

## sPráctica de laboratorio: Servidores Linux

**NOTA: RESPONDER CADA PREGUNTA CON CAPTURAS DE PANTALLA COMPLETAS DONDE SE VEAN LAS SIGLAS DE SU NOMBRE AL LADO DEL RELOJ. EN LUGAR DE COLOCAR UN TEXTO DEBE DE SELECCIONAR EL TEXTO INVESTIGADO EN UNA PÁGINA WEB Y HACERLE CAPTURA DE PANTALLA COMPLETA, PARA LUEGO COLOCARLA AQUÍ. SI SON VARIAS PREGUNTAS, DEBE COLOCAR VARIAS CAPTURAS DE PANTALLAS COMPLETAS. NO COLOCAR INFORMACIÓN ENCONTRADA EN NETACAD, COLOCAR INFORMACIÓN INVESTIGADA EN OTRAS PÁGINAS WEB O LO REALIZADO POR USTED.**

### Objetivos

En esta práctica de laboratorio utilizarán la línea de comandos de Linux para identificar los servidores que están funcionando en una computadora determinada.

#### Parte 1: Servidores

#### Parte 2: Utilizar Telnet para probar los Servicios TCP

### Equipo recomendado

- Máquina virtual Security Workstation

### Instrucciones

#### Parte 1: Servidores

Los servidores son esencialmente programas escritos para proporcionar información específica por solicitud. Los clientes, que también son programas, se comunican con el servidor, hacen la solicitud y aguardan la respuesta del servidor. Se pueden utilizar muchas tecnologías diferentes de comunicación entre clientes y servidores; las más comunes son las redes IP. Esta práctica de laboratorio se enfoca en servidores y clientes basados en redes IP.

#### Paso 1: Accedan a la línea de comandos.

- a. Inicie sesión en la VM Security Workstation con **analyst** como usuario y **cyberops** como contraseña. En toda esta práctica de laboratorio se utiliza la cuenta **analyst** como el usuario ejemplo.

- b. Para acceder a la línea de comando, hagan clic en el icono del **terminal** que se encuentra en el Dock, en la parte inferior de la pantalla de la VM. Se abrirá el emulador de terminales.



### Paso 2: Muestren los servicios que se están ejecutando en este momento.

Se pueden estar ejecutando muchos programas diferentes en una computadora determinada, especialmente en una con el sistema operativo Linux. Muchos programas se ejecutan en segundo plano, por lo que es posible que los usuarios no detecten inmediatamente qué programas se están ejecutando en una computadora determinada. En Linux, a los programas en ejecución también se les llama *procesos*.

**Nota:** El resultado de su comando **ps** será diferente porque se basará en el estado de su VM.

- a. Utilicen el comando **ps** para mostrar todos los programas que se están ejecutando en segundo plano:

```
[analyst@secOps ~]$ sudo ps -elf
[sudo] password for analyst:
 F S UID PID PPID C PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
 4 S root 1 0 0 80 0 - 2250 Sys_ep Feb27 ? 00:00:00 /sbin/init
 1 S root 2 0 0 80 0 - 0 kthrea Feb27 ? 00:00:00 [kthreadd]
 1 S root 3 2 0 80 0 - 0 smpboo Feb27 ? 00:00:00 [ksoftirqd/0]
 1 S root 5 2 0 60 -20 - 0 worker Feb27 ? 00:00:00 [kworker/0:0H]
 1 S root 7 2 0 80 0 - 0 rcu_gp Feb27 ? 00:00:00 [rcu_preempt]
 1 S root 8 2 0 80 0 - 0 rcu_gp Feb27 ? 00:00:00 [rcu_sched]
 1 S root 9 2 0 80 0 - 0 rcu_gp Feb27 ? 00:00:00 [rcu_bh]
 1 S root 10 2 0 -40 - - 0 smpboo Feb27 ? 00:00:00 [migration/0]
 1 S root 11 2 0 60 -20 - 0 rescue Feb27 ? 00:00:00 [lru-add-drain]
 5 S root 12 2 0 -40 - - 0 smpboo Feb27 ? 00:00:00 [watchdog/0]
 1 S root 13 2 0 80 0 - 0 smpboo Feb27 ? 00:00:00 [cpuhp/0]
 5 S root 14 2 0 80 0 - 0 devtmp Feb27 ? 00:00:00 [kdevtmpfs]
 1 S root 15 2 0 60 -20 - 0 rescue Feb27 ? 00:00:00 [netns]
 1 S root 16 2 0 80 0 - 0 watchd Feb27 ? 00:00:00 [khungtaskd]
 1 S root 17 2 0 80 0 - 0 oom_re Feb27 ? 00:00:00 [oom_reaper]
<some output omitted>
```

¿Por qué fue necesario ejecutar **ps** como root (anteponiendo **sudo** al comando)?

