iproute

iproute vs net-tools

Version 1.0

© CC-BY SA4
The C-ALE (Cloud & Container Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the Linux Cloud and Container field of computing.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you’ve learned.

This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at https://cm.osssentials.org/

This material is licensed under CC-BY SA4
# Contents

1 iproute vs net-tools ................................................................. 1
  1.1 Welcome ............................................................................. 2
  1.2 history ............................................................................ 3
  1.3 retirement ....................................................................... 10
  1.4 Labs ............................................................................... 20
Chapter 1

iproutve vs net-tools

1.1 Welcome ................................................................. 2
1.2 history .................................................................. 3
1.3 retirement ............................................................... 10
1.4 Labs ................................................................. 20
1.1 Welcome

Hi, I’m Lee Elston, your presenter for this session.
You can contact me at:
lee@matildasystems.com

I am a course maintainer and instructor for The Linux Foundation and maintain:

- LFS311
- LFS430
- LFS416
1.2 history

- collection of tools for controlling network
- in use since kernel version 2.0
- most familiar networking commands
- uses procfs (/proc) and ioctl system call

**net-tools** are getting a little old: [https://github.com/giftnuss/net-tools](https://github.com/giftnuss/net-tools) last update 2010.
Some of our favorite **net-tools** utilities

Table 1.1: *most commonly used net-tools commands*

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifconfig</td>
<td>display and modify interface</td>
</tr>
<tr>
<td>netstat</td>
<td>print information about the network</td>
</tr>
</tbody>
</table>
The complete list of net-tools:

### Table 1.2: net-tools commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifconfig</td>
<td>display and modify interface</td>
</tr>
<tr>
<td>netstat</td>
<td>print information about the network</td>
</tr>
<tr>
<td>arp</td>
<td>print the arp table</td>
</tr>
<tr>
<td>ether-wake</td>
<td>send magic &quot;Wake-On-LAN&quot; packet</td>
</tr>
<tr>
<td>ipmaddr</td>
<td>add, delete, show multicast addresses</td>
</tr>
<tr>
<td>iptunnel</td>
<td>add, delete, show a tunnel</td>
</tr>
<tr>
<td>mii-daig</td>
<td>network adapter control and monitoring</td>
</tr>
<tr>
<td>mii-tool</td>
<td>show, change interface status &quot;Media Independent Interface (MII)&quot;</td>
</tr>
<tr>
<td>nameif</td>
<td>rename an interface</td>
</tr>
<tr>
<td>plipconfig</td>
<td>Parallel Line Internet Protocol (PLIP)</td>
</tr>
<tr>
<td>route</td>
<td>show, add, delete routes</td>
</tr>
<tr>
<td>slattach</td>
<td>attach a network interface to a serial line</td>
</tr>
</tbody>
</table>
ifconfig

- most used networking command
- similar command everywhere
  - Linux
  - Windows
  - OSX
  - Classic Unix and friends
- variations on the name
  - ipconfig
  - ifconfig
- variations on the output
- variations on command line options
# ifconfig wlp107s0
wlp107s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
  inet 192.168.0.7  netmask 255.255.255.0  broadcast 192.168.0.255
  inet6 fe80::53b1:ef35:6e63:8bab  prefixlen 64  scopeid 0x20<link>
  ether 50:5b:c2:d6:fa:f5  txqueuelen 1000  (Ethernet)
  RX packets 819371  bytes 641332500 (641.3 MB)
  RX errors 0  dropped 0  overruns 0  frame 0
  TX packets 547241  bytes 91562359 (91.5 MB)
  TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
netstat

- second most used networking command
- similar command everywhere
  - Linux
  - Windows
  - OSX
  - Classic Unix and friends
- variations on the output
- variations on command line options
### Output of the netstat command

```bash
$ sudo netstat -4tapn
Active Internet connections (servers and established)

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>PID/Program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>192.168.0.7:53</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1076/named</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>192.168.122.1:53</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1076/named</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>127.0.0.1:53</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1076/named</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>127.0.0.53:53</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1076/named</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:22</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1087/sshd</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:25</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>2306/master</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>127.0.0.1:953</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1076/named</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:445</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>2129/smbd</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:2049</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>-</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:38657</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1088/rpc.mountd</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:514</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>889/rsyslogd</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:111</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
<td>1/init</td>
</tr>
<tr>
<td>tcp</td>
<td>32</td>
<td>0</td>
<td>192.168.0.7:35522</td>
<td>50.19.232.147:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>192.168.0.7:55912</td>
<td>104.16.40.2:443</td>
<td>TIME_WAIT</td>
<td>-</td>
</tr>
<tr>
<td>tcp</td>
<td>1</td>
<td>0</td>
<td>192.168.0.7:34522</td>
<td>66.228.47.22:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>1</td>
<td>0</td>
<td>192.168.0.7:34512</td>
<td>66.228.47.22:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>32</td>
<td>0</td>
<td>192.168.0.7:60100</td>
<td>107.167.110.216:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>1</td>
<td>0</td>
<td>192.168.0.7:34510</td>
<td>66.228.47.22:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>1</td>
<td>0</td>
<td>192.168.0.7:34524</td>
<td>66.228.47.22:443</td>
<td>CLOSE_WAIT</td>
<td>2178/opera</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>192.168.0.7:46868</td>
<td>172.217.13.202:443</td>
<td>TIME_WAIT</td>
<td>-</td>
</tr>
</tbody>
</table>
```
1.3 retirement

