

LINUX FUNDAMENTALS

HTB ACADEMY



Por DamonCDB

- 1. Estructura de Linux**
- 2. Introducción a la Shell**
- 3. Descripción del Prompt**
- 4. Obteniendo ayuda**
- 5. Información del sistema**
- 6. Gestión de usuario**
- 7. Gestión de paquetes**
- 8. Gestión de procesos y servicios**
- 9. Trabajando con servicios web**
- 10. Navegación**
- 11. Trabajando con archivos y directorios**
- 12. Editar archivos**
- 13. Encuentra archivos y directorios**
- 14. Descriptores de archivos y redirecciones**
- 15. Filtrar contenidos**
- 16. Gestión de permisos**
- 17. Atajos**
- 18. Seguridad en Linux**

1 - Estructura de Linux

5 principios sobre Linux.-

- Todo es un archivo.
- Programas pequeños y con un solo propósito.
- Habilidad para encadenar programas juntos para realizar tareas complejas.
- Evita las interfaces de usuario cautivas.
- Datos de configuración almacenados en un archivo de texto.

Componentes.-

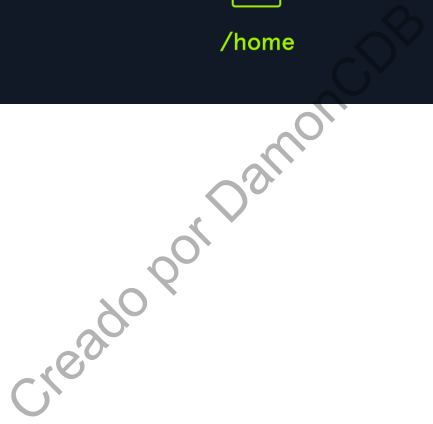
1. Bootloader
2. OS Kernel
3. Daemons
4. Shell del OS
5. Servidor de Gráficos
6. Gestor de ventanas
7. Utilidades

Arquitectura de Linux

- Hardware (periféricos)
- Kernel (núcleo)
- Shell (interfaz de la línea de comandos)
- Utilidad del sistema (funcionalidades)

Jerarquía de archivos

/



2 - Introducción a la Shell

La terminal de Linux proporciona una interfaz de entrada/salida basada en texto entre los usuarios y el núcleo del sistema.

Se pueden ejecutar comandos en la ventana de la terminal.

Emuladores de terminal

Software que emula las funciones de una terminal.

Terminales:

- GNOME Terminal
- XFCE4 Terminal
- XTerm
- ...

Multiplexers (ejecutan más de una terminal en una ventana):

- Tmux
- GNU Screen
- ...

Shell

La más usada en Linux es **Bourne-Again Shell (BASH)**, que forma parte del proyecto GNU. Permite automatizar muchos procesos con scripts más o menos largos.

3 - Descripción del Prompt

El prompt de bash incluye info como el usuario, hostname y directorio actual:

```
<username>@<hostname><current working directory>$
```

El directorio home se marca con un guion curvado (~).

```
<username>@<hostname>[~]$
```

El signo de dólar cambia a almohadilla cuando ingresas como usuario root en el sistema.

```
root@htb[/htb]#
```

El prompt de bash puede personalizarse y cambiarse a nuestro antojo. Para saber más al respecto, mirar en bashrcgenerator (<https://bashrcgenerator.com/>) o en powerline (<https://github.com/powerline/powerline>), que nos permiten adaptarlo a nuestras necesidades.

Creado por DamianoDB

4 - Obteniendo ayuda

Las principales formas de conseguir ayuda es con las páginas **man** y con las funciones de ayuda.

man

Sintaxis:

```
user@htb[/htb]$ man <tool>
```

help

Sintaxis:

```
user@htb[/htb]$ <tool> --help
```

Versión abreviada:

```
user@htb[/htb]$ <tool> -h
```

apropos

Sintaxis:

```
user@htb[/htb]$ apropos <keyword>
```

Otro recurso útil es <https://explainshell.com/>

5 - Información del sistema

Lista de las herramientas necesarias para obtener la información:

- **whoami**: muestra el nombre actual (windows y linux)
- **id**: muestra el id de los usuarios
- **hostname**: nombre del host actual
- **uname**: información básica sobre el sistema operativo y el hardware
- **pwd**: nombre del directorio actual
- **ifconfig**: para ver una dirección de red y/o configurar los parámetros de red
- **ip**: para manipular el routing, los dispositivos de red, interfaces y túneles
- **netstat**: muestra el estado de la red
- **ss**: otra herramienta para investigar sockets
- **ps**: muestra el estado de los procesos
- **who**: muestra quién está loggeado en el sistema
- **env**: crea un entorno o conjunto y ejecuta el comando
- **lsblk**: lista los dispositivos bloqueados
- **lsusb**: lista los dispositivos USB
- **lsdf**: lista los archivos abiertos
- **lspci**: lista los dispositivos PCI

Flags de uname

- **-a, --all**: muestra toda la información
- **-s, --kernel-name**: muestra el nombre del kernel
- **-n, --nodename**: muestra el nombre del nodo del host
- **-r, --kernel-release**: muestra la versión del kernel
- **-m, --machine**: muestra el nombre de la máquina
- **-p, --processor**: muestra el tipo de procesador
- **-i, --hardware-platform**: muestra la plataforma de hardware
- **-o, --operating-system**: muestra el sistema operativo

La flag **-a** omitirá **-p** y **-i** si estos se desconocen.

Conocer la versión del kernel puede ser útil para buscar exploits más eficazmente.

Logging In via SSH

Secure Shell (SSH) es un protocolo que permite a los clientes acceder y ejecutar comandos o acciones en equipos remotos.

No requiere de interfaz gráfica.

SSH Login

ssh [nombre de usuario]@[dirección IP]

Creado por DamonCDB

6 - Gestión de usuario

No es extraño que los usuarios de un grupo específico tengan permisos para ver o editar archivos o directorios específicos.

Ejecución como un usuario

```
user@htb[/htb]$ cat /etc/shadow  
  
cat: /etc/shadow: Permission denied
```

Ejecución como root

```
user@htb[/htb]$ sudo cat /etc/shadow  
  
root:<SNIP>:18395:0:99999:7:::  
daemon:!:17737:0:99999:7:::  
bin:!:17737:0:99999:7:::  
<SNIP>
```

Lista que nos ayudará a entender mejor la gestión de usuarios y a tratar con ello.

- **sudo**: ejecutar comandos como un usuario distinto
- **su**: solicita las credenciales para cambiar a superusuario
- **useradd**: crea un nuevo usuario o actualiza la información de un nuevo usuario por defecto
- **userdel**: elimina una cuenta de usuario y sus archivos relacionados
- **usermod**: modifica una cuenta de usuario
- **adgroup**: añade un grupo al sistema
- **delgroup**: elimina un grupo del sistema
- **passwd**: cambia la contraseña del usuario

7 - Gestión de paquetes

Los paquetes son archivos que contienen binarios de software, archivos de configuración, información sobre las dependencias y mantienen el rastro de actualizaciones.

Las características que la mayoría de sistemas de gestión de paquetes han de tener son:

- Descarga de paquetes
- Resolución de dependencias
- Formato binario estándar del paquete
- Ubicaciones de instalación y configuración comunes
- Funcionalidad y configuración adicional
- Control de calidad

El requisito indispensable es que el software a instalar esté disponible como su correspondiente paquete.

Si el gestor de paquetes reconoce que se necesitan paquetes adicionales para el correcto funcionamiento del paquete que aún no ha sido instalado, se incluirá la dependencia y, o se avisará al administrador o intentará recargar el software desaparecido y lo instalará.

Existen distintos programas de gestión de paquetes que podemos usar:

- **dpkg**: herramienta para instalar, construir, eliminar y gestionar paquetes Debian. Su front-end principal es aptitude
- **apt**: proporciona una interfaz de línea de comandos a alto nivel para el sistema de gestión de paquetes
- **aptitude**: alternativa a apt, interfaz a alto nivel del gestor de paquetes
- **snap**: instala, configura, refresca y elimina paquetes snap (cloud, servidores, escritorio e IoT)
- **gem**: es el front-end de RubyGems, el gestor de paquetes estándar para Ruby
- **pip**: instalador de paquetes Python recomendado para la instalación de paquetes Python no disponibles en archivos Debian. Funciona con repositorios de control de versiones y previene la instalación parcial descargando todos los requisitos antes de comenzar la instalación
- **git**: sistema de control de versiones rápido, escalable y distribuido con un gran conjunto de comandos propios que proporcionan operaciones a alto nivel

Gestor de paquetes avanzado (APT)

Las distribuciones Linux basadas en Debian usan **APT**, que contiene archivos ".deb". La herramienta **dpkg** se usa para instalar estos programas. **APT** hace más sencillo el tener que descargar paquetes adicionales para la instalación de un programa.

Cuando actualizamos un programa o instalamos uno nuevo, el sistema solicita a los repositorios

el paquete deseado. La mayoría de distribuciones Linux usan el repositorio más estable o "main". Este se puede comprobar en `/etc/apt/sources.list`.

```
user@htb[/htb]$ cat /etc/apt/sources.list.d/parrot.list

# parrot repository
# this file was automatically generated by parrot-mirror-selector
deb http://htb.deb.parrot.sh/parrot/ rolling main contrib non-free
#deb-src https://deb.parrot.sh/parrot/ rolling main contrib non-free
deb http://htb.deb.parrot.sh/parrot/ rolling-security main contrib non-free
#deb-src https://deb.parrot.sh/parrot/ rolling-security main contrib non-free
```

APT usa una base de datos llamada APT caché para proporcionar información sobre los paquetes instalados en nuestro sistema:

```
user@htb[/htb]$ apt-cache search impacket

impacket-scripts - Links to useful impacket scripts examples
polenum - Extracts the password policy from a Windows system
python-pcap - Python interface to the libpcap packet capture library (Python 2)
python3-impacket - Python3 module to easily build and dissect network protocols
python3-pcap - Python interface to the libpcap packet capture library (Python 3)
```

Podemos ver info adicional sobre un paquete:

```
user@htb[/htb]$ apt-cache show impacket-scripts

Package: impacket-scripts
Version: 1.4
Architecture: all
Maintainer: Kali Developers <devel@kali.org>
Installed-Size: 13
Depends: python3-impacket (>= 0.9.20), python3-ldap3 (>= 2.5.0), python3-ldapdomaindump
Breaks: python-impacket (<< 0.9.18)
Replaces: python-impacket (<< 0.9.18)
Priority: optional
Section: misc
Filename: pool/main/i/impacket-scripts/impacket-scripts_1.4_all.deb
Size: 2080
<SNIP>
```

