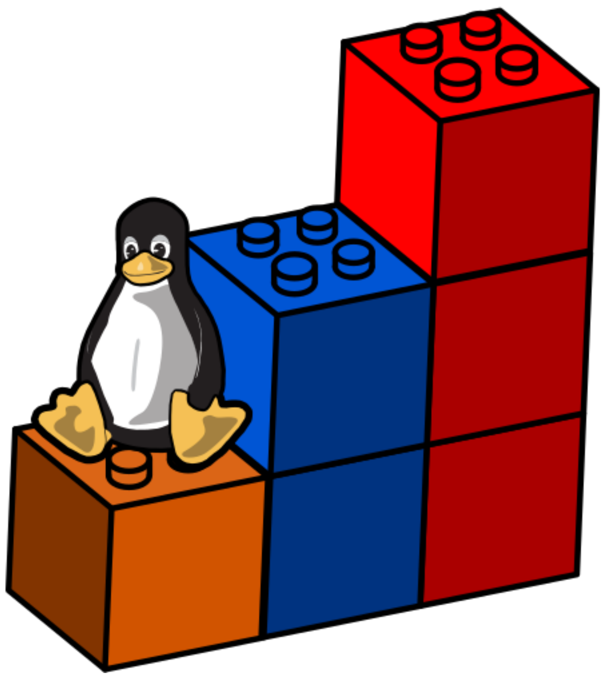
58



Linux PrivEsc

Start AttackBox | Show Split View | Awards | Help

Learn the fundamentals of Linux privilege escalation. From enumeration to exploitation, get hands-on with over 8 different privilege escalation techniques.

×

Chart | Scoreboard | Discuss | Writeups | More

Difficulty: Medium

Active Machine Information

Loading... Loading... Loading... Loading...

6%

Task 1 Introduction

Task 2 What is Privilege Escalation?

Task 3 Enumeration

Start Machine

Note: Launch the target machine attached to this task to follow along.
You can launch the target machine and access it directly from your browser.
Alternatively, you can access it over SSH with the low-privilege user credentials below:

Username: karen
Password: Password1

Enumeration is the first step you have to take once you gain access to any system. You may have accessed the system by exploiting a critical vulnerability that resulted in root-level access or just found a way to send commands using a low privileged account. Penetration testing engagements, unlike CTF machines, don't end once you gain access to a specific system or user privilege level. As you will see, enumeration is as important during the post-compromise phase as it is before.

hostname

The `hostname` command will return the hostname of the target machine. Although this value can easily be changed or have a relatively meaningless string (e.g. Ubuntu-3487340239), in some cases, it can provide information about the target system's role within the corporate network (e.g. SQL-PROD-01 for a production SQL server).

uname -a

Will print system information giving us additional detail about the kernel used by the system. This will be useful when searching for any potential kernel vulnerabilities that could lead to privilege escalation.

/proc/version

The proc filesystem (procfs) provides information about the target system processes. You will find proc on many different Linux flavours, making it an essential tool to have in your arsenal.
Looking at `/proc/version` may give you information on the kernel version and additional data such as whether a compiler (e.g. GCC) is installed.

/etc/issue

Systems can also be identified by looking at the `/etc/issue` file. This file usually contains some information about the operating system but can easily be customized or changes. While on the subject, any file containing system information can be customized or changed. For a clearer understanding of the system, it is always good to look at all of these.

ps Command

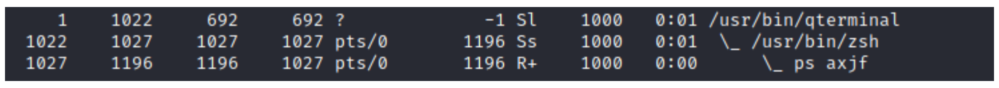
The `ps` command is an effective way to see the running processes on a Linux system. Typing `ps` on your terminal will show processes for the current shell.

The output of the `ps` (Process Status) will show the following;

- PID: The process ID (unique to the process)
- TTY: Terminal type used by the user
- Time: Amount of CPU time used by the process (this is NOT the time this process has been running for)
- CMD: The command or executable running (will NOT display any command line parameter)

The "ps" command provides a few useful options.