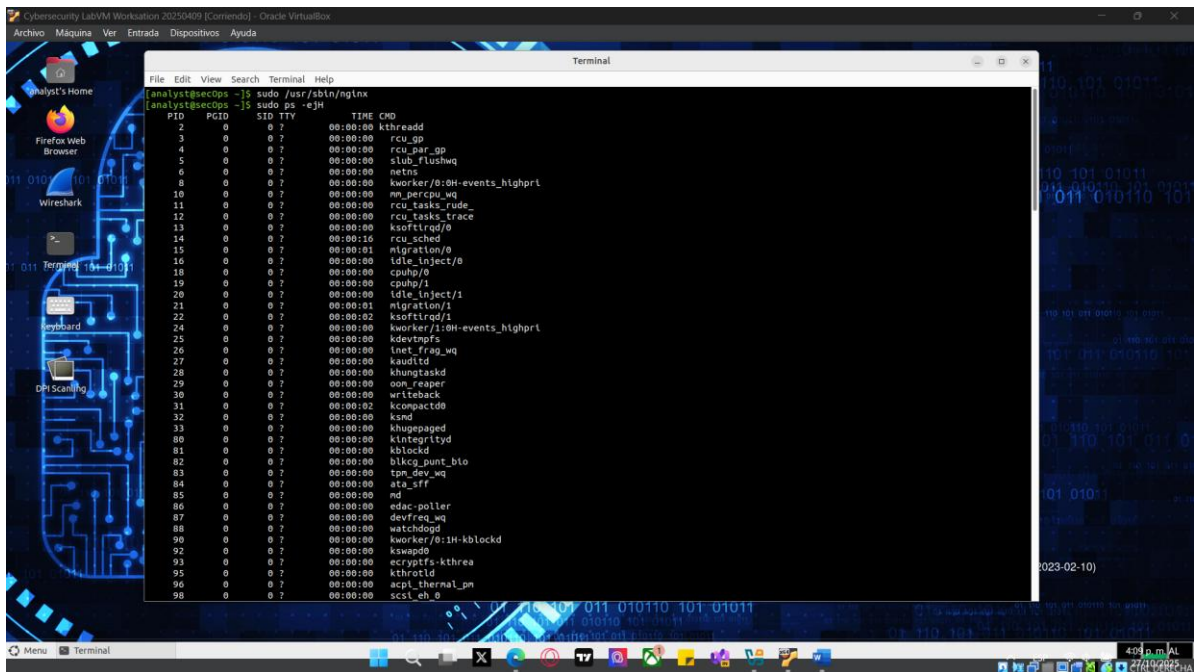
Se usó sudo para poder listar **todos los procesos activos del sistema**, no solo los del usuario analyst, ya que algunos están protegidos y solo el administrador (root) puede verlos completamente.



```
1 1 1 ? 00:00:00 systemd
167 167 167 ? 00:00:01 systemd-journal
193 193 193 ? 00:00:00 systemd-udevd
209 209 209 ? 00:00:00 rsyslogd
210 210 210 ? 00:01:41 java
212 212 212 ? 00:00:01 ovssdb-server
213 213 213 ? 00:00:00 start_pox.sh
224 213 213 ? 00:01:18 python2.7
214 214 214 ? 00:00:00 systemd-logind
216 216 216 ? 00:00:01 dbus-daemon
221 221 221 ? 00:00:05 filebeat
239 239 239 ? 00:00:05 VBoxService
287 287 287 ? 00:00:00 ovs-vswitchd
382 382 382 ? 00:00:00 dhcpcd
387 387 387 ? 00:00:00 lightdm
410 410 410 tty7 00:00:10 Xorg
460 387 387 ? 00:00:00 lightdm
492 492 492 ? 00:00:00 sh
503 492 492 ? 00:00:00 xfce4-session
513 492 492 ? 00:00:00 xfwm4
```

```
517 492 492 ? 00:00:00 Thunar
1592 492 492 ? 00:00:00 thunar-volman
519 492 492 ? 00:00:00 xfce4-panel
554 492 492 ? 00:00:00 panel-6-systray
559 492 492 ? 00:00:00 panel-2-actions
523 492 492 ? 00:00:01 xfdesktop
530 492 492 ? 00:00:00 polkit-gnome-au
395 395 395 ? 00:00:00 nginx
396 395 395 ? 00:00:00 nginx
408 384 384 ? 00:01:58 java
414 414 414 ? 00:00:00 accounts-daemon
418 418 418 ? 00:00:00 polkitd
<some output omitted>
```

¿Cómo representa **ps** la jerarquía de procesos?



