

SOC (Security Operations Center)

Módulo 2



Sistema Operativo – Linux

Ejercicios

Sheila Fernández Cisneros – 24/05/2024

Tarea 1:

Navegar al directorio de trabajo /Documents (comprobar con `pwd`) y crear dos carpetas: `dir1` y `dir2`. Luego, dentro de la carpeta `dir1`, crea un archivo de texto, por ejemplo `file1.txt`, e introduce dentro de él la cadena “Hola Mundo”. Sin moverte de la carpeta `dir1`, crea una copia del archivo `file1.txt` dentro de `dir2`, cambiándole el nombre al archivo, por ejemplo `file1_cp.txt`. Nota: para usar la ruta del directorio padre recuerda usar “..”. Accede al directorio `dir2` y comprueba que el archivo de texto ‘`file1_cp.txt`’ está creado correctamente en este directorio, usando el comando `ls`. Una vez lo hayas comprobado, muestra el contenido del archivo en pantalla (debería salir la frase “Hola Mundo”). Crea el directorio ‘papelera’ dentro de `dir2` y mueve el archivo ‘`file1_cp.txt`’ a este nuevo directorio. Además, crea 2 nuevas carpetas dentro del directorio ‘papelera’. Una vez lo hayas hecho, vuelve hacia atrás y elimina por completo la carpeta papelera junto con todo su contenido interno en un solo comando.

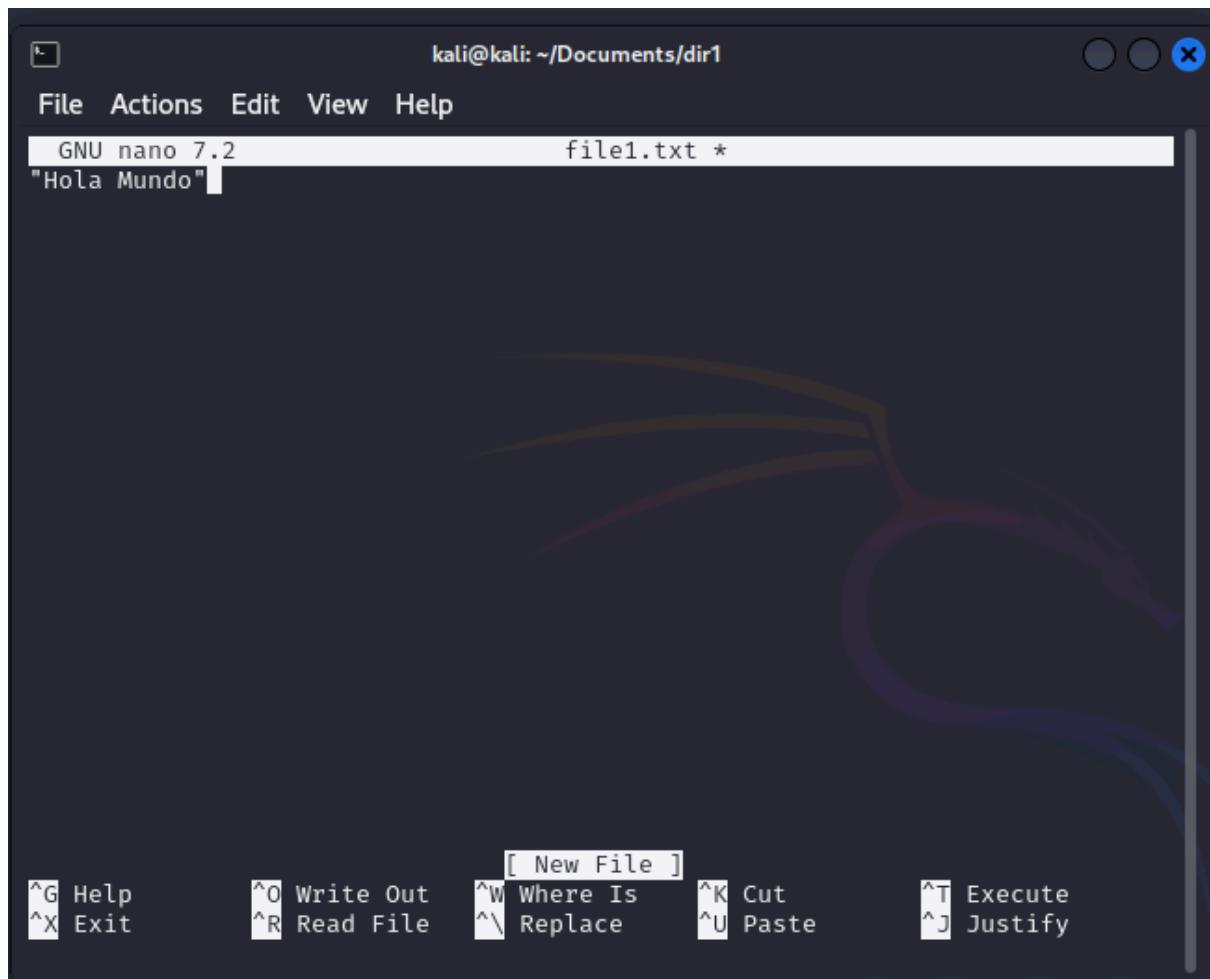
Comandos usados para resolver este ejercicio:

- **cd** : cambiar de directorio.
- **cd ..** : subir un nivel en la jerarquía de directorios (ir al directorio padre).
- **pwd** imprimir el directorio de trabajo actual.
- **mkdir nombre_del_directorio**: crear un nuevo directorio.
- **nano nombre_del_archivo**: abrir el editor de texto nano para editar archivos.
- **cp archivo ruta_destino**: copiar archivos o directorios.
- **ls** : listar los archivos y directorios en el directorio actual.
- **cat archivo**: mostrar el contenido de un archivo por terminal.
- **mv archivo ruta_destino**: mover o renombrar archivos o directorios.
- **rm -rf** : eliminar un directorio y su contenido de forma recursiva y forzada.

The screenshot shows a terminal window titled "kali@kali: ~/Documents". The terminal history is as follows:

```
(kali㉿kali)-[~]
$ cd Documents
(kali㉿kali)-[~/Documents]
$ pwd
/home/kali/Documents
(kali㉿kali)-[~/Documents]
$ mkdir dir1 dir2
(kali㉿kali)-[~/Documents]
$ cd dir1
(kali㉿kali)-[~/Documents/dir1]
$ nano file1.txt
(kali㉿kali)-[~/Documents/dir1]
$ cp file1.txt .. /dir2/file1_cp.txt
(kali㉿kali)-[~/Documents/dir1]
$ cd ..
(kali㉿kali)-[~/Documents]
$ cd dir2
(kali㉿kali)-[~/Documents/dir2]
$ ls
file1_cp.txt
(kali㉿kali)-[~/Documents/dir2]
$ cat file1_cp.txt
"Hello Mundo"
(kali㉿kali)-[~/Documents/dir2]
$ mkdir papelera
(kali㉿kali)-[~/Documents/dir2]
$ mv file1_cp.txt papelera/
(kali㉿kali)-[~/Documents/dir2]
$ mkdir papelera/folder1 papelera/folder2
(kali㉿kali)-[~/Documents/dir2]
$ cd ..
(kali㉿kali)-[~/Documents]
$ rm -rf dir2/papelera
(kali㉿kali)-[~/Documents]
$
```

El editor de texto nano se abre como pantalla emergente. Cuando se acaban de realizar los cambios oportunos nos dirigimos al menú inferior con las indicaciones que debemos tomar para guardar los cambios y salir del programa. En este caso, tras escribir la cadena “Hola Mundo”, se procedió a usar las opciones de teclado: Control O (Write Out), Enter, Control X (Exit). A continuación, se muestran capturas de pantalla con estos pasos.



kali@kali: ~/Documents/dir1

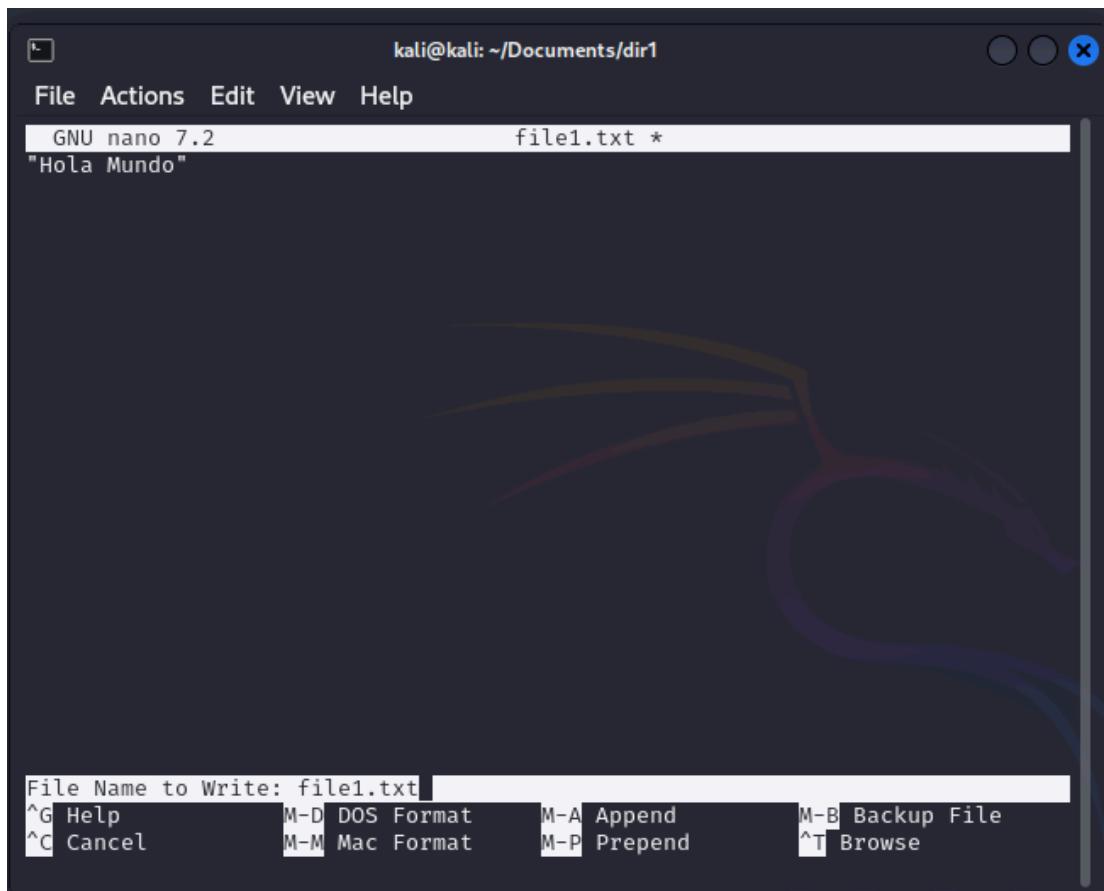
File Actions Edit View Help

GNU nano 7.2 file1.txt *

"Hola Mundo"

File Name to Write: file1.txt]

^G Help ^C Cancel M-D DOS Format M-M Mac Format M-A Append M-P Prepend M-B Backup File ^T Browse



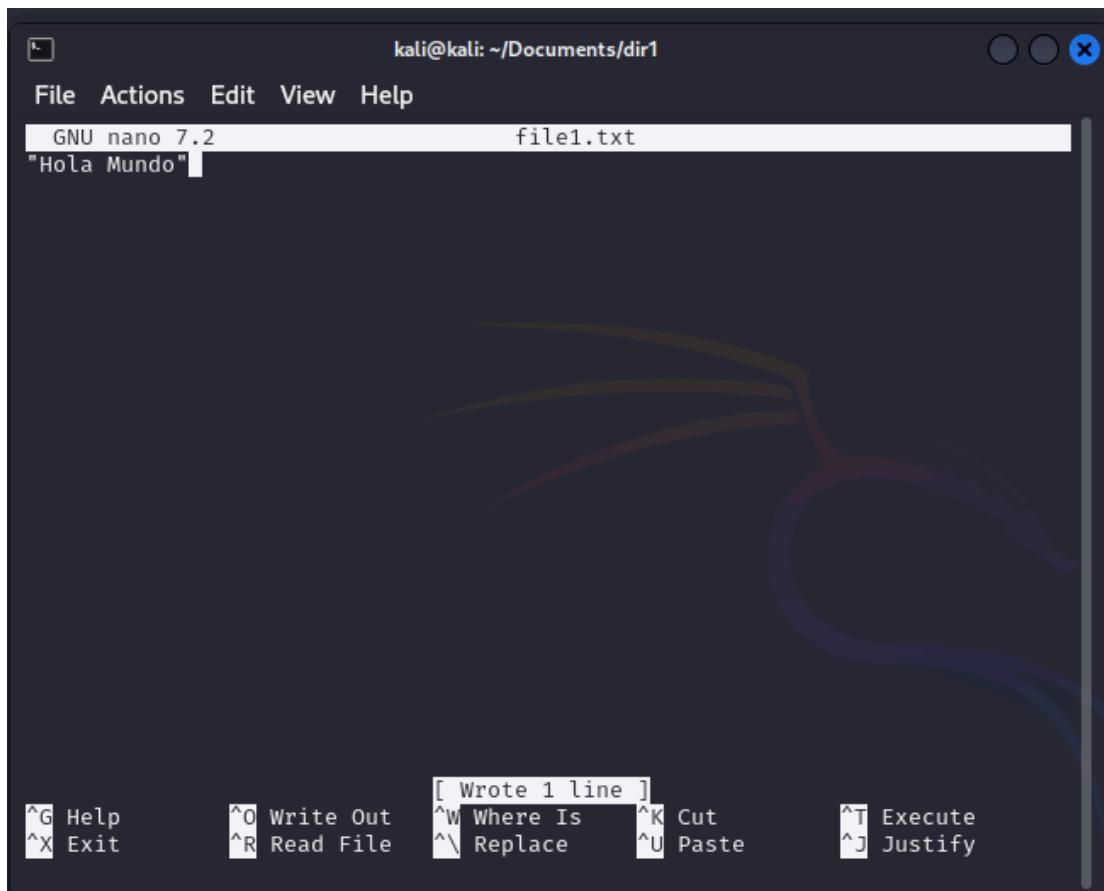
kali@kali: ~/Documents/dir1

File Actions Edit View Help

GNU nano 7.2 file1.txt

"Hola Mundo" [Wrote 1 line]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify



Tarea 2:

Realiza un script que al ejecutarse pregunte al usuario su nombre y tras leerlo, lo salude de forma personalizada. Haz uso de los comandos ‘echo’ y ‘read’ para escribir y leer al host respectivamente.

Para realizar este ejercicio se realizan los siguientes pasos:

1. Abrimos la terminal y navegamos al directorio `Documents`: cd Documents
2. Creamos un script llamado `saludo.sh` con el editor nano en el cual usamos los siguientes comandos:

- **`echo "Por favor, introduce tu nombre:"`**: Este comando imprime un mensaje en la terminal solicitando al usuario que ingrese su nombre.
- **`read nombre`**: Este comando lee la entrada del usuario desde la terminal y almacena ese valor en la variable `nombre`.
- **`echo "Hola, \$nombre! Bienvenida."`**: Este comando utiliza la variable `nombre` para imprimir un saludo personalizado en la terminal. La variable `nombre` se guarda en la memoria temporal del proceso del script mientras se está ejecutando. Cuando el script termina de ejecutarse, esta variable y su contenido se eliminan de la memoria.

Luego, se guarda el archivo y se cierra el editor. En `nano`, esto se hace presionando `Ctrl+O`, `Enter` y luego `Ctrl+X`.

3. Otorgamos permisos de ejecución al script: chmod 777 saludo.sh

Esto hace que el script sea ejecutable por cualquier usuario. Típicamente, se recomienda usar permisos más restrictivos como `chmod +x saludo.sh` para evitar riesgos de seguridad.

4. Ejecutamos el script: ./saludo.sh

Tras iniciar la ejecución del script, éste primero muestra el mensaje "Por favor, introduce tu nombre:", luego espera a que el usuario ingrese su nombre y lo almacena en la variable `nombre`. Finalmente, el script imprime un saludo personalizado usando el nombre ingresado.

```
kali@kali: ~/Documents
File Actions Edit View Help
(kali㉿kali)-[~]
$ cd Documents
(kali㉿kali)-[~/Documents]
$ nano saludo.sh
(kali㉿kali)-[~/Documents]
$ chmod 777 saludo.sh
(kali㉿kali)-[~/Documents]
$ ./saludo.sh
Por favor, introduzca su nombre:
Sheila
Hola, Sheila! Bienvenida.
(kali㉿kali)-[~/Documents]
```

```
kali@kali: ~/Documents
File Actions Edit View Help
GNU nano 7.2                               saludo.sh
#!/bin/bash

#Preguntar por el nombre del usuario
echo "Por favor, introduzca su nombre: "
read nombre #aqui se guarda el nombre ingresado por el usuario en la variable nombre

#saludar al usuario de forma personalizada
echo "Hola, $nombre! Bienvenida."
```



[Read 8 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^L Replace	^U Paste	^J Justify	^/ Go To Line

Tarea 3:

Crear un script donde se saque por pantalla el espacio de los discos del sistema (tamaño total, usado, libre), y el total de lo que ocupan todos los directorios de la raíz del sistema (recuerda que el directorio raíz es `/`). ¿Cuánto ocupa el disco y el directorio con más espacio y cuáles son? Explicar para qué sirven los comandos “`df`”, “`du`” y los diferentes “flags” que se utilizan para realizar el ejercicio.

**NOTA: Recordad que es necesario asignar permisos de ejecución al script, para ello:
chmod +x “nombre_script”**

Para realizar esta tarea, se necesita escribir un script de Bash que utilice los comandos `df` y `du` para mostrar la información del espacio de los discos y los directorios del sistema.

Si echamos un vistazo al manual por terminal usando el comando `man`, obtenemos información de estos comandos:

- `df` muestra la cantidad de espacio disponible en el sistema de archivos que contiene cada nombre de archivo argumento. Si no se proporciona ningún nombre de archivo, se muestra el espacio disponible en todos los sistemas de archivos montados actualmente. El espacio se muestra en bloques de 1K por defecto, a menos que la variable de entorno POSIXLY_CORRECT esté configurada, en cuyo caso se utilizan bloques de 512 bytes.

Si un argumento es el nombre de archivo absoluto de un nodo de dispositivo que contiene un sistema de archivos montado, `df` muestra el espacio disponible en ese sistema de archivos en lugar del sistema de archivos que contiene el nodo de dispositivo. Esta versión de `df` no puede mostrar el espacio disponible en sistemas de archivos no montados, porque en la mayoría de los tipos de sistemas, hacerlo requiere un conocimiento íntimo no portátil de las estructuras del sistema de archivos.

