Linux Commands

List of essential Linux security commands

3 years ago • by David Adams

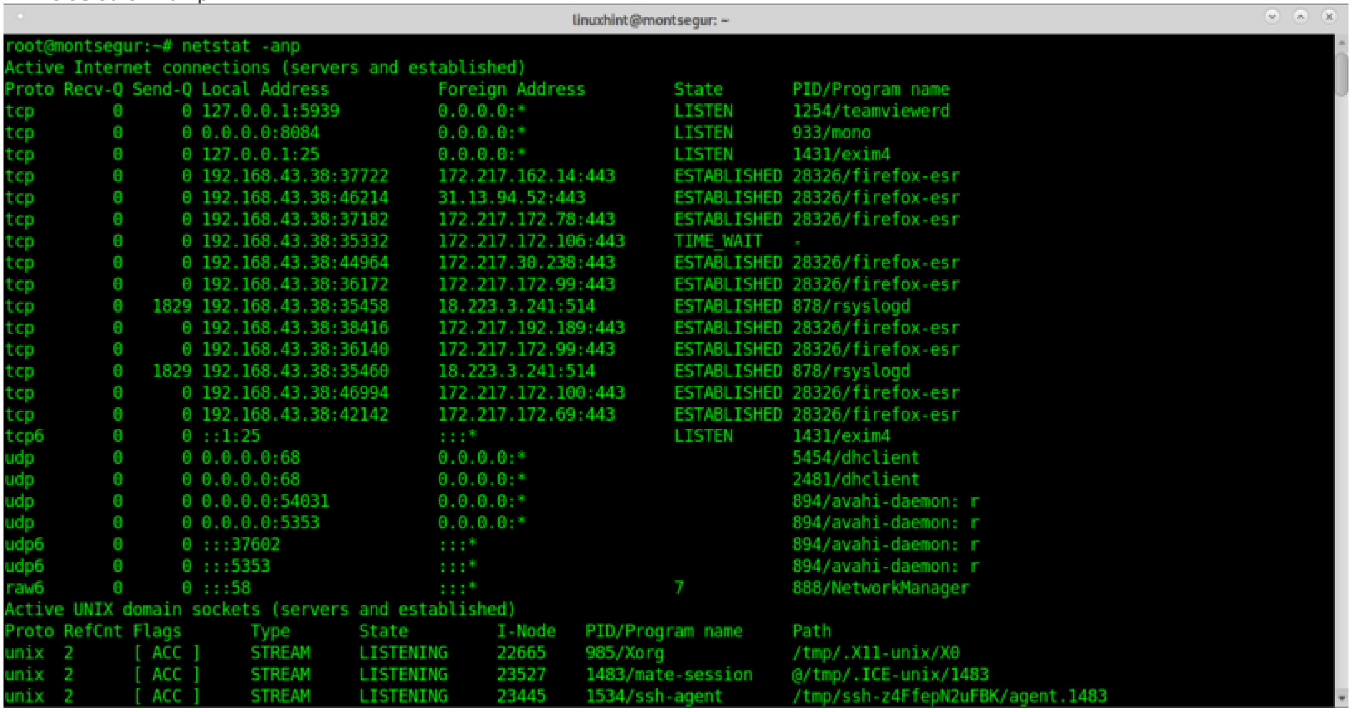
This tutorial shows some of the most basic Linux commands oriented to security.

Using the command netstat to find open ports:

One of the most basic commands to monitor the state of your device is **netstat** which shows the open ports and established connections.

Below an example of the **netstat** with additional options output:

```
# netstat -anp
```



```
root@montsegur:~# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5939          0.0.0.0:*               LISTEN      1254/teamviewerd
tcp        0      0 0.0.0.0:8084            0.0.0.0:*               LISTEN      933/mono
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1431/exim4
tcp        0      0 192.168.43.38:37722     172.217.162.14:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:46214     31.13.94.52:443         ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:37182     172.217.172.78:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:35332     172.217.172.106:443     TIME_WAIT   -
tcp        0      0 192.168.43.38:44964     172.217.30.238:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:36172     172.217.172.99:443      ESTABLISHED 28326/firefox-esr
tcp        0   1829 192.168.43.38:35458     18.223.3.241:514        ESTABLISHED 878/rsyslogd
tcp        0      0 192.168.43.38:38416     172.217.192.189:443     ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:36140     172.217.172.99:443      ESTABLISHED 28326/firefox-esr
tcp        0   1829 192.168.43.38:35460     18.223.3.241:514        ESTABLISHED 878/rsyslogd
tcp        0      0 192.168.43.38:46994     172.217.172.100:443     ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:42142     172.217.172.69:443      ESTABLISHED 28326/firefox-esr
tcp6       0      0 ::1:25                  :::*                    LISTEN      1431/exim4
udp        0      0 0.0.0.0:68              0.0.0.0:*                           5454/dhclient
udp        0      0 0.0.0.0:68              0.0.0.0:*                           2481/dhclient
udp        0      0 0.0.0.0:54031           0.0.0.0:*                           894/avahi-daemon: r
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           894/avahi-daemon: r
udp6       0      0 :::37602                :::*                                894/avahi-daemon: r
udp6       0      0 :::5353                 :::*                                894/avahi-daemon: r
raw6       0      0 :::58                   :::*                    7           888/NetworkManager
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     22665    985/Xorg             /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     23527    1483/mate-session    @/tmp/.ICE-unix/1483
unix  2      [ ACC ]     STREAM     LISTENING     23445    1534/ssh-agent       /tmp/ssh-z4FfepN2uFBK/agent.1483
```

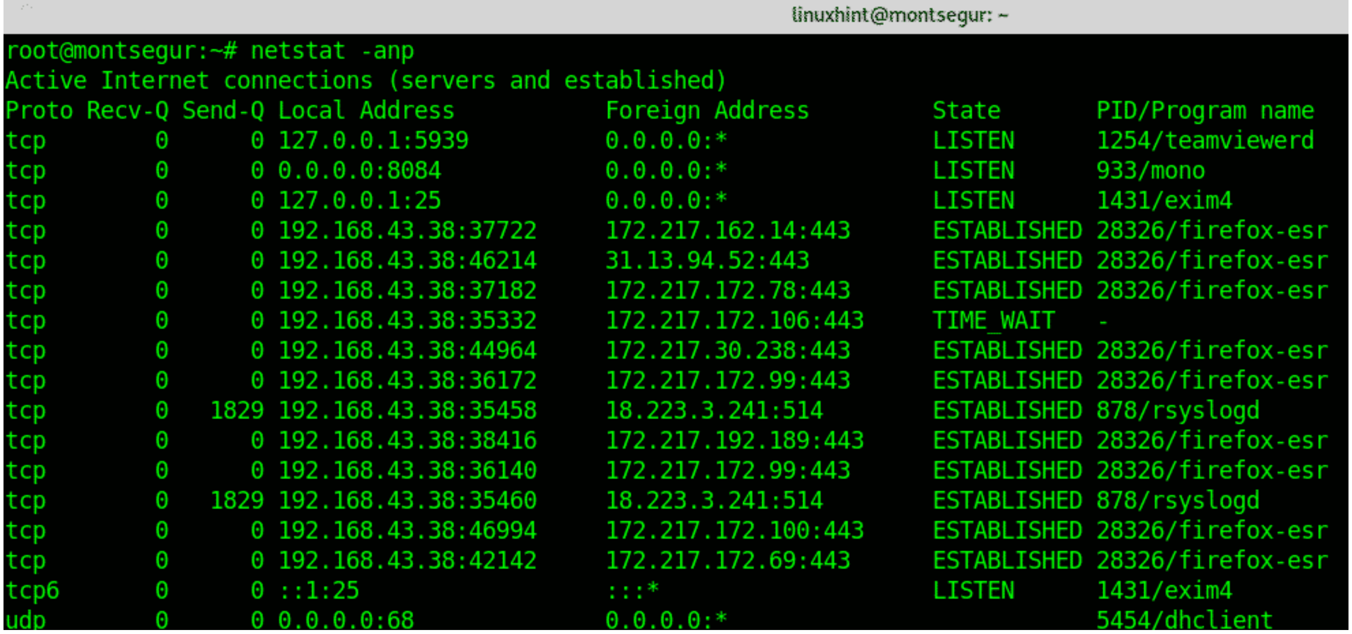
Where:

-a: shows the state for sockets.

-n: shows IP addresses instead of hots.

-p: shows the program establishing the conenction.