- `ps -A` : View all running processes
- `ps axjf` : View process tree (see the tree formation until `ps axjf` is run below)

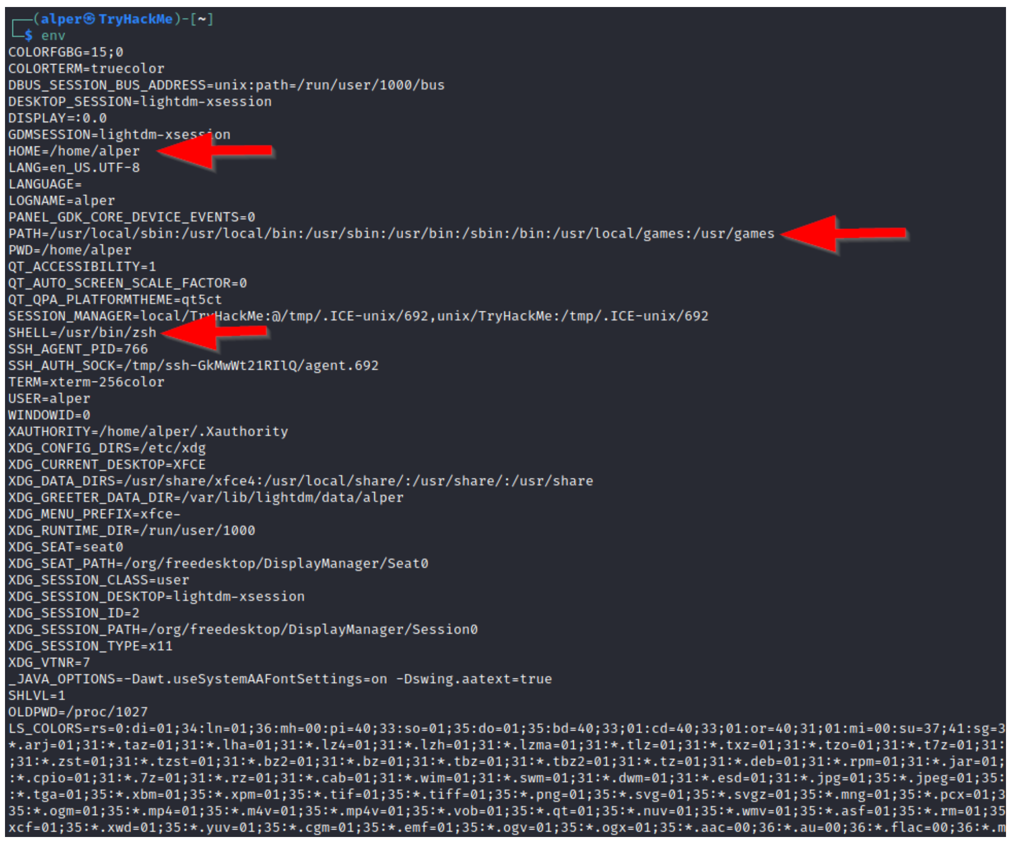


```
    1    1022     692     692 ?             -1 Sl     1000   0:01 /usr/bin/qterminal
 1022    1027    1027    1027 pts/0       1196 Ss     1000   0:01  \_ /usr/bin/zsh
 1027    1196    1196    1027 pts/0       1196 R+     1000   0:00      \_ ps axjf
```

- `ps aux` : The `aux` option will show processes for all users (a), display the user that launched the process (u), and show processes that are not attached to a terminal (x). Looking at the ps aux command output, we can have a better understanding of the system and potential vulnerabilities.

env

The `env` command will show environmental variables.



```
┌──(alper㉿TryHackMe)-[~]
└─$ env
COLORFGBG=15;0
COLORTERM=truecolor
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DESKTOP_SESSION=lightdm-xsession
DISPLAY=:0.0
GDMSESSION=lightdm-xsession
HOME=/home/alper
LANG=en_US.UTF-8
LANGUAGE=
LOGNAME=alper
PANEL_GDK_CORE_DEVICE_EVENTS=0
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
PWD=/home/alper
QT_ACCESSIBILITY=1
QT_AUTO_SCREEN_SCALE_FACTOR=0
QT_QPA_PLATFORMTHEME=qt5ct
SESSION_MANAGER=local/TryHackMe:@/tmp/.ICE-unix/692,unix/TryHackMe:/tmp/.ICE-unix/692
SHELL=/usr/bin/zsh
SSH_AGENT_PID=766
SSH_AUTH_SOCK=/tmp/ssh-GkMwWt21RIlQ/agent.692
TERM=xterm-256color
USER=alper
WINDOWID=0
XAUTHORITY=/home/alper/.Xauthority
XDG_CONFIG_DIRS=/etc/xdg
XDG_CURRENT_DESKTOP=XFCE
XDG_DATA_DIRS=/usr/share/xfce4:/usr/local/share/:/usr/share/:/usr/share
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/alper
XDG_MENU_PREFIX=xfce-
XDG_RUNTIME_DIR=/run/user/1000
XDG_SEAT=seat0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
XDG_SESSION_CLASS=user
XDG_SESSION_DESKTOP=lightdm-xsession
XDG_SESSION_ID=2
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SESSION_TYPE=x11
XDG_VTNR=7
_JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
SHLVL=1
OLDPWD=/proc/1027
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=3
*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:
;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;
:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:
:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;3
35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35
xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m
```

The PATH variable may have a compiler or a scripting language (e.g. Python) that could be used to run code on the target system or leveraged for privilege escalation.