De las flags que presenta este comando, hay dos que nos pueden interesar para este ejercicio y son, `-h` que es indicada como para formatear el resultado para ser leído por el hombre y `T` que nos muestra el tipo de archivo (print file system type).

1. Mostrar el espacio de los discos del sistema: `df -hT`

```

└─(kali㉿kali)-[~/Documents]
└─$ man df

└─(kali㉿kali)-[~/Documents]
└─$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            970296       0   970296   0% /dev
tmpfs           202420     1024   201396   1% /run
/dev/sda1      82083148 19193632 58673968 25% /
tmpfs           1012080       0 1012080   0% /dev/shm
tmpfs            5120        0    5120   0% /run/lock
tmpfs           202416      120   202296   1% /run/user/1000

└─(kali㉿kali)-[~/Documents]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            948M   0   948M   0% /dev
tmpfs           198M  1.0M  197M   1% /run
/dev/sda1       79G   19G   56G  25% /
tmpfs           989M   0   989M   0% /dev/shm
tmpfs            5.0M   0   5.0M   0% /run/lock
tmpfs           198M 120K  198M   1% /run/user/1000

└─(kali㉿kali)-[~/Documents]
└─$ df -hT
Filesystem      Type      Size  Used Avail Use% Mounted on
udev            devtmpfs  948M   0   948M   0% /dev
tmpfs           tmpfs     198M  1.0M  197M   1% /run
/dev/sda1       ext4      79G   19G   56G  25% /
tmpfs           tmpfs     989M   0   989M   0% /dev/shm
tmpfs           tmpfs     5.0M   0   5.0M   0% /run/lock
tmpfs           tmpfs     198M 120K  198M   1% /run/user/1000

```

Como observamos en la imagen, al usar `df` sin argumentos, obtenemos el espacio en disco en bloques de 1K por defecto. No es muy legible para los humanos, pero es útil para scripts y análisis detallados. Se muestra el tamaño del sistema de archivos, el espacio usado, el espacio disponible, el porcentaje de uso y el punto de montaje.

Al añadir la flag `-h`, los resultados que se muestran son cifras convertidas a un formato legible para los humanos (por ejemplo, M para megabytes, G para gigabytes). Esto facilita la lectura y comprensión de la información del espacio en disco.

La flag `-T` además incluye el tipo de sistema de archivos ('Type'). Esto proporciona información adicional sobre qué tipo de sistema de archivos se está utilizando (por ejemplo, 'ext4', 'tmpfs', 'devtmpfs'), lo cual puede ser útil para entender la estructura y características del almacenamiento.

- `du` resume el uso del dispositivo del conjunto de archivos, recursivamente para directorios. Si le añadimos la flag `-s` (separate-dirs) no incluirá el tamaño de los subdirectorios.



```

└──(kali㉿kali)-[~/Documents]
└─$ man du

└──(kali㉿kali)-[~/Documents]
└─$ du
4      ./dir2
8      ./dir1
20      .

└──(kali㉿kali)-[~/Documents]
└─$ du -s
20      .

└──(kali㉿kali)-[~/Documents]
└─$ du -sh
20K      .

```

Al probar el comando `du -sh /*` aparece en el output muchos directorios que muestran que no tenemos permiso para acceder. Entonces usamos la redirección `2>/dev/null` que se utiliza para suprimir errores de acceso a directorios.

¿Qué hace el comando >2/dev/null en Linux? (Ref: <https://es.quora.com/Qu%C3%A9-hace-el-comando-2-dev-null-en-Linux>)

Esto redirecciona la salida estándar de errores (stderr), al dispositivo nulo (/dev/null). En otras palabras, después de este comando ya no verás errores pues todos ellos se irán al “vacío”.

EL operador “>” es para redireccionar la salida de un comando a “otro lado”, generalmente a otro archivo o fichero en Unix (Linux, BSD, AIX, HP-UX y todo sistema operativo de la familia *nix)

> archivo , redirecciona la salida estándar (stdout) a “archivo”
1> archivo , redirecciona igualmente la salida estándar (stdout) a “archivo”
2> archivo , redirecciona la salida estándar de errores (stderr) a “archivo”
&> archivo , redirecciona ambas, stdout y stderr a “archivo”.
/dev/null es el dispositivo nulo, la nada, el vacío.

En la siguiente captura podemos ver en la parte superior las últimas líneas del comando `du -sh /*` y la salida del comando `du -sh /* 2>/dev/null` .

2. Mostrar el espacio ocupado por todos los directorios en la raíz del sistema: `du -sh /* 2>/dev/null`

```
kali@kali: ~
File Actions Edit View Help
du: cannot read directory '/var/lib/docker': Permission denied
du: cannot read directory '/var/lib/php/sessions': Permission denied
du: cannot read directory '/var/lib/samba/usershares': Permission denied
du: cannot read directory '/var/lib/apt/lists/partial': Permission denied
du: cannot read directory '/var/lib/openvas/gnupg': Permission denied
du: cannot read directory '/var/lib/containerd': Permission denied
du: cannot read directory '/var/log/private': Permission denied
du: cannot read directory '/var/log/inetsim': Permission denied
du: cannot read directory '/var/log/speech-dispatcher': Permission denied
du: cannot read directory '/var/log/lightdm': Permission denied
786M    /var
0        /vmlinuz
0        /vmlinuz.old

└──(kali㉿kali)-[~]
$ du -sh /* 2>/dev/null
0        /bin
97M     /boot
0        /dev
13M     /etc
1.4G    /home
0        /initrd.img
0        /initrd.img.old
0        /lib
0        /lib32
0        /lib64
16K     /lost+found
4.0K    /media
4.0K    /mnt
184M    /opt
0        /proc
4.0K    /root
1.2M    /run
0        /sbin
8.0K    /srv
1.1G    /swapfile
0        /sys
64K     /tmp
13G     /usr
786M    /var
0        /vmlinuz
0        /vmlinuz.old
```

3. Calcular el directorio con más espacio ocupado en la raíz del sistema:

`**du -sh /* 2>/dev/null | sort -rh | head -n 1**`: Ordena la salida de `du` en orden descendente por tamaño (`sort -rh`) y muestra el primer resultado (`head -n 1`).

```

└─(kali㉿kali)-[~]
└─$ du -sh /* 2>/dev/null | sort -rh
13G      /usr
1.4G     /home
1.1G     /swapfile
786M    /var
184M    /opt
97M     /boot
13M     /etc
1.2M    /run
64K     /tmp
16K     /lost+found
8.0K    /srv
4.0K    /root
4.0K    /mnt
4.0K    /media
0       /vmlinuz.old
0       /vmlinuz
0       /sys
0       /sbin
0       /proc
0       /lib64
0       /lib32
0       /lib
0       /initrd.img.old
0       /initrd.img
0       /dev
0       /bin

└─(kali㉿kali)-[~]
└─$ du -sh /* 2>/dev/null | sort -rh | head -n 1
13G      /usr

```

4. Mostrar el tamaño total, usado y libre del sistema de archivos raíz: df -hT /

Si además queremos eliminar la primera línea: df -dT / | awk 'NR==2'

```

└─(kali㉿kali)-[~]
└─$ df -hT /
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/sda1        ext4   79G   19G   56G  25% /
 
└─(kali㉿kali)-[~]
└─$ df -hT / | awk 'NR=2'
/dev/sda1        ext4   79G   19G   56G  25% /

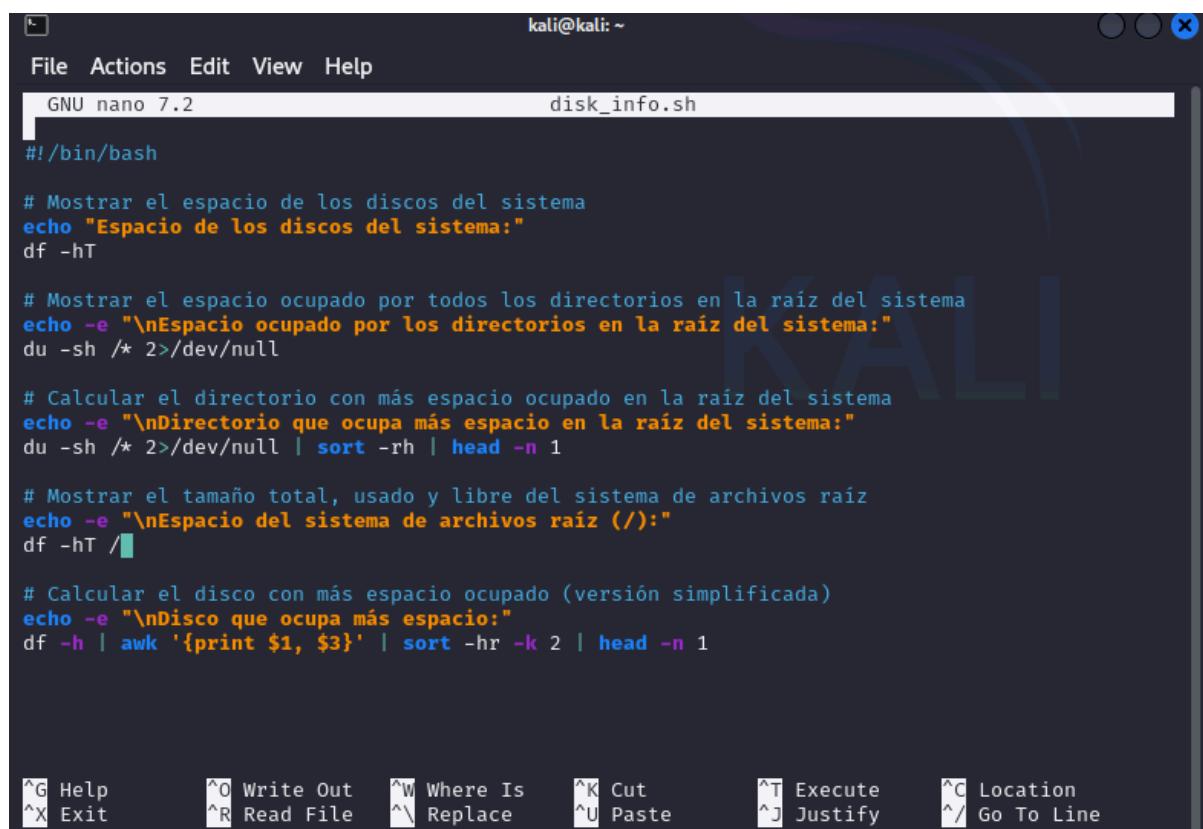
```

5. Calcular el disco con más espacio ocupado: `df -h | awk 'NR>1 {print $1, $3}' | sort -hr -k 2 | head -n 1`
- **Comando df -h:** Muestra el espacio en disco en un formato legible para humanos.
- **Filtrado y procesamiento con awk:** Muestra el nombre del sistema de archivos (columna 1) y el espacio usado (columna 3).
- **Ordenar y seleccionar el mayor uso:**

- **sort -hr -k 2**: Ordena la salida en orden descendente por el valor de la segunda columna (espacio usado).
- **head -n 1**: Muestra solo la primera línea, que corresponde al disco con más espacio ocupado.

```
└─(kali㉿kali)-[~]
$ df -hT | awk '{print $1, $3}' | sort -rh -k 2 | head -n 1
/dev/sda1 79G
└─(kali㉿kali)-[~]
```

Script:



```
kali@kali: ~
File Actions Edit View Help
GNU nano 7.2                         disk_info.sh
#!/bin/bash

# Mostrar el espacio de los discos del sistema
echo "Espacio de los discos del sistema:"
df -hT

# Mostrar el espacio ocupado por todos los directorios en la raíz del sistema
echo -e "\nEspacio ocupado por los directorios en la raíz del sistema:"
du -sh /* 2>/dev/null

# Calcular el directorio con más espacio ocupado en la raíz del sistema
echo -e "\nDirectorio que ocupa más espacio en la raíz del sistema:"
du -sh /* 2>/dev/null | sort -rh | head -n 1

# Mostrar el tamaño total, usado y libre del sistema de archivos raíz
echo -e "\nEspacio del sistema de archivos raíz (/):"
df -hT /

# Calcular el disco con más espacio ocupado (versión simplificada)
echo -e "\nDisco que ocupa más espacio:"
df -h | awk '{print $1, $3}' | sort -hr -k 2 | head -n 1

^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute     ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste      ^J Justify     ^/ Go To Line
```

Ejecución del script:

```

└──(kali㉿kali)-[~]
$ nano disk_info.sh

└──(kali㉿kali)-[~]
$ chmod +x disk_info.sh

└──(kali㉿kali)-[~]
$ ./disk_info.sh
Espacio de los discos del sistema:
Filesystem      Type     Size   Used  Avail Use% Mounted on
udev            devtmpfs  948M    0    948M  0% /dev
tmpfs           tmpfs    198M  1016K  197M  1% /run
/dev/sda1        ext4     79G   19G   56G  25% /
tmpfs           tmpfs    989M    0    989M  0% /dev/shm
tmpfs           tmpfs    5.0M    0    5.0M  0% /run/lock
tmpfs           tmpfs   198M  120K   198M  1% /run/user/1000
-e
Espacio ocupado por los directorios en la raíz del sistema:
0      /bin
97M    /boot
0      /dev
13M   /etc
1.4G  /home
0      /initrd.img
0      /initrd.img.old
0      /lib
0      /lib32
0      /lib64
16K   /lost+found
4.0K  /media
4.0K  /mnt
184M  /opt
0      /proc
4.0K  /root
1.2M  /run
0      /sbin
8.0K  /srv
1.1G  /swapfile
0      /sys
64K   /tmp
13G   /usr
786M  /var
0      /vmlinuz
0      /vmlinuz.old
-e
Directorio que ocupa más espacio en la raíz del sistema:
13G   /usr
-e
Espacio del sistema de archivos raíz (/):
Filesystem      Type   Size  Used  Avail Use% Mounted on
/dev/sda1        ext4   79G   19G   56G  25% /
-e
Disco que ocupa más espacio:
/dev/sda1 19G

└──(kali㉿kali)-[~]

```

Tarea 4:

Los archivos de logs son una herramienta muy interesante en el mundo de la ciberseguridad, ya que permiten comprobar los errores que se están produciendo en

algún servicio, como por ejemplo Apache. En este ejercicio se pide comprobar primeramente los archivos de logs de “apache2” (/var/log/apache2/) que existen, y probar si estos tienen algún log registrado. Tras esto, se debe probar a parar el servicio de apache2 con “systemctl stop” (mirar sintaxis del comando) y una vez parado el servicio, probar a acceder desde el navegador a la dirección <http://localhost:80>. Una vez hecho esto, se debe activar otra vez el servicio de apache2, volver a acceder desde el navegador a la dirección indicada para al final mirar los archivos de logs, e identificar cuáles han sido los logs registrados mientras el servicio de apache2 estaba desactivado y cuáles mientras estaba activado.

Comando systemctl:

El comando ‘systemctl’ es una utilidad de la línea de comandos que se utiliza para examinar y controlar el sistema y los servicios de administración en sistemas basados en Linux que utilizan ‘systemd’ como sistema de inicialización. ‘systemctl’ permite iniciar, detener, reiniciar, habilitar y deshabilitar servicios, así como verificar su estado.

Funciones Principales de `systemctl`

1. Iniciar un servicio: Inicia el servicio especificado.

`sudo systemctl start nombre_del_servicio`

2. Detener un servicio: Detiene el servicio especificado.

`sudo systemctl stop nombre_del_servicio`

3. Reiniciar un servicio: Detiene y luego inicia el servicio especificado.

`sudo systemctl restart nombre_del_servicio`

4. Recargar la configuración de un servicio: Recarga la configuración del servicio sin interrumpir su ejecución.

`sudo systemctl reload nombre_del_servicio`

5. Verificar el estado de un servicio: Muestra el estado actual del servicio, incluyendo si está activo, inactivo o fallado, y proporciona detalles de los logs.

`sudo systemctl status nombre_del_servicio`

6. Habilitar un servicio para que se inicie al arrancar el sistema: Habilita el servicio para que se inicie automáticamente al arrancar el sistema.

`sudo systemctl enable nombre_del_servicio`

7. Deshabilitar un servicio para que no se inicie al arrancar el sistema: Deshabilita el servicio para que no se inicie automáticamente al arrancar el sistema.

`sudo systemctl disable nombre_del_servicio`

8. Verificar si un servicio está habilitado: Comprueba si el servicio está habilitado para iniciarse al arrancar el sistema.

sudo systemctl is-enabled nombre_del_servicio

9. Listar todas las unidades (servicios, sockets, dispositivos, etc.): Lista todas las unidades cargadas actualmente.

sudo systemctl list-units

10. Listar todos los servicios: Lista todas las unidades de tipo servicio cargadas actualmente.

sudo systemctl list-units --type=service

Flags

- `start`: Inicia la unidad especificada (por ejemplo, un servicio).
- `stop`: Detiene la unidad especificada.
- `restart`: Reinicia la unidad especificada.
- `reload`: Recarga la configuración de la unidad especificada sin interrumpir su ejecución.
- `status`: Muestra el estado de la unidad especificada.
- `enable`: Habilita la unidad para que se inicie automáticamente al arrancar el sistema.
- `disable`: Deshabilita la unidad para que no se inicie automáticamente al arrancar el sistema.
- `is-enabled`: Comprueba si la unidad está habilitada para iniciarse al arrancar el sistema.
- `list-units`: Lista todas las unidades cargadas actualmente.
- `list-units --type=service`: Lista todas las unidades de tipo servicio cargadas actualmente.
- `daemon-reload`: Recarga los archivos de configuración del sistema (por ejemplo, después de modificar archivos de configuración de unidades).

1. Comprobar primeramente los archivos de logs de “apache2” (/var/log/apache2/) que existen.

Para ello listamos el contenido del directorio con el comando ls:

```
(kali㉿kali)-[~]
$ ls /var/log/apache2
access.log  error.log  other_vhosts_access.log
```

2. Probar si estos tienen algún log registrado.

Para ver si hay algún registrado, podemos usar un bucle for que itere por los archivos logs. En Bash, un bucle for declara e inicializa implícitamente la variable que usará para iterar.

```
#!/bin/bash

# Bucle for que itera sobre los archivos en /var/log/apache2/
for log in /var/log/apache2/*; do
    # Mostrar el nombre del archivo actual
    echo "Procesando archivo: $log"
    # Mostrar las últimas 10 líneas del archivo actual
    sudo tail -n 10 "$log"
done
```

- **Sintaxis del bucle for:**

```
for variable in lista; do
    # comandos
done
```

- **Declaración de la variable log:**

```
for log in /var/log/apache2/*; do
    o log: Es la variable que el bucle for declara e inicializa.
    o in /var/log/apache2/*: Indica que el bucle iterará sobre cada elemento (en este caso,
    cada archivo) que coincida con el patrón /var/log/apache2/*.
```

- **Inicialización y uso de la variable log:**

- o En cada iteración, la variable log se establece en el siguiente elemento de la lista de archivos en /var/log/apache2/.
- o Dentro del cuerpo del bucle, puedes utilizar \$log para referirte al archivo actual.
- Posteriormente se usa el comando **sudo** seguido de **tail -n 10 \$log** para que nos muestre las últimas 10 líneas de cada archivo log.