and now ......

retirement
new ideas, new tools

With the rapid change in Linux over the years, some utilities that seem to work are still around, especially the tools we all learned early on in our exposure to all the *ix environments. So, here we are, nice new tools and we still prefer the old tools. Time to look at some alternatives.
CHAPTER 1. IPROUTE VS NET-TOOLS

ifconfig vs ip

[root@rt ~]# ifconfig ens3
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.122.199 netmask 255.255.255.0 broadcast 192.168.122.255
  inet6 fe80::5054:ff:fe0b:1cd6 prefixlen 64 scopeid 0x20<link>
  ether 52:54:00:0b:1c:d6 txqueuelen 1000 (Ethernet)
  RX packets 132952 bytes 181326140 (172.9 MiB)
  RX errors 0 dropped 8487 overruns 0 frame 0
  TX packets 75972 bytes 5166820 (4.9 MiB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@rt ~]# ip -s a s ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:0b:1c:d6 brd ff:ff:ff:ff:ff:ff
   inet 192.168.122.199/24 brd 192.168.122.255 scope global dynamic ens3
      valid_lft 2751sec preferred_lft 2751sec
      inet6 fe80::5054:ff:fe0b:1cd6/64 scope link
         valid_lft forever preferred_lft forever
   RX: bytes packets errors dropped overrun mcast
      181342589 133196 0 8674 0 0
   TX: bytes packets errors dropped carrier Collins
      5166820 75972 0 0 0 0
One might notice that some additional flags were passed to the `ip` command to make the output look similar to the `ifconfig` command. This comparison used `ifconfig` and `ip -s a s ens3`.

The options for the command `ip` are:

- `-s`, print the link statistics
- `a`, for address
- `s`, for show
- `ens3` is the interface

Yes, it looks different.
The most common use for `ifconfig` is to check:

- what is the ip-address
- is the adapter up or down

How about a shortcut?

```bash
[root@rt ~]# ip -br a
lo    UNKNOWN      127.0.0.1/8   ::1/128
ens3   UP           192.168.122.199/24 fe80::5054:ff:fe0b:1cd6/64
```
This command uses the flag `-br` for “brief”.
The option `-j` can be used to output the information in “json” format. Adding
one more option `-p”pretty” makes the json output much easier to read.

```
[root@rt ~]# ip -br -j -p a s ens3
[
  {
    "ifname": "ens3",
    "operstate": "UP",
    "addr_info": [
      {
        "local": "192.168.122.199",
        "prefixlen": 24
      },
      {
        "local": "fe80::5054:ff:fe0b:1cd6",
        "prefixlen": 64
      }
    ]
  }
]
```
netstat vs ss

netstat

- From the "man" page:
  Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
- Down a bit in that same "man" page:
  This program is mostly obsolete. Replacement for netstat is ss.

ss

- ss is used to dump socket statistics.
- It allows showing information similar to netstat.
- It can display more TCP and state information than other tools.
Some netstat replacements

- Replacement for `netstat` is `ss`.
- Replacement for `netstat -r` is `ip route`.
- Replacement for `netstat -i` is `ip -s link`.
- Replacement for `netstat -g` is `ip maddr`.
The **ss** program dumps socket statistics.