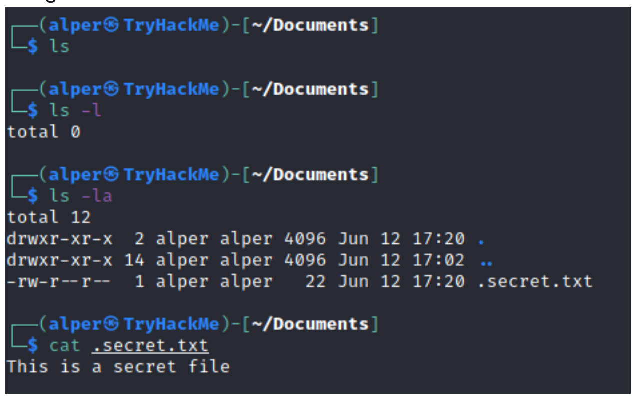
sudo -l

The target system may be configured to allow users to run some (or all) commands with root privileges. The `sudo -l` command can be used to list all commands your user can run using `sudo` .

ls

One of the common commands used in Linux is probably `ls` .

While looking for potential privilege escalation vectors, please remember to always use the `ls` command with the `-la` parameter. The example below shows how the "secret.txt" file can easily be missed using the `ls` or `ls -l` commands.

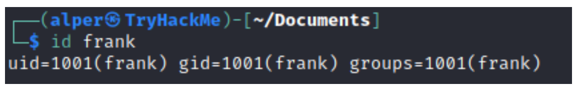


```
┌──(alper㉿TryHackMe)-[~/Documents]
└─$ ls

┌──(alper㉿TryHackMe)-[~/Documents]
└─$ ls -l
total 0

┌──(alper㉿TryHackMe)-[~/Documents]
└─$ ls -la
total 12
drwxr-xr-x  2 alper alper 4096 Jun 12 17:20 .
drwxr-xr-x 14 alper alper 4096 Jun 12 17:02 ..
-rw-r--r--  1 alper alper   22 Jun 12 17:20 .secret.txt

┌──(alper㉿TryHackMe)-[~/Documents]
└─$ cat .secret.txt
This is a secret file
```

Id

The `id` command will provide a general overview of the user's privilege level and group memberships.

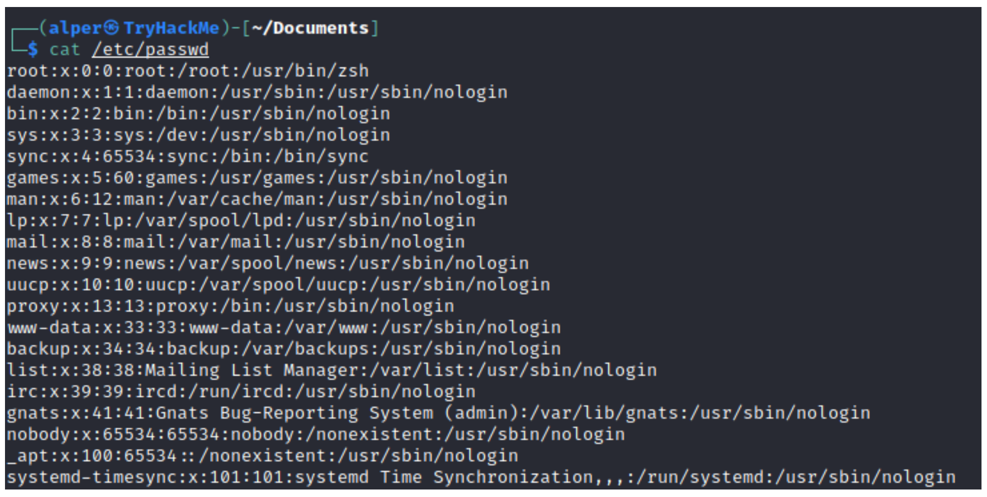
It is worth remembering that the `id` command can also be used to obtain the same information for another user as seen below.



```
┌──(alper㉿TryHackMe)-[~/Documents]
└─$ id frank
uid=1001(frank) gid=1001(frank) groups=1001(frank)
```