An output extract better look:



```
root@montsegur:~# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5939          0.0.0.0:*               LISTEN      1254/teamviewerd
tcp        0      0 0.0.0.0:8084            0.0.0.0:*               LISTEN      933/mono
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1431/exim4
tcp        0      0 192.168.43.38:37722     172.217.162.14:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:46214     31.13.94.52:443         ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:37182     172.217.172.78:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:35332     172.217.172.106:443     TIME_WAIT   -
tcp        0      0 192.168.43.38:44964     172.217.30.238:443      ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:36172     172.217.172.99:443      ESTABLISHED 28326/firefox-esr
tcp        0   1829 192.168.43.38:35458     18.223.3.241:514        ESTABLISHED 878/rsyslogd
tcp        0      0 192.168.43.38:38416     172.217.192.189:443     ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:36140     172.217.172.99:443      ESTABLISHED 28326/firefox-esr
tcp        0   1829 192.168.43.38:35460     18.223.3.241:514        ESTABLISHED 878/rsyslogd
tcp        0      0 192.168.43.38:46994     172.217.172.100:443     ESTABLISHED 28326/firefox-esr
tcp        0      0 192.168.43.38:42142     172.217.172.69:443      ESTABLISHED 28326/firefox-esr
tcp6       0      0 ::1:25                  :::*                    LISTEN      1431/exim4
udp        0      0 0.0.0.0:68              0.0.0.0:*                           5454/dhclient
```

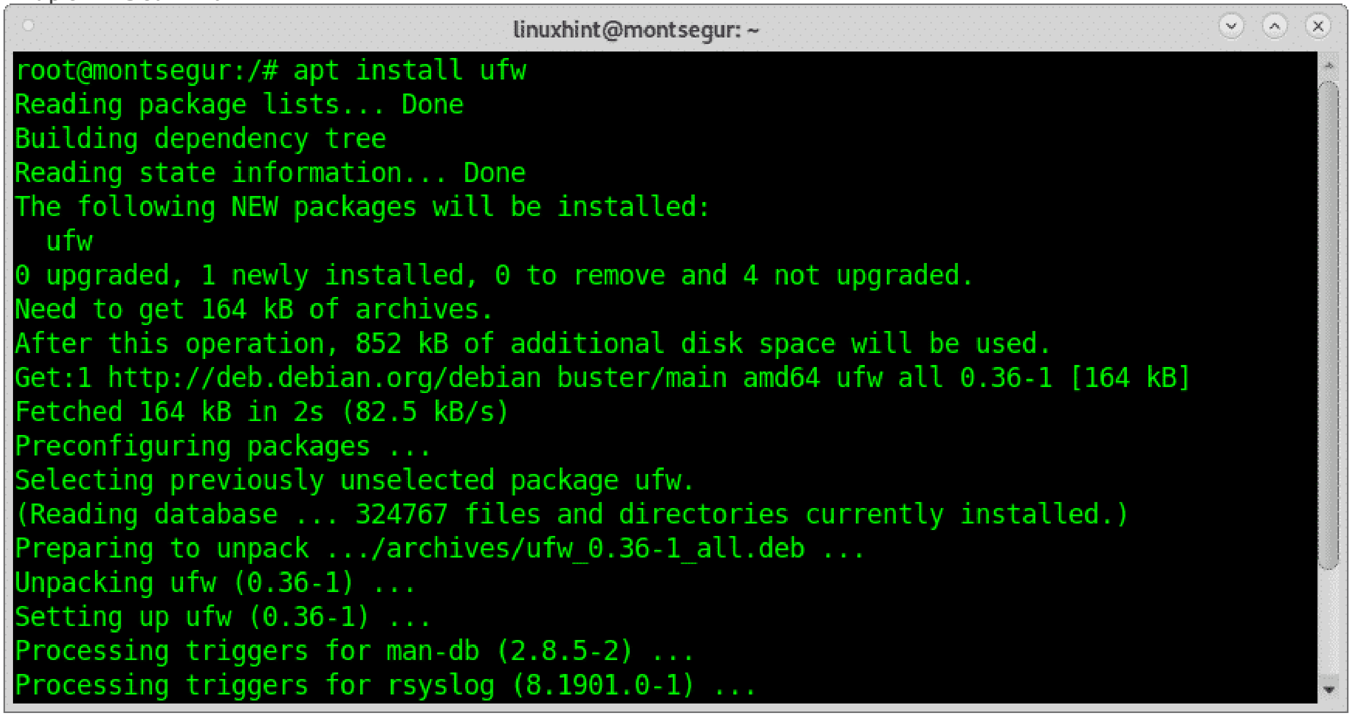
The first column shows the protocol, you can see both TCP and UDP are included, the first screenshot also shows UNIX sockets. If you are suspicious that something is wrong, checking ports is of course mandatory.

Setting basic rules with UFW:

LinuxHint has published great tutorials on UFW and Iptables, here I will focus on a restrictive policy firewall. It is recommended to keep a restrictive policy denying all incoming traffic unless you want it to be allowed.

To install UFW run:

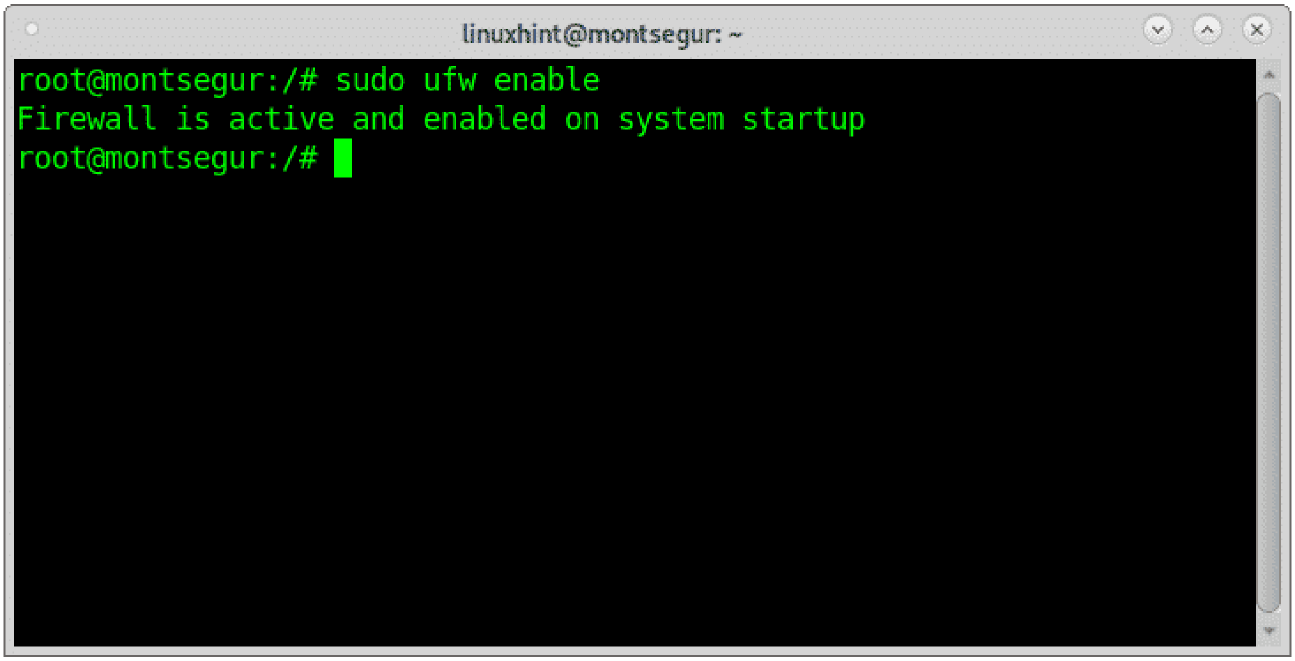
```
# apt install ufw
```



```
root@montsegur:/# apt install ufw
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ufw
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 164 kB of archives.
After this operation, 852 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 ufw all 0.36-1 [164 kB]
Fetched 164 kB in 2s (82.5 kB/s)
Preconfiguring packages ...
Selecting previously unselected package ufw.
(Reading database ... 324767 files and directories currently installed.)
Preparing to unpack .../archives/ufw_0.36-1_all.deb ...
Unpacking ufw (0.36-1) ...
Setting up ufw (0.36-1) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for rsyslog (8.1901.0-1) ...
```

To enable the firewall at startup run:

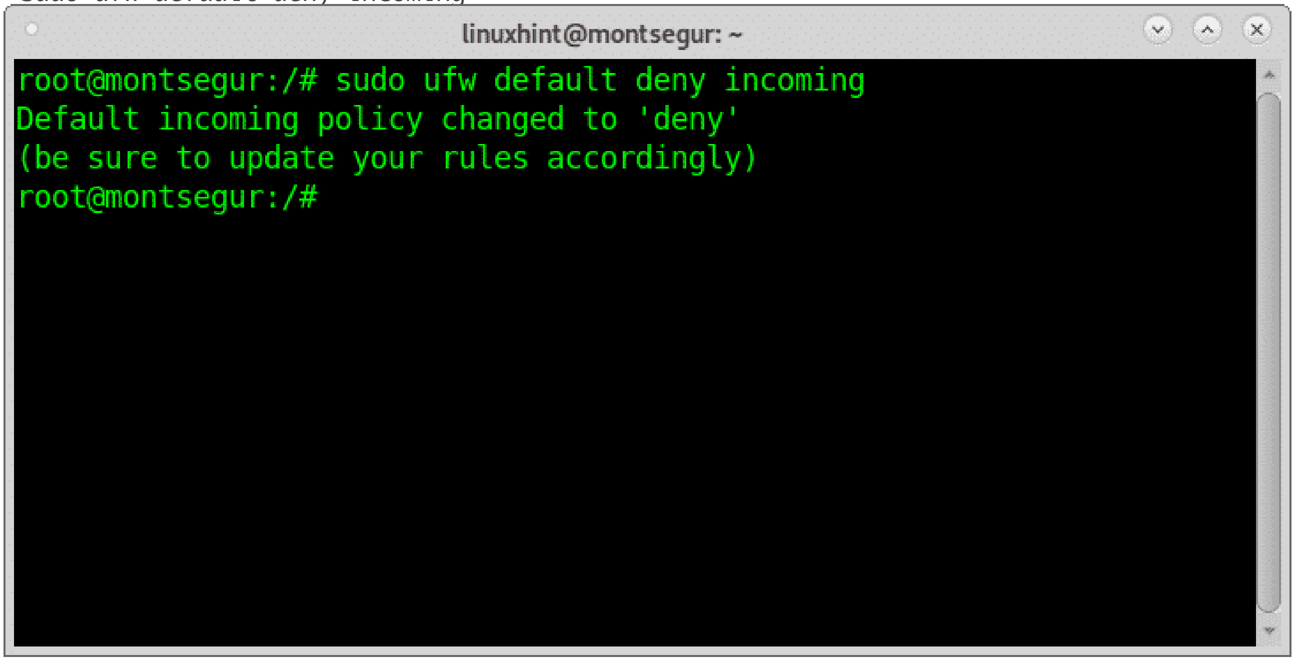
```
# sudo ufw enable
```



```
root@montsegur:/# sudo ufw enable
Firewall is active and enabled on system startup
root@montsegur:/# 
```

Then apply a default restrictive policy by running:

```
#  sudo ufw default deny incoming
```



```
root@montsegur:/# sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@montsegur:/#
```

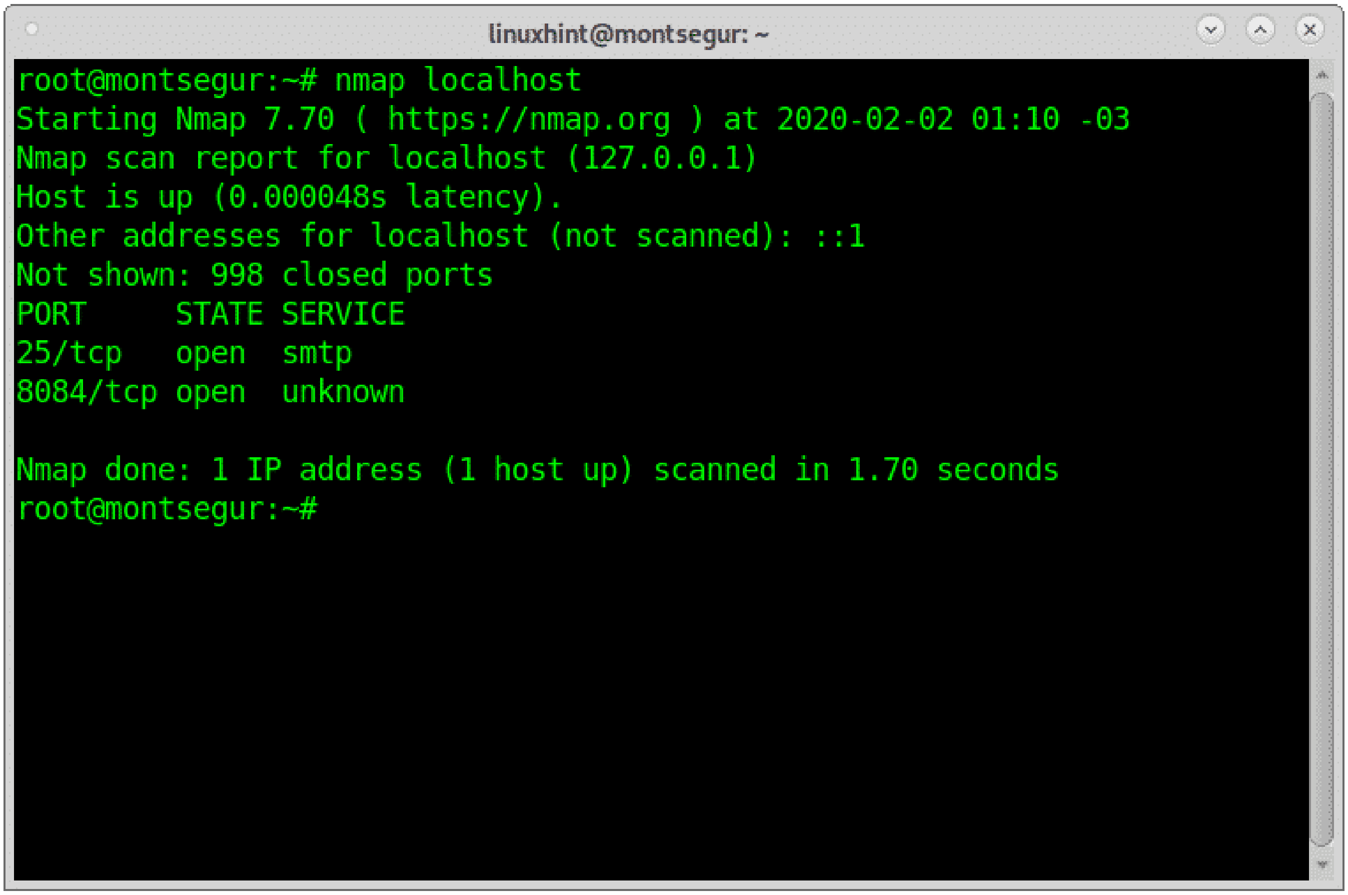
You will need to manually open the ports you want to use by running:

```
# ufw allow <port>
```

Auditing yourself with nmap:

Nmap is, if not the best, one of the best security scanners in the market. It is the main tool used by sysadmins to audit their network security. If you are in a DMZ you can scan your external IP, you can also scan your router or your local host.

A very simple scan against your localhost would be:



```
root@montsegur:~# nmap localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-02-02 01:10 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000048s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT     STATE SERVICE
25/tcp   open  smtp
8084/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 1.70 seconds
root@montsegur:~#
```

As you see the output shows my port 25 and port 8084 are open.

Nmap has a lot of possibilities, including OS, Version detection, vulnerability scans, etc. At LinuxHint we have published a lot of tutorials focused on Nmap and its different techniques. You can find them here.

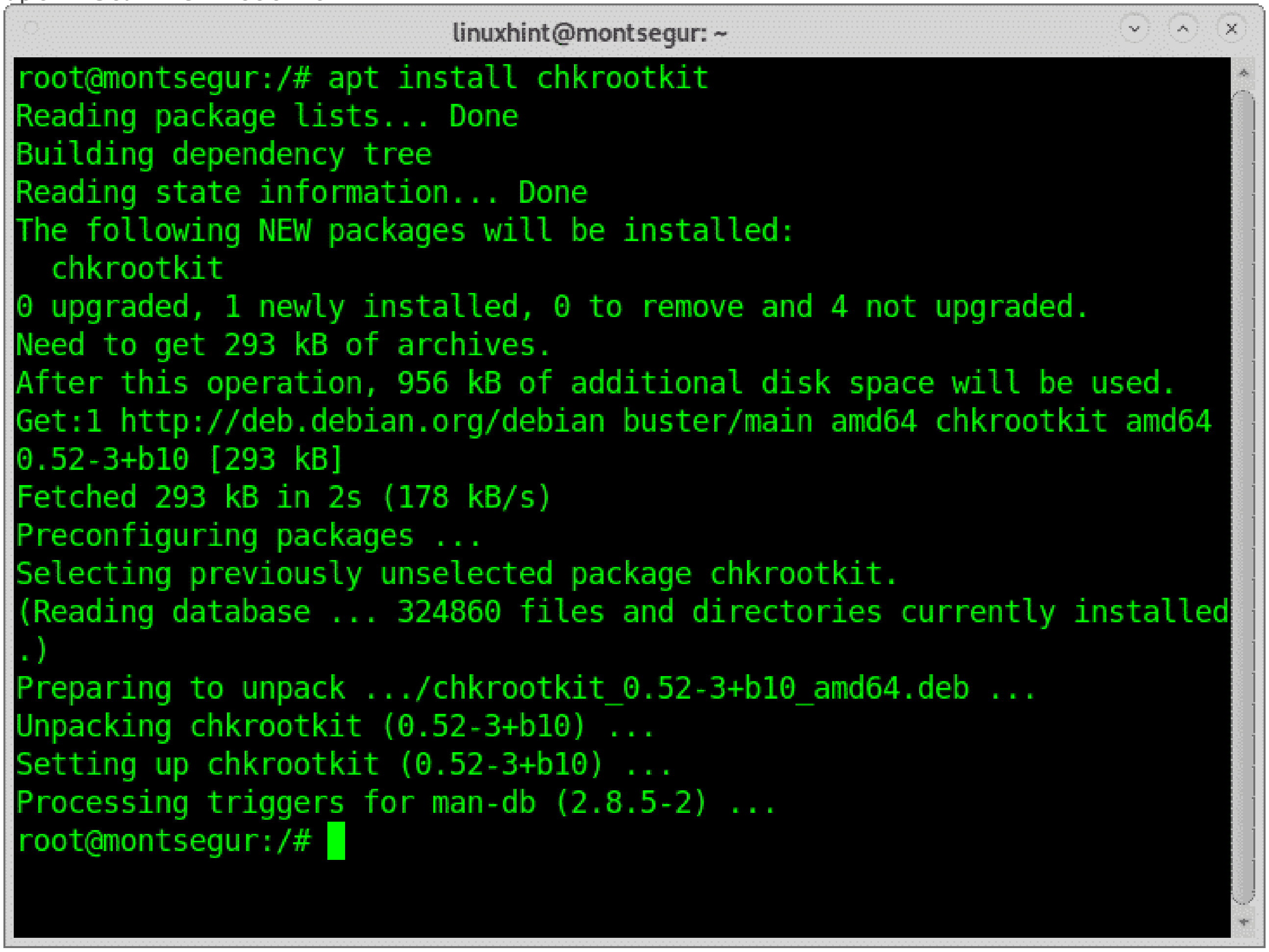
The command chkrootkit to check your system for chrootkit infections:

Rootkits are probably the most dangerous threat to computers. The command chkrootkit

(check rootkit) can help you to detect known rootkits.