- output is similar to **netstat**
- displays more state information than **netstat**
- displays more **TCP** information than **netstat**

Many of the options for **ss** are very similar to **netstat**. As an example:

```
# ss -tuln
```

and

```
# netstat -tuln
```
compare ss and netstat output

These two commands output almost exactly the same data.

```
[root@rt ~]# ss -ltn4
State    Recv-Q  Send-Q Local Address:Port  Peer Address:Port
LISTEN   0      128  0.0.0.0:5355  0.0.0.0:*  
LISTEN   0      128  0.0.0.0:22   0.0.0.0:*  
LISTEN   0       5  127.0.0.1:631  0.0.0.0:*  
LISTEN   0     100  0.0.0.0:25   0.0.0.0:*  
```

```
[root@rt ~]# netstat -ltn4
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State
tcp   0      0  0.0.0.0:5355  0.0.0.0:* LISTEN
tcp   0      0  0.0.0.0:22   0.0.0.0:* LISTEN
tcp   0      0  127.0.0.1:631 0.0.0.0:* LISTEN
tcp   0      0  0.0.0.0:25   0.0.0.0:* LISTEN
```
1.4 Labs

Exercise 1.1: Verify system is ready for experimenting with ip and ss

Confirm that both net-tools and iproute are installed.

Solution 1.1

1. Install required packages:

   - On CentOS:
     
     # yum install iproute iproute-tc net-tools
   
   - On OpenSUSE:
     
     # zypper install net-tools-deprecated net-tools iproute2
   
   - On Ubuntu:
     
     # apt-get install net-tools iproute2

2. Note: the different commands on different distributions may have or require different security for access. Please use root level access.

   Verify the following commands are present and functioning.

   # ip
   # ifconfig
   # netstat
   # ss

3. Take a closer look at the output from ip in comparison to ifconfig. Look at the contents of ifconfig for any adapter except lo:
1.4. LABS

root@ubuntu:~# ifconfig enp1s0
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.122.42 netmask 255.255.255.0 broadcast 192.168.122.255
  inet6 fe80::5054:ff:fe2a:1b3d prefixlen 64 scopeid 0x20<link>
  ether 52:54:00:2a:1b:3d txqueuelen 1000 (Ethernet)
  RX packets 115179 bytes 125395460 (125.3 MB)
  RX errors 0 dropped 28478 overruns 0 frame 0
  TX packets 43973 bytes 3503100 (3.5 MB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

This is the most common display used to look at the IP-address of a network adapter.
The default output of `ip` is quite different. To use the `ip` command on a single adapter requires some more execution flags,

`ip address show link-name` or a shorter version:

`ip add sh link-name` or even:

`ip a s link-name`

Display the default information of an adapter using `ip`:

# ip a s enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:2a:1b:3d brd ff:ff:ff:ff:ff:ff
   inet 192.168.122.42/24 brd 192.168.122.255 scope global dynamic enp1s0
      valid_lft 3294sec preferred_lft 3294sec
   inet6 fe80::5054:ff:fe2a:1b3d/64 scope link
      valid_lft forever preferred_lft forever

4. Comparing the output from the two commands we see immediately that the byte and packet counters are missing.
Add the adapter statistics into the `ip` command with the option `-s`:

root@ubuntu:~# ip -s a s enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:2a:1b:3d brd ff:ff:ff:ff:ff:ff
   inet 192.168.122.42/24 brd 192.168.122.255 scope global dynamic enp1s0
      valid_lft 2069sec preferred_lft 2069sec
   inet6 fe80::5054:ff:fe2a:1b3d/64 scope link
valid_lft forever preferred_lft forever
RX: bytes packets errors dropped overrun mcast
125642456 116866 0 29262 0 0
TX: bytes packets errors dropped carrier collsns
3641594 44499 0 0 0 0

5. Notice that now the ip command displays nearly exactly the same, plus a little more, information than the old ifconfig command.
The ip command also shows the valid_lft and preferred_lft which are the "life time" timers the kernel uses, when they hit zero the address is removed. (see RFC4862 for more information)

Exercise 1.2: Display an incompatibility between ”ip” and ”ifconfig”
The way that ip-aliases are handled has changed and since ifconfig has not been updated for many years there is a potential issue.
Explore using ifconfig and ip to create and display ip-aliases.

Solution 1.2

1. Create two aliases on the primary adapter, not the loopback, one using ifconfig and one using ip.
Locate the adapter.

    root@ubuntu:~# ip -br a
    lo  UNKNOWN   127.0.0.1/8 ::1/128
    enp1s0  UP     192.168.122.42/24 fe80::5054:ff:fe2a:1b3d/64