Comprobamos tras ejecutarlo que no hay ningún contenido en los archivos logs de este directorio.

```
(kali㉿kali)-[~]
$ for log in /var/log/apache2/*; do
echo "Contenido del archivo $log:"
sudo tail -n 10 $log
done
Contenido del archivo /var/log/apache2/access.log:
Contenido del archivo /var/log/apache2/error.log:
Contenido del archivo /var/log/apache2/other_vhosts_access.log:
```

Los siguientes puntos están recogidos en la captura final adjunta al ejercicio.

3. Probar a parar el servicio de apache2 con “systemctl stop”: para realizar esta acción, desde `root` usando **sudo** usamos el comando **systemctl stop apache2**, nos pide la contraseña de sudo y para el servicio apache2.

4. Una vez parado el servicio, probar a acceder desde el navegador a la dirección <http://localhost:80>. Accedemos al navegador con el comando **curl -I url**.

El comando curl -I se utiliza para realizar una solicitud HTTP HEAD a una URL. La solicitud HEAD es similar a una solicitud GET, pero en lugar de devolver el contenido completo del recurso, solo devuelve las cabeceras de respuesta HTTP. Esto puede ser útil para verificar la existencia de un recurso, obtener información sobre el tipo de contenido, tamaño, fecha de modificación, etc., sin descargar el contenido completo.

Comprobamos que no se obtiene nada porque el servicio está parado.

5. Una vez hecho esto, se debe activar otra vez el servicio de apache2, volver a acceder desde el navegador a la dirección indicada para al final mirar los archivos de logs.

Para activarlo de nuevo usamos el comando **systemctl** en este caso con la flag **start**, volvemos a comprobar la url con **curl -I** y en este caso ya vemos que si recibimos información.

Comprobamos los logs con el bucle for anterior y podemos ver que ya si hay registros.

```

└─(kali㉿kali)-[~]
└─$ for log in /var/log/apache2/*; do
echo "Contenido del archivo $log:"
sudo tail -n 10 $log
done
Contenido del archivo /var/log/apache2/access.log:
Contenido del archivo /var/log/apache2/error.log:
Contenido del archivo /var/log/apache2/other_vhosts_access.log:

└─(kali㉿kali)-[~]
└─$ sudo systemctl stop apache2
[sudo] password for kali:

└─(kali㉿kali)-[~]
└─$ curl -I http://localhost:80
curl: (7) Failed to connect to localhost port 80 after 0 ms: Couldn't connect to server

└─(kali㉿kali)-[~]
└─$ sudo systemctl start apache2

└─(kali㉿kali)-[~]
└─$ curl -I http://localhost:80
HTTP/1.1 200 OK
Date: Sat, 01 Jun 2024 11:01:06 GMT
Server: Apache/2.4.58 (Debian)
Last-Modified: Sun, 25 Feb 2024 15:55:18 GMT
ETag: "29cd-61236d1d67a20"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Type: text/html

└─(kali㉿kali)-[~]
└─$ for log in /var/log/apache2/*; do
echo "Contenido del archivo $log:"
sudo tail -n 10 $log
done
Contenido del archivo /var/log/apache2/access.log:
::1 - - [01/Jun/2024:13:01:06 +0200] "HEAD / HTTP/1.1" 200 255 "-" "curl/8.5.0"
Contenido del archivo /var/log/apache2/error.log:
[Sat Jun 01 13:01:00.594344 2024] [mpm_prefork:notice] [pid 98330] AH00163: Apache/2.4.58 (Debian) configured -- resuming normal operations
[Sat Jun 01 13:01:00.594400 2024] [core:notice] [pid 98330] AH00094: Command line: '/usr/sbin/apache2'
Contenido del archivo /var/log/apache2/other_vhosts_access.log:

└─(kali㉿kali)-[~]

```

6. Identificar cuáles han sido los logs registrados mientras el servicio de apache2 estaba desactivado y cuáles mientras estaba activado.

Para identificar claramente los registros en los archivos de logs mientras el servicio de Apache2 estaba desactivado y activado, vamos a revisar las entradas de los logs y explicar cuál corresponde a cada estado del servicio.

- Logs registrados mientras el servicio Apache2 estaba activado

Contenido de `/var/log/apache2/access.log`:

```
::1 - - [01/Jun/2024:13:01:06 +0200] "HEAD / HTTP/1.1" 200 255 "-" "curl/8.5.0"
```

- Fecha y hora: `[01/Jun/2024:13:01:06 +0200]`
- Método HTTP: `HEAD / HTTP/1.1`
- Código de estado HTTP: `200`
- Tamaño de la respuesta: `255`
- User-Agent: `curl/8.5.0`

Esta entrada en `access.log` indica que se realizó una solicitud HEAD al servidor Apache2 desde la dirección `::1` (IPv6 localhost) utilizando `curl`. El servidor respondió con un código de estado `200`, indicando éxito, y la respuesta tuvo un tamaño de `255` bytes. Este registro se generó cuando el servicio Apache2 estaba activado.

Contenido de `/var/log/apache2/error.log`:

```
[Sat Jun 01 13:01:00.594344 2024] [mpm_prefork:notice] [pid 98330] AH00163:  
Apache/2.4.58 (Debian) configured -- resuming normal operations
```

```
[Sat Jun 01 13:01:00.594400 2024] [core:notice] [pid 98330] AH00094: Command line:  
'/usr/sbin/apache2'
```

- Fecha y hora: `[Sat Jun 01 13:01:00.594344 2024]`
- Mensaje: `Apache/2.4.58 (Debian) configured -- resuming normal operations`
- Fecha y hora: `[Sat Jun 01 13:01:00.594400 2024]`
- Mensaje: `Command line: '/usr/sbin/apache2'`

Estas entradas en `error.log` indican que el servidor Apache2 fue configurado y reanudó sus operaciones normales a las `13:01:00` del 1 de junio de 2024. Estos mensajes se generan cuando el servicio Apache2 se inicia.

- Mientras Apache2 estaba desactivado:

No hay registros en los logs de acceso (`access.log`) o de error (`error.log`) que indiquen actividad mientras el servicio Apache2 estaba detenido. El intento de acceder a `http://localhost:80` falló, lo que es consistente con el hecho de que el servicio estaba desactivado.

- Mientras Apache2 estaba activado:

En `access.log`, se registró una solicitud HEAD realizada por `curl` cuando el servicio Apache2 estaba activado.

En `error.log`, se registraron mensajes que indican que Apache2 se configuró y reanudó operaciones normales al ser iniciado.

Estos registros nos permiten identificar claramente los momentos en que Apache2 estaba activado, y no hay registros durante el tiempo que estuvo desactivado.

Tarea 5:

Comprueba las direcciones IP de tu equipo para las diferentes interfaces de red, identificando también cual es la MAC, el MTU, la máscara de red, la dirección de broadcast y la dirección IPv6. Existen diversos comandos que permiten ver esta información. Pon una captura de la información recopilada e intenta comprender el significado de cada uno de estos términos (se verá más en profundidad en el módulo 3).

En primer lugar vamos a definir lo que nos está pidiendo este ejercicio:

- **IP (Internet Protocol) Address:** Un identificador único asignado a cada dispositivo en una red para permitir su comunicación.
- **Interfaces de red:** Conexiones de hardware o software que permiten a un dispositivo comunicarse en una red.
- **MAC (Media Access Control) Address:** Una dirección única de hardware asignada a cada interfaz de red para su identificación en una red local.
- **MTU (Maximum Transmission Unit):** El tamaño máximo de un paquete de datos que puede transmitirse a través de una interfaz de red sin fragmentación.
- **Máscara de red (Subnet Mask):** Un valor que divide una dirección IP en una parte de red y una parte de host, determinando el alcance de la subred.
- **Broadcast Address:** Una dirección que permite enviar datos a todos los dispositivos en una red específica.
- **IPv6 (Internet Protocol version 6) Address:** Una dirección IP de 128 bits diseñada para reemplazar IPv4, proporcionando un espacio de direcciones mucho mayor.

Hay un par de comandos que podemos usar para obtener esta información: `ifconfig` e `ip addr show`.

```
(kali㉿kali)-[~]
$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      inet 17.10.1.1 netmask 255.255.0.0 broadcast 17.10.1.255
        ether 02:10:10:01:00:00 txqueuelen 0 (Ethernet)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.10.1.10 netmask 255.255.255.0 broadcast 10.10.1.255
        inet6 fe80::10:10ff:fe01:10%eth0 brd ff:ff:ff:ff:ff:ff scopeid 0x20<link>
          ether 08:10:10:01:00:00 txqueuelen 1000 (Ethernet)
          RX packets 1 bytes 590 (590.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 29 bytes 3440 (3.3 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
          RX packets 18 bytes 1544 (1.5 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 18 bytes 1544 (1.5 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿kali)-[~]
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 brd ff00::1 scope global dynamic noprefixroute eth0
          valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:10:10:01:00:00 brd ff:ff:ff:ff:ff:ff
      inet 10.10.1.10/24 brd 10.10.1.255 scope global dynamic noprefixroute eth0
        valid_lft 71540sec preferred_lft 71540sec
        inet6 fe80::e539:95c0:b08f:38c4/64 brd ff:ff:ff:ff:ff:ff scope link noprefixroute
          valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:10:10:01:00:00 brd ff:ff:ff:ff:ff:ff
      inet 172.17.0.1/16 brd 172.17.0.1 scope global docker0
        valid_lft forever preferred_lft forever
```

Interpretación de la salida:

Interfaz de bucle invertido (link/loopback interface): Una interfaz de bucle invertido es una dirección IP virtual en un dispositivo que simula una conexión de red física. A diferencia de las interfaces físicas, las interfaces de bucle invertido no están asociadas a hardware específico y siempre se consideran "up" (activas) mientras el dispositivo esté en funcionamiento.

Interfaces de red:

- lo: Es el nombre asignado a la interfaz de bucle invertido en la mayoría de los sistemas Unix y Linux. Esta interfaz siempre está disponible y su dirección IP es 127.X.X.X para IPv4 y ::1 para IPv6.
- eth0: Interfaz Ethernet, utilizada para la comunicación con otras máquinas en la red.
- Docker0: La interfaz docker0 es una interfaz virtual creada por Docker. Actúa como un switch virtual que conecta todos los contenedores en una red predeterminada de Docker a

la máquina anfitriona. Permite la comunicación entre contenedores y con la máquina anfitriona.

Dirección IP (IPv4 e IPv6):

- IPv4:
 - inet 12X.X.X.X Dirección IPv4 de la interfaz de bucle invertido (lo).
 - inet 10.X.X.XX Dirección IPv4 de la interfaz eth0.
 - inet 17X.XX.X.X Dirección IPv4 de la interfaz docker0.
- IPv6:
 - inet6 ::1/1XX Dirección IPv6 de la interfaz de bucle invertido (lo).
 - inet6 fe80::XXXX:XXXX:XXXX:XXXX Dirección IPv6 de enlace local de la interfaz eth0.
 - Por defecto, Docker configura la red docker0 solo con IPv4. Para habilitar IPv6, es necesario ajustar la configuración de Docker.

MAC (Media Access Control) Address:

- lo: No está asociada con ningún hardware de red físico, por tanto no tiene MAC address.
- eth0: ether 08:XX:XX:XX:XX:XX Dirección MAC de la interfaz eth0. Es una dirección única de hardware asignada por el fabricante.
- docker0: 02: XX:XX:XX:XX:XX Dirección MAC de la interfaz docker0.

MTU (Maximum Transmission Unit):

- mtu 65536: MTU de la interfaz lo, indica el tamaño máximo de un paquete que puede manejar la interfaz de bucle invertido.
- mtu 1500: MTU de la interfaz eth0, indica el tamaño máximo de un paquete que puede manejar la interfaz Ethernet.
- mtu 1500: MTU de la interfaz docker0, indica el tamaño máximo de un paquete que puede manejar la interfaz Docker.

Máscara de red (netmask):

- 255.0.0.0 Máscara de red para la interfaz de bucle invertido lo.
- 255.255.255.0 Máscara de red para la interfaz eth0.
- 255.255.0.0 Máscara de red para la interfaz docker0.

Dirección de Broadcast:

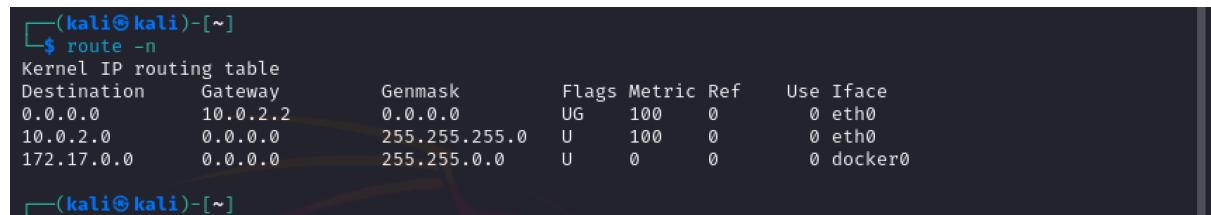
- lo: La interfaz de bucle invertido no tiene una dirección de broadcast porque no necesita ni utiliza la funcionalidad de broadcast. Está diseñada para la comunicación interna dentro del mismo sistema operativo. Permite que los programas y servicios en la misma máquina se comuniquen entre sí sin salir a la red física. No interactúa con otras interfaces.
- eth0: 10.X.X.XXX Dirección de broadcast para la interfaz eth0, utilizada para enviar paquetes a todos los dispositivos en la red.
- docker0: 172.XX.XXX.XXX Dirección broadcast para la interfaz docker0.

Tarea 6:

Usa el comando route -n para ver la tabla de enrutamientos de nuestro equipo actual. En este ejercicio se pide añadir una nueva dirección de red a nuestra tabla de enrutamiento (mirar sintaxis del comando route). La dirección a añadir es la 192.168.100.0, la netmask es 255.255.255.0 y la gateway <mirar Gateway con route -n>. Una vez añadida, vuelve a usar el comando route -n para asegurarte de que se añadió correctamente. Tras esto, vuelve a eliminarla. Si quieres, puedes realizar un script para mostrar las modificaciones sucesivas en las tablas.

El enrutamiento es el proceso de determinar el camino o la ruta que los datos deben seguir a través de una red para llegar desde su origen hasta su destino. Este proceso es esencial para el funcionamiento de las redes de datos, incluida Internet, ya que permite la transmisión eficiente y efectiva de información entre dispositivos en diferentes redes.

Usamos el comando `route -n` para ver la tabla de enrutamiento actual. Este comando se utiliza para mostrar la tabla de enrutamiento del kernel, que contiene información sobre las rutas que los paquetes de red pueden seguir para alcanzar diferentes destinos. El -n es una opción que indica al comando que muestre las direcciones IP en formato numérico, en lugar de resolverlas a nombres de host, lo que hace que la salida sea más rápida y directa.



```
(kali㉿kali)-[~]
$ route -n
Kernel IP routing table
Destination      Gateway        Genmask        Flags Metric Ref    Use Iface
0.0.0.0          10.0.2.2      0.0.0.0        UG    100    0        0 eth0
10.0.2.0         0.0.0.0       255.255.255.0  U     100    0        0 eth0
172.17.0.0        0.0.0.0       255.255.0.0   U     0      0        0 docker0
```

Para añadir una ruta nueva usaremos con sudo el comando:

`sudo route add -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0`

Este comando añadirá una ruta estática a la tabla de enrutamiento del sistema.

- Añade una entrada en la tabla de enrutamiento del sistema para la red 192.168.100.0/24.

- Todo el tráfico destinado a la red 192.168.100.0 (cualquier dirección IP desde 192.168.100.0 hasta 192.168.100.255) será enviado a través del gateway 10.0.2.0.

```
(kali㉿kali)-[~]
└─$ sudo route add -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0
[sudo] password for kali:

(kali㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2       0.0.0.0        UG    100    0        0 eth0
10.0.2.0        0.0.0.0        255.255.255.0   U     100    0        0 eth0
172.17.0.0      0.0.0.0        255.255.0.0    U     0      0        0 docker0
192.168.100.0   10.0.2.0       255.255.255.0   UG    0      0        0 eth0

(kali㉿kali)-[~]
```

Para eliminar la ruta añadida usaremos el mismo comando pero en este caso usando la flag `del`:

`sudo route del -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0`

```
(kali㉿kali)-[~]
└─$ sudo route del -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0

(kali㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2       0.0.0.0        UG    100    0        0 eth0
10.0.2.0        0.0.0.0        255.255.255.0   U     100    0        0 eth0
172.17.0.0      0.0.0.0        255.255.0.0    U     0      0        0 docker0

(kali㉿kali)-[~]
```

Script para automatizar el proceso:



```

kali㉿kali: ~
File Actions Edit View Help
GNU nano 7.2                                     routing.sh *
#!/bin/bash

# Mostrar la tabla de enrutamiento actual
echo "Tabla de enrutamiento actual:"
route -n

# Añadir una nueva ruta
echo "Añadiendo nueva ruta ... "
sudo route add -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0

# Verificar que la ruta se añadió correctamente
echo "Tabla de enrutamiento después de añadir la nueva ruta:"
route -n

# Eliminar la ruta añadida
echo "Eliminando la ruta añadida ... "
sudo route del -net 192.168.100.0 netmask 255.255.255.0 gw 10.0.2.0

# Verificar que la ruta se eliminó correctamente
echo "Tabla de enrutamiento después de eliminar la ruta:"
route -n

```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
 ^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo

Ejecución del script:

```

└─(kali㉿kali)-[~]
$ nano routing.sh

└─(kali㉿kali)-[~]
$ chmod +x routing.sh

└─(kali㉿kali)-[~]
$ ./routing.sh
Tabla de enrutamiento actual:
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2       0.0.0.0       UG    100    0        0 eth0
10.0.2.0        0.0.0.0        255.255.255.0 U      100    0        0 eth0
172.17.0.0      0.0.0.0        255.255.0.0   U      0      0        0 docker0
Añadiendo nueva ruta ...
Tabla de enrutamiento después de añadir la nueva ruta:
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2       0.0.0.0       UG    100    0        0 eth0
10.0.2.0        0.0.0.0        255.255.255.0 U      100    0        0 eth0
172.17.0.0      0.0.0.0        255.255.0.0   U      0      0        0 docker0
192.168.100.0   10.0.2.0       255.255.255.0 UG    0      0        0 eth0
Eliminando la ruta añadida ...
Tabla de enrutamiento después de eliminar la ruta:
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2       0.0.0.0       UG    100    0        0 eth0
10.0.2.0        0.0.0.0        255.255.255.0 U      100    0        0 eth0
172.17.0.0      0.0.0.0        255.255.0.0   U      0      0        0 docker0

```

Tarea 7:

Dentro de la seguridad informática es crucial proteger los sistemas de intrusiones que provoquen ataques maliciosos. Con la finalidad de evitar estos accesos no deseados a nuestros dispositivos se utilizan firewall, ya que actúan como primera línea de defensa filtrando el tráfico de la red y bloqueando accesos no permitidos. En este ejercicio vamos a realizar diferentes configuraciones dentro del firewall haciendo uso de iptables, creando reglas que nos permitan filtrar el tráfico de red. En primer lugar, comprueba y aporta una captura de pantalla de la lista de reglas del firewall presentes inicialmente en el sistema. A continuación, se pide realizar las siguientes modificaciones (para ello se recomienda revisar cómo funciona el comando “iptables”):

Al usar `man iptables`, podemos ver que es una herramienta para filtrar paquetes IPv4.