To install chkrootkit run:

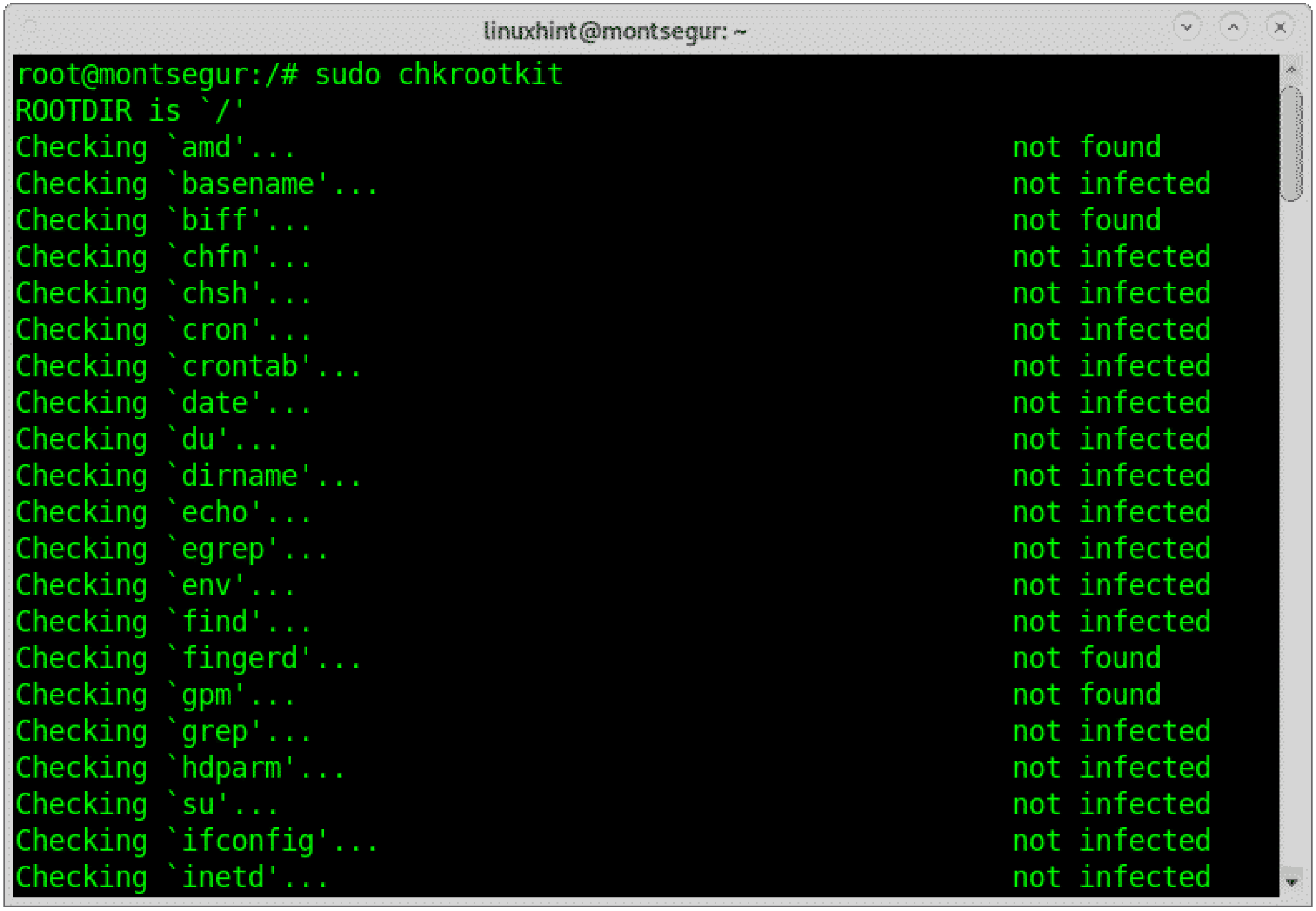
```
# apt install chkrootkit
```



```
root@montsegur:/# apt install chkrootkit
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  chkrootkit
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 293 kB of archives.
After this operation, 956 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 chkrootkit amd64
0.52-3+b10 [293 kB]
Fetched 293 kB in 2s (178 kB/s)
Preconfiguring packages ...
Selecting previously unselected package chkrootkit.
(Reading database ... 324860 files and directories currently installed
.)
Preparing to unpack .../chkrootkit_0.52-3+b10_amd64.deb ...
Unpacking chkrootkit (0.52-3+b10) ...
Setting up chkrootkit (0.52-3+b10) ...
Processing triggers for man-db (2.8.5-2) ...
root@montsegur:/#
```

Then run:

```
# sudo chkrootkit
```

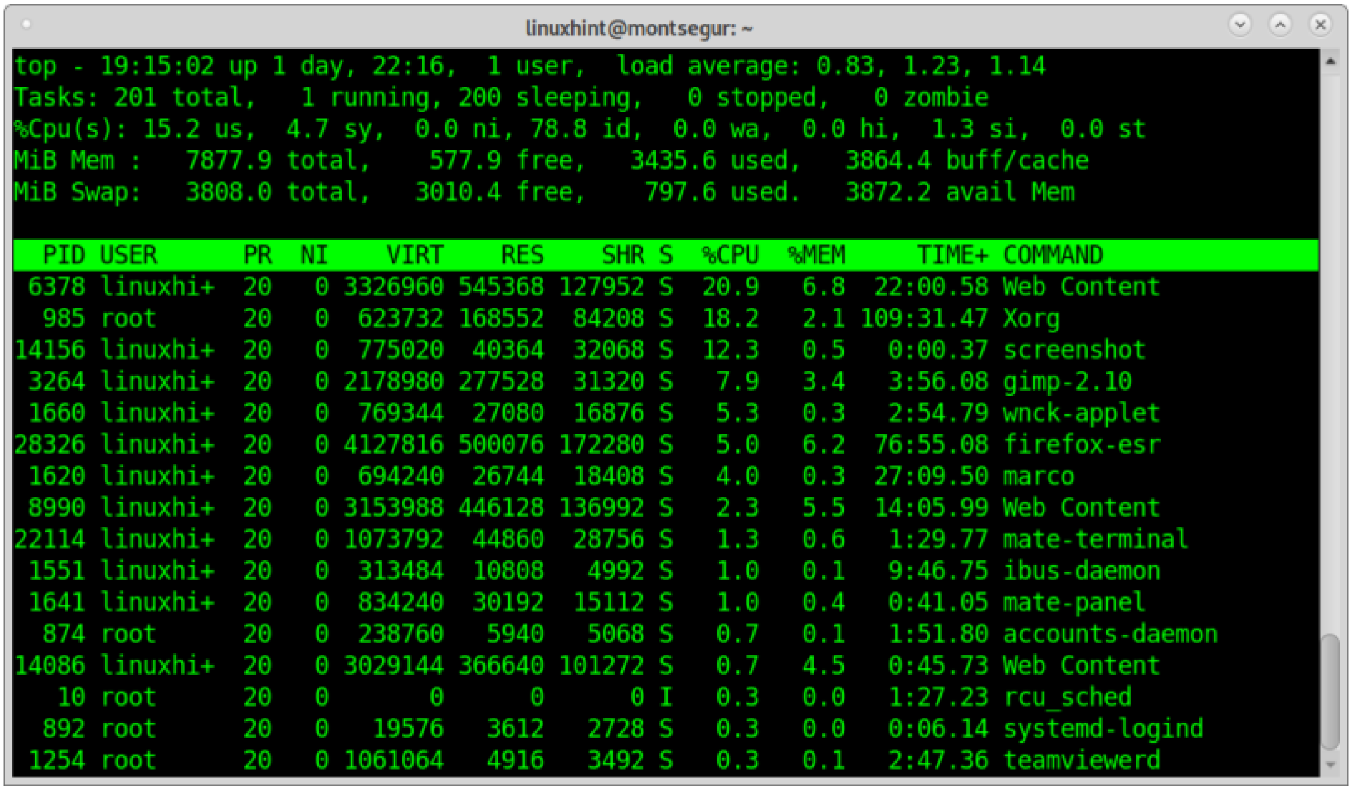


```
root@montsegur:/# sudo chkrootkit
ROOTDIR is `/'
Checking `amd'...                                            not found
Checking `basename'...                                       not infected
Checking `biff'...                                           not found
Checking `chfn'...                                           not infected
Checking `chsh'...                                           not infected
Checking `cron'...                                           not infected
Checking `crontab'...                                        not infected
Checking `date'...                                           not infected
Checking `du'...                                             not infected
Checking `dirname'...                                        not infected
Checking `echo'...                                           not infected
Checking `egrep'...                                          not infected
Checking `env'...                                            not infected
Checking `find'...                                           not infected
Checking `fingerd'...                                        not found
Checking `gpm'...                                            not found
Checking `grep'...                                           not infected
Checking `hdparm'...                                         not infected
Checking `su'...                                             not infected
Checking `ifconfig'...                                       not infected
Checking `inetd'...                                          not infected
```