También podemos listar todos los paquetes instalados:

```
user@htb[/htb]$ apt list --installed

Listing... Done
accountsservice/rolling,now 0.6.55-2 amd64 [installed,automatic]
adapta-gtk-theme/rolling,now 3.95.0.11-1 all [installed]
adduser/rolling,now 3.118 all [installed]
adwaita-icon-theme/rolling,now 3.36.1-2 all [installed,automatic]
aircrack-ng/rolling,now 1:1.6-4 amd64 [installed,automatic]
<SNIP>
```

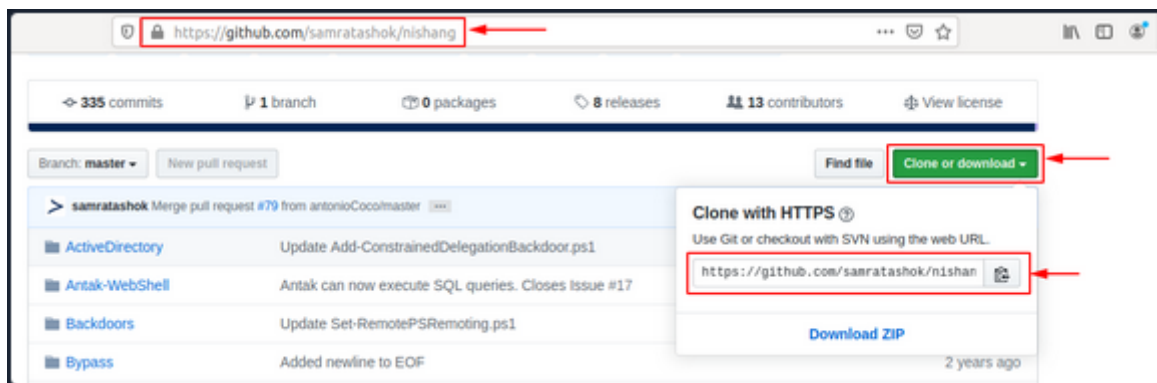
Si echamos de menos algunos paquetes, podemos buscarlos e instalarlos con este comando:

```
user@htb[/htb]$ sudo apt install impacket-scripts -y

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  impacket-scripts
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,080 B of archives.
After this operation, 13.3 kB of additional disk space will be used.
Get:1 https://euro2-emea-mirror.parrot.sh/mirrors/parrot rolling/main amd64
impacket-scripts all 1.4 [2,080 B]
Fetched 2,080 B in 0s (15.2 kB/s)
Selecting previously unselected package impacket-scripts.
(Reading database ... 378459 files and directories currently installed.)
Preparing to unpack .../impacket-scripts_1.4_all.deb ...
Unpacking impacket-scripts (1.4) ...
Setting up impacket-scripts (1.4) ...
Scanning application launchers
Removing duplicate launchers from Debian
Launchers are updated
```

Git

Ahora que tenemos instalado **git**, podemos usarlo para descargar herramientas útiles de Github. Para ello navegaremos al **repositorio del proyecto** y copiaremos el enlace de Github usando git para descargarlo.



De todos modos, antes de descargar el proyecto, deberemos crear una carpeta para el mismo:

```
user@htb[/htb]$ mkdir ~/nishang/ && git clone
https://github.com/samratashok/nishang.git ~/nishang

Cloning into '/opt/nishang/'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 1691 (delta 4), reused 6 (delta 2), pack-reused 1676
Receiving objects: 100% (1691/1691), 7.84 MiB | 4.86 MiB/s, done.
Resolving deltas: 100% (1055/1055), done.
```

DPKG

Podemos descargar programas y herramientas de repositorios por separado:

```
user@htb[/htb]$ wget
http://archive.ubuntu.com/ubuntu/pool/main/s/strace/strace_4.21-1ubuntu1_amd64.deb

--2020-05-15 03:27:17--
http://archive.ubuntu.com/ubuntu/pool/main/s/strace/strace_4.21-1ubuntu1_amd64.deb
Resolving archive.ubuntu.com (archive.ubuntu.com)... 91.189.88.142, 91.189.88.152,
2001:67c:1562::18, ...
Connecting to archive.ubuntu.com (archive.ubuntu.com)|91.189.88.142|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 333388 (326K) [application/x-debian-package]
Saving to: 'strace_4.21-1ubuntu1_amd64.deb'

strace_4.21-1ubuntu1_amd64.deb      100%
[=====>] 325,57K
.-KB/s    in 0,1s

2020-05-15 03:27:18 (2,69 MB/s) - 'strace_4.21-1ubuntu1_amd64.deb' saved
[333388/333388]
```

Después podemos usar tanto **apt** como **dpkg** para instalar el paquete:

```
user@htb[/htb]$ sudo dpkg -i strace_4.21-1ubuntu1_amd64.deb

(Reading database ... 154680 files and directories currently installed.)
Preparing to unpack strace_4.21-1ubuntu1_amd64.deb ...
Unpacking strace (4.21-1ubuntu1) over (4.21-1ubuntu1) ...
Setting up strace (4.21-1ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

Con esto habremos instalado la herramienta y podemos comprobar que funciona correctamente:

```
user@htb[/htb]$ strace -h

usage: strace [-CdffhiqrsttvVwxy] [-I n] [-e expr]...
             [-a column] [-o file] [-s strsize] [-P path]...
             -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]
or: strace -c[dfw] [-I n] [-e expr]... [-O overhead] [-S sortby]
             -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]

Output format:
  -a column      alignment COLUMN for printing syscall results (default 40)
  -i             print instruction pointer at time of syscall
```

8 - Gestión de procesos y servicios

Dos tipos de servicios:

- Internos: servicios relevantes requeridos para el inicio del sistema.
- Instalados por el usuario.

También están los **daemons** que se identifican con la letra **d** al final del nombre del programa (sshd, systemd).

systemd monitoriza y cuida del arranque ordenado y de detener otros servicios. Todos los procesos tienen asignado un PID que se puede ver bajo **/proc/** con el número correspondiente. Un proceso puede tener un proceso padre (PPID).

También podemos usar **update-rc.d** para gestionar los enlaces del script de inicio SysV.

Systemctl

Podemos arrancar el servicio con el siguiente comando:

```
user@htb[/htb]$ systemctl start ssh
```

Después podemos comprobar que corre sin errores:

```
user@htb[/htb]$ systemctl status ssh
```

```
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Thu 2020-05-14 15:08:23 CEST; 24h ago
   Main PID: 846 (sshd)
   Tasks: 1 (limit: 4681)
   CGroup: /system.slice/ssh.service
           └─846 /usr/sbin/sshd -D
```

```
Mai 14 15:08:22 inlane systemd[1]: Starting OpenBSD Secure Shell server...
Mai 14 15:08:23 inlane sshd[846]: Server listening on 0.0.0.0 port 22.
Mai 14 15:08:23 inlane sshd[846]: Server listening on :: port 22.
Mai 14 15:08:23 inlane systemd[1]: Started OpenBSD Secure Shell server.
Mai 14 15:08:30 inlane systemd[1]: Reloading OpenBSD Secure Shell server.
Mai 14 15:08:31 inlane sshd[846]: Received SIGHUP; restarting.
Mai 14 15:08:31 inlane sshd[846]: Server listening on 0.0.0.0 port 22.
Mai 14 15:08:31 inlane sshd[846]: Server listening on :: port 22.
```


Para añadir OpenSSH al script SysV para decirle al sistema que arranque este servicio después de encender, lo hacemos con:

```
user@htb[/htb]$ systemctl enable ssh

Synchronizing state of ssh.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
```

Una vez reiniciado el sistema, OpenSSH se abrirá automáticamente. Podemos comprobarlo con la herramienta **ps**:

```
user@htb[/htb]$ ps -aux | grep ssh

root      846  0.0  0.1  72300  5660 ?        Ss   Mai14   0:00 /usr/sbin/sshd -D
```

También podemos usar **systemctl** para listar todos los servicios:

```
user@htb[/htb]$ systemctl list-units --type=service

UNIT                                LOAD    ACTIVE SUB
DESCRIPTION
accounts-daemon.service            loaded active running
Accounts Service
acpid.service                      loaded active running
ACPI event daemon
apache2.service                    loaded active running
The Apache HTTP Server
apparmor.service                   loaded active exited
AppArmor initialization
apport.service                     loaded active exited
LSB: automatic crash repor
avahi-daemon.service               loaded active running
Avahi mDNS/DNS-SD Stack
bolt.service                       loaded active running
Thunderbolt system service
```

Si un servicio no arranca por un error, podemos ver el problema con **journalctl**:

```
user@htb[/htb]$ journalctl -u ssh.service --no-pager

-- Logs begin at Wed 2020-05-13 17:30:52 CEST, end at Fri 2020-05-15 16:00:14 CEST.
--
Mai 13 20:38:44 inlane systemd[1]: Starting OpenBSD Secure Shell server...
```

```

Mai 13 20:38:44 inlane sshd[2722]: Server listening on 0.0.0.0 port 22.
Mai 13 20:38:44 inlane sshd[2722]: Server listening on :: port 22.
Mai 13 20:38:44 inlane systemd[1]: Started OpenBSD Secure Shell server.
Mai 13 20:39:06 inlane sshd[3939]: Connection closed by 10.22.2.1 port 36444
[preauth]
Mai 13 20:39:27 inlane sshd[3942]: Accepted password for master from 10.22.2.1 port
36452 ssh2
Mai 13 20:39:27 inlane sshd[3942]: pam_unix(sshd:session): session opened for user
master by (uid=0)
Mai 13 20:39:28 inlane sshd[3942]: pam_unix(sshd:session): session closed for user
master
Mai 14 02:04:49 inlane sshd[2722]: Received signal 15; terminating.
Mai 14 02:04:49 inlane systemd[1]: Stopping OpenBSD Secure Shell server...
Mai 14 02:04:49 inlane systemd[1]: Stopped OpenBSD Secure Shell server.
-- Reboot --

```

Matar un proceso

Un proceso puede tener estos estados:

- Corriendo
- Esperando (a un evento o recurso del sistema)
- Parado
- Zombie (parado pero continúa teniendo una entrada en la tabla de procesos)

Los procesos se pueden controlar con **kill**, **pgrep** y **killall**. Para interactuar con un proceso, debemos mandarle una señal. Podemos ver todas las señales con el comando:

```
user@htb[/htb]$ kill -l
```

```

1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD    18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG     24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF    28) SIGWINCH   29) SIGIO      30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX

```

Las más usadas son:

1. SIGHUP: se envía a un proceso cuando la terminal que controla es cerrada
2. SIGINT: se envía cuando un usuario pulsa **Ctrl + C** durante el control de la terminal para interrumpir un proceso
3. SIGQUIT: cuando un usuario pulsa **Ctrl + D** para quitar
4. SIGKILL: mata un proceso inmediatamente sin operaciones de limpieza
5. SIGTERM: termina un programa
6. SIGSTOP: detiene el programa, no se puede retomar
7. SIGTSTP: cuando un usuario pulsa **Ctrl + Z** para suspender un servicio. El usuario lo puede retomar después.

Si un programa se congela, podemos forzar su muerte con:

```
user@htb[/htb]$ kill 9 <PID>
```

Procesos a segundo plano

A veces es necesario para continuar usando la sesión actual para interactuar con el sistema o iniciar otros procesos. Lo podemos hacer con **Ctrl + Z**. Mandamos una señal SIGTSTP al núcleo, que suspende el proceso.