Para ver la lista de reglas del firewall presentes en el sistema usaremos las flags -L -v:

- **-L:** Esta opción indica a iptables que debe listar todas las reglas en las cadenas de la tabla seleccionada (por defecto, la tabla filter, que es la tabla principal usada para el filtrado de paquetes).
- **-v:** Esta opción activa el modo "verbose" (detallado), lo que significa que iptables mostrará información adicional sobre cada regla en las cadenas, incluyendo contadores de paquetes y bytes.

```

└─(kali㉿kali)-[~]
$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
  0    0 DOCKER-USER  all -- any   any   anywhere       anywhere
  0    0 DOCKER-ISOLATION-STAGE-1 all -- any   any   anywhere       anywhere
  0    0 ACCEPT      all -- any   docker0 anywhere       anywhere      ctstate RELATED,
ESTABLISHED
  0    0 DOCKER      all -- any   docker0 anywhere       anywhere
  0    0 ACCEPT      all -- docker0 !docker0 anywhere       anywhere
  0    0 ACCEPT      all -- docker0 docker0 anywhere       anywhere

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination

Chain DOCKER (1 references)
pkts bytes target     prot opt in     out      source          destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
pkts bytes target     prot opt in     out      source          destination
  0    0 DOCKER-ISOLATION-STAGE-2 all -- docker0 !docker0 anywhere       anywhere
  0    0 RETURN      all -- any   any   anywhere       anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
pkts bytes target     prot opt in     out      source          destination
  0    0 DROP        all -- any   docker0 anywhere       anywhere
  0    0 RETURN      all -- any   any   anywhere       anywhere

Chain DOCKER-USER (1 references)
pkts bytes target     prot opt in     out      source          destination
  0    0 RETURN      all -- any   any   anywhere       anywhere

└─(kali㉿kali)-[~]
$ 

```

a) Probar a hacer PING a la dirección 8.8.8.8 (Google). Bloquear dicha dirección. Para ello hay que tener en cuenta que la regla tiene que bloquear el tráfico que sale de nuestra máquina (OUTPUT), y que la dirección que queremos bloquear es la de destino. Comprobar si se ha aplicado correctamente haciendo PING de nuevo.

El comando ping es una herramienta de red que se utiliza para probar la conectividad entre dos dispositivos en una red.

Probamos a hacer PING a 8.8.8.8:

```
(kali㉿kali)-[~]
└─$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=30.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=32.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=32.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=31.5 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=93.3 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=63 time=156 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=63 time=139 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=63 time=158 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=63 time=25.1 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=63 time=26.3 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=63 time=25.4 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=63 time=25.7 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=63 time=27.0 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=63 time=29.7 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=63 time=24.7 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=63 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=63 time=25.1 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=63 time=27.3 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=63 time=23.2 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=63 time=29.7 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=63 time=29.3 ms
64 bytes from 8.8.8.8: icmp_seq=22 ttl=63 time=30.3 ms
64 bytes from 8.8.8.8: icmp_seq=23 ttl=63 time=26.6 ms
64 bytes from 8.8.8.8: icmp_seq=24 ttl=63 time=34.3 ms
64 bytes from 8.8.8.8: icmp_seq=25 ttl=63 time=31.0 ms
64 bytes from 8.8.8.8: icmp_seq=26 ttl=63 time=42.9 ms
64 bytes from 8.8.8.8: icmp_seq=27 ttl=63 time=24.5 ms
64 bytes from 8.8.8.8: icmp_seq=28 ttl=63 time=29.2 ms
64 bytes from 8.8.8.8: icmp_seq=29 ttl=63 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=30 ttl=63 time=30.0 ms
64 bytes from 8.8.8.8: icmp_seq=31 ttl=63 time=29.8 ms
^C
— 8.8.8.8 ping statistics —
31 packets transmitted, 31 received, 0% packet loss, time 30094ms
rtt min/avg/max/mdev = 23.241/42.815/158.361/37.457 ms
```

Añadimos una regla para bloquear el tráfico hacia 8.8.8.8 usando el comando **sudo iptables -A OUTPUT -d 8.8.8.8 -j DROP** y lo comprobamos que se ha creado con **sudo iptables -L -v**.

- **-A:** Significa "append" (añadir). Añade una nueva regla al final de la cadena especificada. En este caso, la cadena especificada es OUTPUT.
- **OUTPUT:** Especifica la cadena a la que se está añadiendo la regla. La cadena OUTPUT maneja el tráfico de salida desde la máquina local.
- **-d:** significa "destination" (destino). Especifica que la regla se aplica a los paquetes que tienen como destino la dirección IP 8.8.8.8.
- **-j DROP:** -j significa "jump" (saltar). Especifica la acción que debe tomarse con los paquetes que coinciden con esta regla. DROP significa que los paquetes serán descartados sin ninguna notificación.

```

└──(kali㉿kali)-[~]
$ sudo iptables -A OUTPUT -d 8.8.8.8 -j DROP

└──(kali㉿kali)-[~]
$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
0     0   DOCKER-USER  all  --  any    any     anywhere       anywhere
0     0   DOCKER-ISOLATION-STAGE-1 all  --  any    any     anywhere       anywhere
0     0   ACCEPT     all  --  any    docker0 anywhere       anywhere      ctstate RELATED,
ESTABLISHED
0     0   DOCKER     all  --  any    docker0 anywhere       anywhere
0     0   ACCEPT     all  --  docker0 !docker0 anywhere       anywhere
0     0   ACCEPT     all  --  docker0 docker0  anywhere       anywhere

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
0     0   DROP       all  --  any    any     anywhere       dns.google

Chain DOCKER (1 references)
pkts bytes target     prot opt in     out      source          destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
pkts bytes target     prot opt in     out      source          destination
0     0   DOCKER-ISOLATION-STAGE-2 all  --  docker0 !docker0 anywhere       anywhere
0     0   RETURN     all  --  any    any     anywhere       anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
pkts bytes target     prot opt in     out      source          destination
0     0   DROP       all  --  any    docker0 anywhere       anywhere
0     0   RETURN     all  --  any    any     anywhere       anywhere

Chain DOCKER-USER (1 references)
pkts bytes target     prot opt in     out      source          destination
0     0   RETURN     all  --  any    any     anywhere       anywhere

└──(kali㉿kali)-[~]
$ 

```

Comprobamos que efectivamente se ha cortado el tráfico haciendo ping de nuevo a 8.8.8.8:

```

└──(kali㉿kali)-[~]
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
— 8.8.8.8 ping statistics —
10 packets transmitted, 0 received, 100% packet loss, time 9202ms

└──(kali㉿kali)-[~]
$ 

```

Si quisieramos que estas reglas perduraran tras reiniciar el sistema, al ser Kali una distribución basada en Debian, se podrían guardar las reglas usando '**iptables-persistent**'. '**iptables-persistent**' es una herramienta que guarda las reglas de iptables y las restaura automáticamente al iniciar el sistema. Este guarda las reglas actuales para asegurarse de que se restauren después de un reinicio.

Usaremos este comando para guardar las reglas actuales de iptables en los archivos /etc/iptables/rules.v4 (para IPv4) y /etc/iptables/rules.v6 (para IPv6).

sudo netfilter-persistent save

Para asegurarnos de que el servicio que restaura las reglas está activo, se puede reiniciar el servicio con netfilter-persistent usando el siguiente comando.

```
sudo systemctl restart netfilter-persistent
```

b) Otra opción que se permite es la de bloquear puertos (tanto origen como destino). Probar a crear una regla que bloquee el tráfico a través de los puertos 80 (HTTP) y 443 (HTTPS). Tened en cuenta que van sobre TCP y que el puerto que hay que bloquear es el de destino. Para comprobar si ha funcionado, se debe intentar acceder a alguna web como “Wikipedia”.

- Para bloquear el tráfico hacia los puertos 80 y 443, añadimos las reglas a iptables para bloquear el tráfico de salida (OUTPUT) hacia los puertos 80 y 443.
- Bloquear el puerto 80 (HTTP)

```
sudo iptables -A OUTPUT -p tcp --dport 80 -j DROP
```

- **-p tcp:** -p significa "protocol" (protocolo). Especifica que la regla se aplica a los paquetes que utilizan el protocolo TCP.
- **--dport 80:** --dport significa "destination port" (puerto de destino). Especifica que la regla se aplica a los paquetes que tienen como puerto de destino el puerto 80 (HTTP).
- Bloquear el puerto 443 (HTTPS)

```
sudo iptables -A OUTPUT -p tcp --dport 443 -j DROP
```

- Verificamos las reglas de iptables:

```
sudo iptables -L -v
```

```

[~] (kali㉿kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --dport 80 -j DROP
[sudo] password for kali:

[~] (kali㉿kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --dport 443 -j DROP

[~] (kali㉿kali)-[~]
└─$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
0      0   ACCEPT    all   --  any    docker0  anywhere        anywhere
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
0      0   DOCKER-USER all   --  any    docker0  anywhere        anywhere
0      0   DOCKER-ISOLATION-STAGE-1 all   --  any    any    docker0 !docker0 anywhere        anywhere
0      0   ACCEPT    all   --  any    docker0  anywhere        anywhere
ESTABLISHED
0      0   DOCKER    all   --  any    docker0  anywhere        anywhere
0      0   ACCEPT    all   --  any    docker0 !docker0 anywhere        anywhere
0      0   ACCEPT    all   --  any    docker0  docker0 anywhere        anywhere
Chain OUTPUT (policy ACCEPT 1 packets, 66 bytes)
pkts bytes target     prot opt in     out      source          destination
10     840  DROP     all   --  any    any    anywhere        dns.google
0      0   DROP     tcp   --  any    any    anywhere        anywhere          tcp dpt:http
0      0   DROP     tcp   --  any    any    anywhere        anywhere          tcp dpt:https

Chain DOCKER (1 references)
pkts bytes target     prot opt in     out      source          destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
pkts bytes target     prot opt in     out      source          destination
0      0   DOCKER-ISOLATION-STAGE-2 all   --  docker0 !docker0 anywhere        anywhere
0      0   RETURN   all   --  any    any    anywhere        anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
pkts bytes target     prot opt in     out      source          destination
0      0   DROP     all   --  any    docker0  anywhere        anywhere
0      0   RETURN   all   --  any    any    anywhere        anywhere

Chain DOCKER-USER (1 references)
pkts bytes target     prot opt in     out      source          destination
0      0   RETURN   all   --  any    any    anywhere        anywhere

```

También se puede comprobar de forma mas sencilla si las reglas se han añadido correctamente:

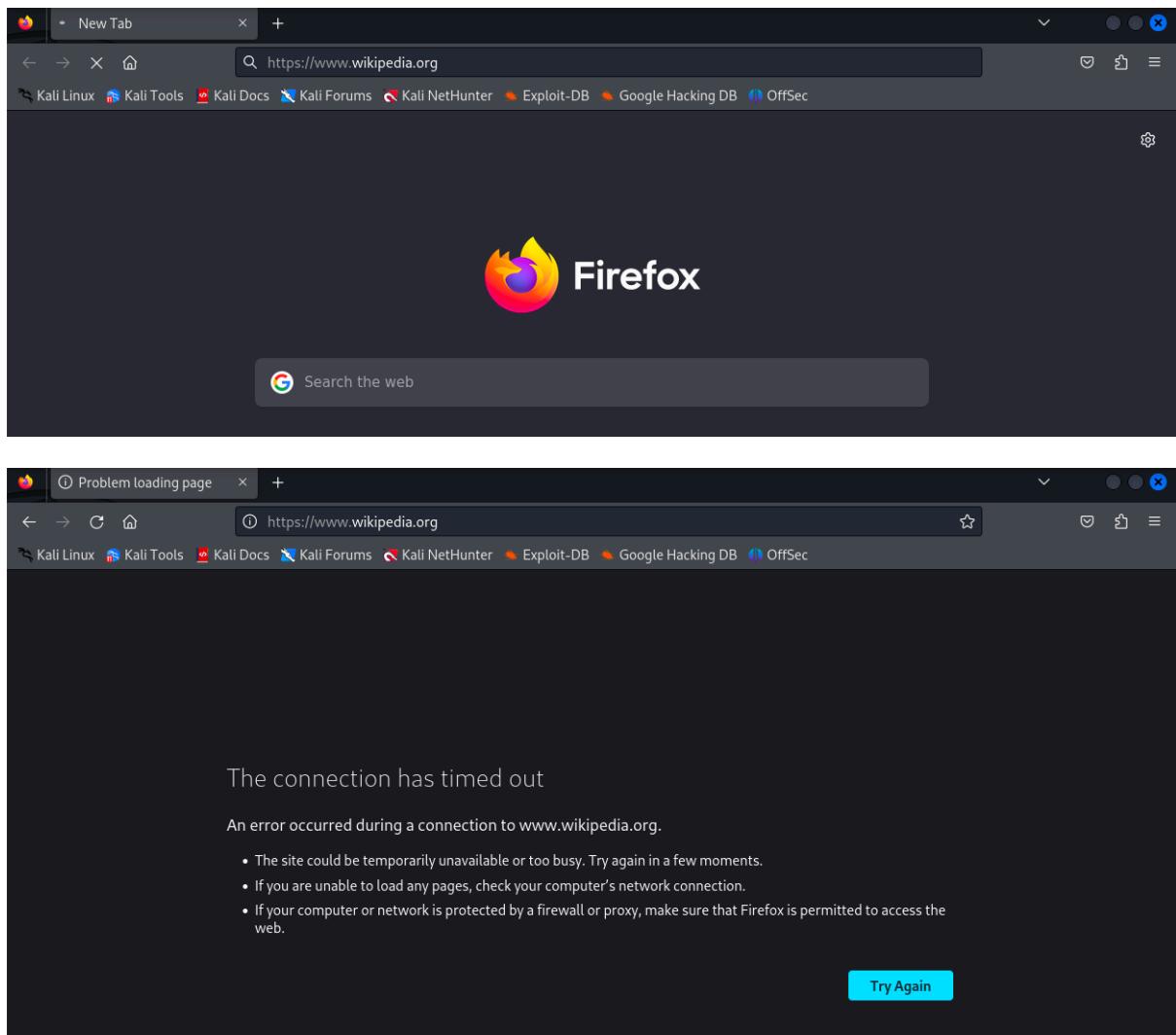
```

[~] (kali㉿kali)-[~]
└─$ sudo iptables -L -v
Chain OUTPUT (policy ACCEPT 736 packets, 92401 bytes)
pkts bytes target     prot opt in     out      source          destination
10     840  DROP     all   --  any    any    anywhere        dns.google
124    7440  DROP    tcp   --  any    any    anywhere        anywhere          tcp dpt:http
692   41520  DROP    tcp   --  any    any    anywhere        anywhere          tcp dpt:https

[~] (kali㉿kali)-[~]
└─$ 

```

Comprobamos en el navegador que efectivamente no puede acceder a la web de la Wikipedia:



c) También es interesante bloquear el tráfico que entra a nuestra máquina desde una dirección fuente. Para poder hacer la prueba, vamos a bloquear el tráfico entrante proveniente de la dirección de origen 127.0.0.1 (localhost). Tras crear la regla, probar a hacer PING a dicha dirección IP.

Para bloquear el tráfico entrante proveniente de la dirección de origen 127.0.0.1 (localhost) usando iptables. Añadimos una regla a iptables para bloquear el tráfico entrante (INPUT) desde la dirección 127.0.0.1.

sudo iptables -A INPUT -s 127.0.0.1 -j DROP

- -s significa "source" (origen). Especifica que la regla se aplica a los paquetes que tienen como dirección de origen 127.0.0.1 (localhost).

Verificamos que se ha realizado correctamente usando ping.

```
(kali㉿kali)-[~]
$ sudo iptables -A INPUT -s 127.0.0.1 -j DROP
[sudo] password for kali:

(kali㉿kali)-[~]
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
^C
--- 127.0.0.1 ping statistics ---
22 packets transmitted, 0 received, 100% packet loss, time 21472ms
```

Si queremos comprobar si se ha agregado la regla correctamente, también podemos usar iptables -L INPUT -v. En la salida del comando, vemos que la dirección de origen aparece como localhost en lugar de 127.0.0.1. Esto es una característica de iptables que convierte direcciones IP conocidas en sus nombres de host correspondientes cuando se visualizan las reglas.

```
(kali㉿kali)-[~]
$ sudo iptables -L INPUT -v
Chain INPUT (policy ACCEPT 6 packets, 1378 bytes)
 pkts bytes target     prot opt in     out     source               destination
      22  1848 DROP       all   --  any    any    localhost           anywhere
```

Para bloquear el tráfico entrante desde 127.0.0.1 (localhost), la regla se ha añadido correctamente y funciona como se esperaba.

Para verificar que la regla está funcionando correctamente, hacemos ping a 127.0.0.1 como se puede ver en la captura anterior y se observa que no se recibe respuestas, lo que indica que los paquetes están siendo descartados.

d) Una opción interesante que permite iptables es crear reglas que acepten/bloqueen tráfico que entra por una interfaz concreta. En este caso se pide bloquear el tráfico de entrada en la interfaz “eth0”. Para comprobar si la regla está bien, se puede probar a ejecutar el siguiente comando “curl <http://example.com>”. Si devuelve el código HTML, la regla no se ha aplicado bien.

En este ejercicio nos pide usar curl sin la flag -I como había usado con anterioridad. La diferencia entre ambas es que curl devuelve el contenido completo de la respuesta (cuerpo y encabezados) y curl -I devuelve solo los encabezados de la respuesta.

- `curl` sin opciones adicionales realiza una solicitud HTTP GET completa al servidor y devuelve el contenido de la respuesta.
- `curl -I` realiza una solicitud HTTP HEAD al servidor y devuelve solo los encabezados de la respuesta.