Using the command **top** to check processes taking most of your resources:

To get a fast view on running resources you can use the command top, on the terminal run:

```
# top
```



```
top - 19:15:02 up 1 day, 22:16,  1 user,  load average: 0.83, 1.23, 1.14
Tasks: 201 total,   1 running, 200 sleeping,   0 stopped,   0 zombie
%Cpu(s): 15.2 us,  4.7 sy,  0.0 ni, 78.8 id,  0.0 wa,  0.0 hi,  1.3 si,  0.0 st
MiB Mem :   7877.9 total,    577.9 free,   3435.6 used,   3864.4 buff/cache
MiB Swap:   3808.0 total,   3010.4 free,    797.6 used.   3872.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6378 linuxhi+  20   0 3326960 545368 127952 S  20.9   6.8  22:00.58 Web Content
  985 root      20   0  623732 168552  84208 S  18.2   2.1 109:31.47 Xorg
14156 linuxhi+  20   0  775020  40364  32068 S  12.3   0.5   0:00.37 screenshot
 3264 linuxhi+  20   0 2178980 277528  31320 S   7.9   3.4   3:56.08 gimp-2.10
 1660 linuxhi+  20   0  769344  27080  16876 S   5.3   0.3   2:54.79 wnck-applet
28326 linuxhi+  20   0 4127816 500076 172280 S   5.0   6.2  76:55.08 firefox-esr
 1620 linuxhi+  20   0  694240  26744  18408 S   4.0   0.3  27:09.50 marco
 8990 linuxhi+  20   0 3153988 446128 136992 S   2.3   5.5  14:05.99 Web Content
22114 linuxhi+  20   0 1073792  44860  28756 S   1.3   0.6   1:29.77 mate-terminal
 1551 linuxhi+  20   0  313484  10808   4992 S   1.0   0.1   9:46.75 ibus-daemon
 1641 linuxhi+  20   0  834240  30192  15112 S   1.0   0.4   0:41.05 mate-panel
  874 root      20   0  238760   5940   5068 S   0.7   0.1   1:51.80 accounts-daemon
14086 linuxhi+  20   0 3029144 366640 101272 S   0.7   4.5   0:45.73 Web Content
   10 root      20   0       0      0      0 I   0.3   0.0   1:27.23 rcu_sched
  892 root      20   0   19576   3612   2728 S   0.3   0.0   0:06.14 systemd-logind
 1254 root      20   0 1061064   4916   3492 S   0.3   0.1   2:47.36 teamviewerd
```

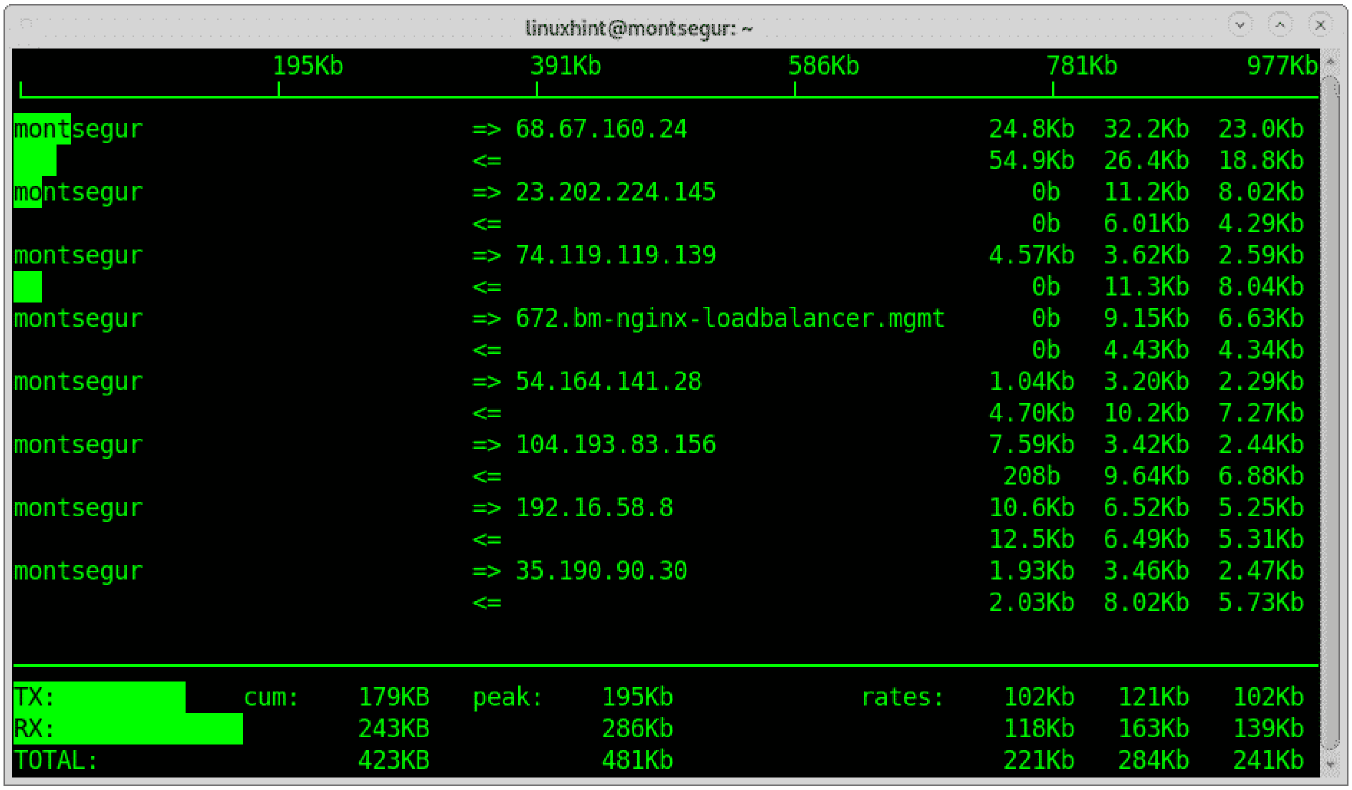
The command **iftop** to monitor your network traffic:

Another great tool to monitor your traffic is iftop,

```
# sudo iftop  <interface>
```