```
user@htb[/htb]$ ping -c 10 www.hackthebox.eu

user@htb[/htb]$ vim tmpfile
[Ctrl + Z]
[2]+  Stopped                  vim tmpfile
```

Podemos ver los procesos en segundo plano con:

```
user@htb[/htb]$ jobs

[1]+  Stopped                  ping -c 10 www.hackthebox.eu
[2]+  Stopped                  vim tmpfile
```

Ctrl + Z suspende el proceso, y no podrá ser ejecutar después. Para seguir corriéndolo en segundo plano, usaremos el comando **bg**:

```
user@htb[/htb]$ bg

user@htb[/htb]$
--- www.hackthebox.eu ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 113482ms
```

```
[ENTER]  
[1]+  Exit 1                  ping -c 10 www.hackthebox.eu
```

Otra opción es indicar automáticamente el proceso con un **&** al final del comando:

```
user@htb[/htb]$ ping -c 10 www.hackthebox.eu &  
  
[1] 10825  
PING www.hackthebox.eu (172.67.1.1) 56(84) bytes of data.
```

Una vez termine el proceso, veremos los resultados:

```
user@htb[/htb]$  
  
--- www.hackthebox.eu ping statistics ---  
10 packets transmitted, 0 received, 100% packet loss, time 9210ms  
  
[ENTER]  
[1]+  Exit 1                  ping -c 10 www.hackthebox.eu
```

Procesos a primer plano

Podemos usar el comando **jobs** para listar todos los procesos en segundo plano. Estos no requieren interacción del usuario, y podemos usar la misma sesión de shell sin esperar a que acaben. Cuando uno termine, se nos notificará en la terminal:

```
user@htb[/htb]$ jobs  
  
[1]+  Running                  ping -c 10 www.hackthebox.eu &
```

Si queremos traer un proceso de segundo a primer plano e interactuar con él de nuevo, usaremos el comando **fg ID**:

```
user@htb[/htb]$ fg 1  
ping -c 10 www.hackthebox.eu  
  
--- www.hackthebox.eu ping statistics ---  
10 packets transmitted, 0 received, 100% packet loss, time 9206ms
```

Ejecutar múltiples comandos

Existen tres posibilidades para ello:

- Punto y coma (;)
- Doble ampersand (&&)
- Pipes (|)

La diferencia radica en el tratamiento de los procesos y depende de si el proceso previo se ha completado con éxito o con errores.

El punto y coma (;) separa los comandos y los ejecuta ignorando los resultados previos de los comandos:

```
user@htb[/htb]$ echo '1'; echo '2'; echo '3'
```

```
1
2
3
```

```
user@htb[/htb]$ echo '1'; ls MISSING_FILE; echo '3'
```

```
1
ls: cannot access 'MISSING_FILE': No such file or directory
3
```

Si usamos el doble ampersand (&&), si hay un error en uno de los comandos, los siguientes no se ejecutarán y todo el proceso se parará:

```
user@htb[/htb]$ echo '1' && ls MISSING_FILE && echo '3'
```

```
1
ls: cannot access 'MISSING_FILE': No such file or directory
```

Los pipes (|) dependen no solo de la operación sin errores del proceso anterior, sino también de los resultados de los procesos anteriores. Hablaremos de ellos en la sección **Descriptores y redirecciones de archivos**.

9 - Trabajando con servicios Web

Para un servidor web Apache, podemos usar los módulos apropiados, los cuales pueden encriptar la comunicación entre navegador y servidor (mod_ssl), usarlo como servidor proxy (mod_proxy) o realizar complejas manipulaciones de los datos de cabecera HTTP (mod_headers) y URLs (mod_rewrite).

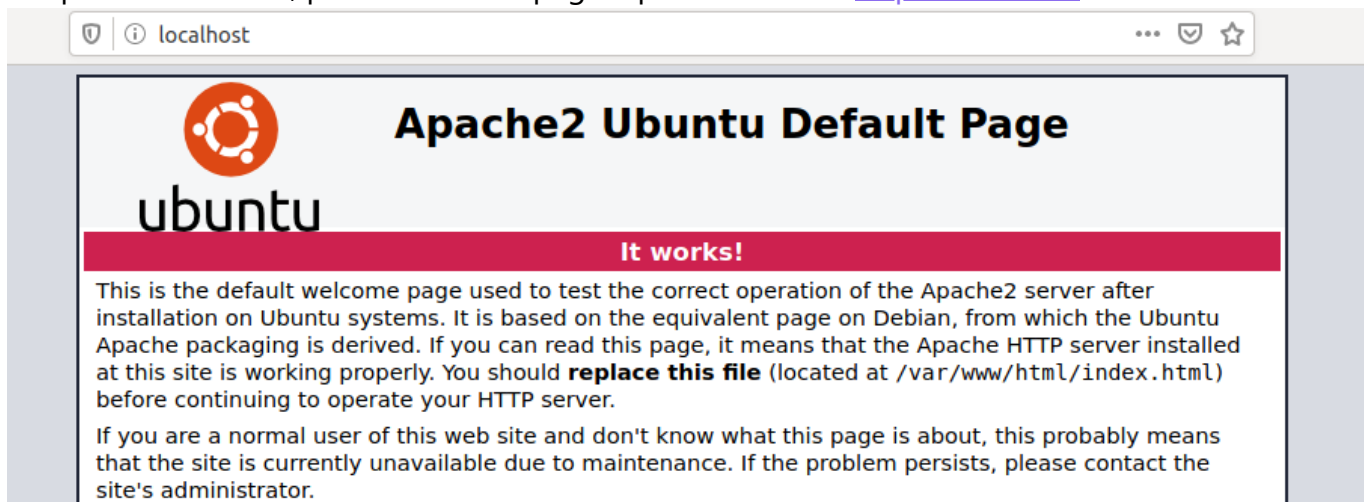
Apache ofrece la posibilidad de crear páginas web de forma dinámica usando lenguajes del lado del servidor. Los más usados: PHP, Perl, Ruby. Aunque hay otros que también se pueden usar para ello: Python, JavaScript, Lua, .NET.

Para instalar el servidor web Apache:

```
user@htb[/htb]$ apt install apache2 -y

Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.
Need to get 95,1 kB of archives.
After this operation, 535 kB of additional disk space will be used.
Get:1 http://de.archive.ubuntu.com/ubuntu bionic-updates/main amd64 apache2 amd64
2.4.29-1ubuntu4.13 [95,1 kB]
Fetched 95,1 kB in 0s (270 kB/s)
<SNIP>
```

Después de iniciarlo, podemos ver la página por defecto en <http://localhost>.



CURL

cURL es una herramienta que permite transferir archivos en protocolos como HTTP, HTTPS, FTP, SFTP, FTPS o SCP. Ofrece la posibilidad de controlar y testear sitios web remotamente. También permite ver las peticiones individuales en la comunicación entre cliente y servidor.

```
user@htb[/htb]$ curl http://localhost

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
...SNIP...
```

Nos permite inspeccionar el código fuente del sitio web y obtener información de él.

Wget

Alternativa a curl. Permite descargar archivos de servidores FTP y HTTP directamente desde la terminal (buen gestor de descargas). La diferencia con curl es que el contenido del sitio web se descarga y almacena localmente.

```
user@htb[/htb]$ wget http://localhost

--2020-05-15 17:43:52--  http://localhost/
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10918 (11K) [text/html]
Saving to: 'index.html'

index.html          100%[=====>]  10,66K
--.-KB/s    in 0s

2020-05-15 17:43:52 (33,0 MB/s) - 'index.html' saved [10918/10918]
```

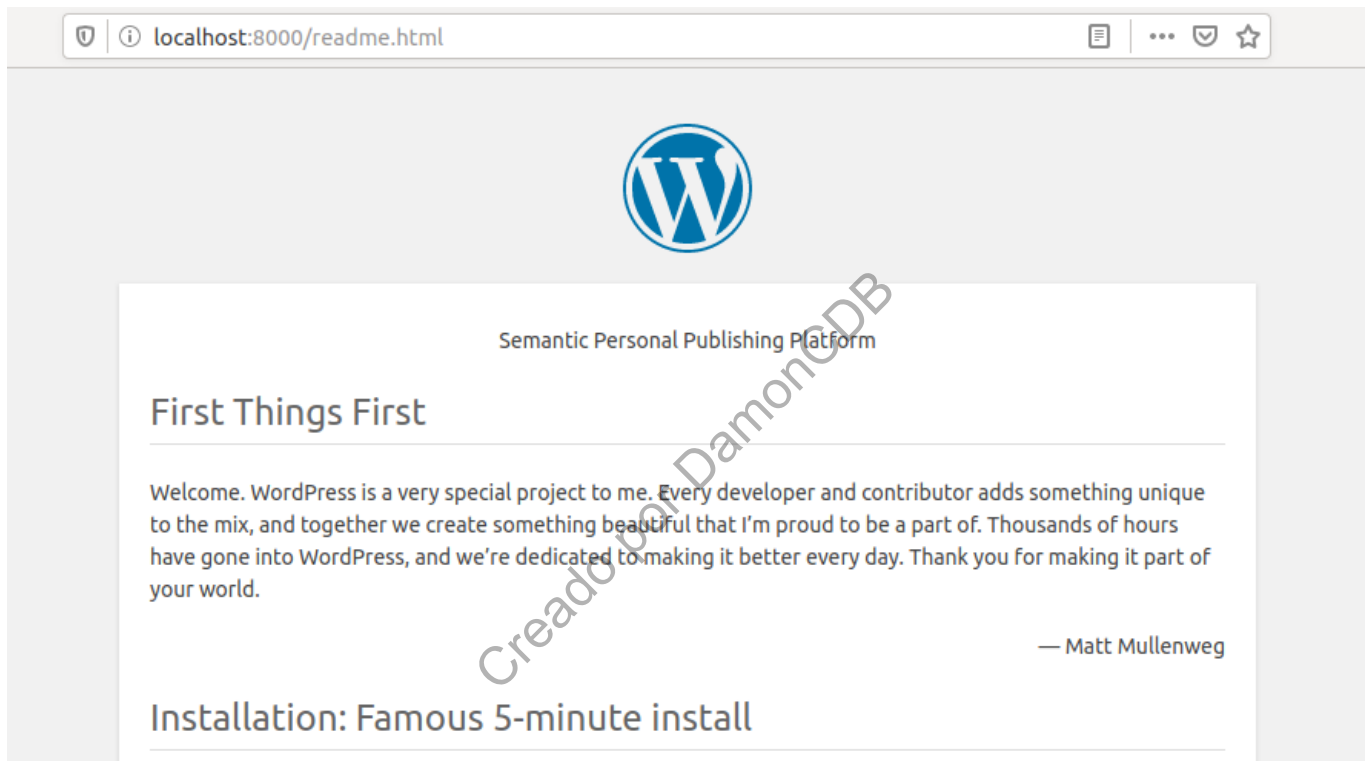
Python 3

En este caso, es en el directorio raíz del servidor web dónde se ejecuta el comando para arrancar el servidor.

En este ejemplo, estamos en un directorio donde se ha instalado WordPress y contiene "readme.html". Veamos si podemos acceder a él mediante Python 3.