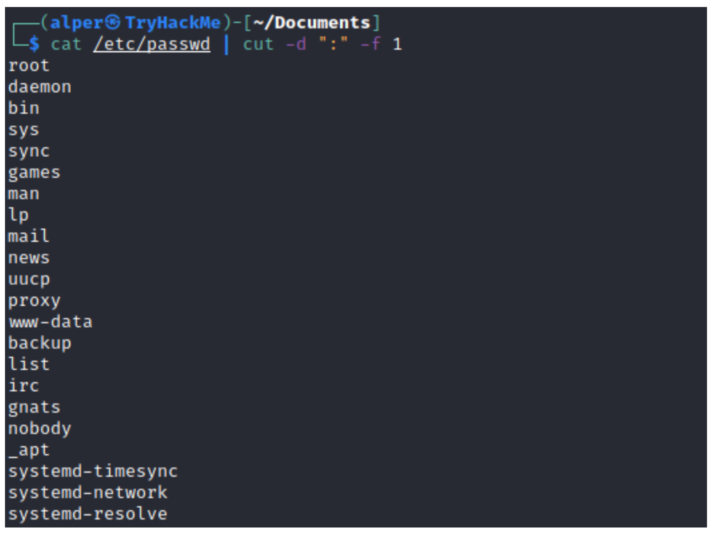
/etc/passwd

Reading the `/etc/passwd` file can be an easy way to discover users on the system.



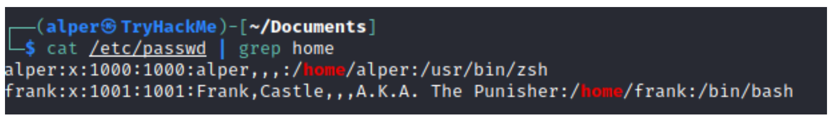
```
┌──(alper㉿TryHackMe)-[~/Documents]
└─$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
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
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
```

While the output can be long and a bit intimidating, it can easily be cut and converted to a useful list for brute-force attacks.



```
┌──(alper㉿TryHackMe)-[~/Documents]
└─$ cat /etc/passwd | cut -d ":" -f 1
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-timesync
systemd-network
systemd-resolve
```

Remember that this will return all users, some of which are system or service users that would not be very useful. Another approach could be to grep for "home" as real users will most likely have their folders under the "home" directory.



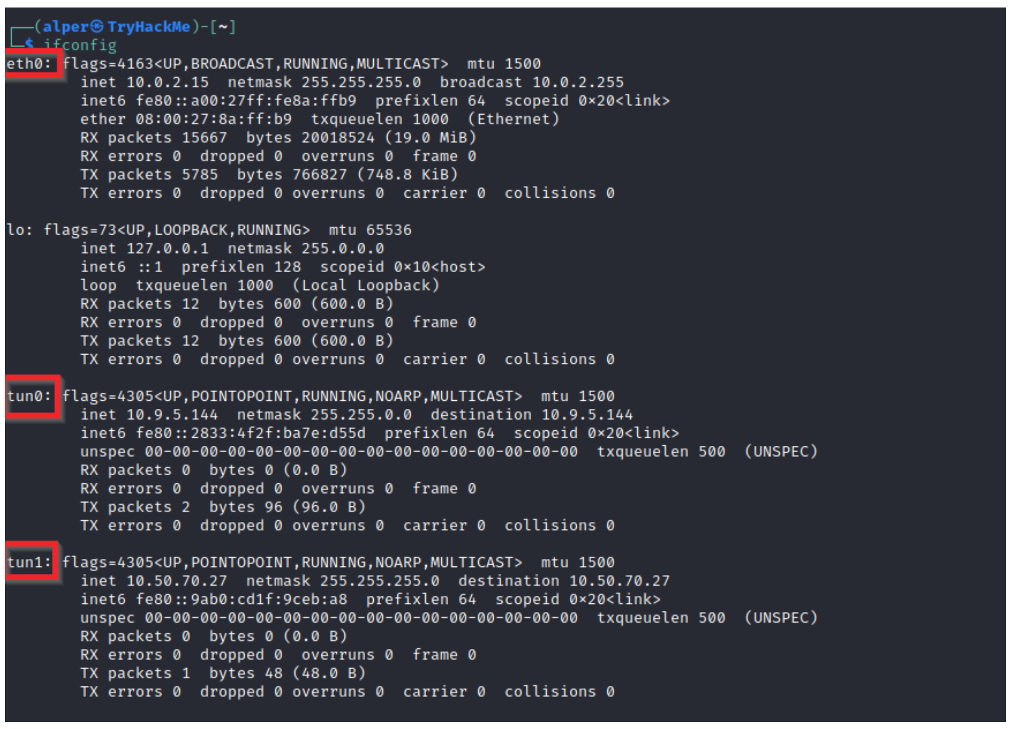
```
┌──(alper㉿TryHackMe)-[~/Documents]
└─$ cat /etc/passwd | grep home
alper:x:1000:1000:alper,,,:/home/alper:/usr/bin/zsh
frank:x:1001:1001:Frank,Castle,,,A.K.A. The Punisher:/home/frank:/bin/bash
```

history

Looking at earlier commands with the `history` command can give us some idea about the target system and, albeit rarely, have stored information such as passwords or usernames.