El comando **ps -ejH** muestra la jerarquía de procesos **por indentación**, donde:

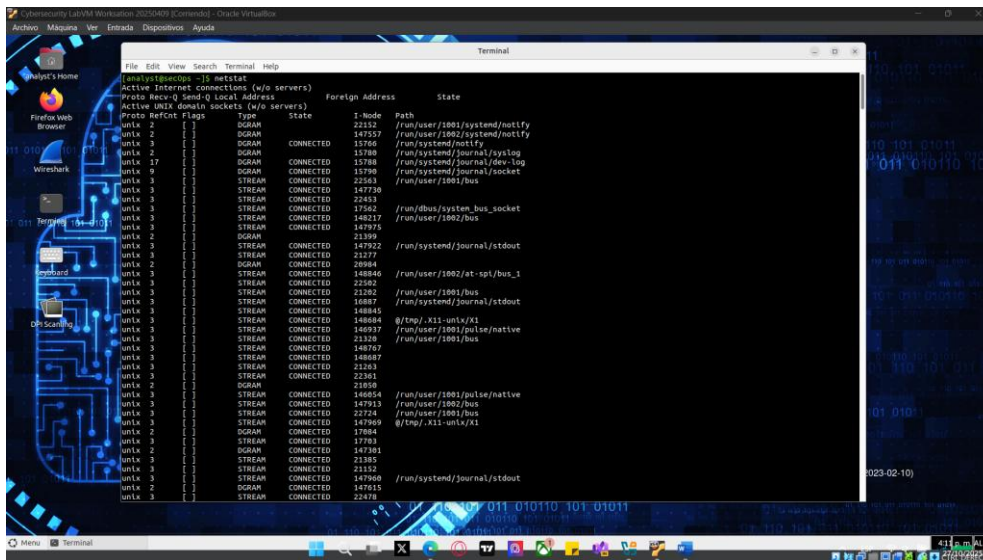
- Cada nivel más sangrado representa un **proceso hijo**.
  - Las columnas PID, PGID, y SID ayudan a identificar las relaciones entre procesos.
  - Esto permite visualizar la estructura jerárquica completa de cómo los procesos se agrupan y se lanzan en el sistema operativo.
- c. Como ya se mencionó, los servidores son programas esencialmente y, a menudo, son iniciados por el propio sistema en el momento del arranque. La tarea que realiza un servidor se llama *servicio*. De esa forma, un servidor web proporciona servicios web.

El comando **netstat** es una excelente herramienta para identificar más fácilmente a los servidores de red que se están ejecutando en una computadora. El poder de **netstat** radica en su capacidad para mostrar conexiones de red.

**Nota:** Su resultado puede ser diferente dependiendo de la cantidad de conexiones de red abiertas en su VM.

En la ventana del terminal escriban **netstat**.

```
[analyst@secOps ~]$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost.localdo:48746 localhost.local:wap-wsp ESTABLISHED
tcp 0 0 localhost.localdo:48748 localhost.local:wap-wsp ESTABLISHED
tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48748 ESTABLISHED
tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48746 ESTABLISHED
tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48744 ESTABLISHED
tcp6 0 0 localhost.localdo:48744 localhost.local:wap-wsp ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags Type State I-Node Path
unix 3 [ ] DGRAM 8472 /run/systemd/notify
unix 2 [ ] DGRAM 8474 /run/systemd/cgroups-agent<some output omitted>
```



Como puede verse arriba, **netstat** devuelve gran cantidad de información cuando se utiliza sin opciones. Se pueden utilizar muchas opciones para filtrar la salida de **netstat** y darle formato, lo que la vuelve más útil.

- d. Utilicen **netstat** con las opciones **-tunap** para ajustar la salida de **netstat**. Observe que **netstat** permite agrupar varias opciones bajo el mismo signo “-”.