Para resolver el ejercicio, añadimos una regla a iptables para bloquear todo el tráfico entrante en la interfaz eth0:

sudo iptables -A INPUT -i eth0 -j DROP

- **-i** significa "input interface" (interfaz de entrada). Esta opción indica que la regla se aplica a los paquetes que llegan a través de la interfaz de red eth0. Solo se procesarán los paquetes que ingresen por esta interfaz específica.

Para ver la lista de reglas del firewall después de añadir la nueva regla:

sudo iptables -L INPUT -v

Probamos ejecutar el siguiente comando para verificar si la regla se ha aplicado correctamente usaremos como ejemplo esta url:

curl <https://es.wikipedia.org/>

```
└─(kali㉿kali)-[~]
$ sudo iptables -A INPUT -i eth0 -j DROP

└─(kali㉿kali)-[~]
$ sudo iptables -L INPUT -v

Chain INPUT (policy ACCEPT 30 packets, 6706 bytes)
 pkts bytes target     prot opt in     out      source               destination
   22  1848 DROP       all  --  any    any      localhost           anywhere
   24  4652 DROP       all  --  eth0   any      anywhere           anywhere
```

```
└─(kali㉿kali)-[~]
$ curl https://es.wikipedia.org/
curl: (6) Could not resolve host: es.wikipedia.org
```

e) Por último, es interesante comprobar que sucede si se tienen dos reglas que “se contradicen”. Se puede probar a crear una regla igual que la anterior, pero que acepte el tráfico entrante por la interfaz “eth0”. ¿Qué sucede ahora? ¿Y si primero se crea la que acepta el tráfico y luego la que lo bloquea?

Añadimos una regla que acepte el tráfico entrante en la interfaz eth0:

sudo iptables -A INPUT -i eth0 -j ACCEPT

Para ver las reglas en el orden en que se aplican:

sudo iptables -L INPUT -v --line-numbers

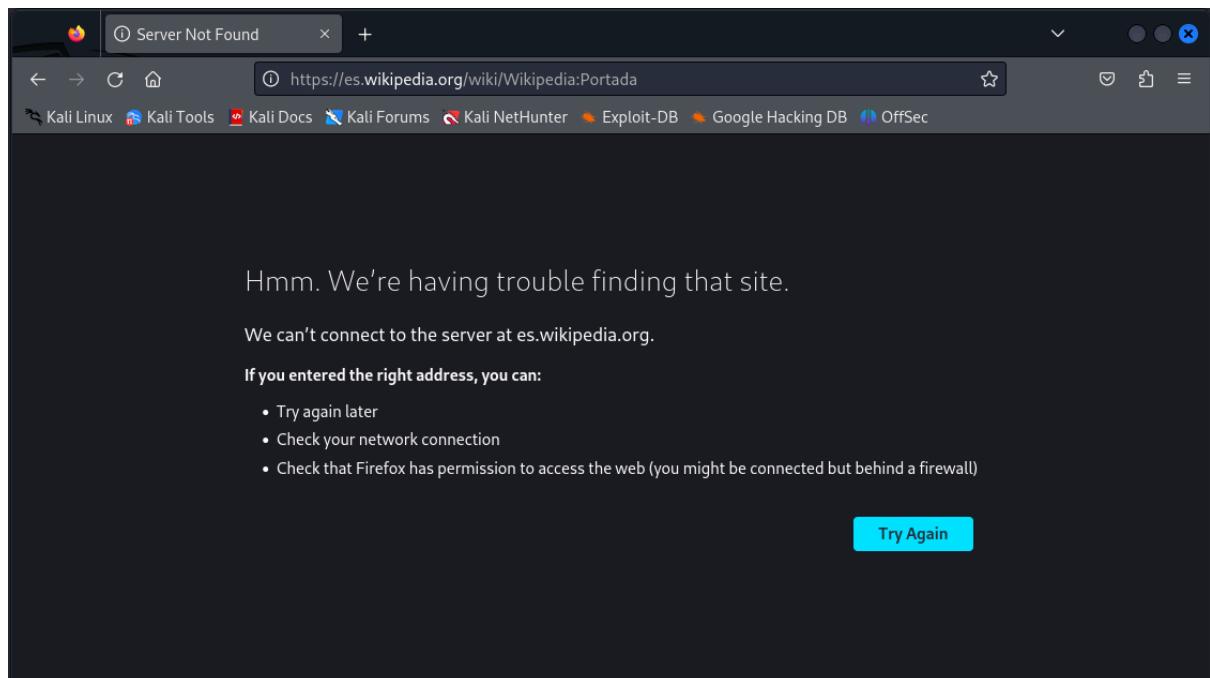
Probamos conectividad: **curl https://es.wikipedia.org/**

```
└─(kali㉿kali)-[~]
$ sudo iptables -A INPUT -i eth0 -j ACCEPT

└─(kali㉿kali)-[~]
$ sudo iptables -L INPUT -v --line-numbers

Chain INPUT (policy ACCEPT 30 packets, 6706 bytes)
num  pkts bytes target     prot opt in     out    source          destination
1      22  1848 DROP       all   --  any    any    localhost      anywhere
2     354 76896 DROP       all   --  eth0   any    anywhere      anywhere
3        0     0 ACCEPT     all   --  eth0   any    anywhere      anywhere
```

```
└─(kali㉿kali)-[~]
$ curl https://es.wikipedia.org/
curl: (6) Could not resolve host: es.wikipedia.org
```



Comprobamos que tiene prioridad la regla DROP por tanto no tenemos conectividad.

Para invertir el orden de las reglas, primero tenemos que eliminar todas las reglas ya que como hemos visto hay otra regla con prioridad 1 que puede bloquear las demás y volvemos a probar.

Si las quisiéramos eliminar de una en una las eliminamos así:

```
sudo iptables -D INPUT -i eth0 -j DROP
```

```
sudo iptables -D INPUT -i eth0 -j ACCEPT
```

Como las quiero eliminar todas, uso el comando: **sudo iptables -F**

```
└─(kali㉿kali)-[~]
└─$ sudo iptables -F

└─(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers

Chain INPUT (policy ACCEPT 30 packets, 6706 bytes)
num  pkts bytes target     prot opt in     out      source          destination

└─(kali㉿kali)-[~]
└─$ 
```

Y las volvemos a añadir del mismo modo que antes pero en orden inverso.

Verificamos de nuevo: **sudo iptables -L INPUT -v --line-numbers**

Probamos la conectividad otra vez: **curl <https://es.wikipedia.org/>**

Como podemos ver en la captura de pantalla, tras eliminar todas las reglas de entrada, se crean las reglas de aceptar y drop en este orden. Comprobamos que el tráfico entrante funciona.

```
└─(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination

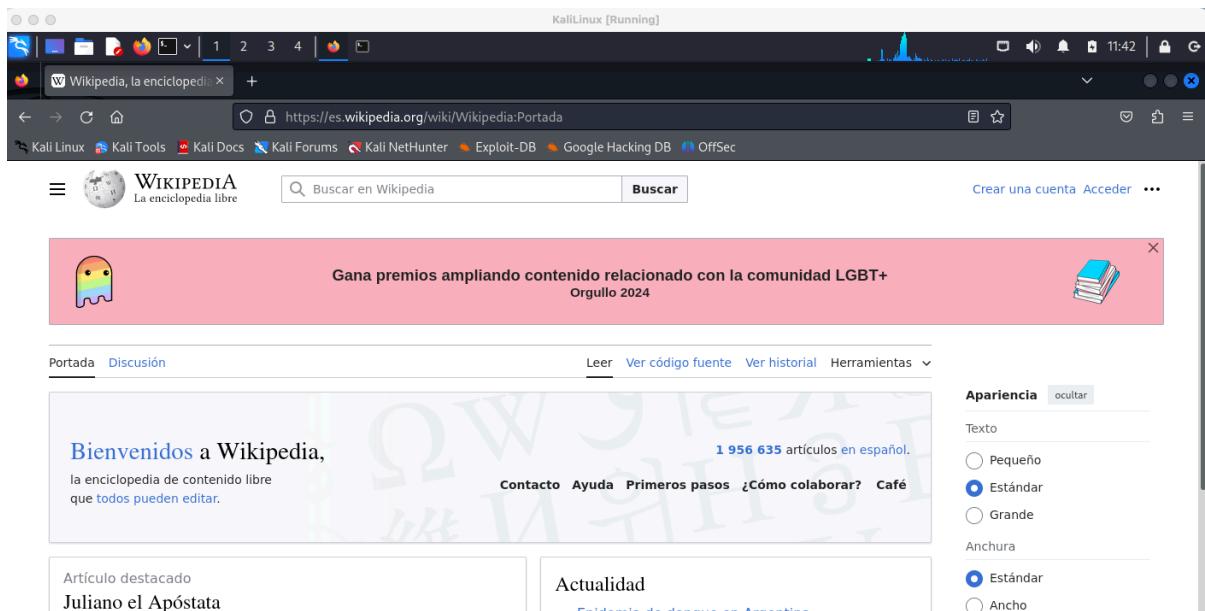
└─(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -i eth0 -j ACCEPT

└─(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      0      0 ACCEPT     all   --  eth0    any    anywhere        anywhere

└─(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -i eth0 -j DROP

└─(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      0      0 ACCEPT     all   --  eth0    any    anywhere        anywhere
2      0      0 DROP       all   --  eth0    any    anywhere        anywhere

└─(kali㉿kali)-[~]
└─$ curl https://es.wikipedia.org/
└─(kali㉿kali)-[~]
```



Por las dudas, debido a que anteriormente teníamos otra regla como prioritaria, vuelvo a realizar la parte anterior siendo el orden de las reglas DROP primero y luego accept.

Ahora si podemos decir que efectivamente el orden de prioridad de las reglas importa, podemos ver que cuando las reglas son contradictorias, la que tiene prioridad es la que actúa primero, de modo que si posteriormente se ha creado otra que intenta hacer lo contrario se verá bloqueada por la anterior. Es por ello muy importante comprobar las reglas de firewall presentes antes de crear una nueva.

```

└──(kali㉿kali)-[~]
└─$ sudo iptables -F
[sudo] password for kali:

└──(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination
      0     0 ACCEPT      all  --  eth0    any      anywhere            anywhere
      1     0     0 DROP       all  --  eth0    any      anywhere            anywhere

└──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -i eth0 -j DROP

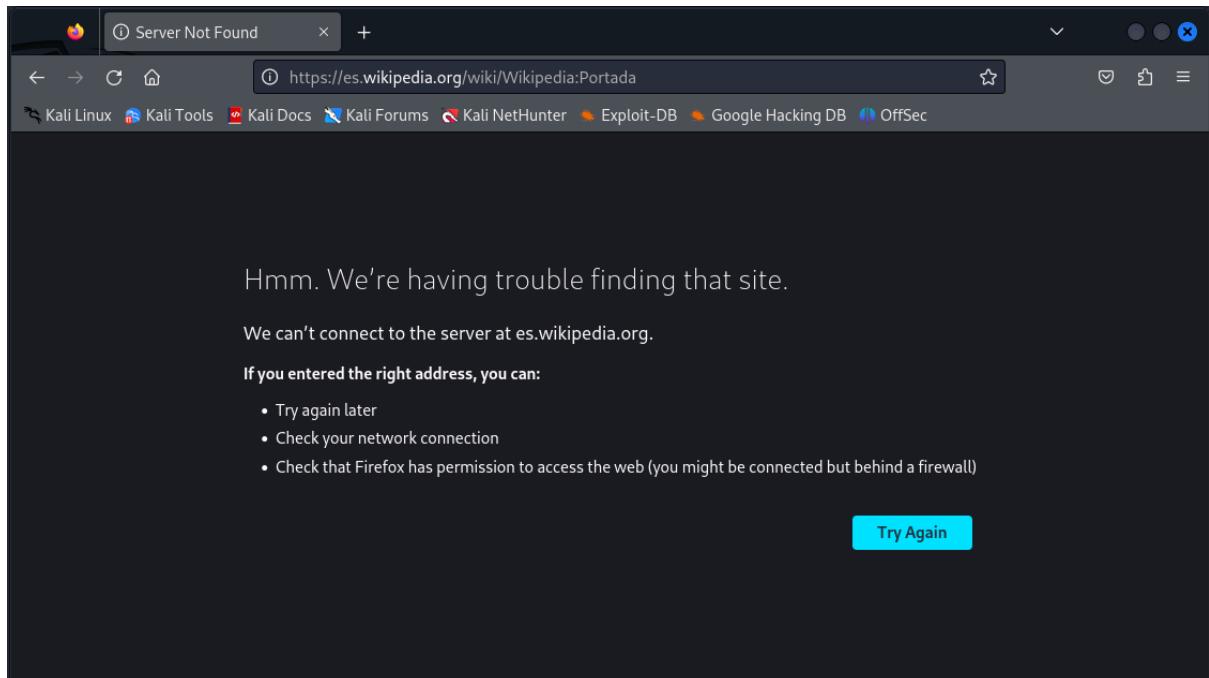
└──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -i eth0 -j ACCEPT

└──(kali㉿kali)-[~]
└─$ sudo iptables -L INPUT -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination
1     0     0 DROP       all  --  eth0    any      anywhere            anywhere
2     0     0 ACCEPT      all  --  eth0    any      anywhere            anywhere

└──(kali㉿kali)-[~]
└─$ curl https://es.wikipedia.org/
curl: (6) Could not resolve host: es.wikipedia.org

└──(kali㉿kali)-[~]
└─$ █

```



NOTA: se recomienda borrar todas las reglas creadas una vez finalizado el ejercicio. Para ello: “`sudo iptables -F`” si se quieren borrar todas. Si se quiere borrar solamente una, utilizar “`-D`” junto con la regla para eliminarla.

Tarea 8:

En este ejercicio se pretende ver de forma práctica todo lo relacionado con los permisos. Para ello, se pide primero añadir un nuevo usuario desde la terminal con el comando “adduser” <nombre de usuario>, llamado “usuario1” y con contraseña “usuario1”. Tras crear este usuario, se pide cambiar los permisos al archivo “file1.txt” creado anteriormente para que solo el usuario propietario (kali en este caso) tenga permisos (para cambiar permisos se usa el comando “chmod”). Una vez cambiados los permisos de dicho archivo, se debe cambiar de usuario con el comando “su usuario1”, comprobar que se ha cambiado correctamente el usuario (indicar el comando con el que se puede comprobar esto) e intentar mostrar el contenido del archivo al que se le han cambiado los permisos. RECOMENDACIÓN: realizar diferentes tipos de pruebas aparte de las que se exigen en el ejercicio (dar solo permisos de lectura o escritura, etc.).

Vamos a realizar el ejercicio paso a paso.

1. Creamos el usuario1 con contraseña usuario1 como se indica en el enunciado.

The screenshot shows a terminal window titled "kali@kali: ~". The command \$ sudo adduser usuario1 is run, followed by a password prompt for the root user. The user information is then entered, including full name, room number, work phone, home phone, and other details. A confirmation question is asked, and the user responds with 'y'. The terminal then shows that the user has been added to the 'users' group. The session ends with the prompt \$.

```
(kali㉿kali)-[~]
$ sudo adduser usuario1
[sudo] password for kali:
info: Adding user `usuario1' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `usuario1' (1001) ...
info: Adding new user `usuario1' (1001) with group `usuario1 (1001)' ...
info: Creating home directory `/home/usuario1' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for usuario1
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
info: Adding new user `usuario1' to supplemental / extra groups `users' ...
info: Adding user `usuario1' to group `users' ...
(kali㉿kali)-[~]
```

2. Cambiamos los permisos del archivo file1.txt de manera que solo el usuario kali tenga permisos para dicho archivo con ‘chmod’ y comprobamos con ‘ls -l’.

```
[kali㉿kali)-[~]
$ cd Documents/dir1

[kali㉿kali)-[~/Documents/dir1]
$ ls
file1.txt

[kali㉿kali)-[~/Documents/dir1]
$ chmod 600 file1.txt

[kali㉿kali)-[~/Documents/dir1]
$ ls -l file1.txt
-rw——— 1 kali kali 13 May 24 12:27 file1.txt

[kali㉿kali)-[~/Documents/dir1]
$
```

El comando `chmod` en Unix/Linux se utiliza para cambiar los permisos de archivos y directorios. Los permisos determinan quién puede leer, escribir o ejecutar un archivo o directorio. Los permisos se pueden especificar de dos maneras: usando una notación simbólica o usando una notación numérica (octal). A continuación, te explico ambas.

Para entender por qué se ha usado la opción **600** en este caso explicamos como funciona el comando `chmod`. Hay dos modos de notación, la simbólica y la numérica.

- Notación Simbólica: La notación simbólica usa letras para representar los permisos y operadores para modificarlos:
 - Letras para permisos:
 - `r`: Permiso de lectura (read)
 - `w`: Permiso de escritura (write)
 - `x`: Permiso de ejecución (execute)
 - Letras para usuarios:
 - `u`: Usuario (propietario del archivo)
 - `g`: Grupo (grupo al que pertenece el archivo)
 - `o`: Otros (todos los demás usuarios)
 - `a`: Todos (usuario, grupo y otros)
 - Operadores:
 - `+`: Añadir permiso
 - `-`: Quitar permiso
 - `=:` Establecer permiso exactamente

Ejemplos:

1. Añadir permiso de lectura para otros: chmod o+r file.txt
2. Quitar permiso de escritura para el grupo: chmod g-w file.txt
3. Establecer permisos de lectura y ejecución para todos: chmod a=rx file.txt

- Notación Numérica (Octal): En la notación numérica, los permisos se representan con números. Cada permiso se representa por un número y los permisos para usuario, grupo y otros se combinan en un número de tres dígitos.

- Valores numéricos:

- 'r' (read) = 4
- 'w' (write) = 2
- 'x' (execute) = 1

Se suman los valores para obtener los permisos deseados. Por ejemplo:

- '7' = 'rwx' ($4 + 2 + 1$)
- '6' = 'rw-' ($4 + 2$)
- '5' = 'r-x' ($4 + 1$)
- '4' = 'r--'
- '3' = '-wx' ($2 + 1$)
- '2' = '-w-'
- '1' = '--x'
- '0' = '---'

Ejemplos:

1. Permisos 'rwxr-xr--': chmod 754 file.txt

Esto significa:

- Usuario: 'rwx' (7)
- Grupo: 'r-x' (5)
- Otros: 'r--' (4)

2. Permisos 'rw-rw-r--': chmod 664 file.txt

Esto significa:

- Usuario: 'rw-' (6)

- Grupo: `rw-` (6)
- Otros: `r--` (4)

Resumiendo:

- Simbólica:
 - `chmod u+x file.txt` (añadir permiso de ejecución al usuario)
 - `chmod g-w file.txt` (quitar permiso de escritura al grupo)
 - `chmod o=r file.txt` (establecer permiso de solo lectura para otros)
- Numérica:
 - `chmod 755 file.txt` (rwxr-xr-x)
 - `chmod 644 file.txt` (rw-r--r--)
 - `chmod 600 file.txt` (rw-----)

Estas son las formas de utilizar `chmod` para cambiar los permisos de archivos y directorios en Unix/Linux, tanto con notación simbólica como numérica.

