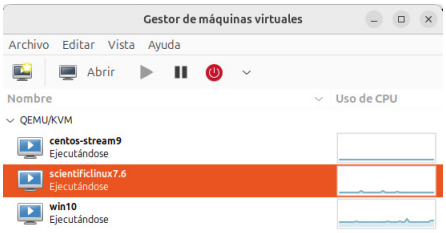

Fig. 3. Ventana de Virt Manager

A continuación se muestra como se configuran las maquinas virtuales desde el bash:

Fig. 4. Configuración de la maquina virtual.

```
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo qemu-system-x86_64 -enable-kvm -m 4G -smp 2 \
    -drive file=wlndows.qcow2,format=qcow2 \
    -cdrom "/home/miguelitron/Descargas/Wln10_22H2_Spanish_Mexico_x64v1.lso" \
    -boot d \
    -netdev user,id=net0 -device e1000,netdev=net0 \
    -vga qxl
```

Fig. 5. Configuración de la maquina virtual.

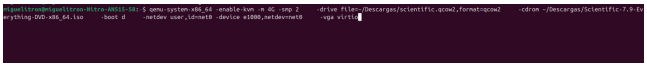

Fig. 6. Configuración de la maquina virtual.

Las líneas de comando configuran máquinas virtuales en QEMU con KVM en Ubuntu, emulando una arquitectura x86-64 con KVM habilitado, asignando 4 GB de RAM y 2 núcleos de CPU. Se montan discos en formato QCOW2 y se utilizan archivos ISO de instalación, configurando el arranque desde el CD. La red se establece con -netdev user,id=net0 y un adaptador e1000, mientras que la visualización se optimiza con -vga virtio o -vga qxl.

Una vez preparadas las maquinas virtuales, se procede con el segundo objetivo el cual es mapear la red con nmap. El siguiente comando nos instala nmap, con el cual haremos un scanning de red entre las maquinas virtuales.

```
miguelitron@miguelitron-Nitro-AN515-58:~$ sudo apt update && sudo apt install nmap -y
```

Fig. 7. Instalación de nmap para el mapeo de red entre Maquinas virtuales

A continuación se ejecuta un comando para saber las direcciones de las maquinas activas y de instancia local:

A continuación se muestra la configuración de cada una de estas maquinas virtuales.

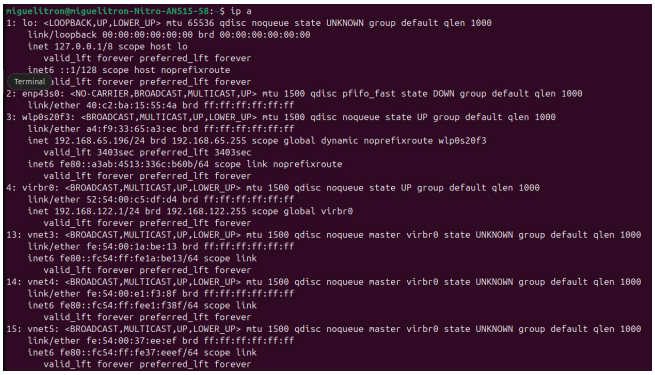

Fig. 8. Comando ip a, muestra maquinas virtuales activas

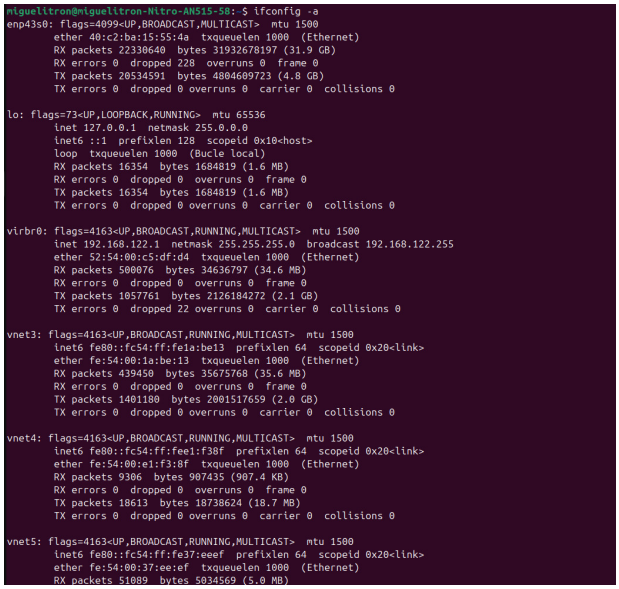

Fig. 9. Muestra la configuración de red de cada maquina virtual y la configuración local

A continuación se muestra un comando para ver la tabla de enrutamiento.

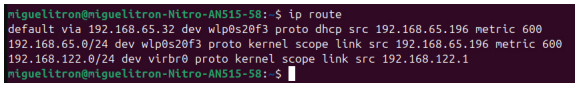

Fig. 10. Muestra la tabla de enrutamiento

El comando “ip route” se utiliza para mostrar la tabla de enrutamiento del sistema, indicando cómo se encamina el tráfico hacia diferentes redes o direcciones IP. En la salida observada, se detalla la ruta por defecto (default) que dirige todo el tráfico a la puerta de enlace 192.168.65.32 a través de la interfaz wlp0s20f3, con un origen en la dirección 192.168.65.196. Además, se listan rutas específicas para las redes 192.168.65.0/24 y 192.168.122.0/24, donde se señala la interfaz correspondiente (wlp0s20f3 y virbr0, respectivamente), la dirección IP de origen y el tipo de protocolo que ha definido estas rutas (en este caso, DHCP o kernel).