In my case:

```
# sudo iftop wlp3s0
```



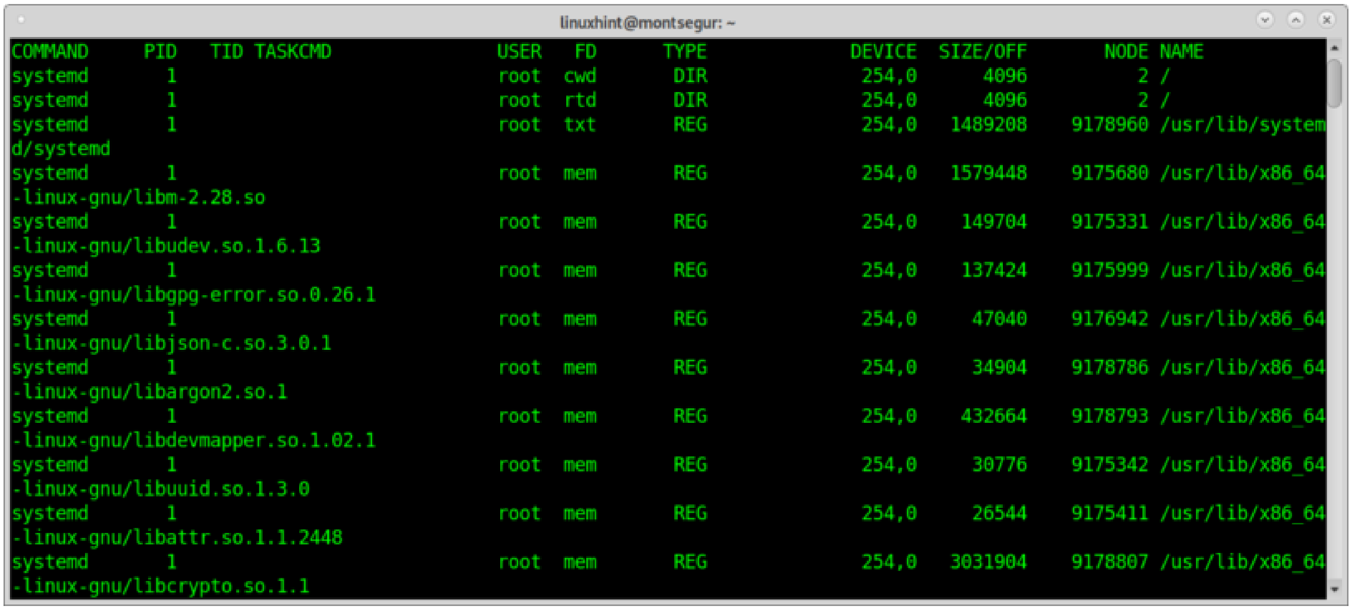
```
                 195Kb            391Kb            586Kb            781Kb         977Kb
└─────────────────┴────────────────┴────────────────┴────────────────┴──────────────────
montsegur                       => 68.67.160.24                     24.8Kb  32.2Kb  23.0Kb
                                <=                                  54.9Kb  26.4Kb  18.8Kb
montsegur                       => 23.202.224.145                      0b   11.2Kb  8.02Kb
                                <=                                     0b   6.01Kb  4.29Kb
montsegur                       => 74.119.119.139                   4.57Kb  3.62Kb  2.59Kb
                                <=                                     0b   11.3Kb  8.04Kb
montsegur                       => 672.bm-nginx-loadbalancer.mgmt      0b   9.15Kb  6.63Kb
                                <=                                     0b   4.43Kb  4.34Kb
montsegur                       => 54.164.141.28                    1.04Kb  3.20Kb  2.29Kb
                                <=                                  4.70Kb  10.2Kb  7.27Kb
montsegur                       => 104.193.83.156                   7.59Kb  3.42Kb  2.44Kb
                                <=                                   208b   9.64Kb  6.88Kb
montsegur                       => 192.16.58.8                      10.6Kb  6.52Kb  5.25Kb
                                <=                                  12.5Kb  6.49Kb  5.31Kb
montsegur                       => 35.190.90.30                     1.93Kb  3.46Kb  2.47Kb
                                <=                                  2.03Kb  8.02Kb  5.73Kb

────────────────────────────────────────────────────────────────────────────────────────
TX:             cum:   179KB   peak:   195Kb          rates:   102Kb   121Kb   102Kb
RX:                    243KB           286Kb                   118Kb   163Kb   139Kb
TOTAL:                 423KB           481Kb                   221Kb   284Kb   241Kb
```

The command lsof (list open file) to check for files<>processes association:

Upon being suspicious something is wrong, the command **lsof** can list you the open processes and to which programs are they associated, on the console run:

```
# lsof
```

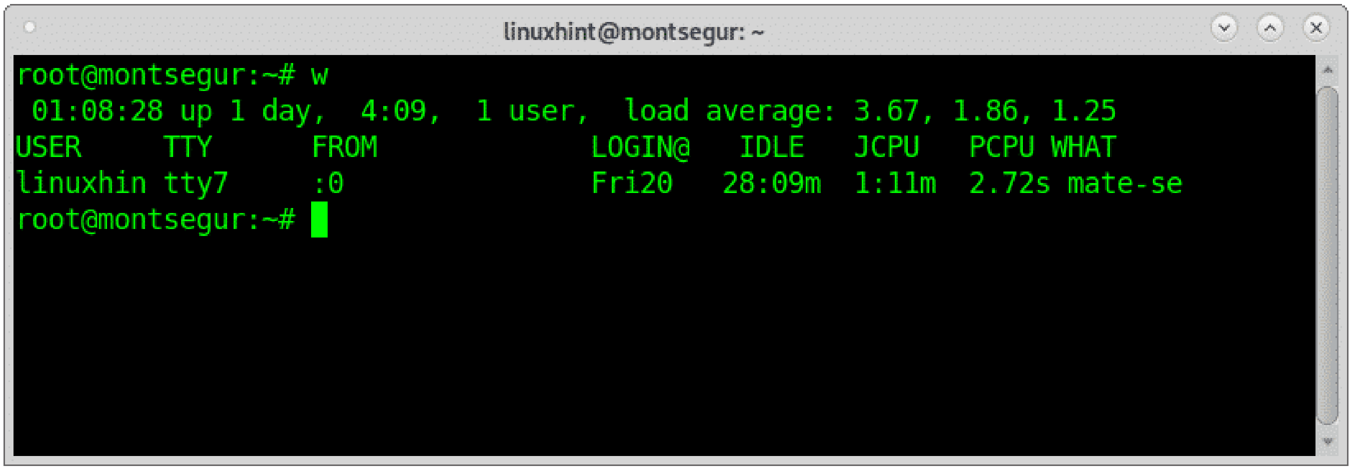


```
COMMAND     PID   TID TASKCMD               USER   FD      TYPE             DEVICE  SIZE/OFF       NODE NAME
systemd       1                             root  cwd       DIR              254,0      4096          2 /
systemd       1                             root  rtd       DIR              254,0      4096          2 /
systemd       1                             root  txt       REG              254,0   1489208    9178960 /usr/lib/system
d/systemd
systemd       1                             root  mem       REG              254,0   1579448    9175680 /usr/lib/x86_64
-linux-gnu/libm-2.28.so
systemd       1                             root  mem       REG              254,0    149704    9175331 /usr/lib/x86_64
-linux-gnu/libudev.so.1.6.13
systemd       1                             root  mem       REG              254,0    137424    9175999 /usr/lib/x86_64
-linux-gnu/libgpg-error.so.0.26.1
systemd       1                             root  mem       REG              254,0     47040    9176942 /usr/lib/x86_64
-linux-gnu/libjson-c.so.3.0.1
systemd       1                             root  mem       REG              254,0     34904    9178786 /usr/lib/x86_64
-linux-gnu/libargon2.so.1
systemd       1                             root  mem       REG              254,0    432664    9178793 /usr/lib/x86_64
-linux-gnu/libdevmapper.so.1.02.1
systemd       1                             root  mem       REG              254,0     30776    9175342 /usr/lib/x86_64
-linux-gnu/libuuid.so.1.3.0
systemd       1                             root  mem       REG              254,0     26544    9175411 /usr/lib/x86_64
-linux-gnu/libattr.so.1.1.2448
systemd       1                             root  mem       REG              254,0   3031904    9178807 /usr/lib/x86_64
-linux-gnu/libcrypto.so.1.1
```

The who and w to know who is logged into your device:

Additionally, to know how to defend your system it is mandatory to know how to react before you are suspicious your system has been hacked. One of the first commands to run before such situation are ***w*** or ***who*** which will show what users are logged into your system and through what terminal. Let’s begin with the command **w:**

```
# w
```



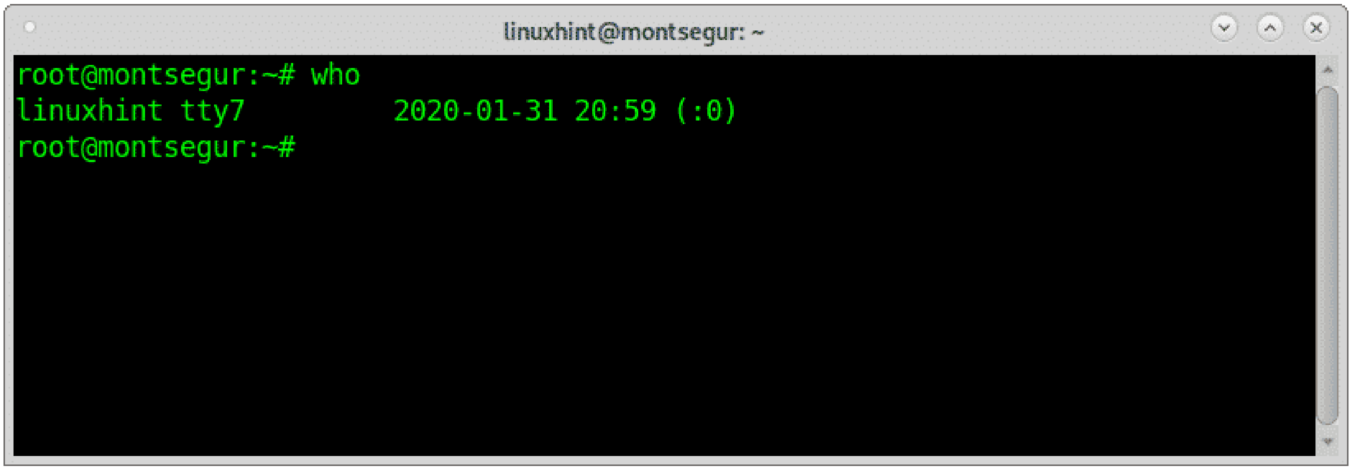
```
root@montsegur:~# w
 01:08:28 up 1 day,  4:09,  1 user,  load average: 3.67, 1.86, 1.25
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
linuxhin tty7     :0               Fri20   28:09m  1:11m  2.72s mate-se
root@montsegur:~#
```

Note: commands "w" and "who" may not show users logged from pseudo terminals like Xfce terminal or MATE terminal.

