6 tcpdump network traffic filter options

https://www.redhat.com/sysadmin/tcpdump-part-one

1. Option -D

`tcpdump` with `-D` provides a list of devices from which you can capture traffic. This option identifies what devices `tcpdump` knows about. Once you see this list, you can decide which interface you want to capture the traffic on. It also tells you if the interface is Up, Running, and whether it is a Loopback interface, as you can see below:

```
# tcpdump -D
1.tun0 [Up, Running]
2.wlp0s20f3 [Up, Running]
3.lo [Up, Running, Loopback]
4.any (Pseudo-device that captures on all interfaces) [Up, Running]
5.virbr0 [Up]
6.docker0 [Up]
7.enp0s31f6 [Up]
```

2. Option -c X

The `-c` option captures **X** number of packets and then stops. Otherwise, `tcpdump` will keep running indefinitely. So when you want to capture only a small sample set of packets, you can use this option. However, if there is no activity on the interface, `tcpdump` keeps waiting.

```
# tcpdump -c 5 -i any
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
16:19:22.128996 ARP, Request who-has _gateway tell 192.168.86.81, length 28
```

```
16:19:22.130560 IP 172.217.222.189.https > kkulkarni.58810: Flags [P.], seq 3506342975:3506343029, ack 2537104576, win 377, options [nop,nop,TS val 4137065873 ecr 75405758], length 54
16:19:22.130642 IP kkulkarni.58810 > 172.217.222.189.https: Flags [.], ack 54, win 501, options [nop,nop,TS val 75422756 ecr 4137065873], length 016:19:22.131198 IP ovpn-3-80.rdu2.redhat.com.36380 > infoblox-trust01.intranet.prod.int.rdu2.redhat.com.domain: 53320+ PTR? 1.86.168.192.in-addr.arpa. (43)
16:19:22.131395 IP kkulkarni.53013 > ovpn-rdu2-alt.redhat.com.https: UDP, length 95
5 packets captured
49 packets received by filter
37 packets dropped by kernel
```

3. Option -n

It is usually easier to work if you use IP addresses instead of names, such as **kkulkarni.53013** as shown in the above output. You can use `-n` for this.

```
# tcpdump -c 5 -i any -n
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
16:20:21.523375 IP 172.217.9.206.https > 192.168.86.31.34288: Flags [P.], seq 723352132:723352349, ack 2124268216, win 1059, options [nop,nop,TS val 2934032467 ecr 824781066], length 217
16:20:21.563992 IP 192.168.86.31.34288 > 172.217.9.206.https: Flags [.], ack 217, win 12654, options [nop,nop,TS val 824783221 ecr 2934032467], length 0
16:20:22.956717 IP 192.168.86.83.mdns > 224.0.0.251.mdns: 0 [2q] [1au] PTR (QU)? _companion-link._tcp.local. PTR (QU)? _homekit._tcp.local. (88)16:20:22.956839 IP 192.168.86.83.mdns > 224.0.0.251.mdns: 0*- [0q] 2/0/3 (Cache flush)
16:20:22.956932 IP6 fe80::2:8c40:fdea:5a16.mdns > ff02::fb.mdns: 0*- [0q] 2/0/3 (Cache flush) PTR local., (Cache flush) PTR local. (214)
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

4. Option -s

`tcpdump` with `-sXXX` helps you control the capture size. On the second line in the previous output you can see it says capture size 262144 bytes, which is much larger than the packet. You can use `-s` to change the capture size. If you just want to inspect the packet headers, then you can use a

smaller size for the capture. See the example below:

```
# tcpdump -c 5 -i any -n -s64
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 64 bytes
16:24:39.909994 IP 10.22.3.80.46368 > 10.11.200.20.ldap: Flags [.], ack 2583785634, win 502, options
[nop,nop,TS[|tcp]>
16:24:39.910118 IP 192.168.86.31.53013 > 66.187.232.72.https: UDP, length 76
16:24:39.981646 IP 192.168.86.111.mdns > 224.0.0.251.mdns: 0 [5a] [28q] [1n] [1au][|domain]
16:24:39.983954 IP 192.168.86.111.mdns > 224.0.0.251.mdns: 0*- [0q] 2/0/1[|domain]
16:24:40.186150 IP 192.168.86.111.mdns > 224.0.0.251.mdns: 0 [1n] [1au][|domain]
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

5. Port captures

`tcpdump` allows you to specify network packets that are either using some port **X** as source or destination. For example, to capture DNS traffic, you can use `port 53`. You could prefix the **port** keyword with **src**/**dst** as `src port 53` or `dst port 53` and filter it even further.

```
# tcpdump -i any port 53 -n
16:49:58.979410 IP 10.22.3.80.46391 > 10.11.5.19.domain: 31741+ A? youtube.com. (29)
16:49:58.979450 IP 10.22.3.80.46391 > 10.11.5.19.domain: 4579+ AAAA? youtube.com. (29)
16:49:58.985835 IP 10.11.5.19.domain > 10.22.3.80.44202: 8898 NXDomain 0/1/0 (154)
16:49:58.986761 IP 10.22.3.80.38074 > 10.11.5.19.domain: 43241+ PTR? 31.86.168.192.in-addr.arpa. (44)
16:49:59.015164 IP 10.11.5.19.domain > 10.22.3.80.38074: 43241 NXDomain 0/1/0 (122)
16:49:59.015209 IP 10.11.5.19.domain > 10.22.3.80.46391: 4579 1/0/0 AAAA 2607:f8b0:4004:810::200e (57)
16:49:59.015231 IP 10.11.5.19.domain > 10.22.3.80.46391: 31741 1/0/0 A 172.217.15.78 (45)
16:49:59.015831 IP 10.22.3.80.51955 > 10.11.5.19.domain: 2503+ PTR? 1.122.168.192.in-addr.arpa. (44)
16:49:59.041490 IP 10.11.5.19.domain > 10.22.3.80.51955: 2503 NXDomain 0/1/0 (122)
```

6. Option -w

If you want to write the output of `tcpdump` to a file, use the option `-w .pcap` to write to a file. If you want to see how many packages were written, you can add `-v`.

```
# tcpdump -c 4 -i any port 53 -w dns.pcap -v
tcpdump: data link type LINUX_SLL2
dropped privs to tcpdump
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
4 packets captured
24 packets received by filter
0 packets dropped by kernel
```

Revision #1
Created 23 December 2023 15:17:04 by ColtM
Updated 7 August 2024 23:24:39 by ColtM