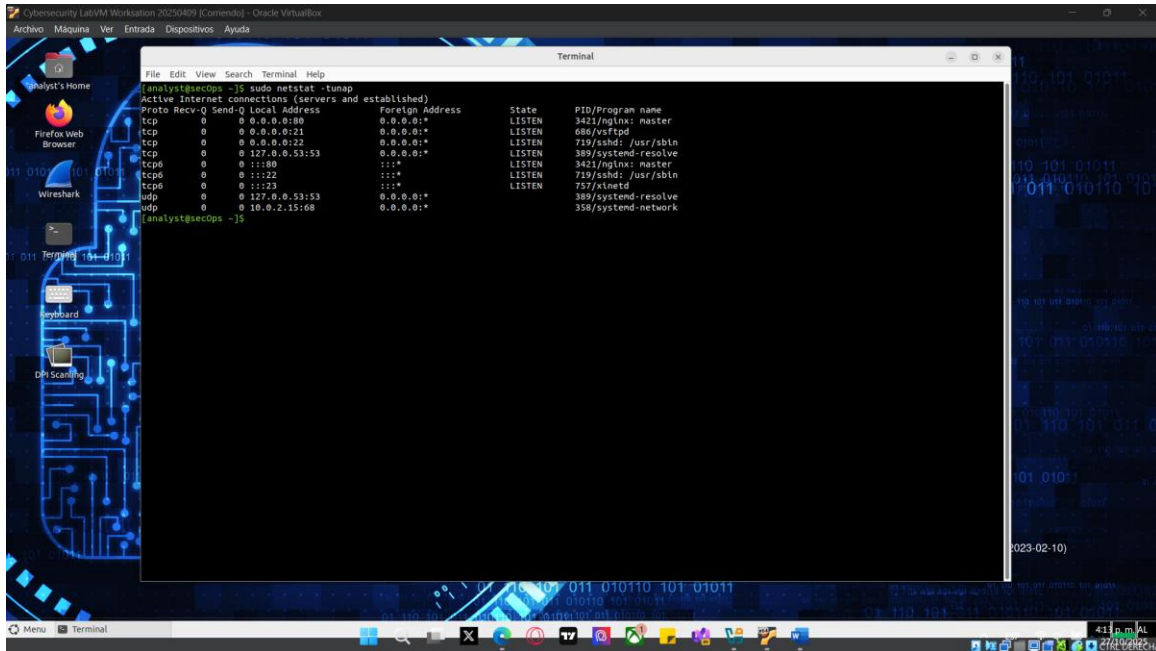
La información correspondiente al servidor nginx está resaltada.

```
[analyst@secOps ~]$ sudo netstat -tunap
[sudo] password for analyst:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:6633 0.0.0.0:* LISTEN 257/python2.7
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 395/nginx: master
```

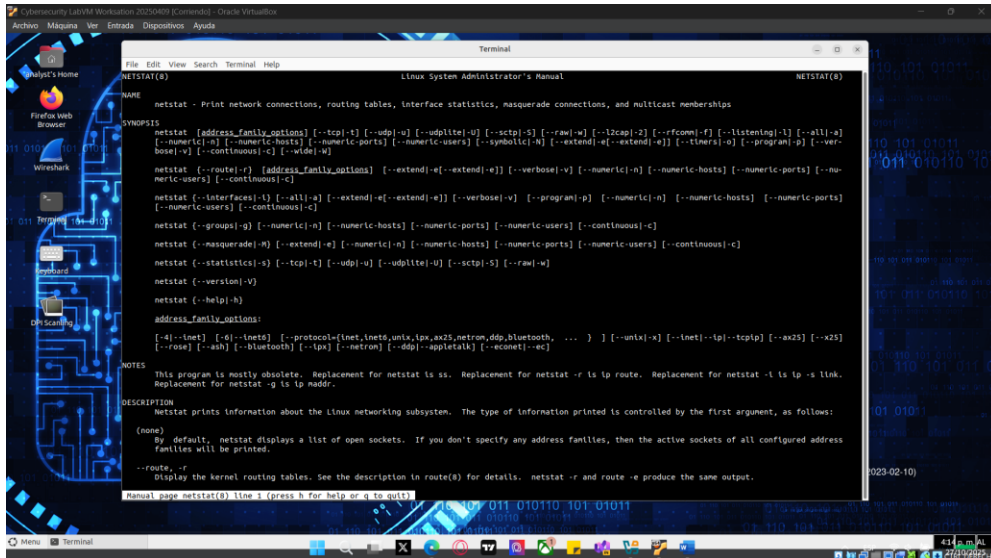


## Práctica de laboratorio: Servidores Linux

```
tcp 0 0 0.0.0.0:21 0.0.0.0:* LISTEN 279/vsftpd
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 277/sshd: /usr/bin
tcp6 0 0 :::22 :::* LISTEN 277/sshd: /usr/bin
udp 0 0 192.168.1.15:68 0.0.0.0:* 237/systemd-network
```