```
user@htb[/htb]$ python3 -m http.server
```

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



Aquí podemos ver las peticiones hechas en los eventos del servidor web Python 3:

```
user@htb[/htb]$ python3 -m http.server
```

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
127.0.0.1 - - [15/May/2020 17:56:29] "GET /readme.html HTTP/1.1" 200 -  
127.0.0.1 - - [15/May/2020 17:56:29] "GET /wp-admin/css/install.css?ver=20100228  
HTTP/1.1" 200 -  
127.0.0.1 - - [15/May/2020 17:56:29] "GET /wp-admin/images/wordpress-logo.png  
HTTP/1.1" 200 -  
127.0.0.1 - - [15/May/2020 17:56:29] "GET /wp-admin/images/wordpress-logo.svg?  
ver=20131107 HTTP/1.1" 200 -
```


10 - Navegación

- **pwd**: muestra el directorio en el que estamos

```
cry011t3@htb[~]$ pwd  
  
/home/cry011t3
```

- **ls**: lista archivos y directorios en un directorio

```
cry011t3@htb[~]$ ls  
  
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

```
cry011t3@htb[~]$ ls -l  
  
total 32  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:37 Desktop  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Documents  
drwxr-xr-x 3 cry011t3 cry011t3 4096 Nov 15 03:26 Downloads  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Music  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Pictures  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Public  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Templates  
drwxr-xr-x 2 cry011t3 cry011t3 4096 Nov 13 17:34 Videos
```

Se le puede pasar como parámetro una ruta para ver lo que hay en ella:

```
cry011t3@htb[~]$ ls -l /var/  
  
total 52  
drwxr-xr-x 2 root root 4096 Mai 15 18:54 backups  
drwxr-xr-x 18 root root 4096 Nov 15 16:55 cache  
drwxrwsrwt 2 root whoopsie 4096 Jul 25 2018 crash  
drwxr-xr-x 66 root root 4096 Mai 15 03:08 lib  
drwxrwsr-x 2 root staff 4096 Nov 24 2018 local  
<SNIP>
```

- **cd**: para navegar por los directorios. Podemos ir carpeta por carpeta o directamente a una ruta.

```
cry011t3@htb[~]$ cd /dev/shm
```

```
cry011t3@htb[/dev/shm]$
```

Si estábamos en el directorio home, podemos volver a él con:

```
cry011t3@htb[/dev/shm]$ cd -
```

```
cry011t3@htb[~]$
```

- La shell nos permite autocompletar una ruta pulsando la tecla **TAB**:

```
cry011t3@htb[~]$ cd /dev/s [TAB 2x]
```

```
shm/ snd/
```

- Con **ls -la** vemos los directorios, archivos y archivos ocultos en el directorio que deseemos:

```
cry011t3@htb[/dev/shm]$ ls -la
```

```
total 0
```

```
drwxrwxrwt  2 root root  40 Mai 15 18:31 .
```

```
drwxr-xr-x 17 root root 4000 Mai 14 20:45 ..
```

Los archivos ocultos se representan con un punto delante.

El punto solo representa el directorio actual. Los dos puntos seguidos representan el directorio padre, al que podemos saltar con el comando:

```
cry011t3@htb[/dev/shm]$ cd ..
```

```
cry011t3@htb[/dev]$
```

- **clear**: limpia la pantalla de la terminal.

11 - Trabajando con archivos y directorios

Crear, mover y copiar

Podemos crear un archivo vacío con **touch**, o un directorio vacío con **mkdir**:

```
user@htb[/htb]$ touch info.txt
```

```
user@htb[/htb]$ mkdir Storage
```

Si queremos especificar directorios dentro del directorio, **mkdir** tiene la opción **-p** para añadir directorios padre:

```
user@htb[/htb]$ mkdir -p Storage/local/user/documents
```

Podemos ver la estructura de árbol de un directorio con **tree**:

```
user@htb[/htb]$ tree .  
  
.  
├── info.txt  
└── Storage  
    ├── local  
    │   └── user  
    │       └── documents  
  
4 directories, 1 file
```

También podemos crear archivos en los directorios especificando la ruta. El truco es usar el punto (**.**) para decirle al sistema que queremos empezar a contar desde el directorio actual:

```
user@htb[/htb]$ touch ./Storage/local/user/userinfo.txt
```

```
userB@htb[/htb]$ tree .  
  
.  
├── info.txt  
└── Storage  
    ├── local  
    │   └── user
```

```
├─ documents
└─ userinfo.txt
```

4 directories, 2 files

Podemos mover y renombrar archivos y directorios con **mv**:

```
user@htb[/htb]$ mv <file/directory> <renamed file/directory>
```

Renombrar archivo:

```
user@htb[/htb]$ mv info.txt information.txt
```

Creamos un archivo:

```
user@htb[/htb]$ touch readme.txt
```

Movemos archivos a un directorio específico:

```
user@htb[/htb]$ mv information.txt readme.txt Storage/
```

```
user@htb[/htb]$ tree .
```

```
.
├─ Storage
│   ├── information.txt
│   ├── local
│   │   └─ user
│   │       ├── documents
│   │       └─ userinfo.txt
└─ readme.txt
```

4 directories, 3 files

Si queremos copiar *readme.txt* al directorio *local/*:

```
user@htb[/htb]$ cp Storage/readme.txt Storage/local/
```

```
user@htb[/htb]$ tree .
```

```
.
├─ Storage
│   └─ information.txt
```

```
├─ local
│   ├── readme.txt
│   └── user
│       ├── documents
│       └── userinfo.txt
└─ readme.txt
```

4 directories, 4 files

Creado por DamonCDB

12 - Editar archivos

Los dos principales editores para archivos son **Vi / Vim** y **Nano**. Nano es un poco más sencillo de entender.

Editor Nano

Podemos crear un fichero directamente con Nano especificando el nombre del fichero como parámetro del comando.

```
user@htb[/htb]$ nano notes.txt
```

Esto abrirá el editor con el archivo creado y podremos insertar el texto en él:

```
GNU nano 2.9.3                               notes.txt

Here we can type everything we want and make our notes.

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur
Pos      M-U Undo
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To
Line  M-E Redo
```

El símbolo "^" equivale a la tecla **Ctrl**.

Con **Ctrl + W** realizamos una búsqueda en el texto:

```
GNU nano 2.9.3                               notes.txt

Here we can type everything we want and make our notes.

Search:  notes
^G Get Help      M-C Case Sens  M-B Backwards  M-J FullJstify ^W Beg of Par  ^Y First
Line  ^P PrevHstory
^C Cancel        M-R Regexp     ^R Replace      ^T Go To Line  ^O End of Par  ^V Last
Line  ^N NextHstory
```

Para saltar a la siguiente coincidencia, volvemos a pulsar la misma combinación de teclas:

```
GNU nano 2.9.3                               notes.txt

Here we can type everything we want and make our notes.

Search [we]:
^G Get Help      M-C Case Sens  M-B Backwards  M-J FullJstify ^W Beg of Par  ^Y First
Line  ^P PrevHstory
^C Cancel        M-R Regexp      ^R Replace      ^T Go To Line  ^O End of Par  ^V Last
Line  ^N NextHstory
```

Podemos guardar el archivo con **Ctrl + O** y confirmar el nombre del archivo con **Enter**:

```
GNU nano 2.9.3                               notes.txt

Here we can type everything we want and make our notes.

File Name to Write: notes.txt
^G Get Help      M-C Case Sens  M-B Backwards  M-J FullJstify ^W Beg of Par  ^Y First
Line  ^P PrevHstory
^C Cancel        M-R Regexp      ^R Replace      ^T Go To Line  ^O End of Par  ^V Last
Line  ^N NextHstory
```

Una vez guardado, pulsaremos **Ctrl + X** para salir de Nano.

De vuelta en la shell

Para ver el contenido de un archivo, usamos **cat**:

```
user@htb[/htb]$ cat notes.txt

Here we can type everything we want and make our notes.
```

Hay archivos en Linux que juegan un papel fundamental, entre ellos */etc/passwd*.

VIM

Editor de código abierto para todo tipo de textos ASCII, como Nano. Es un clon del anterior, Vi. Vim proporciona una interfaz a programas externos como **grep**, **awk** o **sed**. Menos propenso a errores.

Vim sigue los principios de Unix.

```
user@htb[/htb]$ vim
```

Vim puede distinguir entre texto y entrada de comandos. Ofrece seis modos que harán nuestro trabajo más sencillo:

- Normal: todas las entradas se consideran como editor de comandos. Después de iniciarlo, solemos estar en modo normal.
- Insert: con algunas excepciones, todos los caracteres introducidos se insertan en el buffer.
- Visual: se usa para marcar una parte contigua del texto, que se resaltará visualmente. La zona resaltada se puede editar de varias formas, borrando, copiando o reemplazándola.
- Command: permite introducir comandos de una sola línea en la parte inferior del editor. Se puede usar para ordenar, reemplazar secciones de texto o borrarlas.
- Replace: el texto introducido sobrescribirá los caracteres de texto hasta que no haya caracteres antiguos en la posición del cursor. Después se añadirá el nuevo texto introducido.

Cuando tenemos abierto Vim, podemos escribir `":q"` para cerrarlo.