En esta parte del ejercicio, se ha usado chmod 600 para darle todos los permisos al usuario actual para este archivo eliminando los permisos de los demás. El comando chmod 600 establece los permisos de un archivo de la siguiente manera:

Desglose de Permisos

- **Usuario (propietario):**
 - **Lectura (r):** Permitido
 - **Escritura (w):** Permitido
 - **Ejecución (x):** No permitido
- **Grupo:**
 - **Lectura (r):** No permitido
 - **Escritura (w):** No permitido
 - **Ejecución (x):** No permitido
- **Otros:**
 - **Lectura (r):** No permitido
 - **Escritura (w):** No permitido
 - **Ejecución (x):** No permitido

Resumen de chmod 600

- **6:** Permisos para el usuario propietario (rw-):
 - Lectura (r) = 4
 - Escritura (w) = 2
 - Sin ejecución (-) = 0
 - **Total:** 6 (4 + 2)

- **0:** Permisos para el grupo (---):
 - Sin lectura (-) = 0
 - Sin escritura (-) = 0
 - Sin ejecución (-) = 0
 - **Total:** 0
- **0:** Permisos para otros (---):
 - Sin lectura (-) = 0
 - Sin escritura (-) = 0
 - Sin ejecución (-) = 0
 - **Total:** 0

3. Cambiamos el usuario a usuario1 y lo comprobamos con el comando `whoami` que es el comando en Unix/Linux que se utiliza para mostrar el nombre del usuario actual que ha iniciado sesión en el sistema.

```
(kali㉿kali)-[~/Documents/dir1]
$ su usuario1
Password:
((usuario1㉿kali))-[/home/kali/Documents/dir1]
$ whoami
usuario1

((usuario1㉿kali))-[/home/kali/Documents/dir1]
$
```

4. Intentamos mostrar el contenido del archivo file1.txt con el comando `cat` y comprobamos que no tenemos permisos para este archivo desde el usuario1.

```
(usuario1㉿kali)-[/home/kali/Documents/dir1]
$ cat file1.txt
cat: file1.txt: Permission denied

((usuario1㉿kali))-[/home/kali/Documents/dir1]
$
```

5. Pruebas de lo explicado anteriormente.

En los ejemplos podemos comprobar que `chmod 400` le da permisos de solo escritura al usuario. Comprobamos con `ls -l` los permisos del archivo y cambiando de usuario. Probamos también con `chmod 200` que solo da permisos de escritura al usuario y con `chmod 660` que da permisos de lectura y escritura al usuario y grupo, pero no a otros. Hacemos pruebas de escritura con echo para ver que funciona.

```

└─(kali㉿kali)-[~/Documents/dir1]
  $ chmod 400 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ ls -l file1.txt
  -r----- 1 kali kali 13 May 24 12:27 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ cat file1.txt
  "Hola Mundo"

└─(kali㉿kali)-[~/Documents/dir1]
  $ chmod 200 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ ls -l
  total 4
  --w---- 1 kali kali 13 May 24 12:27 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ cat file1.txt
  cat: file1.txt: Permission denied

└─(kali㉿kali)-[~/Documents/dir1]
  $ su usuario1
  Password:
  └─(usuario1㉿kali)-[/home/kali/Documents/dir1]
    $ cat file1.txt
    cat: file1.txt: Permission denied

└─(usuario1㉿kali)-[/home/kali/Documents/dir1]
  $ ls -l file1.txt
  --w---- 1 kali kali 13 May 24 12:27 file1.txt

└─(usuario1㉿kali)-[/home/kali/Documents/dir1]
  $ su kali
  Password:
  └─(kali㉿kali)-[~/Documents/dir1]
    $ chmod 660 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ ls -l file1.txt
  -rw-rw--- 1 kali kali 13 May 24 12:27 file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ su usuario1
  Password:
  └─(usuario1㉿kali)-[/home/kali/Documents/dir1]
    $ echo "Texto de prueba" > file1.txt
    bash: file1.txt: Permission denied

└─(usuario1㉿kali)-[/home/kali/Documents/dir1]
  $ su kali
  Password:
  └─(kali㉿kali)-[~/Documents/dir1]
    $ echo "Texto de prueba" > file1.txt

└─(kali㉿kali)-[~/Documents/dir1]
  $ cat file1.txt
  Texto de prueba

```

Tarea 9:

Investiga acerca del comando netstat en terminal de Linux y las posibilidades que ofrece (puedes usar el comando netstat -h para ver la ayuda). Tras esto, usa el comando adecuado para listar las conexiones de red en estado de escucha (listening), lo cual incluye puertos abiertos en el equipo. Probar y comentar para qué sirven los flags -s, -t, -u y -r.

El comando `netstat` (network statistics) en Linux es una herramienta de línea de comandos que se utiliza para monitorear y analizar el estado de las conexiones de red, las interfaces de red y las rutas de la tabla de enrutamiento del sistema. A continuación, se describen algunas de las opciones más comunes y útiles que ofrece netstat.

```
(kali㉿kali)-[~/Documents/dir1]
$ netstat -h
usage: netstat [-vWnNcCF] [<Af>] -r          netstat {-V|--version|-h|--help}
               netstat [-vWnNcaeol] [<Socket> ...]
               netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw] }

-r, --route           display routing table
-i, --interfaces     display interface table
-g, --groups          display multicast group memberships
-s, --statistics      display networking statistics (like SNMP)
-M, --masquerade     display masqueraded connections

-v, --verbose         be verbose
-W, --wide             don't truncate IP addresses
-n, --numeric          don't resolve names
--numeric-hosts       don't resolve host names
--numeric-ports        don't resolve port names
--numeric-users        don't resolve user names
-N, --symbolic        resolve hardware names
-e, --extend            display other/more information
-p, --programs         display PID/Program name for sockets
-o, --timers           display timers
-c, --continuous       continuous listing

-l, --listening        display listening server sockets
-a, --all              display all sockets (default: connected)
-F, --fib              display Forwarding Information Base (default)
-C, --cache            display routing cache instead of FIB
-Z, --context          display SELinux security context for sockets

<Socket> { -t|tcp } { -u|udp } { -U|udplite } { -S|sctp } { -w|raw }
           { -x|unix } --ax25 --ipx --netrom
<AF> Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
List of possible address families (which support routing):
  inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
  netrom (AMPR NET/ROM) rose (AMPR ROSE) ipx (Novell IPX)
  ddp (Appletalk DDP) x25 (CCITT X.25)
```

1. Listar las conexiones de red en estado de escucha (listening): `netstat -tuln`

Las flags utilizadas:

-t: Muestra las conexiones TCP.

-u: Muestra las conexiones UDP.

-l: Muestra solo los sockets en estado de escucha (listening).

-n: Muestra direcciones y números de puerto en formato numérico.

- **TCP (Transmission Control Protocol):** Es un protocolo de comunicación que establece una conexión fiable y orientada a la conexión entre dos dispositivos, garantizando la entrega correcta y en orden de los datos enviados.
- **UDP (User Datagram Protocol):** Es un protocolo de comunicación que permite el envío de mensajes (datagramas) sin establecer una conexión previa, sin garantía de entrega, orden o integridad, lo que lo hace más rápido pero menos fiable que TCP.

En este caso solo tenemos un puerto en modo escucha de conexión TCP.

```
(kali㉿kali)-[~/Documents/dir1]
$ netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 127.0.0.1:22              0.0.0.0:*              LISTEN
```

Probamos con las flags -s, -t, -u, -r.

1. `netstat -s`: muestra estadísticas de las diferentes familias de protocolos. Este comando muestra estadísticas detalladas de los protocolos de red, incluyendo TCP, UDP, ICMP y más. Es útil para diagnosticar problemas de red y entender el tráfico general de la red.
- **ICMP (Internet Control Message Protocol)**: Es un protocolo utilizado para enviar mensajes de control y diagnóstico sobre el estado de la red, como errores y respuestas a solicitudes de ping, ayudando a gestionar y solucionar problemas de red.

```
(kali㉿kali)-[~/Documents/dir1]
└─$ netstat -s
Ip:
  Forwarding: 1
  1663 total packets received
  1 with invalid addresses
  0 forwarded
  0 incoming packets discarded
  1388 incoming packets delivered
  1283 requests sent out
Icmp:
  0 ICMP messages received
  0 input ICMP message failed
  ICMP input histogram:
  0 ICMP messages sent
  0 ICMP messages failed
  ICMP output histogram:
Tcp:
  39 active connection openings
  0 passive connection openings
  2 failed connection attempts
  1 connection resets received
  1 connections established
  836 segments received
  813 segments sent out
  0 segments retransmitted
  0 bad segments received
  14 resets sent
Udp:
  554 packets received
  0 packets to unknown port received
  0 packet receive errors
  464 packets sent
  0 receive buffer errors
  0 send buffer errors
UdpLite:
TcpExt:
  27 TCP sockets finished time wait in fast timer
  22 delayed acks sent
  11 packet headers predicted
  91 acknowledgments not containing data payload received
  261 predicted acknowledgments
  TCPBacklogCoalesce: 6
  1 connections aborted due to timeout
  TCPRecvCoalesce: 57
  TCPAutoCorking: 3
  TCPOrigDataSent: 314
  TCPKeepAlive: 48
  TCPDelivered: 348
IpExt:
  OutMcastPkts: 10
  InOctets: 4742630
  OutOctets: 145209
  OutMcastOctets: 1590
  InNoECTPkts: 4233
MPTcpExt:
└─(kali㉿kali)-[~/Documents/dir1]
```

2. `netstat -t`: muestra únicamente las conexiones TCP. Este comando lista solo las conexiones TCP activas. Es útil cuando deseas enfocarte únicamente en el tráfico TCP.

```
(kali㉿kali)-[~/Documents/dir1]
└─$ netstat -t
Active Internet connect[redacted] servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 10.0.2.15:38546        93.243.107.34.bc.:https ESTABLISHED
└─(kali㉿kali)-[~/Documents/dir1]
```

3. `netstat -u`: muestra únicamente las conexiones UDP. Este comando lista solo las conexiones UDP activas. Es útil para monitorear servicios que usan UDP, como DNS o DHCP.

- **DNS (Domain Name System)**: Es un sistema que traduce nombres de dominio legibles por humanos (como www.example.com) en direcciones IP numéricas que las computadoras utilizan para identificar y comunicarse con servidores en Internet.
- **DHCP (Dynamic Host Configuration Protocol)**: Es un protocolo de red que asigna automáticamente direcciones IP y otros parámetros de configuración de red a los dispositivos en una red para que puedan comunicarse eficientemente.

```
(kali㉿kali)-[~/Documents/dir1]
$ netstat -u
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
udp      0      0 10.0.2.15:bootpc          10.0.2.2:bootps       ESTABLISHED

(kali㉿kali)-[~/Documents/dir1]
```

4. `netstat -r`: muestra la tabla de enrutamiento del kernel. Este comando muestra las rutas de red que el sistema está utilizando. Es útil para entender cómo se enrutan los paquetes a través de la red y para diagnosticar problemas de enrutamiento.

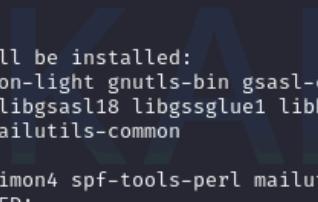
```
(kali㉿kali)-[~/Documents/dir1]
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
default         10. [REDACTED]    0.0.0.0        UG        0 0          0 eth0
10. [REDACTED]   0.0.0.0        255.255.255.0  U         0 0          0 eth0
172. [REDACTED]  0.0.0.0        255.255.0.0   U         0 0          0 docker0

(kali㉿kali)-[~/Documents/dir1]
```

Tarea 10:

Instala la herramienta chkrootkit (`sudo apt install chkrootkit`) y ejecútala como superusuario. ¿Qué es lo que hace esta herramienta?

La herramienta **chkrootkit** es una herramienta de seguridad que se utiliza para comprobar si un sistema Unix/Linux ha sido comprometido mediante un rootkit. Un rootkit es un conjunto de software malicioso que permite a un atacante mantener el acceso a un sistema comprometido sin ser detectado. chkrootkit escanea el sistema en busca de firmas conocidas de rootkits y otras actividades sospechosas.

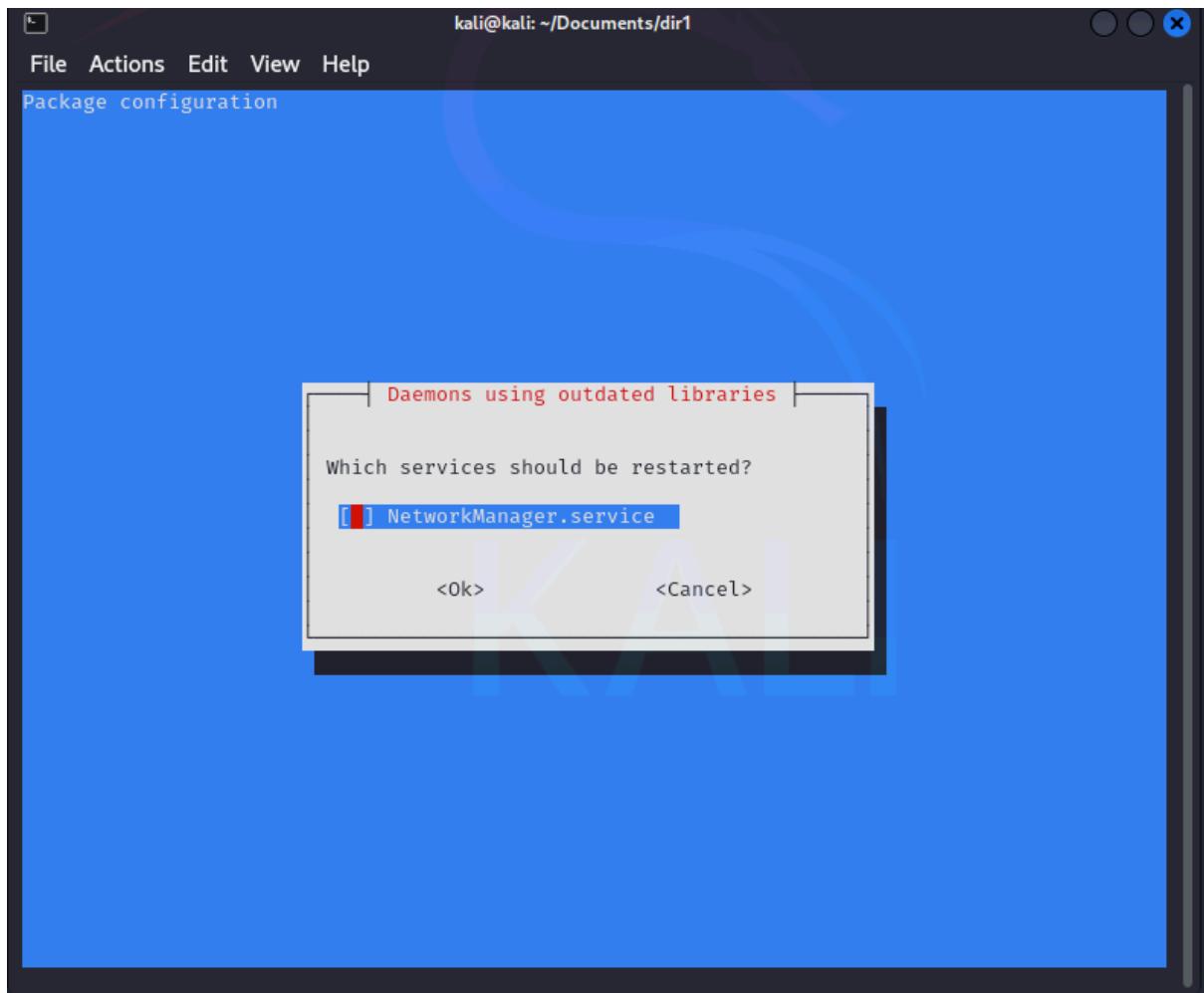


```
kali@kali: ~/Documents/dir1
File Actions Edit View Help
└─(kali㉿kali)-[~/Documents/dir1]
└─$ sudo apt update
[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [47.0 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [114 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [257 kB]
Fetched 67.3 MB in 10s (6,472 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1561 packages can be upgraded. Run 'apt list --upgradable' to see them.

└─(kali㉿kali)-[~/Documents/dir1]
└─$ sudo apt install chkrootkit

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  exim4-base exim4-config exim4-daemon-light gnutls-bin gsasl-common guile-3.0-libs
  libgnutls-dane0t64 libgnutls30t64 libgsasl18 libgssglue1 libhogweed6t64 libmailutils9t64
  libnettle8t64 libntlm0 mailutils mailutils-common
Suggested packages:
  exim4-doc-html | exim4-doc-info eximon4 SPF-tools-perl mailutils-mh mailutils-doc
The following packages will be REMOVED:
  libgnutls-dane0 libgnutls30 libhogweed6 libnettle8
The following NEW packages will be installed:
  chkrootkit exim4-base exim4-config exim4-daemon-light gsasl-common guile-3.0-libs
  libgnutls-dane0t64 libgnutls30t64 libgsasl18 libgssglue1 libhogweed6t64 libmailutils9t64
  libnettle8t64 libntlm0 mailutils mailutils-common
The following packages will be upgraded:
  gnutls-bin
1 upgraded, 16 newly installed, 4 to remove and 1560 not upgraded.
Need to get 14.8 MB of archives.
```

Al instalarlo, emerge una ventana donde procedemos a confirmar.