¿Qué significan las opciones **-t**, **-u**, **-n**, **-a** y **-p** en **netstat**? (utilicen **man netstat** para responder)



¿El orden de las opciones es importante para **netstat**?

netstat -tunap

netstat -ptuna

netstat -anutp

Los clientes se conectarán a un puerto y, a través del protocolo correcto, solicitarán información a un servidor. En la salida de **netstat** que se muestra arriba se ven ciertos servicios que están escuchando puertos específicos. Las columnas interesantes son las siguientes:

- o En la primera columna se ve el protocolo de Capa 4 en uso (UDP o TCP, en este caso).
- o En la tercera columna se utiliza el formato **<DIRECCIÓN:PUERTO>** para dar formato a la dirección IP local y al puerto en el que se puede establecer una conexión con un servidor específico. La dirección IP 0.0.0.0 significa que el servidor está escuchando a todas las direcciones IP configuradas en la computadora.
- o En la cuarta columna se utiliza el mismo formato de sockets **<DIRECCIÓN:PUERTO>** para exponer la dirección y el puerto del dispositivo presente en el extremo remoto de la conexión. 0.0.0.0:\* significa que en este momento ningún dispositivo remoto está utilizando la conexión.
- o En la quinta columna se puede ver el estado de la conexión.
- o En la sexta columna se muestra el ID de proceso (PID) del proceso responsable de la conexión. También se muestra el nombre abreviado asociado con el proceso.

Si nos basamos en el resultado de **netstat** que se muestra en el punto (d), ¿cuál es el protocolo de Capa 4, el estado de conexión y el PID del proceso que se están ejecutando en el puerto 80?

Aunque los números de puerto son solo una convención, ¿pueden intuir qué clase de servicio se está ejecutando en el puerto 80 de TCP?

puerto TCP **80** es la convención estándar para **HTTP** (servidores web). Es decir, normalmente en ese puerto corre un servidor web (por ejemplo apache, nginx, httpd, IIS, etc.), que atiende peticiones HTTP no cifradas.

- e. En ocasiones, resulta útil cruzar la información provista por **netstat** con **ps**. En función de la salida del punto (d), se sabe que hay un proceso con el **PID 395** vinculado al puerto 80 de TCP. En este ejemplo se utiliza el puerto 395. Utilizar **ps** y **grep** para generar una lista con todas las líneas de resultado de **ps** que contengan **PID 395**. Reemplazar 395 con el número PID para su instancia de ejecución particular de nginx:

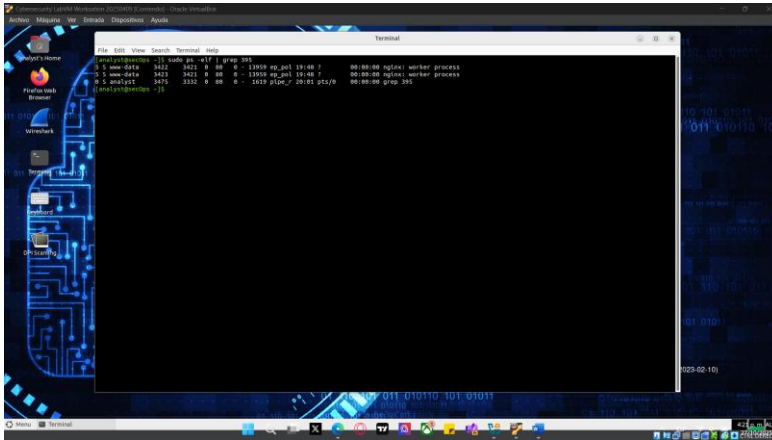
```
[analyst@secOps ~]$ sudo ps -elf | grep 395
[sudo] password for analyst:
1 S root 395 1 0 80 0 - 1829 19:33 ? 00:00:00 nginx: master process /usr/bin/nginx
5 S http 396 395 0 80 0 - 1866 19:33 ? 00:00:00 nginx: worker process
0 S analyst 3789 1872 0 80 0 - 1190 19:53 pts/0 00:00:00 grep 395
```

En el resultado anterior, el comando **ps** se canaliza a través del comando **grep** para filtrar solamente las líneas que contienen el número 395. El resultado son tres líneas con texto acomodado.

La primera línea muestra un proceso en propiedad del usuario **root** (tercera columna), iniciado por otro proceso con PID 1 (quinta columna), a las 19:33 (duodécima columna)

La segunda línea muestra un proceso con PID 396, en propiedad del usuario **http**, iniciado por el proceso 395, a las 19:33.