```
1 $  
~  
~  
~ VIM - Vi IMproved  
~  
~ version 8.0.1453  
~ by Bram Moolenaar et al.  
~ Modified by pkg-vim-maintainers@lists.alioth.debian.org  
~ Vim is open source and freely distributable
```


13 - Encuentra archivos y directorios

Importancia de la búsqueda

Una vez tenemos acceso a un sistema Linux, es crucial encontrar los archivos de configuración, scripts del administrador y otros archivos y directorios.

Which

Esta herramienta devuelve la ruta al archivo o enlace que debería ser ejecutado. Permite determinar si programas específicos como **cURL**, **netcat**, **wget**, **python**, están en el sistema:

```
user@htb[/htb]$ which python

/usr/bin/python
```

Si el programa no existe, no se mostrarán resultados.

Find

Además de la función de encontrar ficheros y carpetas, también tiene la función de filtrar los resultados por parámetros como su tamaño o fecha:

```
user@htb[/htb]$ find <location> <options>
```

```
user@htb[/htb]$ find / -type f -name *.conf -user root -size +20k -newermt 2020-03-03 -exec ls -al {} \; 2>/dev/null
```

```
-rw-r--r-- 1 root root 136392 Apr 25 20:29 /usr/src/linux-headers-5.5.0-1parrot1-  
amd64/include/config/auto.conf  
-rw-r--r-- 1 root root 82290 Apr 25 20:29 /usr/src/linux-headers-5.5.0-1parrot1-  
amd64/include/config/tristate.conf  
-rw-r--r-- 1 root root 95813 May 7 14:33 /usr/share/metasploit-  
framework/data/jtr/repeats32.conf  
-rw-r--r-- 1 root root 60346 May 7 14:33 /usr/share/metasploit-  
framework/data/jtr/dynamic.conf  
-rw-r--r-- 1 root root 96249 May 7 14:33 /usr/share/metasploit-  
framework/data/jtr/dumb32.conf  
-rw-r--r-- 1 root root 54755 May 7 14:33 /usr/share/metasploit-  
framework/data/jtr/repeats16.conf  
-rw-r--r-- 1 root root 22635 May 7 14:33 /usr/share/metasploit-
```

```
framework/data/jtr/korelogic.conf
-rwxr-xr-x 1 root root 108534 May  7 14:33 /usr/share/metasploit-
framework/data/jtr/john.conf
-rw-r--r-- 1 root root 55285 May  7 14:33 /usr/share/metasploit-
framework/data/jtr/dumb16.conf
-rw-r--r-- 1 root root 21254 May  2 11:59
/usr/share/doc/sqlmap/examples/sqlmap.conf
-rw-r--r-- 1 root root 25086 Mar  4 22:04 /etc/dnsmasq.conf
-rw-r--r-- 1 root root 21254 May  2 11:59 /etc/sqlmap/sqlmap.conf
```

Veamos las opciones del comando:

- `-type f`: define el tipo de objeto buscado, en este caso un archivo (`f`)
- `-name asterisco.conf`: con name indicamos el nombre del archivo que buscamos. el asterisco.conf busca todos los archivos de extensión conf
- `-user root`: filtra todos los archivos cuyo propietario es el usuario root
- `-size +20k`: filtra los archivos localizados y especifica que solo queremos los mayores a 20 KiB
- `-newermt 2020-03-03`: solo se mostrarán los ficheros más nuevos que la fecha especificada
- `-exec ls -al {} \;`: ejecuta el comando indicado, usando las llaves como lugares en los que almacenar el resultado. La barra invertida escapa el siguiente carácter para que no sea interpretado por la shell.
- `2>/dev/null`: redirección de STDERR al dispositivo nulo. Asegura que no haya errores a mostrar en la terminal. **No** debe ser una opción del comando 'find'

Locate

Este comando nos ofrece una forma más rápida de buscar en el sistema. Funciona con una base de datos local que contiene toda la info acerca de los archivos y carpetas existentes. Podemos actualizar la base de datos con:

```
user@htb[/htb]$ sudo updatedb
```

Si ahora buscamos todos los archivos ".conf", verás que el resultado se muestra más rápido:

```
user@htb[/htb]$ locate *.conf

/etc/GeoIP.conf
/etc/NetworkManager/NetworkManager.conf
/etc/UPower/UPower.conf
/etc/adduser.conf
<SNIP>
```

Esta herramienta no tiene filtros que podamos utilizar. Su uso dependerá de qué estamos buscando.

Creado por DamonCDB

14 - Descriptores de archivos y redirecciones

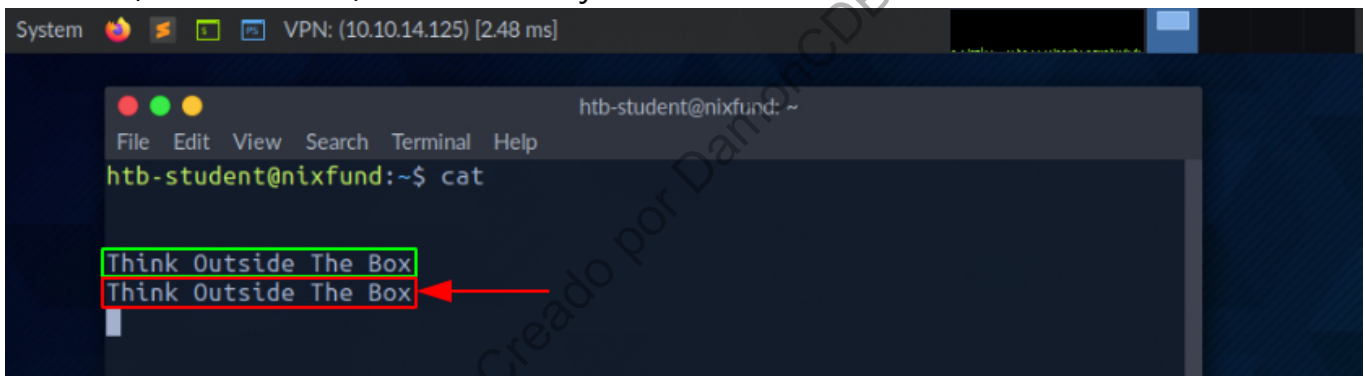
Descriptor de archivo: indicador de conexión mantenido por el núcleo para realizar operaciones de I/O.

Por defecto, los tres primeros descriptores en Linux son:

1. Data Stream para Input -> **STDIN - 0**
2. Data Stream para Output -> **STDOUT - 1**
3. Data Stream para Output que se refiere a un error ocurrido -> **STDERR - 2**

STDIN y STDOUT

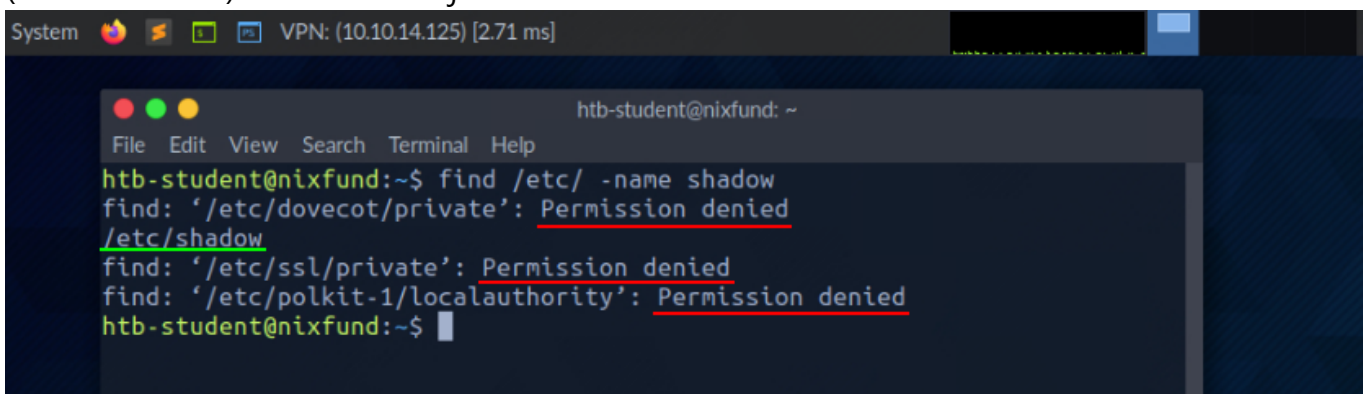
Cuando ejecutamos **cat**, le damos al programa la entrada estándar (**STDIN - FD 0**), marcada en verde. Tan pronto como confirmamos nuestra entrada con **ENTER**, nos devuelve la salida por terminal (**STDOUT - FD 1**), marcada en rojo:



```
System [2.48 ms]
htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ cat
Think Outside The Box
Think Outside The Box
```

STDOUT y STDERR

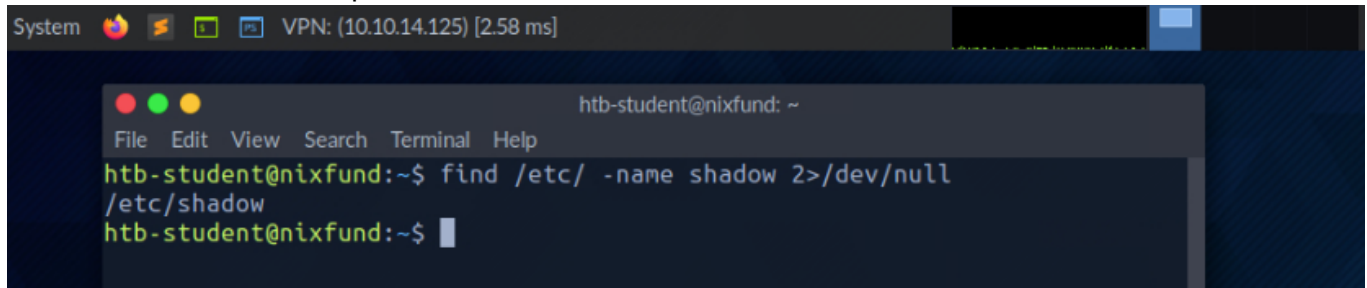
En este ejemplo usando el comando **find**, vemos la salida marcada en verde y el error estándar (**STDERR - FD 2**) marcado en rojo:



```
System [2.71 ms]
htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ find /etc/ -name shadow
find: '/etc/dovecot/private': Permission denied
/etc/shadow
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
htb-student@nixfund:~$
```

Podemos redireccionar el descriptor del error a **"/dev/null"**. De esta forma redireccionamos los

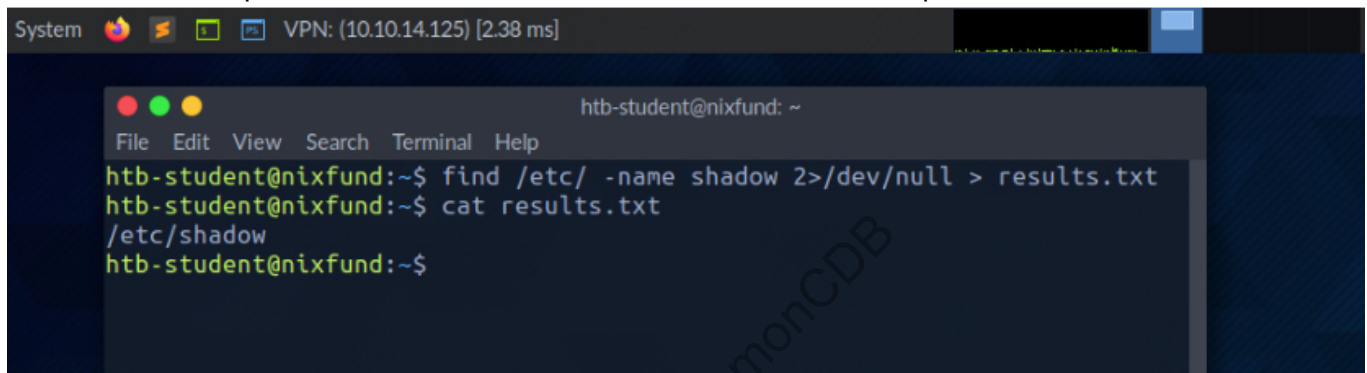
errores resultantes al dispositivo nulo, el cual descarta esos datos:



```
System [Icons] VPN: (10.10.14.125) [2.58 ms]
htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ find /etc/ -name shadow 2>/dev/null
/etc/shadow
htb-student@nixfund:~$
```

Redireccionar STDOUT a un archivo

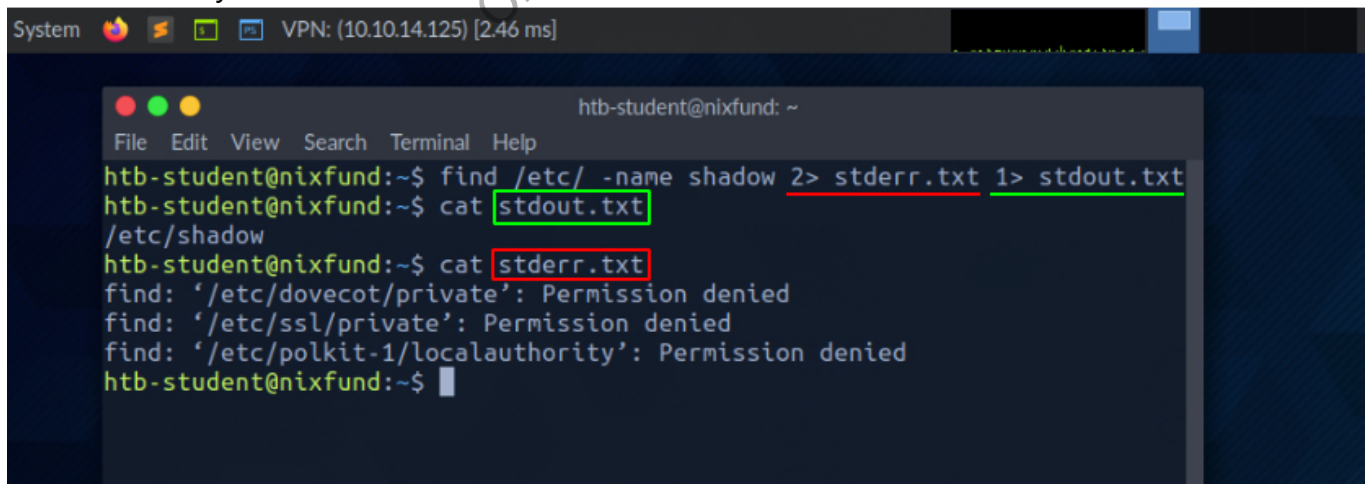
Podemos ver que los errores ahora no aparecen. El único resultado que aparece es la salida estándar, la cual podemos redireccionar a un archivo **results.txt** que contendrá solo esta salida:



```
System [Icons] VPN: (10.10.14.125) [2.38 ms]
htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ find /etc/ -name shadow 2>/dev/null > results.txt
htb-student@nixfund:~$ cat results.txt
/etc/shadow
htb-student@nixfund:~$
```

Redireccionar STDOUT y STDERR a archivos separados

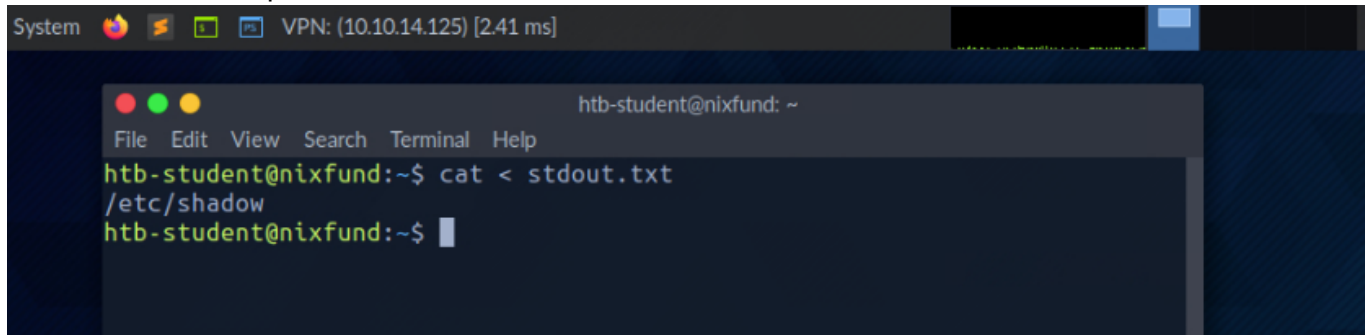
El símbolo > sirve para redireccionar el resultado. En este caso se muestra cómo redireccionar el error estándar y la salida estándar a archivos distintos:



```
System [Icons] VPN: (10.10.14.125) [2.46 ms]
htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ find /etc/ -name shadow 2> stderr.txt 1> stdout.txt
htb-student@nixfund:~$ cat stdout.txt
/etc/shadow
htb-student@nixfund:~$ cat stderr.txt
find: '/etc/dovecot/private': Permission denied
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
htb-student@nixfund:~$
```

Redireccionar STDIN

El símbolo < sirve para redireccionar un archivo a la entrada estándar:

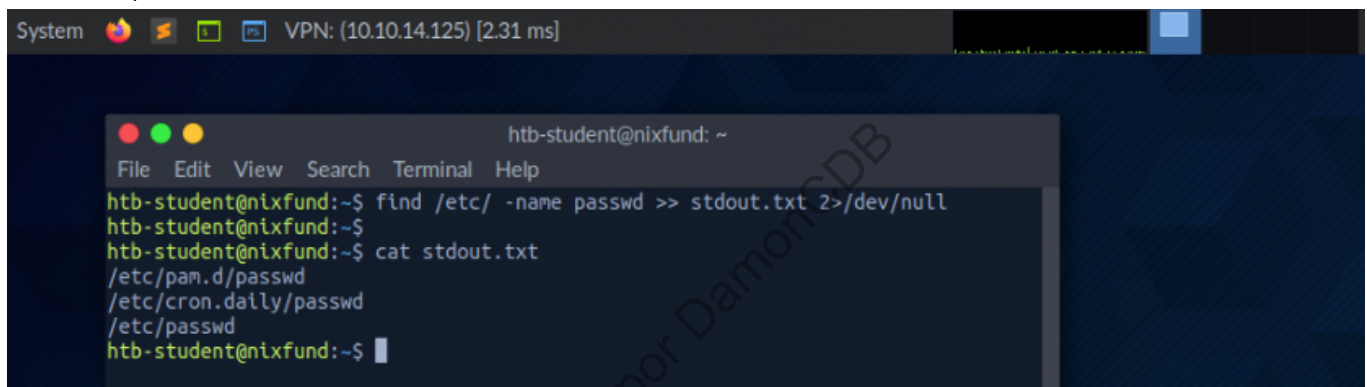


```
System [Icons] VPN: (10.10.14.125) [2.41 ms]

htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ cat < stdout.txt
/etc/shadow
htb-student@nixfund:~$
```

Redireccionar STDOUT y añadirlo a un archivo

Cuando usamos > para redireccionar la salida, se crea un archivo nuevo. Si el archivo ya existe, se sobrescribirá sin pedir confirmación. Si queremos que la salida se agregue a un archivo ya existente, usaremos >>:

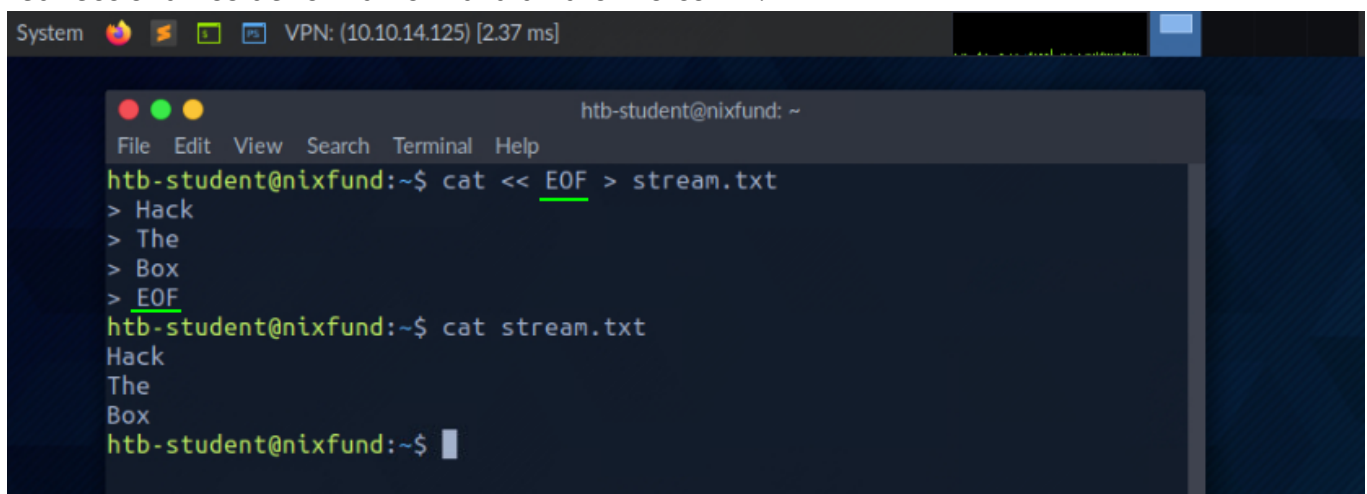