Seleccionamos 'Enter' y se obtiene esta salida.



```
kali@kali: ~/Documents/dir1
File Actions Edit View Help
update-alternatives: using /usr/bin/dotlock.mailutils to provide /usr/bin/dotlock (dotlock) in
auto mode
update-alternatives: using /usr/bin/mail.mailutils to provide /usr/bin/mailx (mailx) in auto mode
Setting up exim4-base (4.97-8) ...
exim: DB upgrade, deleting hints-db
update-rc.d: As per Kali policy, exim4 init script is left disabled.
Created symlink /etc/systemd/system/timers.target.wants/exim4-base.timer → /usr/lib/systemd/system/exim4-base.timer.
exim4-base.service is a disabled or a static unit, not starting it.
Setting up exim4-daemon-light (4.97-8) ...
exim4.service is a disabled or a static unit, not starting it.
Processing triggers for doc-base (0.11.2) ...
Processing 3 added doc-base files ...
Processing triggers for libc-bin (2.37-12) ...
Processing triggers for man-db (2.12.0-3) ...
Processing triggers for kali-menu (2023.4.7) ...
Scanning processes ...
Scanning candidates ...
Scanning linux images ...

Running kernel seems to be up-to-date.

Restarting services ...
Service restarts being deferred:
  systemctl restart NetworkManager.service

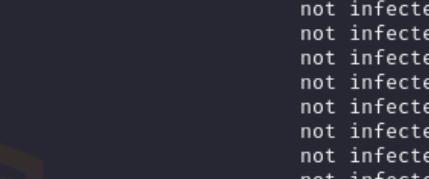
No containers need to be restarted.

User sessions running outdated binaries:
  kali @ session #2: xfce4-session[1085]

No VM guests are running outdated hypervisor (qemu) binaries on this host.

└─(kali㉿kali)-[~/Documents/dir1]
└─$
```

Ejecutamos la herramienta con sudo:



```
└─(kali㉿kali)-[~/Documents/dir1]
└─$ sudo chkrootkit

ROOTDIR is '/'
Checking `amd' ...                                not found
Checking `basename' ...                            not infected
Checking `biff' ...                               not found
Checking `chfn' ...                               not infected
Checking `chsh' ...                               not infected
Checking `cron' ...                               not infected
Checking `crontab' ...                            not infected
Checking `date' ...                               not infected
Checking `du' ...                                 not infected
Checking `dirname' ...                            not infected
Checking `echo' ...                               not infected
Checking `egrep' ...                             not infected
Checking `env' ...                                not infected
```

Obtenemos los siguientes resultados al final del examen:

```

Checking `crim' ...
Checking `rexedcs' ...
Checking `sniffer' ...

WARNING: Output from ifpromisc:
lo: not promisc and no packet sniffer sockets
eth0: PACKET SNIFFER(/usr/sbin/NetworkManager[548])
docker0: not promisc and no packet sniffer sockets

Checking `w55808' ...                               not found
Checking `wted' ...                                not found
Checking `scalper' ...                             not found
Checking `slapper' ...                            not found
Checking `z2' ...                                 not found
Checking `chkutmp' ...                           not found
Checking `OSX_RSPLUG' ...                         not tested

└─(kali㉿kali)-[~/Documents/dir1]

```

En la salida de `chkrootkit` se obtiene información sobre el estado de las interfaces de red y la posible presencia de "sniffers" de paquetes. Vamos a desglosar lo que significa cada parte de la salida:

- `lo: not promisc and no packet sniffer sockets`:

`lo`: Esta es la interfaz de bucle invertido (loopback).

`not promisc`: La interfaz no está en modo promiscuo. El modo promiscuo permite que la interfaz de red capture todos los paquetes que pasan por ella, no solo los destinados a su dirección MAC.

`no packet sniffer sockets`: No se han encontrado sockets de sniffer de paquetes activos en esta interfaz.

- `eth0: PACKET SNIFFER(/usr/sbin/NetworkManager[548])`:

`eth0`: Esta es la interfaz Ethernet.

`PACKET SNIFFER(/usr/sbin/NetworkManager[548])`: Se ha detectado un sniffer de paquetes en esta interfaz, que está siendo ejecutado por `NetworkManager`. NetworkManager es una herramienta legítima para la gestión de conexiones de red en muchos sistemas Linux, por lo que esta entrada generalmente no es motivo de preocupación a menos que tengamos razones para sospechar una actividad maliciosa.

- `docker0: not promisc and no packet sniffer sockets`:

`docker0`: Esta es la interfaz de red utilizada por Docker para la comunicación entre contenedores.

`not promisc`: La interfaz no está en modo promiscuo.

`no packet sniffer sockets`: No se han encontrado sockets de sniffer de paquetes activos en esta interfaz.

Decido investigar más acerca de 'NetworkManager'.

- Verificar 'NetworkManager' con el comando: **ps aux | grep NetworkManager**

El comando `ps aux | grep NetworkManager` se utiliza para buscar procesos relacionados con "NetworkManager" en un sistema basado en Unix/Linux. Busca y muestra todos los procesos relacionados con "NetworkManager" que están actualmente en ejecución en tu sistema. Esto es útil para verificar si un servicio o proceso específico está activo.

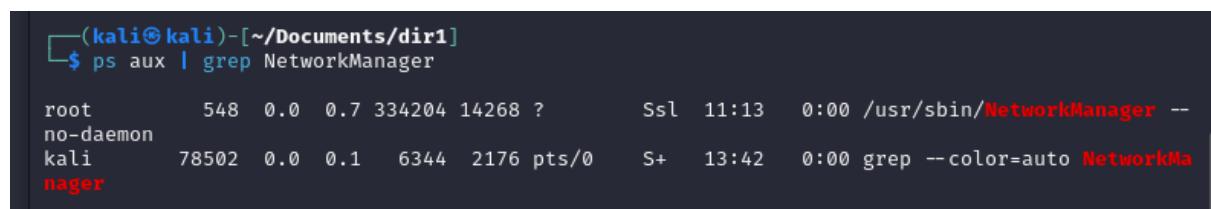
1. **'ps aux'**: Este comando muestra una lista de los procesos que se están ejecutando en el sistema.

- 'ps': Es el comando que muestra información sobre los procesos.
- 'a': Muestra todos los procesos que pertenecen a todos los usuarios.
- 'u': Muestra la información de los procesos en un formato más detallado y legible, incluyendo el usuario que está ejecutando el proceso, el uso de CPU, el uso de memoria, el tiempo de ejecución y el comando que inició el proceso.
- 'x': Muestra los procesos que no están conectados a una terminal de control.

2. **'|'**: Es un operador de tubería que toma la salida del comando anterior ('ps aux') y la pasa como entrada al siguiente comando ('grep NetworkManager').

3. **'grep NetworkManager'**: Este comando filtra la salida de 'ps aux' y muestra solo las líneas que contienen la cadena "NetworkManager".

- 'grep': Es un comando de búsqueda que busca texto en la entrada.
- 'NetworkManager': Es la cadena de texto que estás buscando en la lista de procesos.



```
(kali㉿kali)-[~/Documents/dir1]
$ ps aux | grep NetworkManager
root      548  0.0  0.7 334204 14268 ?        Ssl  11:13   0:00 /usr/sbin/NetworkManager --
no-daemon
kali     78502  0.0  0.1   6344   2176 pts/0    S+   13:42   0:00 grep --color=auto NetworkMa
nager
```

La salida del comando 'ps aux | grep NetworkManager' muestra que el proceso 'NetworkManager' está corriendo como se esperaba y que no parece haber actividad maliciosa asociada a él.

- El proceso 'NetworkManager' está corriendo con PID 548 bajo el usuario 'root'. Esto es normal y esperado en un sistema que usa 'NetworkManager' para la gestión de redes.

- No hay señales de actividad maliciosa o inusual en relación con 'NetworkManager'.

Dado que 'NetworkManager' es una herramienta legítima para gestionar las conexiones de red y su ejecución como proceso del sistema ('root') es normal, no hay razones para preocuparse por la salida de 'chkrootkit' que menciona un "packet sniffer" asociado a 'NetworkManager'.

Sin embargo, si quisiéramos revisar los logs de 'NetworkManager', se puede usar:

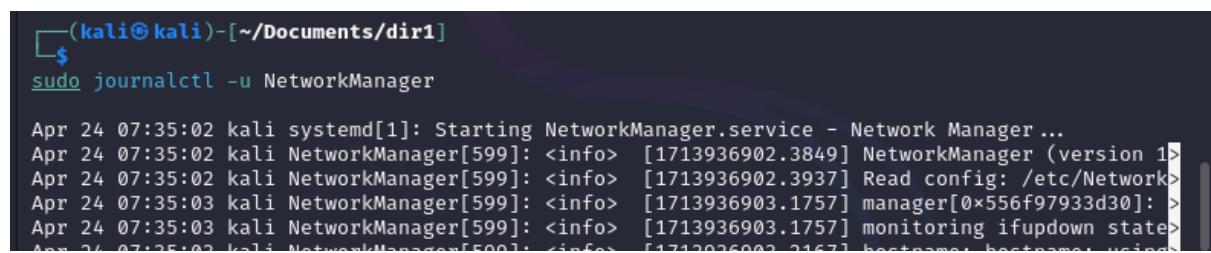
sudo journalctl -u NetworkManager

El comando se utiliza para visualizar los registros de log (logs) del sistema relacionados con el servicio "NetworkManager".

- **journalctl**: Es una herramienta de línea de comandos que se utiliza para consultar y mostrar los logs almacenados por systemd-journald. Esta herramienta permite a los administradores del sistema acceder y analizar los registros del sistema y de los servicios.
- **-u NetworkManager**: Especifica que solo se muestren los registros relacionados con el servicio "NetworkManager".
- **-u o --unit**: Esta opción de journalctl filtra los logs por unidad de servicio. En este caso, NetworkManager es el nombre de la unidad de servicio que queremos consultar.

Este comando mostrará los registros de 'NetworkManager', proporcionando más información sobre su actividad y cualquier evento relevante. Útil para:

- Diagnosticar problemas de red.
- Verificar el estado y eventos del servicio "NetworkManager".
- Identificar errores o advertencias relacionadas con la gestión de redes.



```
(kali㉿kali)-[~/Documents/dir1]
$ sudo journalctl -u NetworkManager

Apr 24 07:35:02 kali systemd[1]: Starting NetworkManager.service - Network Manager ...
Apr 24 07:35:02 kali NetworkManager[599]: <info>  [1713936902.3849] NetworkManager (version 1>
Apr 24 07:35:02 kali NetworkManager[599]: <info>  [1713936902.3937] Read config: /etc/Network>
Apr 24 07:35:03 kali NetworkManager[599]: <info>  [1713936903.1757] manager[0x556f97933d30]: >
Apr 24 07:35:03 kali NetworkManager[599]: <info>  [1713936903.1757] monitoring ifupdown state>
Apr 24 07:35:03 kali NetworkManager[599]: <info>  [1713936903.2167] bestnameset bestnameset usin>
```

Todo normal.

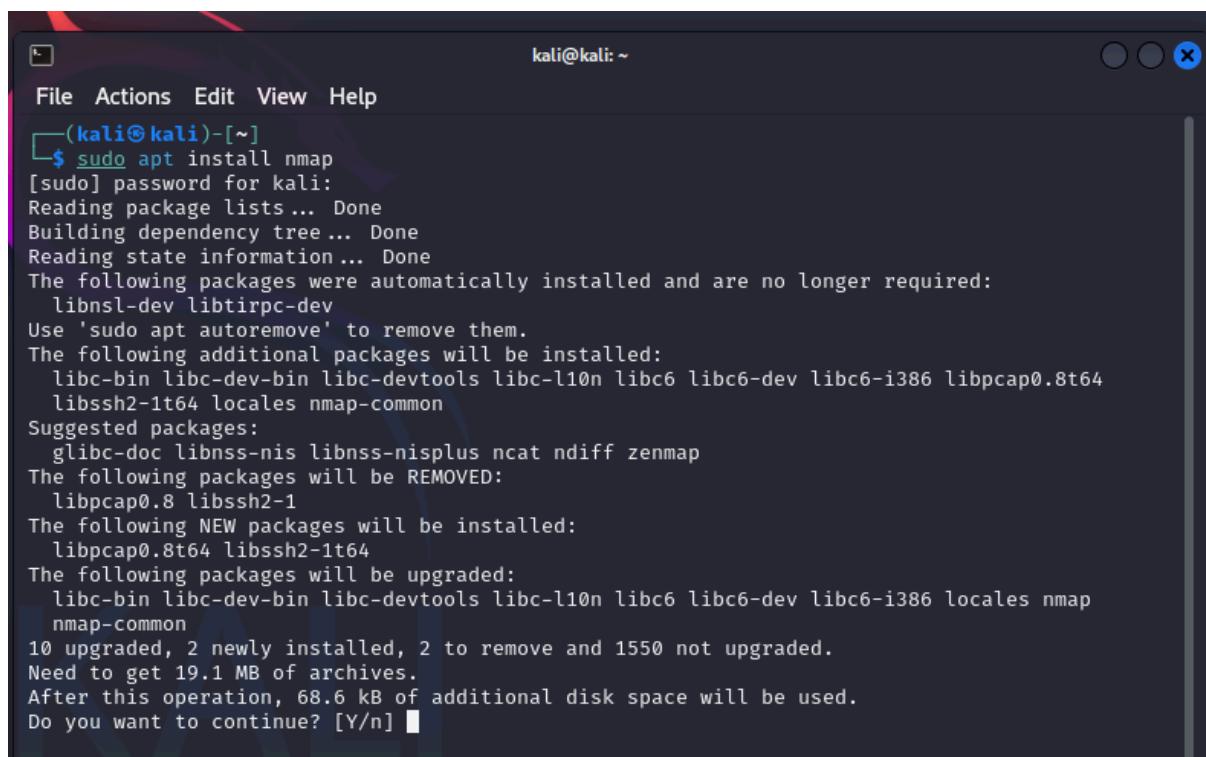
Tarea 11:

La herramienta nmap suele ser muy común entre los atacantes. Ejecuta el comando nmap seguido de tu dirección IP o una a la que tengas acceso en la red y aporta una captura de pantalla sobre el resultado obtenido. ¿Cómo se podrían ver todos los equipos conectados a la misma red en la que se encuentra tu equipo? Investiga sobre las posibilidades que ofrece esta herramienta. Si no dispones de la herramienta recuerda usar el comando ‘sudo apt install nmap’.

En este ejercicio lo primero que vamos a hacer es instalar la herramienta con el comando:

sudo apt install nmap

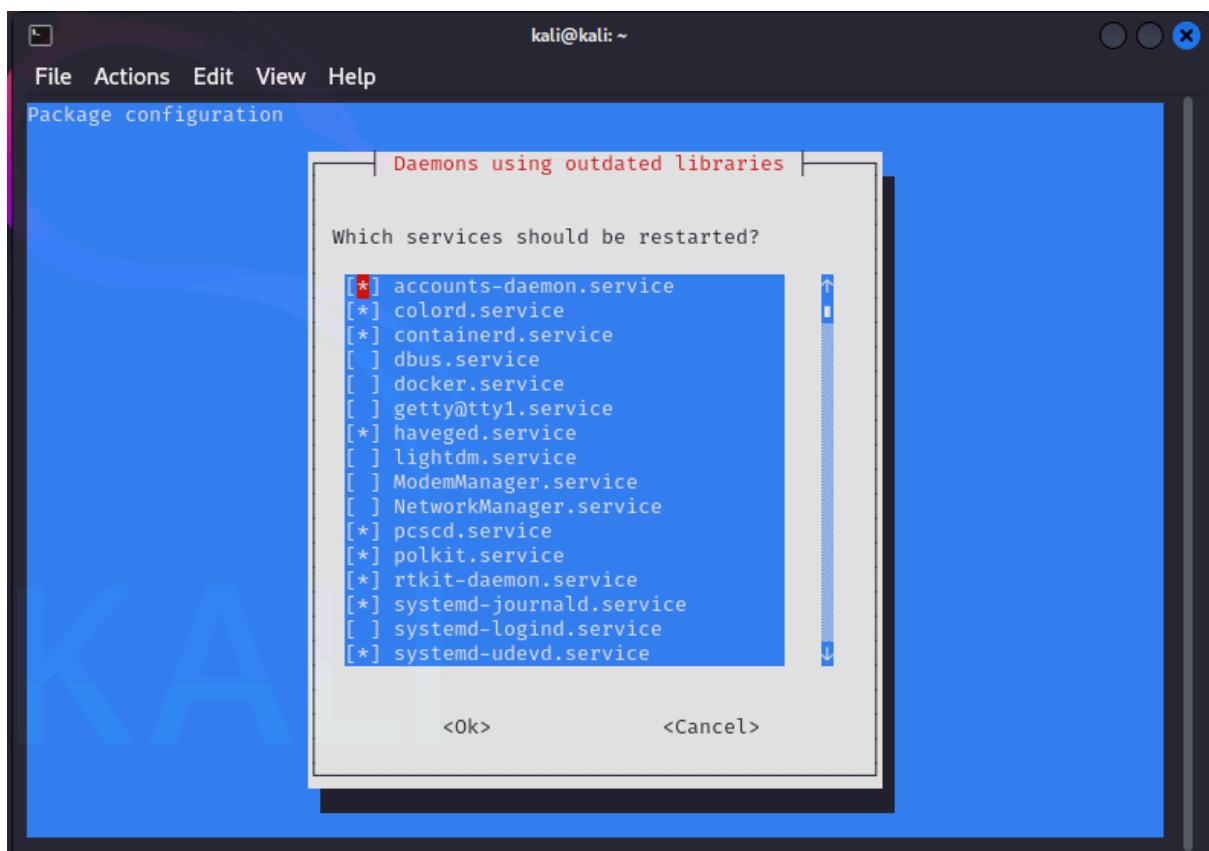
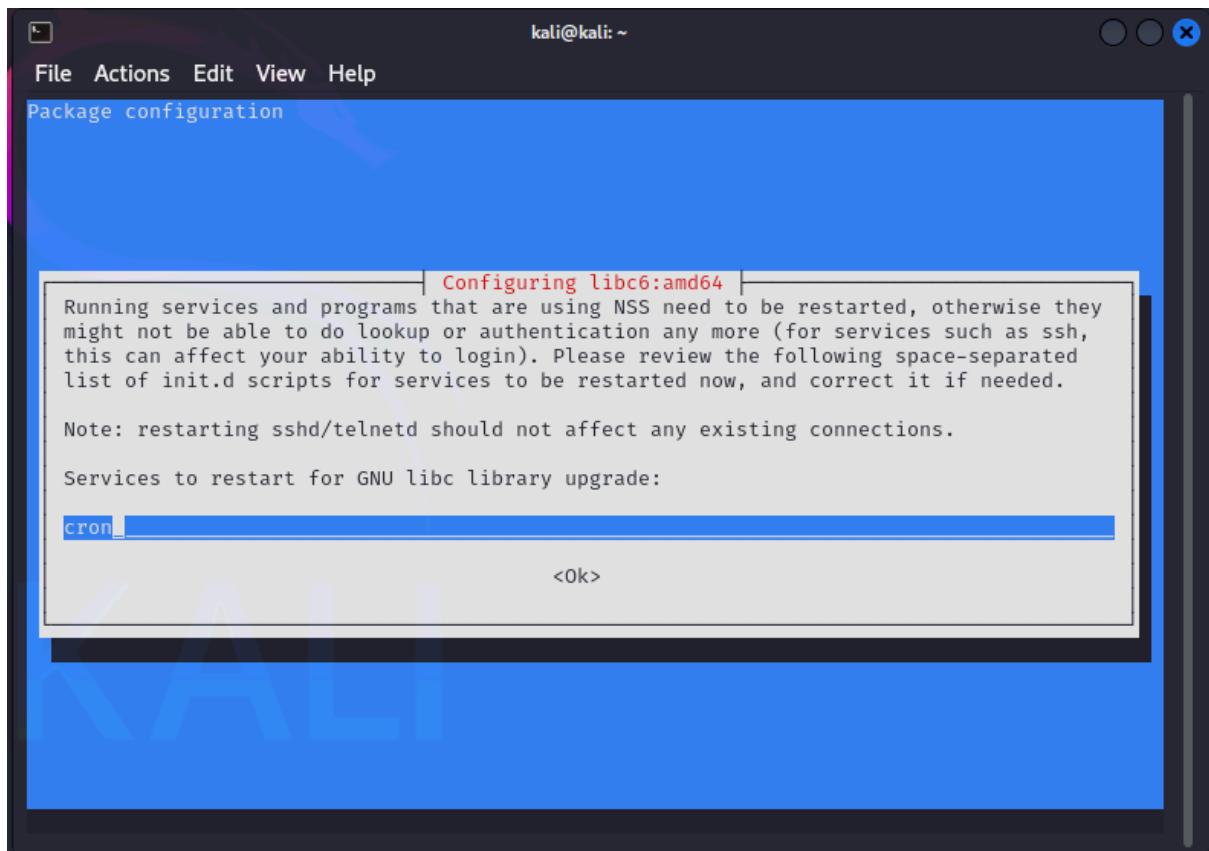
`nmap` (Network Mapper) es una herramienta de código abierto utilizada para el escaneo y la auditoría de seguridad de redes. Es muy popular entre los administradores de sistemas para descubrir dispositivos y servicios en una red, así como para identificar posibles vulnerabilidades.