The column called **USER** displays the **username**, the screenshot above shows the only user logged is linuxhint, the column **TTY** shows the terminal (tty7), the third column **FROM** displays the user address, in this scenario there are not remote users logged in but if they were logged in you could see IP addresses there. The **LOGIN@** column specifies the time in which the user logged in, the column **JCPU** summarizes the minutes of process executed in the terminal or TTY. the **PCPU** displays the CPU used by the process listed in the last column **WHAT**.

While **w** equals to executing **uptime**, **who** and **ps -a** together another alternative, despite with less information is the command "**who**":

```
# who
```



```
root@montsegur:~# who
linuxhint tty7         2020-01-31 20:59 (:0)
root@montsegur:~#
```

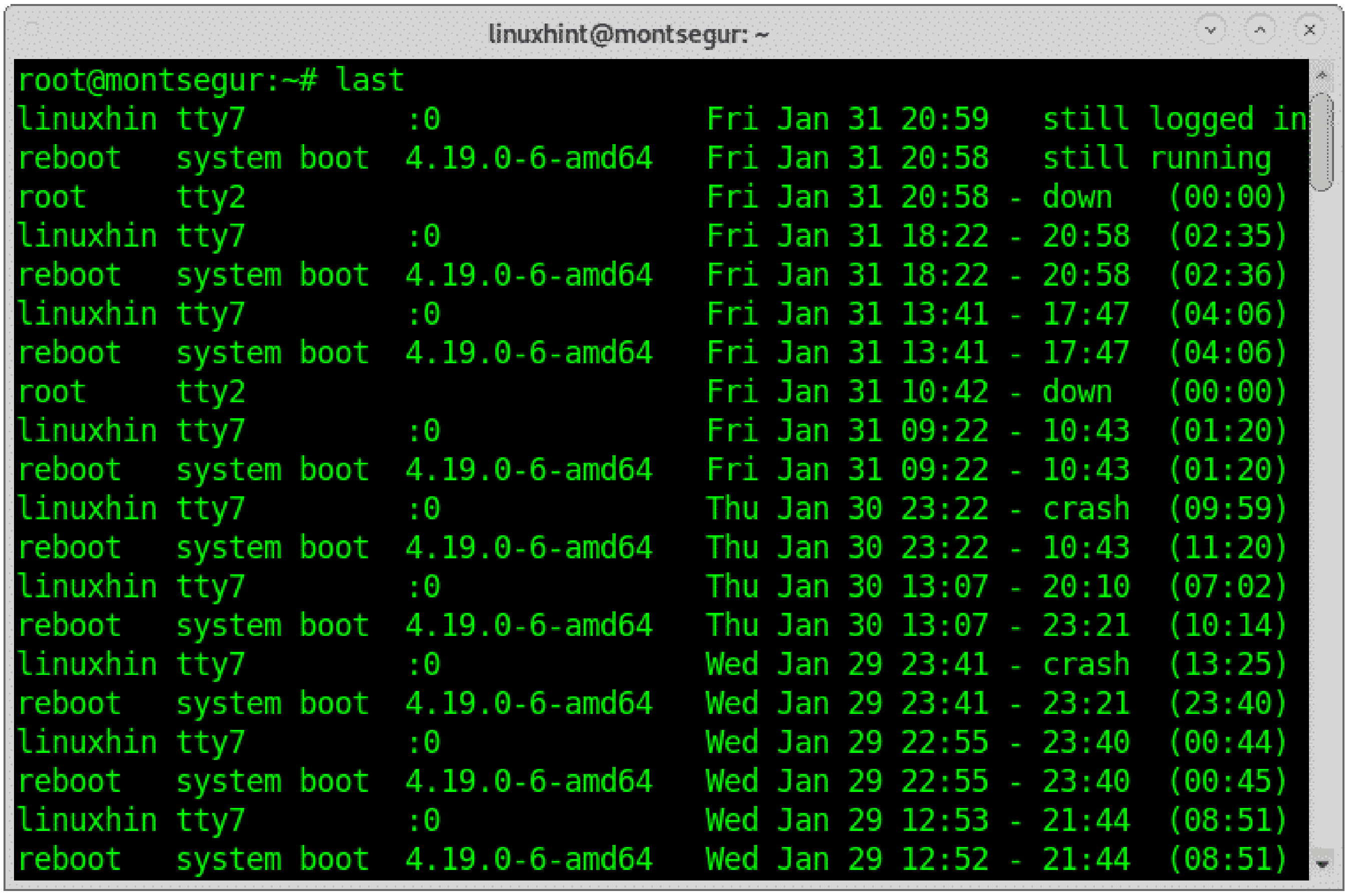
The command ***last*** to check the login activity:

Other way to supervise users' activity is through the command "last" which allows to read the file **wtmp** which contains information on login access, login source, login time, with features to improve specific login events, to try it run:

Checking the login activity with the command **last**:

The command last reads the file **wtmp** to find information on login activity, you can print it by running:

```
# last
```



```
root@montsegur:~# last
linuxhin tty7         :0               Fri Jan 31 20:59   still logged in
reboot   system boot  4.19.0-6-amd64   Fri Jan 31 20:58   still running
root     tty2                          Fri Jan 31 20:58 - down   (00:00)
linuxhin tty7         :0               Fri Jan 31 18:22 - 20:58  (02:35)
reboot   system boot  4.19.0-6-amd64   Fri Jan 31 18:22 - 20:58  (02:36)
linuxhin tty7         :0               Fri Jan 31 13:41 - 17:47  (04:06)
reboot   system boot  4.19.0-6-amd64   Fri Jan 31 13:41 - 17:47  (04:06)
root     tty2                          Fri Jan 31 10:42 - down   (00:00)
linuxhin tty7         :0               Fri Jan 31 09:22 - 10:43  (01:20)
reboot   system boot  4.19.0-6-amd64   Fri Jan 31 09:22 - 10:43  (01:20)
linuxhin tty7         :0               Thu Jan 30 23:22 - crash  (09:59)
reboot   system boot  4.19.0-6-amd64   Thu Jan 30 23:22 - 10:43  (11:20)
linuxhin tty7         :0               Thu Jan 30 13:07 - 20:10  (07:02)
reboot   system boot  4.19.0-6-amd64   Thu Jan 30 13:07 - 23:21  (10:14)
linuxhin tty7         :0               Wed Jan 29 23:41 - crash  (13:25)
reboot   system boot  4.19.0-6-amd64   Wed Jan 29 23:41 - 23:21  (23:40)
linuxhin tty7         :0               Wed Jan 29 22:55 - 23:40  (00:44)
reboot   system boot  4.19.0-6-amd64   Wed Jan 29 22:55 - 23:40  (00:45)
linuxhin tty7         :0               Wed Jan 29 12:53 - 21:44  (08:51)
reboot   system boot  4.19.0-6-amd64   Wed Jan 29 12:52 - 21:44  (08:51)
```

Checking your SELinux status and enable it if needed:

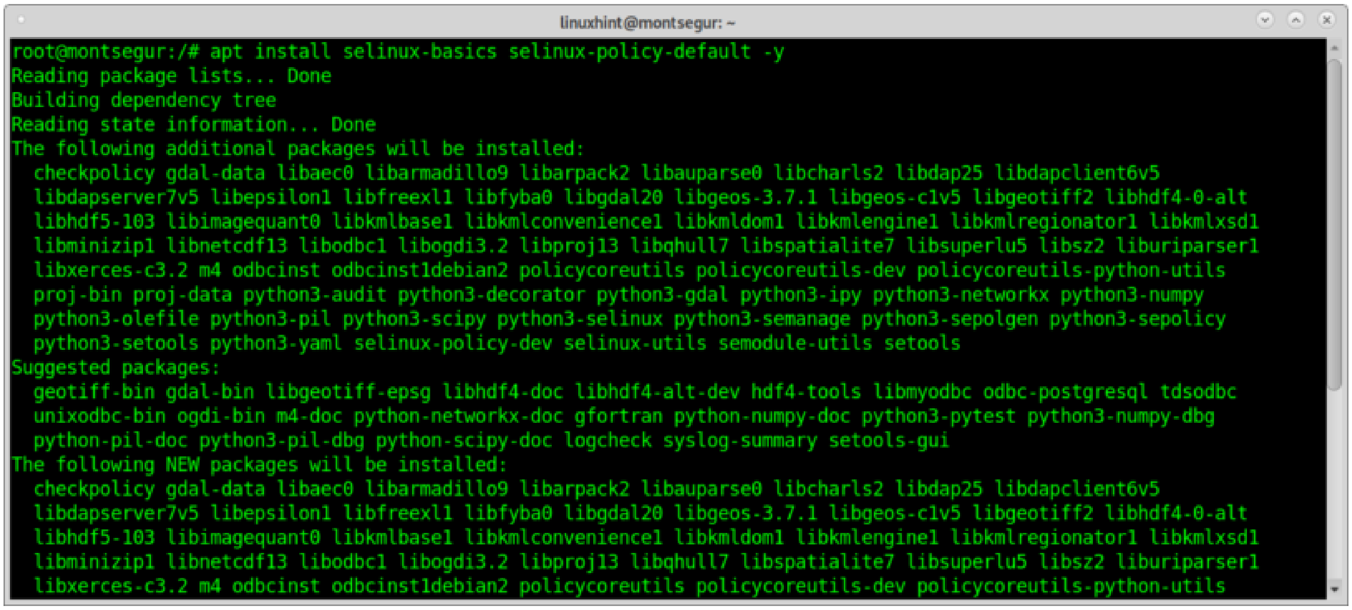
SELinux is restriction system which improves any Linux security, it comes by default on some Linux distributions, it is widely explained here on linuxhint.