En la tercera línea se muestra un proceso que es propiedad del usuario **analyst**, con el PID 3789, iniciado por un proceso con el PID 1872, como el comando **grep 395**.



El PID 395 del proceso es **nginx**. ¿Cómo se pudo llegar a esa conclusión a partir de la salida anterior?

En la salida del comando:

```
sudo ps -elf | grep 395
```

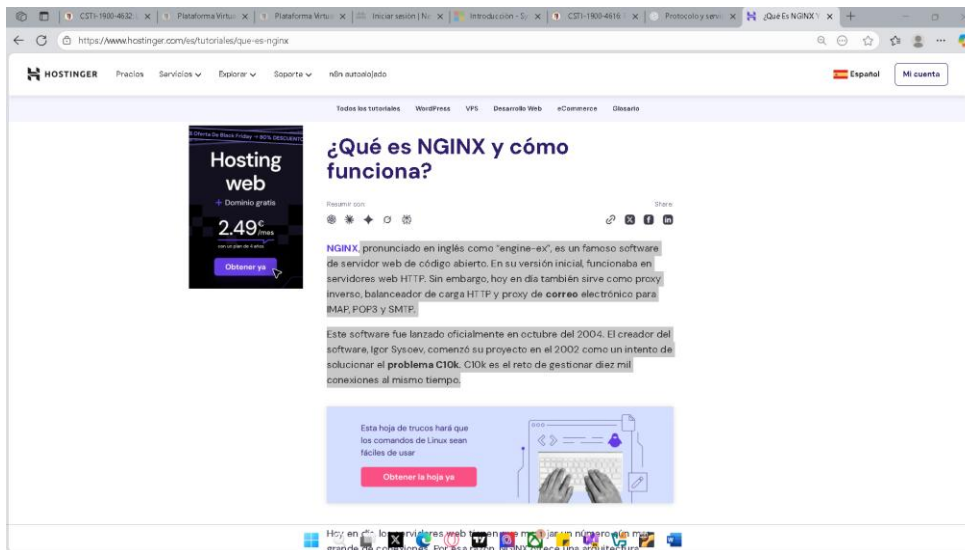
aparece una línea similar a:

```
5 S root      395    1 0 80  0 - 7836 ep_pol 19:33 ?      00:00:00 nginx: master process /usr/sbin/nginx
```

De esta línea se puede deducir que:

- En la **columna del comando ejecutado** se muestra `nginx: master process /usr/sbin/nginx`.
- Esto indica que el proceso cuyo **PID es 395** corresponde al **proceso maestro (master process)** del servidor **nginx**.

¿Qué es **nginx**? ¿Cuál es su función? (Busquen nginx en Google)



En la segunda línea vemos que el proceso 396 es propiedad de un usuario de nombre http y que tiene el número de proceso 395 como proceso matriz. ¿Qué significa? ¿Es común este comportamiento?

La línea indica algo como:

```
5 S http      396  395  0  80  0 - 13959 ep  pol 19:33 ?  00:00:00 nginx: worker process
```

Esto significa:



- El proceso con **PID 396** fue iniciado por el proceso con **PID 395**, es decir, **395 es su proceso padre (parent process)**.
- El proceso **396 pertenece al usuario http** (o `www-data` en algunos sistemas).
- En `nginx`, el proceso **maestro (master process)** corre como **root**, y crea varios **procesos hijo (worker processes)** bajo usuarios no privilegiados como `http` o `www-data` para manejar las conexiones de red de forma segura.

¿Por qué vemos `grep 395` en la última línea?

La última línea:

```
0 S analyst 3475 3332 0 80 0 - 1619 pipe_r 20:01 pts/0 00:00:00 grep 395
```

muestra el propio **comando grep 395** que el usuario ejecutó.

Esto ocurre porque `grep` también es un proceso del sistema en ejecución mientras busca el número 395 en la salida de `ps`.

Por lo tanto, `ps` lista **todos** los procesos activos — incluyendo el mismo comando `grep` que está filtrando los resultados.

## Parte 2: Utilizar Telnet para probar servicios TCP

Telnet es una aplicación de shell remoto simple. Telnet se considera inseguro ya que no proporciona cifrado. Los administradores que optan por utilizar Telnet para administrar dispositivos de red y servidores en forma remota dejan expuestas las credenciales para iniciar sesión en ese servidor, ya que Telnet transmitirá los datos de la sesión como texto sin cifrar. Si bien no se recomienda utilizar Telnet como aplicación de shell remoto, puede ser muy útil para pruebas rápidas o para recopilar información sobre servicios TCP.