El comando “sudo nmap -sV -O 192.168.122.0/24” realiza un escaneo de toda la subred 192.168.122.0/24 para detectar hosts activos, puertos abiertos y sistemas operativos. En la salida se observan tres máquinas: 192.168.122.16 y 192.168.122.32, ambas con el puerto 22 (SSH) abierto y con coincidencias de Linux en distintas versiones de kernel, y 192.168.122.169, que muestra el puerto 5357 (HTTPAPI de Microsoft) y se identifica como un sistema Windows (posiblemente desde XP hasta Windows Server 2019). Además, las direcciones MAC indican que se trata de interfaces de red virtuales (QEMU), lo cual sugiere que estas máquinas están ejecutándose como máquinas virtuales.

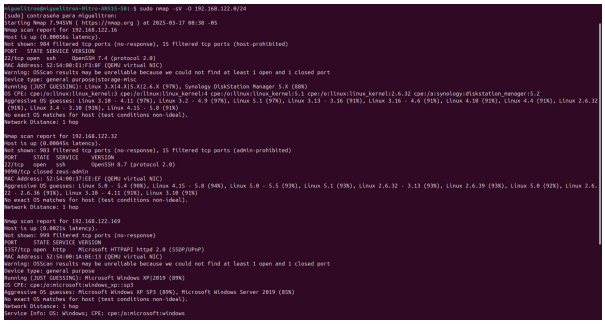

Fig. 11. Muestra las maquinas virtuales

A continuación se muestra el cron o crontab, con una funcion programa, para el github esta completo.

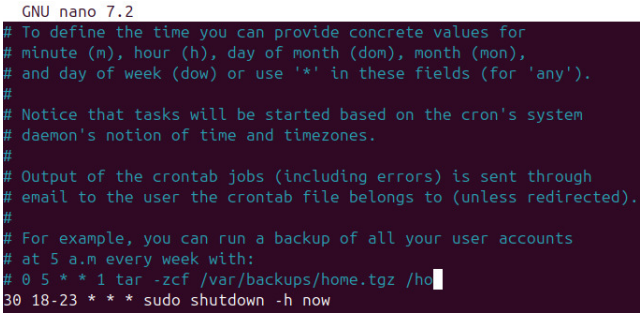

Fig. 12. Muestra las maquinas virtuales

A. *Desarrollo en git*

Se desarrollo el programa y se subieron los requerimientos a github incluyendo este documento. en el siguiente link:
https://github.com/MiguelXx7/tarea1

XII. Resultados

La instalación y configuración de las máquinas virtuales en QEMU permitió evaluar la compatibilidad y el rendimiento de cada sistema operativo en un entorno virtualizado. La ejecución de nmap facilitó el mapeo de la red interna, identificando los dispositivos conectados y sus respectivos puertos abiertos, lo que proporciona una visión clara de la infraestructura local. A nivel global, el análisis con nmap mostró la accesibilidad de nuestro sistema desde el exterior,

lo que permite evaluar posibles vulnerabilidades. Además, la implementación de tareas automatizadas con cron y crontab demostró la eficiencia de la programación en sistemas Linux, ejecutando procesos periódicos sin intervención manual. Finalmente, la documentación estructurada y la subida del proyecto a GitHub garantizaron la trazabilidad y disponibilidad del trabajo, lo que facilita futuras referencias y mejoras.

XIII. Conclusiones

- Instalación de máquinas virtuales en QEMU: La virtualización de Windows, CentOS y Scientific Linux con QEMU permitió evaluar el desempeño y compatibilidad de cada sistema operativo dentro de un entorno controlado. La configuración adecuada de cada máquina virtual facilitó la interacción con el hardware y la red del equipo anfitrión.
- Análisis de red con nmap: El escaneo de la red interna permitió identificar los dispositivos conectados y sus puertos abiertos, proporcionando información clave sobre la comunicación entre las máquinas virtuales y el sistema anfitrión. A nivel global, el análisis reflejó la visibilidad del sistema en la red externa, lo que es crucial para evaluar la exposición a posibles amenazas.
- Automatización con cron y crontab: La implementación de tres tareas automáticas demostró la utilidad de cron en la ejecución programada de procesos. Esto optimizó la gestión de tareas recurrentes y minimizó la necesidad de intervención manual, mejorando la eficiencia operativa del sistema.
- Las señales de se restan, en el amplificador de resta , se ve demostrado en el osciloscopio además en los resultados es evidente la resta de un v_2 a un v_1.
- Documentación y repositorio en GitHub: La estructuración del proyecto en un repositorio GitHub dentro de la carpeta "Tarea 1" garantizó la organización y trazabilidad del desarrollo. Además, la documentación en Overleaf permitió una presentación clara y profesional de los procedimientos y resultados obtenidos.

References

[1] Documentacion oficial de qemu. https://www.qemu.org/documentation/

[2] Guia de uso de nmap. https://www.geeksforgeeks.org/crontab-in-linux-with-examples/

[3] Manual de cron y crontab. https://man7.org/linux/man-pages/man5/crontab.5.html

[4] Documentación de GitHub y Overleaf: https://wiki.qemu.org/Documentation