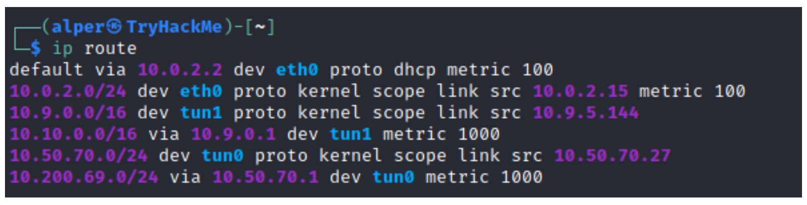
ifconfig

The target system may be a pivoting point to another network. The `ifconfig` command will give us information about the network interfaces of the system. The example below shows the target system has three interfaces (eth0, tun0, and tun1). Our attacking machine can reach the eth0 interface but can not directly access the two other networks.



```
┌──(alper㉿TryHackMe)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe8a:ffb9  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:8a:ff:b9  txqueuelen 1000  (Ethernet)
        RX packets 15667  bytes 20018524 (19.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5785  bytes 766827 (748.8 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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 12  bytes 600 (600.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 600 (600.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.9.5.144  netmask 255.255.0.0  destination 10.9.5.144
        inet6 fe80::2833:4f2f:ba7e:d55d  prefixlen 64  scopeid 0×20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 96 (96.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.50.70.27  netmask 255.255.255.0  destination 10.50.70.27
        inet6 fe80::9ab0:cd1f:9ceb:a8  prefixlen 64  scopeid 0×20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 48 (48.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This can be confirmed using the `ip route` command to see which network routes exist.

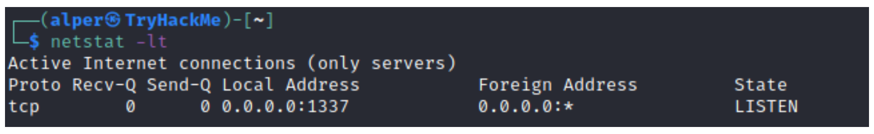