You can check your SELinux status by running:

```
# sestatus
```

If you get a command not found error, you can install SELinux by running:

```
#   apt install selinux-basics selinux-policy-default -y
```



```
root@montsegur:/# apt install selinux-basics selinux-policy-default -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  checkpolicy gdal-data libaec0 libarmadillo9 libarpack2 libauparse0 libcharls2 libdap25 libdapclient6v5
  libdapserver7v5 libepsilon1 libfreexl1 libfyba0 libgdal20 libgeos-3.7.1 libgeos-c1v5 libgeotiff2 libhdf4-0-alt
  libhdf5-103 libimagequant0 libkmlbase1 libkmlconvenience1 libkmldom1 libkmlengine1 libkmlregionator1 libkmlxsd1
  libminizip1 libnetcdf13 libodbc1 libogdi3.2 libproj13 libqhull7 libspatialite7 libsuperlu5 libsz2 liburiparser1
  libxerces-c3.2 m4 odbcinst odbcinst1debian2 policycoreutils policycoreutils-dev policycoreutils-python-utils
  proj-bin proj-data python3-audit python3-decorator python3-gdal python3-ipy python3-networkx python3-numpy
  python3-olefile python3-pil python3-scipy python3-selinux python3-semanage python3-sepolgen python3-sepolicy
  python3-setools python3-yaml selinux-policy-dev selinux-utils semodule-utils setools
Suggested packages:
  geotiff-bin gdal-bin libgeotiff-epsg libhdf4-doc libhdf4-alt-dev hdf4-tools libmyodbc odbc-postgresql tdsodbc
  unixodbc-bin ogdi-bin m4-doc python-networkx-doc gfortran python-numpy-doc python3-pytest python3-numpy-dbg
  python-pil-doc python3-pil-dbg python-scipy-doc logcheck syslog-summary setools-gui
The following NEW packages will be installed:
  checkpolicy gdal-data libaec0 libarmadillo9 libarpack2 libauparse0 libcharls2 libdap25 libdapclient6v5
  libdapserver7v5 libepsilon1 libfreexl1 libfyba0 libgdal20 libgeos-3.7.1 libgeos-c1v5 libgeotiff2 libhdf4-0-alt
  libhdf5-103 libimagequant0 libkmlbase1 libkmlconvenience1 libkmldom1 libkmlengine1 libkmlregionator1 libkmlxsd1
  libminizip1 libnetcdf13 libodbc1 libogdi3.2 libproj13 libqhull7 libspatialite7 libsuperlu5 libsz2 liburiparser1
  libxerces-c3.2 m4 odbcinst odbcinst1debian2 policycoreutils policycoreutils-dev policycoreutils-python-utils
```

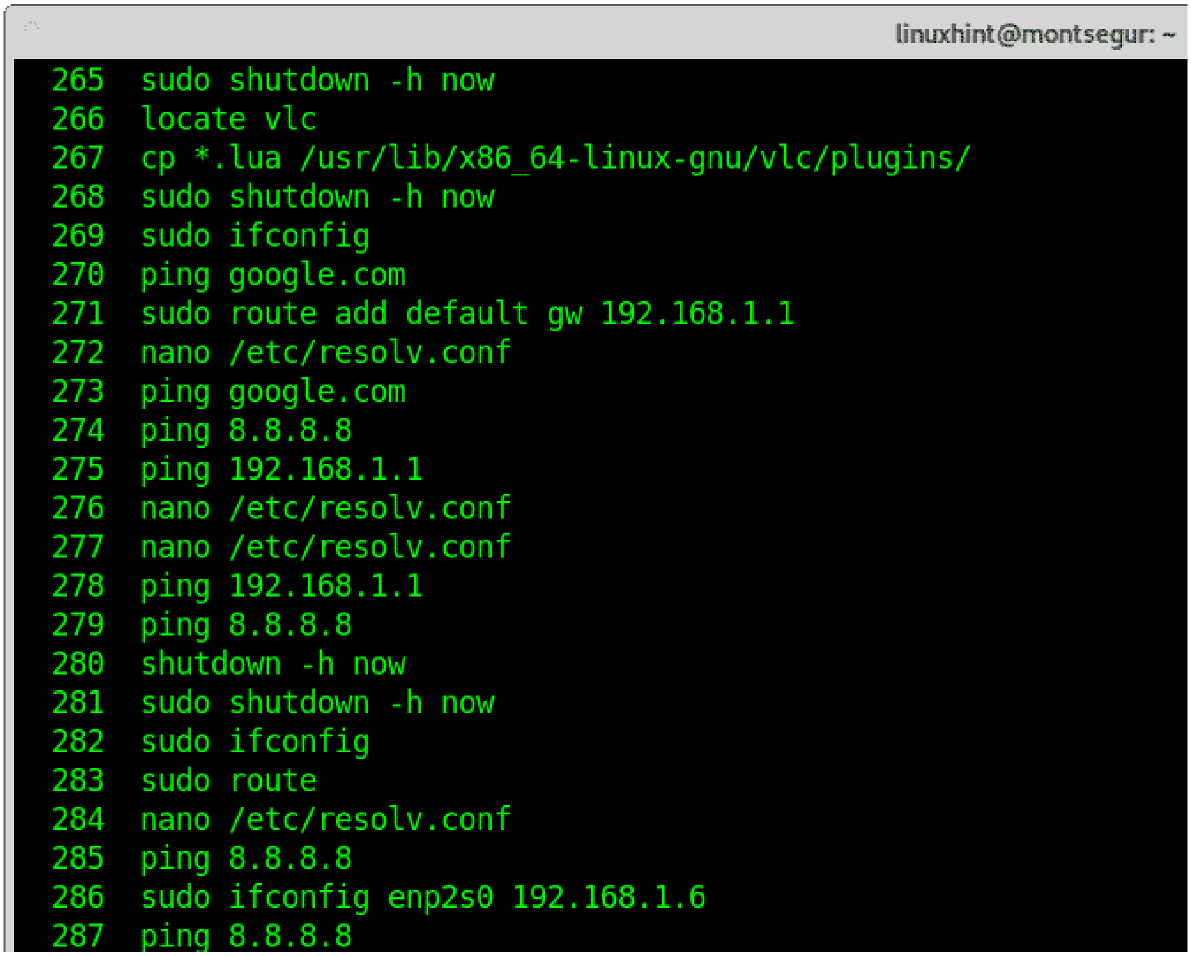
Then run:

```
# selinux-activate
```

Check any user activity using the command **history**:

At any time, you can check any user activity (if you are root) by using the command history logged as the user you want to monitor:

```
# history
```

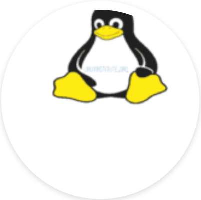


```
 265  sudo shutdown -h now
 266  locate vlc
 267  cp *.lua /usr/lib/x86_64-linux-gnu/vlc/plugins/
 268  sudo shutdown -h now
 269  sudo ifconfig
 270  ping google.com
 271  sudo route add default gw 192.168.1.1
 272  nano /etc/resolv.conf
 273  ping google.com
 274  ping 8.8.8.8
 275  ping 192.168.1.1
 276  nano /etc/resolv.conf
 277  nano /etc/resolv.conf
 278  ping 192.168.1.1
 279  ping 8.8.8.8
 280  shutdown -h now
 281  sudo shutdown -h now
 282  sudo ifconfig
 283  sudo route
 284  nano /etc/resolv.conf
 285  ping 8.8.8.8
 286  sudo ifconfig enp2s0 192.168.1.6
 287  ping 8.8.8.8
```

The command history reads the file bash_history of each user. Of course, this file can be adulterated, and you as root can read this file directly without invoking the command history. Yet, if you want to monitor activity running is recommended.

I hope you found this article on essential Linux security commands useful. Keep following LinuxHint for more tips and updates on Linux and networking.

ABOUT THE AUTHOR



David Adams

David Adams is a System Admin and writer that is focused on open source technologies, security software, and computer systems.

View all posts

RELATED LINUX HINT POSTS

- Check Listening Ports on Linux
- How to Use Topgrade to Update Packages in Linux
- Linux sysfs File System
- OProfile Tutorial
- Syslog Tutorial
- How to Create a New File Using Linux Touch Command
- Stealth Scans With Nmap

Linux Hint LLC, editor@linuxhint.com
1309 S Mary Ave Suite 210, Sunnyvale, CA
94087