El protocolo Telnet funciona en el puerto 23, por medio de TCP de manera predeterminada. El cliente **telnet**, sin embargo, permite especificar otro puerto. Si se cambia el puerto y se establece una conexión con un servidor, el cliente **telnet** permite que un analista especializado en redes evalúe rápidamente la naturaleza de un servidor específico comunicándose directamente con él.

**Nota:** Es altamente recomendable que utilicen **ssh** como aplicación de shell remoto en lugar de **telnet**.

- a. En la Parte 1, descubrimos que **nginx** se estaba ejecutando y que se le había asignado el puerto 80 de TCP. Aunque con una breve búsqueda en internet se revela que **nginx** es un servidor web liviano, ¿qué podría hacer un `analyst` para estar seguro de eso? ¿Qué sucedería si un atacante cambiara el nombre de un programa de malware por el de **nginx**, simplemente para que se pareciera al servidor web tan utilizado? Utilicen **telnet** para conectar el host local en el puerto 80 de TCP:

```
[analyst@secOps ~]$ telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
```

- b. Presionen algunas teclas del teclado. Cualquiera funcionará. Después de presionar algunas teclas, pulsen `INTRO`. A continuación se incluye la salida completa, incluido el establecimiento de la conexión Telnet y las teclas aleatorias que se presionaron (`fdsafsdaf`, en este caso):

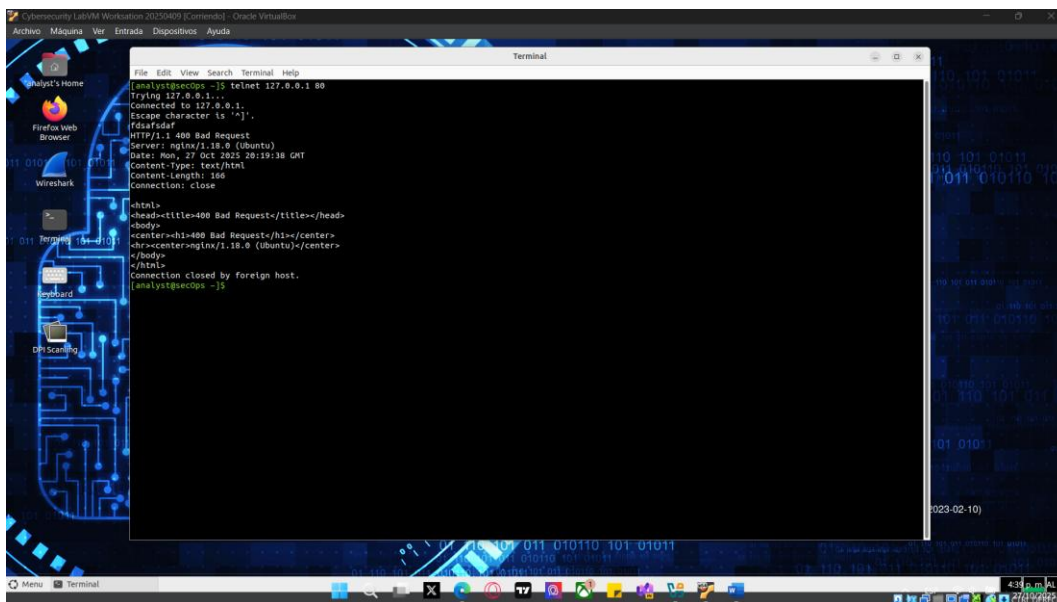
```
fdsafsdaf
HTTP/1.1 400 Bad Request
Server: nginx/1.16.1
Date: Tue, 28 Apr 2020 20:09:37 GMT
Content-Type: text/html
Content-Length: 173
```

```
Connection: close
```

```
<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.16.1</center>
</body>
</html>
```

```
Connection closed by foreign host.
```

Gracias al protocolo Telnet se estableció una conexión TCP de texto sin cifrar (a cargo del cliente Telnet) directamente al servidor nginx, que está escuchando al puerto 80 de TCP de 127.0.0.1. Esta conexión nos permite enviar datos directamente al servidor. Como nginx es un servidor web, no comprende la secuencia de letras aleatorias que se le envió y devuelve un error en formato de página web.



¿Por qué el error se envió como página web?

Nginx es un servidor web que usa HTTP. Al enviar texto aleatorio que no cumple con el formato HTTP, devuelve un **error 400 Bad Request** en **HTML**, como es normal en servidores web.