A screenshot of a terminal window titled "kali@kali: ~". The window shows the command \$ sudo apt install nmap being run, followed by the output of the package manager. The output details the installation process, including packages to be removed, new packages to be installed, and upgraded packages. It also indicates the amount of disk space required and asks for confirmation to continue.

```
File Actions Edit View Help
[(kali㉿kali)-~]
$ sudo apt install nmap
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libnsl-dev libtirpc-dev
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libc-bin libc-dev-bin libc-devtools libc-l10n libc6 libc6-dev libc6-i386 libpcap0.8t64
  libssh2-1t64 locales nmap-common
Suggested packages:
  glibc-doc libnss-nis libnss-nisplus ncat ndiff zenmap
The following packages will be REMOVED:
  libpcap0.8 libssh2-1
The following NEW packages will be installed:
  libpcap0.8t64 libssh2-1t64
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-devtools libc-l10n libc6 libc6-dev libc6-i386 locales nmap
  nmap-common
10 upgraded, 2 newly installed, 2 to remove and 1550 not upgraded.
Need to get 19.1 MB of archives.
After this operation, 68.6 kB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

Nos aparece una ventana emergente, aceptamos y continuamos.



Posteriormente, obtenemos las IP a las que tenemos acceso, esto puede hacerse mediante el comando: `ip a` que no es mas que una abreviatura del comando que usamos con anterioridad: `ip addr`

```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:22:48:9c brd ff:ff:ff:ff:ff:ff
    inet 10.██████ brd 10.██████ scope global dynamic noprefixroute eth0
        valid_lft 86285sec preferred_lft 86285sec
    inet6 fe80::e539:95c0:b08f:38c4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:e3:b7:28:3b brd ff:ff:ff:ff:ff:ff
    inet 172.██████ brd 172.██████ scope global docker0
        valid_lft forever preferred_lft forever
```

Procedemos a hacer `nmap` a la `ip` a la que tenemos acceso.

```
(kali㉿kali)-[~]
$ nmap 10.██████
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-02 13:58 CEST
Nmap scan report for 10.██████
Host is up (0.000090s latency).
All 1000 scanned ports on 10.██████ are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 1 IP address (1 host up) scanned in 13.09 seconds
```

Procedemos a ver todos los dispositivos conectados a la misma red con la flag `-sP`. Esta opción especifica un "Ping Scan" o escaneo de ping. Cuando se ejecuta `nmap -sP` seguido de una dirección IP o un rango de direcciones, nmap envía paquetes ICMP Echo Request (solicitudes de ping) a cada dirección en el rango especificado. Si un host responde a estos paquetes, nmap lo marca como "activo". Además de ICMP, nmap puede intentar hacer una resolución ARP en redes locales, lo que puede detectar hosts incluso si están configurados para no responder a solicitudes ICMP.

ARP, o **Address Resolution Protocol** (Protocolo de Resolución de Direcciones), es un protocolo utilizado en redes de computadoras para mapear direcciones IP (Internet Protocol) a direcciones MAC (Media Access Control) en una red de área local (LAN). Este mapeo es esencial para la comunicación entre dispositivos en una red Ethernet.

En una red Ethernet, los dispositivos se comunican entre sí usando direcciones MAC. Sin embargo, las aplicaciones y servicios de red utilizan direcciones IP para identificar

dispositivos. ARP actúa como un intermediario que traduce direcciones IP a direcciones MAC, permitiendo la correcta entrega de paquetes de datos.



```
(kali㉿kali)-[~]
$ nmap -sP 10.██████
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-02 13:59 CEST
Nmap scan report for 10.██████
Host is up (0.0019s latency).
Nmap scan report for 10.██████
Host is up (0.0026s latency).
Nmap scan report for 10.██████
Host is up (0.0024s latency).
Nmap scan report for 10.██████
Host is up (0.000075s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 15.46 seconds
```

Posibilidades que Ofrece Nmap

Nmap es una herramienta muy poderosa con muchas opciones. Aquí hay algunas de las funcionalidades más comunes:

- **Detección de Sistemas Operativos:**

```
nmap -O 192.168.1.10
```

Identifica el sistema operativo que se está ejecutando en el dispositivo escaneado.

- **Escaneo de Puertos Específicos:**

```
nmap -p 22,80,443 192.168.1.10
```

Escanea solo los puertos especificados (22, 80, 443).

- **Escaneo de Servicios y Versiones:**

```
nmap -sV 192.168.1.10
```

Detecta los servicios que se están ejecutando y sus versiones.

- **Escaneo de Puertos UDP:**

```
nmap -sU 192.168.1.10
```

Escanea puertos UDP.

- **Escaneo Completo de Puertos:**

```
nmap -p- 192.168.1.10
```

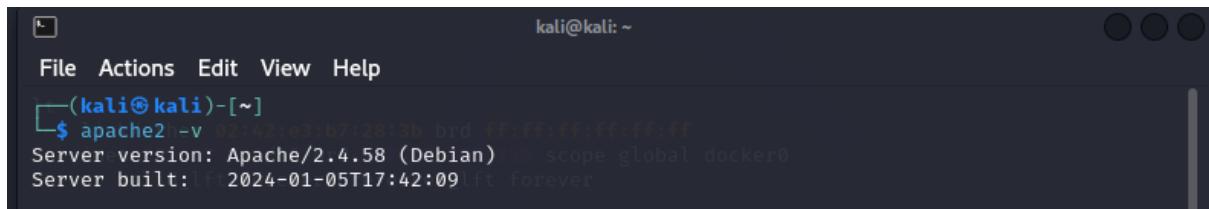
Escanea todos los puertos (0-65535).

En resumen, Nmap es una herramienta esencial para el análisis y la auditoría de redes. Permite descubrir hosts activos en una red, identificar servicios y puertos abiertos, determinar sistemas operativos y más. Podemos usar `nmap` para escanear nuestra propia red y entender mejor cómo está configurada, lo nos ayudará a mejorar la seguridad de nuestros sistemas.

Tarea 12:

(OPCIONAL) Creación de un servidor web personalizado con Apache. Hay una guía en los vídeos. Recordatorio: necesario instalar apache2 en el sistema: sudo apt install apache2. El firewall tiene que permitir el tráfico HTTP y HTTPS (puertos 80 y 443). Verificar el funcionamiento en el navegador (puedes navegar a localhost o la IP 127.0.0.1).

Primero, comprobamos si tenemos apache2 instalado con el comando `apache2 -v`, seguramente si porque en ejercicios anteriores se ha visto.



```
kali@kali: ~
File Actions Edit View Help
[(kali㉿kali)-[~]]$ apache2 -v
Server version: Apache/2.4.58 (Debian)
Server built: 2024-01-05T17:42:09 CEST
```

Esto confirma que tenemos el servicio Apache2 instalado en el sistema. Ahora vamos a proceder con los pasos para asegurarnos de que el servidor web Apache esté configurado correctamente y funcionando.

1. Verificar el Estado del Servicio Apache2

Verificamos que el servicio Apache2 esté activo y corriendo con el comando:

```
sudo systemctl status apache2
```



```
[(kali㉿kali)-[~]]$ sudo systemctl status apache2
[sudo] password for kali: https://nmap.org/
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: inactive (dead)
     Docs: https://httpd.apache.org/docs/2.4/
```

La salida nos indica que el servicio Apache2 está instalado, pero no está activo ni habilitado.

Iniciamos el servicio: sudo systemctl start apache2

Verificamos el estado del servicio: sudo systemctl status apache2

```
(kali㉿kali)-[~]
$ sudo systemctl start apache2
[sudo] password for kali:

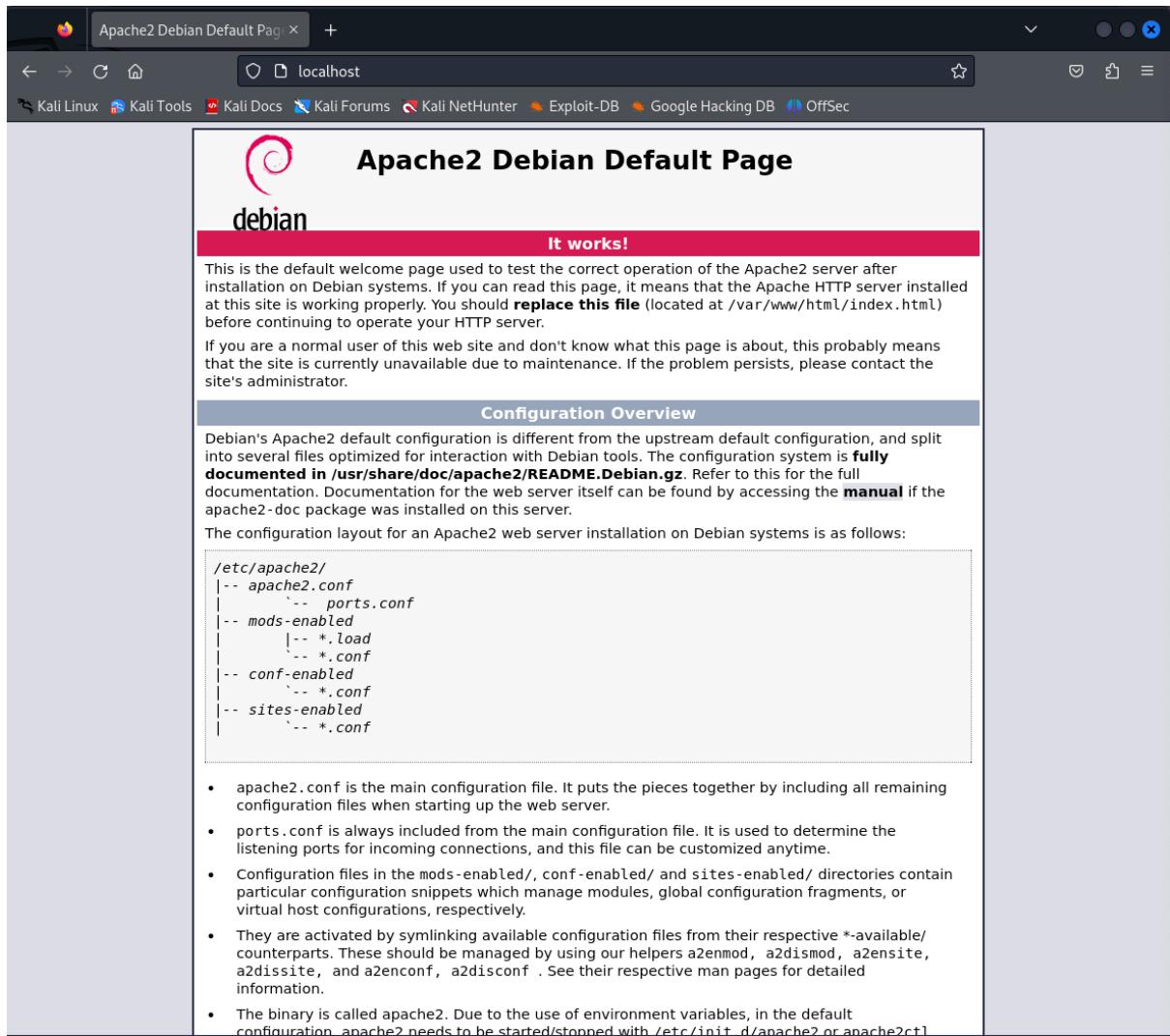
(kali㉿kali)-[~]
$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Sun 2024-06-02 17:47:09 CEST; 5min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 25642 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 25658 (apache2)
    Tasks: 6 (limit: 2274)
   Memory: 18.8M (peak: 19.1M)
      CPU: 155ms
     CGroup: /system.slice/apache2.service
             ├─25658 /usr/sbin/apache2 -k start
             ├─25661 /usr/sbin/apache2 -k start
             ├─25662 /usr/sbin/apache2 -k start
             ├─25663 /usr/sbin/apache2 -k start
             ├─25664 /usr/sbin/apache2 -k start
             └─25665 /usr/sbin/apache2 -k start
Jun 02 17:47:09 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Jun 02 17:47:09 kali apachectl[25657]: AH00558: apache2: Could not reliably determine the ser>
Jun 02 17:47:09 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
Lines 1-20/20 (END) ... skipping ...
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Sun 2024-06-02 17:47:09 CEST; 5min ago
     Host: Docs: https://httpd.apache.org/docs/2.4/
   Process: 25642 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 25658 (apache2)
    Tasks: 6 (limit: 2274)
   Memory: 18.8M (peak: 19.1M)
      CPU: 155ms
     CGroup: /system.slice/apache2.service
             ├─25658 /usr/sbin/apache2 -k start
Starting Nmap 7.8.0 ( https://nmap.org ) at 2024-06-02 13:59 CEST
Nmap scan report for 10.0.2.4
Host is up (0.000s latency).
             ├─25661 /usr/sbin/apache2 -k start
             ├─25662 /usr/sbin/apache2 -k start
             ├─25663 /usr/sbin/apache2 -k start
             ├─25664 /usr/sbin/apache2 -k start
             └─25665 /usr/sbin/apache2 -k start
Nmap scan report for 10.0.2.4
Jun 02 17:47:09 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Jun 02 17:47:09 kali apachectl[25657]: AH00558: apache2: Could not reliably determine the ser>
Jun 02 17:47:09 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
~$ nmap done: 256 IP addresses (4 hosts up) scanned in 15.46 seconds
~$
```

Efectivamente está activado.

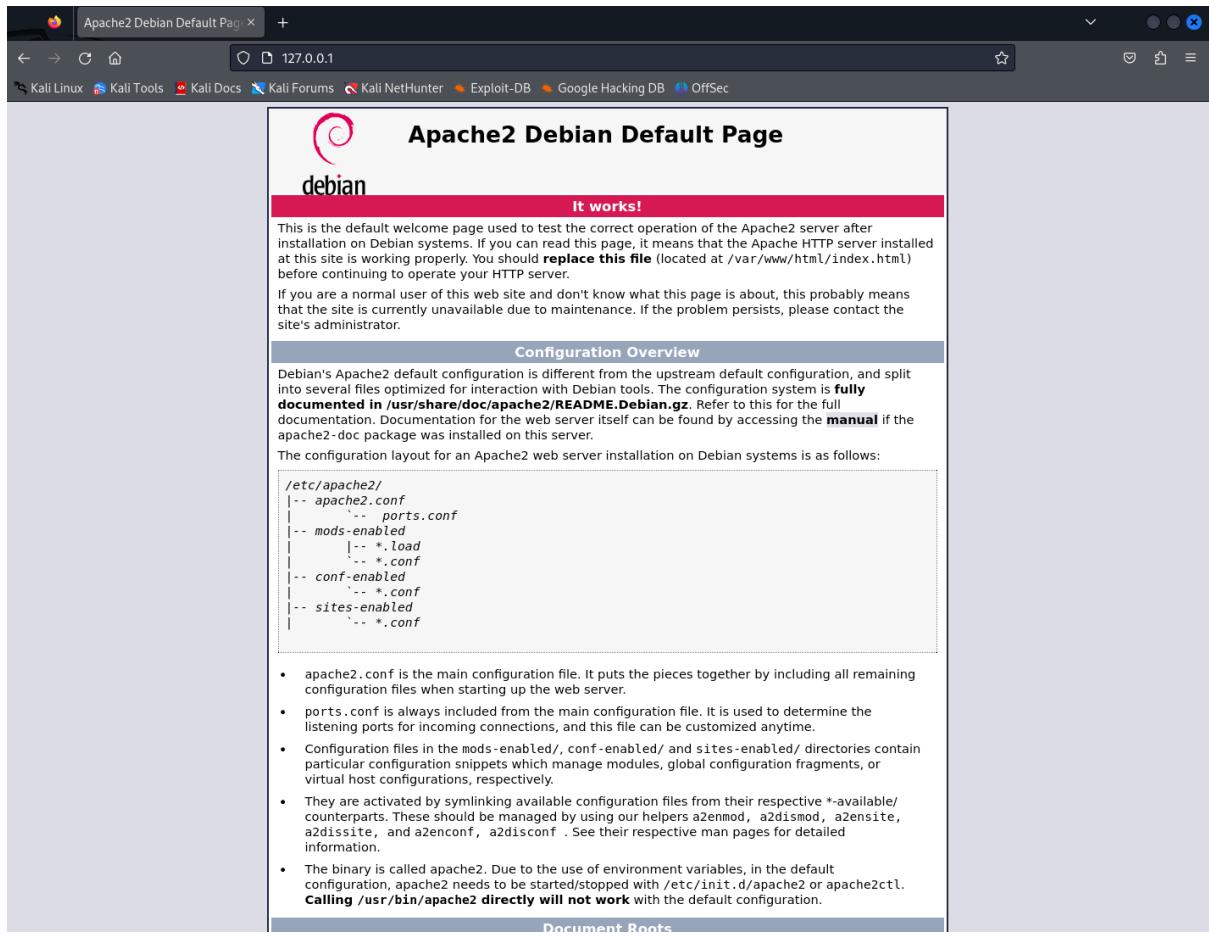
Probar el Servidor Web en el Navegador

1. **Abrir un navegador web** en tu máquina.
2. **Navegar a localhost o 127.0.0.1**: En la barra de direcciones del navegador, escribe <http://localhost> o <http://127.0.0.1> y presiona Enter.

Comprobamos que funciona correctamente con <http://localhost>



Igualmente funciona correctamente con <http://127.0.0.1>



Y con `https://localhost`. Para que esta conexión funcione ha sido necesario hacer reboot y volver a hacerlo.