```
┌──(alper㉿TryHackMe)-[~]
└─$ ip route
default via 10.0.2.2 dev eth0 proto dhcp metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
10.9.0.0/16 dev tun1 proto kernel scope link src 10.9.5.144
10.10.0.0/16 via 10.9.0.1 dev tun1 metric 1000
10.50.70.0/24 dev tun0 proto kernel scope link src 10.50.70.27
10.200.69.0/24 via 10.50.70.1 dev tun0 metric 1000
```

netstat

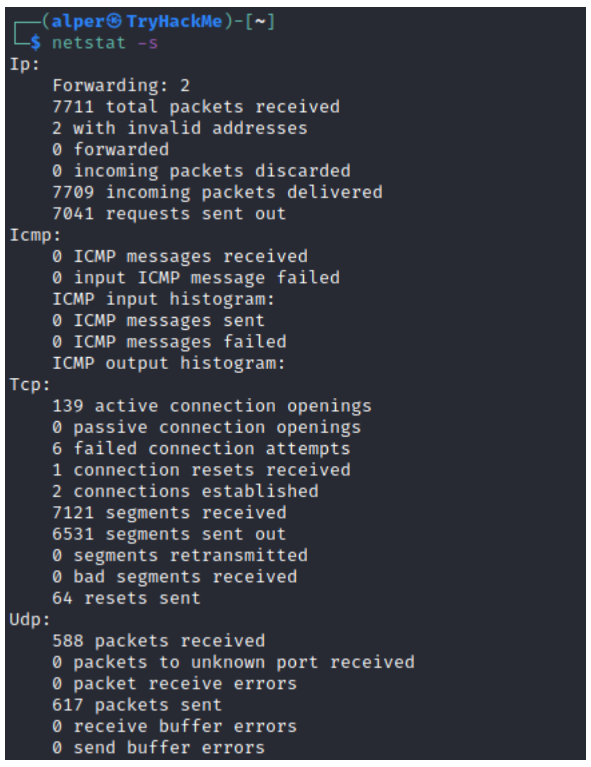
Following an initial check for existing interfaces and network routes, it is worth looking into existing communications. The `netstat` command can be used with several different options to gather information on existing connections.

- `netstat -a` : shows all listening ports and established connections.
- `netstat -at` or `netstat -au` can also be used to list TCP or UDP protocols respectively.
- `netstat -l` : list ports in "listening" mode. These ports are open and ready to accept incoming connections. This can be used with the "t" option to list only ports that are listening using the TCP protocol (below)



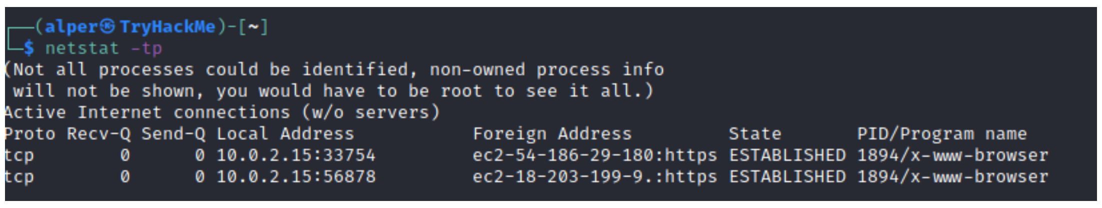
```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:1337            0.0.0.0:*               LISTEN
```

- `netstat -s` : list network usage statistics by protocol (below) This can also be used with the `-t` or `-u` options to limit the output to a specific protocol.



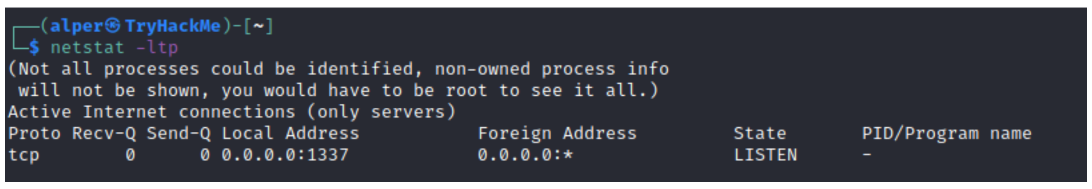
```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -s
Ip:
    Forwarding: 2
    7711 total packets received
    2 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    7709 incoming packets delivered
    7041 requests sent out
Icmp:
    0 ICMP messages received
    0 input ICMP message failed
    ICMP input histogram:
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
Tcp:
    139 active connection openings
    0 passive connection openings
    6 failed connection attempts
    1 connection resets received
    2 connections established
    7121 segments received
    6531 segments sent out
    0 segments retransmitted
    0 bad segments received
    64 resets sent
Udp:
    588 packets received
    0 packets to unknown port received
    0 packet receive errors
    617 packets sent
    0 receive buffer errors
    0 send buffer errors
```

- `netstat -tp` : list connections with the service name and PID information.



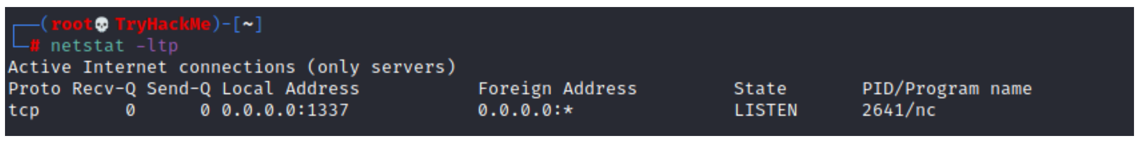
```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -tp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 10.0.2.15:33754         ec2-54-186-29-180:https ESTABLISHED 1894/x-www-browser
tcp        0      0 10.0.2.15:56878         ec2-18-203-199-9.:https ESTABLISHED 1894/x-www-browser
```

This can also be used with the `-l` option to list listening ports (below)



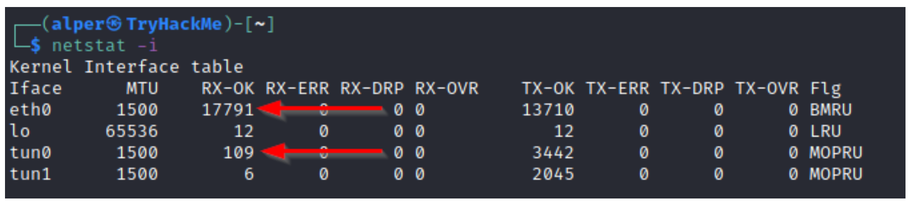
```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -ltp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:1337            0.0.0.0:*               LISTEN      -
```

We can see the "PID/Program name" column is empty as this process is owned by another user.
Below is the same command run with root privileges and reveals this information as 2641/nc (netcat)



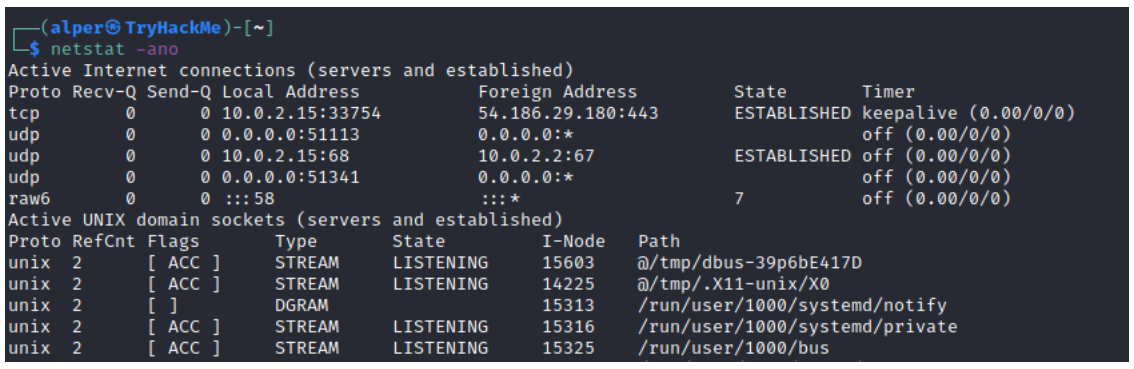
```
┌──(root💀 TryHackMe)-[~]
└─# netstat -ltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:1337            0.0.0.0:*               LISTEN      2641/nc
```

- `netstat -i` : Shows interface statistics. We see below that "eth0" and "tun0" are more active than "tun1".