```
System [Icons] VPN: (10.10.14.125) [2.31 ms]

htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ find /etc/ -name passwd >> stdout.txt 2>/dev/null
htb-student@nixfund:~$ cat stdout.txt
/etc/pam.d/passwd
/etc/cron.daily/passwd
/etc/passwd
htb-student@nixfund:~$
```

Redireccionar el flujo STDIN a un archivo

Usando << añadiremos nuestra entrada estándar a un flujo. Podemos usar la función **End-Of-File (EOF)** de los archivos de sistema Linux, el cual define el final de la entrada. Después lo redireccionamos de forma normal a un archivo con >:

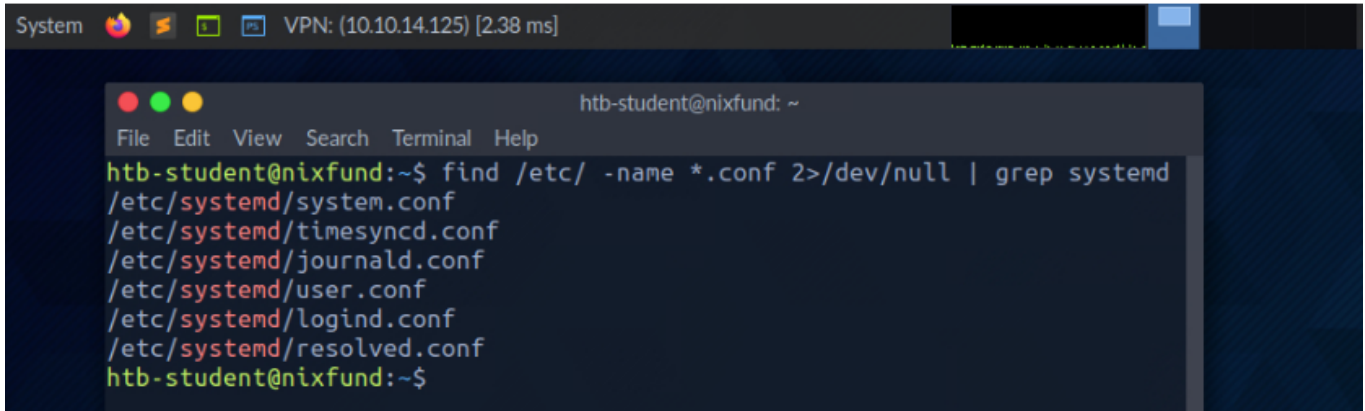


```
System [Icons] VPN: (10.10.14.125) [2.37 ms]

htb-student@nixfund: ~
File Edit View Search Terminal Help
htb-student@nixfund:~$ cat << EOF > stream.txt
> Hack
> The
> Box
> EOF
htb-student@nixfund:~$ cat stream.txt
Hack
The
Box
htb-student@nixfund:~$
```

Pipes

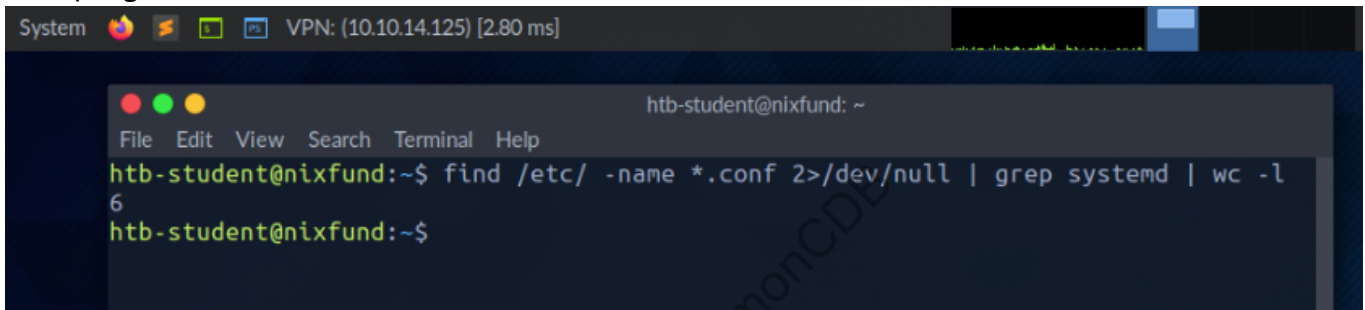
Otra forma de redireccionar STDOUT es mediante el uso de pipes (|), muy útiles cuando queremos usar la salida de un programa como entrada de otro (por ejemplo con **grep** o **find**):



A terminal window titled 'htb-student@nixfund: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command `find /etc/ -name *.conf 2>/dev/null | grep systemd` is entered. The output lists six files: `/etc/systemd/system.conf`, `/etc/systemd/timesyncd.conf`, `/etc/systemd/journald.conf`, `/etc/systemd/user.conf`, `/etc/systemd/logind.conf`, and `/etc/systemd/resolved.conf`. The prompt returns to `htb-student@nixfund:~$`.

```
htb-student@nixfund:~$ find /etc/ -name *.conf 2>/dev/null | grep systemd
/etc/systemd/system.conf
/etc/systemd/timesyncd.conf
/etc/systemd/journald.conf
/etc/systemd/user.conf
/etc/systemd/logind.conf
/etc/systemd/resolved.conf
htb-student@nixfund:~$
```

Las redirecciones no solo funcionan una vez. Podemos usar el resultado para redireccionarlo a otro programa:



A terminal window titled 'htb-student@nixfund: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command `find /etc/ -name *.conf 2>/dev/null | grep systemd | wc -l` is entered. The output is the number `6`. The prompt returns to `htb-student@nixfund:~$`.

```
htb-student@nixfund:~$ find /etc/ -name *.conf 2>/dev/null | grep systemd | wc -l
6
htb-student@nixfund:~$
```

Creado por DamonCDL

15 - Filtrar contenidos

Para leer los archivos no necesitamos obligatoriamente un editor. Tenemos las herramientas **more** y **less**. Son paginadores que nos permiten desplazarnos por el archivo en una vista interactiva.

More

```
user@htb[/htb]$ more /etc/passwd
```

Después de leer el contenido usando **cat** y redireccionarlo a **more**, automáticamente nos situaremos al principio del archivo:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
<SNIP>
--More--
```

Con la tecla **Q** podemos salir del paginador. La salida se mantendrá en la terminal.

Less

Se comporta de una forma similar a **more**:

```
user@htb[/htb]$ less /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
<SNIP>
:
```

Al cerrar con **Q**, vemos que, a diferencia de **more**, la salida no se mantiene en la terminal.

Head

Si queremos ver solo las primeras líneas de un archivo, usaremos el comando **head**, el cual mostrará las diez primeras líneas, si no es especifica lo contrario:

```
user@htb[/htb]$ head /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Tail

Si solo queremos ver las últimas diez líneas, usaremos **tail**:

```
user@htb[/htb]$ tail /etc/passwd

miredo:x:115:65534:./var/run/miredo:/usr/sbin/nologin
usbmux:x:116:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:117:119:RealtimeKit,,,:/proc:/usr/sbin/nologin
nm-openvpn:x:118:120:NetworkManager
OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
nm-openconnect:x:119:121:NetworkManager OpenConnect
plugin,,,:/var/lib/NetworkManager:/usr/sbin/nologin
pulse:x:120:122:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
beef-xss:x:121:124:./var/lib/beef-xss:/usr/sbin/nologin
lightdm:x:122:125:Light Display Manager:/var/lib/lightdm:/bin/false
do-agent:x:998:998:./home/do-agent:/bin/false
user6:x:1000:1000:,,,:/home/user6:/bin/bash
```

Sort

Dependiendo de los resultados, en ocasiones estos no aparecen ordenados. Si necesitamos ordenarlos alfabética o numéricamente, usaremos **sort**:

```
user@htb[/htb]$ cat /etc/passwd | sort

_apt:x:104:65534:./nonexistent:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

```
bin:x:2:2:bin:/bin:/usr/sbin/nologin
cry011t3:x:1001:1001:./home/cry011t3:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
dovecot:x:114:117:Dovecot mail server,,,:/usr/lib/dovecot:/usr/sbin/nologin
dovenull:x:115:118:Dovecot login user,,,:/nonexistent:/usr/sbin/nologin
ftp:x:113:65534:./srv/ftp:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
htb-student:x:1002:1002:./home/htb-student:/bin/bash
<SNIP>
```

Grep

Si solo queremos buscar un patrón específico en los resultados, usaremos **grep**:

```
user@htb[/htb]$ cat /etc/passwd | grep "/bin/bash"

root:x:0:0:root:/root:/bin/bash
mrb3n:x:1000:1000:mrb3n:/home/mrb3n:/bin/bash
cry011t3:x:1001:1001:./home/cry011t3:/bin/bash
htb-student:x:1002:1002:./home/htb-student:/bin/bash
```

Otra posibilidad es excluir resultados específicos, con la opción **"-v"** de **grep**:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\\|nologin"

root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
postgres:x:111:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user6:x:1000:1000:,,,:/home/user6:/bin/bash
```

Cut

Algunos resultados pueden estar separados por delimitadores. Con **cut** podemos indicar el delimitador con la opción **"-d"** y definir con **"-f"** la posición en la línea en la que queremos la salida:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\\|nologin" | cut -d":" -f1

root
sync
mrb3n
```

```
cry011t3
htb-student
```

Tr

Podemos reemplazar ciertos caracteres de una línea con caracteres definidos por nosotros con la herramienta **tr**. Como primera opción definimos el carácter a reemplazar, y como segunda opción definimos el carácter con el que reemplazarlo:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\|nologin" | tr ":" " "

root x 0 0 root /root /bin/bash
sync x 4 65534 sync /bin /bin/sync
mrb3n x 1000 1000 mrb3n /home/mrb3n /bin/bash
cry011t3 x 1001 1001 /home/cry011t3 /bin/bash
htb-student x 1002 1002 /home/htb-student /bin/bash
```

Column

Con esta herramienta se muestran los resultados en forma tabular usando la opción **"-t"**:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\|nologin" | tr ":" " " | column -t

root      x  0      0      root      /root      /bin/bash
sync      x  4      65534  sync      /bin        /bin/sync
mrb3n     x  1000    1000    mrb3n     /home/mrb3n /bin/bash
cry011t3  x  1001    1001    /home/cry011t3 /bin/bash
htb-student x 1002    1002    /home/htb-student /bin/bash
```

Awk

Sirve para mantener una salida lo más simple posible para ordenar sus resultados. Con **(\$1)** muestra el primero y con **(\$NF)** muestra el último resultado de la línea:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\|nologin" | tr ":" " " | awk
'{print $1, $NF}'

root /bin/bash
sync /bin/sync
mrb3n /bin/bash
cry011t3 /bin/bash
htb-student /bin/bash
```

Sed

Una de sus principales funciones es sustituir texto. Usaremos **sed** con patrones formados por expresiones regulares (regex) y los reemplazaremos con otro patrón definido previamente. La opción "s" al principio significa el comando a sustituir. Después especificamos el patrón a reemplazar. Después de la barra ("/"), introducimos el patrón que usaremos para reemplazar. Finalmente usamos la opción "g", la cual indica que reemplacemos todas las coincidencias:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\|nologin" | tr ":" " " | awk  
'{print $1, $NF}' | sed 's/bin/HTB/g'  
  
root /HTB/bash  
sync /HTB/sync  
mrb3n /HTB/bash  
cry011t3 /HTB/bash  
htb-student /HTB/bash
```

Wc

En ocasiones será útil saber cuántas coincidencias tenemos. Para evitar tener que contarlas manualmente, usaremos **wc**. Con la opción "-l" especificamos que solo queremos contar las líneas:

```
user@htb[/htb]$ cat /etc/passwd | grep -v "false\|nologin" | tr ":" " " | awk  
'{print $1, $NF}' | wc -l
```

5

Práctica

Tómate tu tiempo para experimentar con estas herramientas. Echa un vistazo a sus páginas [man](#) o a sus opciones de ayuda. La mejor forma de familiarizarse con ellas es practicar.