2. Add an alias using ifconfig

    root@ubuntu:~# ifconfig enp1s0:1 172.16.42.1 up

3. Add an alias using ip

    root@ubuntu:~# ip addr add 172.16.44.1/24 dev enp1s0
4. Use the `ip` command to view all the adapters

```bash
root@ubuntu:~# ip -s a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
    RX: bytes packets errors dropped overrun mcast
       141300 1379 0 0 0 0
    TX: bytes packets errors dropped carrier collsns
       141300 1379 0 0 0 0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:2a:1b:3d brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.42/24 brd 192.168.122.255 scope global dynamic enp1s0
       valid_lft 3285sec preferred_lft 3285sec
    inet 172.16.42.1/16 brd 172.16.255.255 scope global enp1s0:1
       valid_lft forever preferred_lft forever
    inet 172.16.44.1/24 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe2a:1b3d/64 scope link
       valid_lft forever preferred_lft forever
    RX: bytes packets errors dropped overrun mcast
       126045077 119897 0 31128 0 0
    TX: bytes packets errors dropped carrier collsns
       3718401 45030 0 0 0 0
```

Notice that both of the aliases are listed in the output.

5. Use `ifconfig` to view all the adapters.

```bash
root@ubuntu:~# ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.122.42 netmask 255.255.255.0 broadcast 192.168.122.255
    inet6 fe80::5054:ff:fe2a:1b3d/64 prefixlen 64 scopeid 0x20<link>
```

iproute
ether 52:54:00:2a:1b:3d txqueuelen 1000 (Ethernet)
RX packets 120027 bytes 126053820 (126.0 MB)
RX errors 0 dropped 31216 overruns 0 frame 0
TX packets 45039 bytes 3719025 (3.7 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp1s0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
   inet 172.16.42.1 netmask 255.255.0.0 broadcast 172.16.255.255
   ether 52:54:00:2a:1b:3d txqueuelen 1000 (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
   inet 127.0.0.1 netmask 255.0.0.0
   inet6 ::1 prefixlen 128 scopeid 0x10<host>
   loop txqueuelen 1000 (Local Loopback)
RX packets 1379 bytes 141300 (141.3 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1379 bytes 141300 (141.3 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Notice the alias 172.16.44.1 is missing.

**Exercise 1.3: Some ss examples**

Using the `man ss` as a reference, experiment with the commands as shown and others more specific to your interest.

1. Display all TCPv4 sockets.

   `ss -t -a -4`

   

<table>
<thead>
<tr>
<th>State</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTEN</td>
<td>0</td>
<td>100</td>
<td>0.0.0.0:imaps</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>LISTEN</td>
<td>0</td>
<td>64</td>
<td>0.0.0.0:nfs</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>LISTEN</td>
<td>0</td>
<td>128</td>
<td>0.0.0.0:37089</td>
<td>0.0.0.0:*</td>
</tr>
</tbody>
</table>

iproute © CC-BY SA4
1.4. LABS

<table>
<thead>
<tr>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0:36353</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:pop3s</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:pop3</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:imap2</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:48431</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:sunrpc</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>127.0.0.53%lo:domain</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:ssh</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>127.0.0.1:ipp</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:smtp</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>0.0.0.0:38335</td>
<td>0.0.0.0:*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1:37604</td>
<td>127.0.0.1:ssh</td>
</tr>
<tr>
<td>127.0.0.1:ssh</td>
<td>127.0.0.1:37604</td>
</tr>
</tbody>
</table>

2. Display all ipv4 UDP sockets

```
ss -ua4
```

<table>
<thead>
<tr>
<th>State</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:40477</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:ipp</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:54961</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:39620</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:35662</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:nfs</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>127.0.0.53%lo:domain</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>192.168.122.42%enp1s0:bootpc</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:sunrpc</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:mdns</td>
<td>0.0.0.0:*</td>
</tr>
<tr>
<td>UNCONN</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:50559</td>
<td>0.0.0.0:*</td>
</tr>
</tbody>
</table>

3. Display all the established connections that have a source or destination on port 21 (ssh)

```
# ss -o state established '( dport = :ssh or sport = :ssh )'
```

<table>
<thead>
<tr>
<th>Netid</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
</tr>
</thead>
</table>
4. An active web server is required for this example. This example shows packet with status **time-wait**, destination port of **http** or **https** and destination **ip network**.

```
# ss -t -o state time-wait '(' dport = :http or dport = :https ')' dst 192.168.122/24
```

<table>
<thead>
<tr>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address:Port</th>
<th>Peer Address:Port</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>192.168.122.42:49460</td>
<td>192.168.122.42:http</td>
<td>timer:(timewait,54sec,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>192.168.122.42:49466</td>
<td>192.168.122.42:http</td>
<td>timer:(timewait,56sec,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>192.168.122.42:49464</td>
<td>192.168.122.42:http</td>
<td>timer:(timewait,55sec,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>192.168.122.42:49462</td>
<td>192.168.122.42:http</td>
<td>timer:(timewait,54sec,0)</td>
</tr>
</tbody>
</table>