```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500    17791      0      0 0         13710      0      0      0 BMRU
lo       65536       12      0      0 0            12      0      0      0 LRU
tun0      1500      109      0      0 0          3442      0      0      0 MOPRU
tun1      1500        6      0      0 0          2045      0      0      0 MOPRU
```

The `netstat` usage you will probably see most often in blog posts, write-ups, and courses is `netstat -ano` which could be broken down as follows;
- `-a` : Display all sockets
- `-n` : Do not resolve names
- `-o` : Display timers



```
┌──(alper㉿TryHackMe)-[~]
└─$ netstat -ano
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       Timer
tcp        0      0 10.0.2.15:33754         54.186.29.180:443       ESTABLISHED keepalive (0.00/0/0)
udp        0      0 0.0.0.0:51113           0.0.0.0:*                           off (0.00/0/0)
udp        0      0 10.0.2.15:68            10.0.2.2:67             ESTABLISHED off (0.00/0/0)
udp        0      0 0.0.0.0:51341           0.0.0.0:*                           off (0.00/0/0)
raw6       0      0 :::58                   :::*                    7           off (0.00/0/0)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     15603    @/tmp/dbus-39p6bE417D
unix  2      [ ACC ]     STREAM     LISTENING     14225    @/tmp/.X11-unix/X0
unix  2      [ ]         DGRAM                    15313    /run/user/1000/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     15316    /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     15325    /run/user/1000/bus
```

find Command

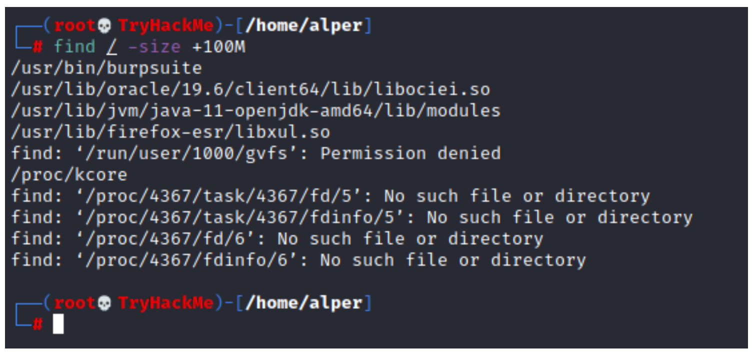
Searching the target system for important information and potential privilege escalation vectors can be fruitful. The built-in "find" command is useful and worth keeping in your arsenal.

Below are some useful examples for the "find" command.

Find files:

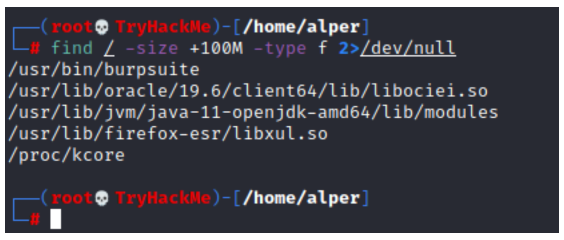
- `find . -name flag1.txt` : find the file named "flag1.txt" in the current directory
- `find /home -name flag1.txt` : find the file names "flag1.txt" in the /home directory
- `find / -type d -name config` : find the directory named config under "/"
- `find / -type f -perm 0777` : find files with the 777 permissions (files readable, writable, and executable by all users)
- `find / -perm a=x` : find executable files
- `find /home -user frank` : find all files for user "frank" under "/home"
- `find / -mtime 10` : find files that were modified in the last 10 days
- `find / -atime 10` : find files that were accessed in the last 10 day
- `find / -cmin -60` : find files changed within the last hour (60 minutes)
- `find / -amin -60` : find files accesses within the last hour (60 minutes)
- `find / -size 50M` : find files with a 50 MB size

This command can also be used with (+) and (-) signs to specify a file that is larger or smaller than the given size.