Aunque el servidor reportó un error y dio por finalizada la conexión, pudimos descubrir mucho. Descubrimos que:

- 1) El **nginx** con el PID 395 de hecho es un servidor web.
- 2) The version of **nginx** is 1.16.1.
- 3) The network stack of our Security Workstation VM is fully functional all the way to Layer 7.

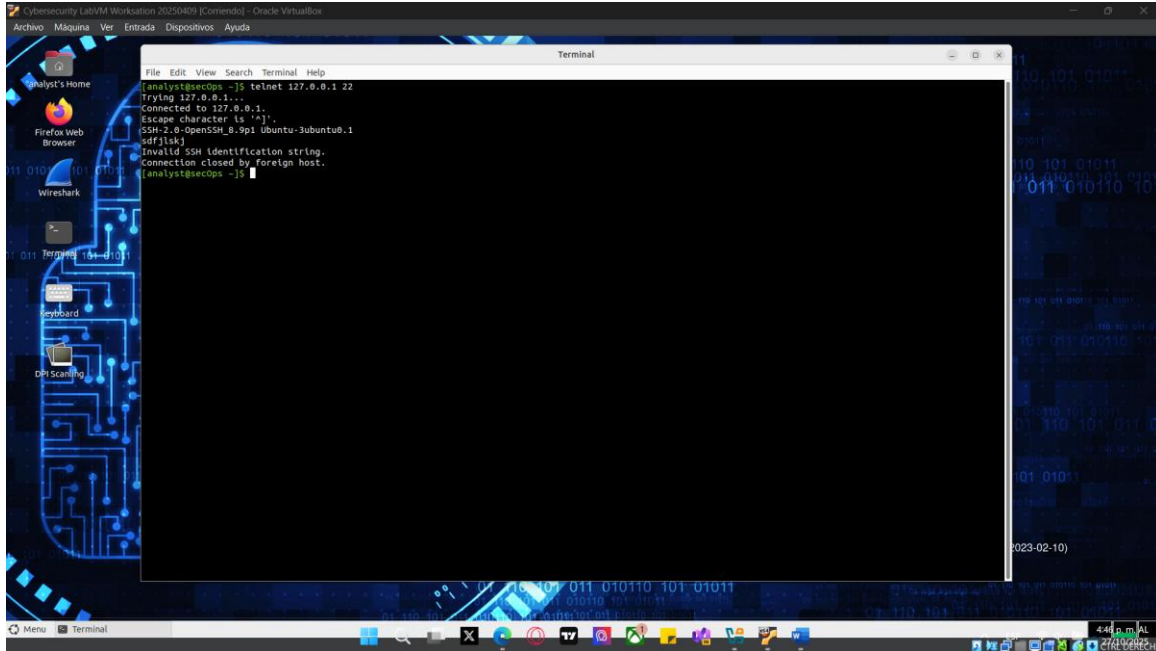
No todos los servicios son iguales. Algunos servicios están diseñados para aceptar datos sin formato y no se cancelan si se introduce basura desde el teclado. A continuación vemos un ejemplo de esa clase de servicio:

- c. Si miramos la salida de **netstat** que presentamos anteriormente, podremos ver un proceso conectado al puerto 22. Utilicen Telnet para conectarse con él.

El puerto 22 de TCP está asignado a un servicio SSH. SSH permite que un administrador se conecte con una computadora remota en forma segura.

A continuación se detalla la salida:

```
[analyst@secOps ~]$ telnet 127.0.0.1 22
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_8.2
sdfjlskj
Invalid SSH identification string.
Connection closed by foreign host.
```



Utilicen Telnet para conectarse con el puerto 68. ¿Qué ocurre? Explique.

SSH no acepta texto arbitrario → la conexión se cierra.

## Preguntas de reflexión

1. ¿Cuáles son las ventajas de utilizar netstat?

**Permite ver puertos abiertos y conexiones activas.**

**Muestra qué procesos están escuchando en cada puerto.**

**Ayuda a identificar servicios sospechosos o inesperados.**

**Útil para diagnóstico de red y monitoreo de seguridad.**

2. ¿Cuáles son las ventajas de utilizar Telnet? ¿Es seguro?

**Ventajas de utilizar Telnet y seguridad:**

- Permite **probar la conectividad a un puerto TCP** de forma directa.
- Permite **enviar datos manualmente** a un servicio y observar su respuesta.
- Es útil para **diagnóstico rápido de servicios** como HTTP o SSH (aunque no para autenticarse).

**Seguridad:**

- **No es seguro**, porque transmite todo **sin cifrar**, incluyendo contraseñas.
- Hoy en día se recomienda usar **SSH** para conexiones remotas seguras.