16 - Gestión de permisos

Se puede asignar permisos a usuarios y a grupos.

Cada usuario puede ser miembro de distintos grupos, y cada membresía otorga al usuario permisos específicos.

Los permisos para usuarios y grupos que definen a un archivo son definidos por sus respectivos propietarios.

Cuando un usuario quiere acceder al contenido de un directorio en Linux, primero debe ir a ese directorio, lo cual requiere que tenga permisos para ejecutar acciones en él. Sin ellos, se le denegará el permiso.

```
cry011t3@htb[/htb]$ ls -l

drw-rw-r-- 3 cry011t3 cry011t3  4096 Jan 12 12:30 scripts

cry011t3@htb[/htb]$ ls -al mydirectory/

ls: cannot access 'mydirectory/script.sh': Permission denied
ls: cannot access 'mydirectory/..': Permission denied
ls: cannot access 'mydirectory/subdirectory': Permission denied
ls: cannot access 'mydirectory/.': Permission denied
total 0
d????????? ? ? ? ?      ? .
d????????? ? ? ? ?      ? ..
-????????? ? ? ? ?      ? script.sh
d????????? ? ? ? ?      ? subdirectory
```

Los archivos de ejecución en un directorio solo permiten entrar en él y acceder al contenido del directorio, no modificar archivos internos.

Para ello, el usuario tiene que tener permisos de ejecución en el archivo en concreto que quiere ejecutar. Para modificar el contenido de un directorio, el usuario necesita permisos de escritura en el directorio.

El sistema de permisos en Linux se basa en el sistema octal. Hay tres tipos de permisos:

- (r) - Lectura
- (w) - Escritura

- Estos permisos pueden ser ajustados por el propietario, grupo y otros usuarios que tengan los permisos correspondientes:

```

cry011t3@htb[/htb]$ ls -l /etc/passwd

-rwx rw- r--  1 root root 1641 May  4 23:42 /etc/passwd
- --- --- --- | | | | |
| | | | | | | | | Date
| | | | | | | | | File Size
| | | | | | | | | Group
| | | | | | | | | User
| | | | | | | | | Number of hard links
| | | | | | | | | Permission of others (read)
| | | | | | | | | Permissions of the group (read, write)
| | | | | | | | | Permissions of the owner (read, write, execute)
| | | | | | | | | File type (- = File, d = Directory, l = Link, ... )

```

Cambiar permisos

chmod -> cambia los permisos indicando el usuario o grupo (**u** - propietario, **g** - Grupo, **o** - otros, **a** - Todos los usuarios) y añadiendo un + o un - para agregar o quitar los permisos indicados anteriormente:

```
cry011t3@htb[/htb]$ ls -l shell
```

```
-rwxr-x--x  1 cry011t3 htbteam 0 May  4 22:12 shell
```

```
cry011t3@htb[/htb]$ chmod a+r shell && ls -l shell
```

```
-rwxr-xr-x  1 cry011t3 htbteam 0 May  4 22:12 shell
```

También podemos asignar permisos de lectura para todos usando la notación octal:

```
cry011t3@htb[/htb]$ chmod 754 shell && ls -l shell
```

```
-rwxr-xr--  1 cry011t3 htbteam 0 May  4 22:12 shell
```

- Representaciones asociadas con los distintos permisos para entender mejor cómo asignarlos:

```
Binary Notation:      4 2 1 | 4 2 1 | 4 2 1
-----
```

Binary Representation:	1 1 1		1 0 1		1 0 0

Octal Value:	7		5		4

Permission Representation:	r w x		r - x		r - -

Cambiar propietario

Para ello usamos **chown**:

```
cry011t3@htb[/htb]$ chown <user>:<group> <file/directory>
```

```
cry011t3@htb[/htb]$ chown root:root shell && ls -l shell
```

```
-rwxr-xr--  1 root root 0 May  4 22:12 shell
```

SUID y GUID

También se pueden configurar permisos especiales para archivos mediante los bits **Set User ID (SUID)** y el **Set Group ID (GUID)**. Estos permiten a los usuarios correr programas con los derechos de otro usuario. Ello se indica mediante la letra "s" en lugar de la "x". Cuando se ejecuta dicho programa, estamos usando el SUID/GUID del propietario del archivo.

En ocasiones los administradores no están familiarizados con ello y asignan estos bits a cualquiera, lo cual conlleva un gran riesgo de seguridad.

Más información sobre esto y otras aplicaciones se puede encontrar en [GTFObins](https://gtfobins.github.io/).

Sticky Bit

Es un tipo de permisos que se puede asignar a los directorios. Proporciona una capa extra de seguridad controlando la eliminación y renombramiento de archivos en el directorio. Normalmente se usa en directorios compartidos por múltiples usuarios para prevenir accidentes.

Un ejemplo típico es configurar así los directorios de usuarios del directorio **home**. Configurar el sticky bit en un directorio asegura que solo el propietario del archivo, del directorio o el usuario root pueden cambiar los archivos de su interior.

Cuando está configurado, se representa mediante la letra "t" en la lista de permisos del directorio.


```
cry011t3@htb[/htb]$ ls -l
```

```
drw-rw-r-t 3 cry011t3 cry011t3  4096 Jan 12 12:30 scripts  
drw-rw-r-T 3 cry011t3 cry011t3  4096 Jan 12 12:32 reports
```

Si la T es mayúscula significa que ningún otro usuario tiene archivos de ejecución en el directorio, por lo que tampoco puede ver el contenido del mismo ni ejecutar programas en él.

Creado por DamonCDB

17 - Atajos

Autocompletar

La tecla **TAB** inicia el autocompletado de comandos y directorios en **STDIN**

Movimiento del cursor

CTRL + A: mueve el cursor al principio de la línea actual

CTRL + E: mueve el cursor al final de la línea actual

CTRL + <- o ->: salta al principio de la palabra actual/previa

ALT + B/F: salta a la palabra anterior/siguiente

Borrar la línea actual

CTRL + U: borra todo desde la posición del cursor hasta el principio de la línea

CTRL + K: borra todo desde la posición del cursor hasta el final de la línea

CTRL + W: borra la palabra que precede a la posición del cursor

Pegar contenido eliminado

CTRL + Y: pega el texto o palabra borrado

Finalizar tarea

CTRL + C: finaliza la tarea/proceso actual enviando una señal **SIGINT**. El proceso finalizará sin pedirnos confirmación

Fin de archivo (EOF)

CTRL + D: cerrar la entrada **STDIN** se conoce como EOF o Transmisión EOF

Limpiar terminal

CTRL + L: limpia la terminal

Mandar proceso a segundo plano

CTRL + Z: suspende el proceso actual con una señal **SIGTSTP**

Buscar por el historial de comandos

CTRL + R: busca a lo largo del historial de comandos el comando que le indiquemos
Flechas arriba y abajo: ir al comando previo/siguiente en el historial

Cambiar entre aplicaciones

ALT + TAB: cambia entre aplicaciones abiertas

Zoom

CTRL + +: acercar

CTRL + -: alejar

Creado por DamonCDB

18 - Seguridad en Linux

Todos los sistemas tienen inherente un riesgo de intrusión. Los sistemas Linux son menos proclives a que les afecten los virus que afectan a Windows y no presentan una superficie de ataque tan amplia como Active Directory. Aún así, es necesario tener ciertos fundamentos a la hora de securizar Linux:

- Mantener actualizados los paquetes del sistema operativo:

```
user@htb[/htb]$ apt update && apt dist-upgrade
```

- Podemos usar **iptables** para restringir el tráfico desde y hacia nuestro host.
 - Si tenemos abierto **SSH**, debemos deshabilitar el login con contraseña y el logging a usuario root mediante **SSH**.
 - Es conveniente evitar loggarse y administrar el sistema como root.
 - El acceso de los usuarios debe seguir el principio de privilegios mínimos.
 - **fail2ban**. Esta herramienta cuenta la cantidad de intentos de inicio de sesión fallidos, y si el usuario ha alcanzado el máximo de intentos permitido, es baneado.
 - Es importante auditar de cuando en cuando el sistema para asegurarse de que no hay problemas que puedan facilitar la escalada de privilegios (desactualización del kernel, problemas en permisos a los usuarios, archivos editables y servicios o tareas programas mal configuradas).
 - **SELinux** y **AppArmor** son módulos de seguridad que se pueden usar para políticas de seguridad de control de accesos. **SELinux** proporciona controles de acceso granulares.
 - **Snort**, **chkrootkit**, **rkhunter**, **Lynis** y otros pueden contribuir a la seguridad de nuestro Linux.
 - Eliminar o deshabilitar todo software o servicio innecesario.
 - Eliminar todos los servicios que recaigan en mecanismos de autenticación sin cifrar.
 - Asegurarse de que NTP está habilitado y Syslog está corriendo.
 - Cada usuario ha de tener su propia cuenta.
 - Reforzar el uso de contraseñas robustas.
 - Configurar caducidad de contraseñas y restringir el uso de contraseñas antiguas.
 - Bloquear cuentas de usuario después de intentos de login fallidos.
 - Deshabilitar todos los binarios SUID/SGID no buscados.
- Hay más pasos a seguir, la seguridad es un proceso, no un producto.
Cuanto más familiarizado esté el administrador con el sistema, mejor podrá securizarlo.

TCP Wrappers

TCP Wrapper es un mecanismo de seguridad usado en sistemas Linux que permite al administrador del sistema controlar qué servicios tienen acceso permitido al sistema. Puede bloquear servicios o IPs.

Sus archivos de configuración son:

- `/etc/hosts.allow`
- `/etc/hosts.deny`

El primero especifica los permitidos y el segundo los prohibidos.

Se pueden editar para añadir reglas específicas.

`/etc/hosts.allow`

```
user@htb[/htb]$ cat /etc/hosts.allow

# Allow access to SSH from the local network
sshd : 10.129.14.0/24

# Allow access to FTP from a specific host
ftpd : 10.129.14.10

# Allow access to Telnet from any host in the inlanefreight.local domain
telnetd : .inlanefreight.local
```

`/etc/hosts.deny`

```
user@htb[/htb]$ cat /etc/hosts.deny

# Deny access to all services from any host in the inlanefreight.com domain
ALL : .inlanefreight.com

# Deny access to SSH from a specific host
sshd : 10.129.22.22

# Deny access to FTP from hosts with IP addresses in the range of 10.129.22.0 to 10.129.22.255
ftpd : 10.129.22.0/24
```

Es importante recordar que el orden de las reglas es importante, ya que se aplican en ese orden. Los TCPWrappers no son un sustituto del firewall, solo pueden controlar el acceso a los servicios, no a los puertos.