```
┌──(root💀 TryHackMe)-[/home/alper]
└─# find / -size +100M
/usr/bin/burpsuite
/usr/lib/oracle/19.6/client64/lib/libociei.so
/usr/lib/jvm/java-11-openjdk-amd64/lib/modules
/usr/lib/firefox-esr/libxul.so
find: '/run/user/1000/gvfs': Permission denied
/proc/kcore
find: '/proc/4367/task/4367/fd/5': No such file or directory
find: '/proc/4367/task/4367/fdinfo/5': No such file or directory
find: '/proc/4367/fd/6': No such file or directory
find: '/proc/4367/fdinfo/6': No such file or directory

┌──(root💀 TryHackMe)-[/home/alper]
└─# █
```

The example above returns files that are larger than 100 MB. It is important to note that the "find" command tends to generate errors which sometimes makes the output hard to read. This is why it would be wise to use the "find" command with "-type f 2>/dev/null" to redirect errors to "/dev/null" and have a cleaner output (below).

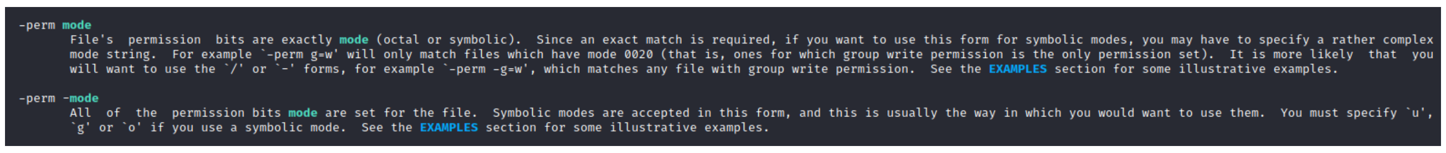


```
┌──(root💀 TryHackMe)-[/home/alper]
└─# find / -size +100M -type f 2>/dev/null
/usr/bin/burpsuite
/usr/lib/oracle/19.6/client64/lib/libociei.so
/usr/lib/jvm/java-11-openjdk-amd64/lib/modules
/usr/lib/firefox-esr/libxul.so
/proc/kcore

┌──(root💀 TryHackMe)-[/home/alper]
└─# █
```

Folders and files that can be written to or executed from:

- `find / -writable -type d 2>/dev/null` : Find world-writeable folders
- `find / -perm -222 -type d 2>/dev/null` : Find world-writeable folders
- `find / -perm -o w -type d 2>/dev/null` : Find world-writeable folders

The reason we see three different "find" commands that could potentially lead to the same result can be seen in the manual document. As you can see below, the perm parameter affects the way "find" works.



```
-perm mode
       File's  permission  bits are exactly mode (octal or symbolic).  Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex
       mode string.  For example `-perm g=w' will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set).  It is more likely  that  you
       will want to use the `/' or `-' forms, for example `-perm -g=w', which matches any file with group write permission.  See the EXAMPLES section for some illustrative examples.

-perm -mode
       All  of  the  permission bits mode are set for the file.  Symbolic modes are accepted in this form, and this is usually the way in which you would want to use them.  You must specify `u',
       `g' or `o' if you use a symbolic mode.  See the EXAMPLES section for some illustrative examples.
```

- `find / -perm -o x -type d 2>/dev/null` : Find world-executable folders

Find development tools and supported languages:

- `find / -name perl*`
- `find / -name python*`
- `find / -name gcc*`

Find specific file permissions:

Below is a short example used to find files that have the SUID bit set. The SUID bit allows the file to run with the privilege level of the account that owns it, rather than the account which runs it. This allows for an interesting privilege escalation path,we will see in more details on task 6. The example below is given to complete the subject on the "find" command.

- `find / -perm -u=s -type f 2>/dev/null` : Find files with the SUID bit, which allows us to run the file with a higher privilege level than the current user.

General Linux Commands

As we are in the Linux realm, familiarity with Linux commands, in general, will be very useful. Please spend some time getting comfortable with commands such as `find` , `locate` , `grep` , `cut` , `sort` , etc.

Answer the questions below

What is the hostname of the target system?

Answer format: ******** Submit

What is the Linux kernel version of the target system?

Answer format: *.*.*************** Submit

What Linux is this?

Answer format: ****** **.** *** Submit

What version of the Python language is installed on the system?

Answer format: *.*.* Submit

What vulnerability seem to affect the kernel of the target system? (Enter a CVE number)

Answer format: ************* Submit

Task 4 Automated Enumeration Tools

Task 5 Privilege Escalation: Kernel Exploits

Task 6 Privilege Escalation: Sudo

Task 7 Privilege Escalation: SUID

Task 8 Privilege Escalation: Capabilities

Task 9 Privilege Escalation: Cron Jobs

Task 10 Privilege Escalation: PATH

Task 11 ◯ Privilege Escalation: NFS

Task 12 ◯ Capstone Challenge

Created by

tryhackme and

basaranalper

This is a **free** room, which means anyone can deploy virtual machines in the room (without being subscribed)! 1884 users are in here and this room is 129 days old.