Cybersecurity
Everett HS
Mr. Plotnick

TCPDump Tutorial

Imagine if you could listen to all the conversations being held in the lunchroom at the same time. Network packet analysis essentially allows your computer to see all the messages being sent along the wire. You can look at messages being sent between any two points, filter to look for specific messages and output your captures to a text file for deeper analysis.

TCPDump is a packet capturing program. It is a command line utility so everything you type in will be at the system prompt **$**

You will need to log into the classroom Linux server to use this program.

Note: Options for using the program are CASE SENSITIVE. Carefully read the instructions since a capital D is not the same as a lowercase d.

Let's get started

tcpdump -D

This will list all the available interfaces (hardware on the computer that is capable of sending or receiving network data).

Here is an example from my computer:

```
nplotnick@ehs_cyber1:~$ tcpdump -D
1.enp2s0 [Up, Running]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.nflog (Linux netfilter log (NFLOG) interface)
5.nfqueue (Linux netfilter queue (NFQUEUE) interface)
6.usbmon1 (USB bus number 1)
7.usbmon2 (USB bus number 2)
8.usbmon3 (USB bus number 3)
9.usbmon4 (USB bus number 4)
```

In the above, we will be focusing on **enp2s0** as this is the name of Ethernet port on the computer. On some computers the name of the ports may be different.

The Ethernet port is the one on the back of the computer with the cable running into the wall jack. This wire connects the computer to the building network and we can look at virtually all the traffic being sent.

Capture some packets

tcpdump -i enp2s0

Your screen should immediately start filling up with lines of data that looks like this:

```
14:48:19.666006 IP 10.1.11.109.mdns > 224.0.0.251.mdns: 0 TXT (QU)? RICOH MP
6503 [002673EA818C]._uscan._tcp.local. (64)
14:48:19.666010 IP6 fe80::1846:51b5:b562:37c0.mdns > ff02::fb.mdns: 0 TXT (QU)?
RICOH MP 6503 [002673EA818C]._uscan._tcp.local. (64)
14:48:19.666086 IP ehs_cyber1.ssh > 10.1.7.39.40428: Flags [P.], seq 480244:480448,
ack 397, win 501, options [nop,nop,TS val 1513742659 ecr 3834977648], length 204
```

Make it stop!

<ctrl> c

This will stop the capturing.

Limiting the capture to a set number of packets

As you can see, there are thousands of packets captured every second. In most cases, a network manager or cybersecurity specialist will just need to capture a sample of packets to do their work.

tcpdump -c 10

In the above example, we are only capturing 10 packets. Try this and see what your results are.

Your results should look like this:

```
=50484248-4338-3135-3938-40b03423294b" "kind=document,envelope,photo"
"PaperMax=legal-A4" "TLS=1.2" "usb_MFG=Hewlett-Packard" "usb_MDL=HP LaserJet
M402n" "mac=40:b0:34:23:29:4b" "print_wfds=T" "mopria_certified=1.2", (Cache flush)
TXT "UUID=50484248-4338-3135-3938-40b03423294b" "txtvers=1" "qtotal=1" "ty=HP
LaserJet M402n" "product=(HP LaserJet M402n)" "priority=40"
"adminurl=http://KathleenMcCormack.local." "note=Guidance Office 1709"
"Transparent=T" "Binary=T" "TBCP=T", (Cache flush) TXT
"UUID=50484248-4338-3135-3938-40b03423294b" "txtvers=1" "qtotal=1" "rp=RAW"
"ty=HP LaserJet M402n" "product=(HP LaserJet M402n)" "priority=50"
"adminurl=http://KathleenMcCormack.local." "note=Guidance Office 1709"
"Transparent=T" "Binary=T" "TBCP=T" (1326)
14:55:57.633914 ARP, Request who-has 10.1.3.246 tell 10.1.5.117, length 46
14:55:57.633946 IP6 fe80::56bf:64ff:fe82:6ada > ff02::1:ff64:826a: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff64:826a, length 24
14:55:57.634196 IP6 fe80::56bf:64ff:fe82:6ada > ff02::1:ff64:826a: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff64:826a, length 24
14:55:57.637181 IP6 fe80::56bf:64ff:fe82:6ada > ff02::1:ff64:826a: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff64:826a, length 24
14:55:57.646837 IP ehs_cyber1.ssh > 10.1.7.39.40428: Flags [P.], seq
3397029772:3397030832, ack 3416998134, win 501, options [nop,nop,TS val
1514200638 ecr 3835435385], length 1060
14:55:57.646885 IP6 fe80::56bf:64ff:fe82:6ada > ff02::1:ff64:826a: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff64:826a, length 24

10 packets captured
176 packets received by filter
158 packets dropped by kernel
```

What are the dropped packets and why are we getting them?

There are many thousands of packets being transmitted on the network every second. The computer can only **parse** (examine and analyze) a finite number of packets at a time. We can increase the **buffer** (amount of memory allocated to the capture) with another option in the command.

Increasing the packet buffer

tcpdump -c 10 -B 4096

```
10 packets captured
40 packets received by filter
12 packets dropped by kernel
1 packet dropped by interface
```

As you can see, increasing the buffer to 4096 greatly reduced the number of packets that were dropped.

Note the switches are a lowercase c and a uppercase B.

Numbering the packets

There are many switches that can be combined when capturing packets. For example, you can have tcpdump number each packet.

tcpdump --number -c 5

```
1  16:14:06.909652 IP6 fe80::e4d:e9ff:fe9e:3e99 > ff02::2:ff59:4ff1: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::2:ff59:4ff1, length 24
2  16:14:06.909662 IP6 fe80::e4d:e9ff:fe9e:3e99 > ff02::1:ff9e:3e99: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff9e:3e99, length 24
3  16:14:06.909730 IP 10.1.4.243.52974 > 239.255.255.250.1900: UDP, length 125
4  16:14:06.909856 IP6 fe80::82a:dcb0:f98b:135f > ff02::2:ff73:7ff7: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::2:ff73:7ff7, length 24
 5  16:14:06.909860 IP6 fe80::82a:dcb0:f98b:135f > ff02::1:ff8b:135f: HBH ICMP6,
multicast listener reportmax resp delay: 0 addr: ff02::1:ff8b:135f, length 24
```

As you can see, I captured 5 packets and they are numbered. This may make it easier to read the information.

Simplify the output

We can use the -q command line switch to reduce the amount of information being displayed.

tcpdump -c 10 -q

```
$ tcpdump -c 10 -q
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp2s0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:17:32.695187 IP ehs_cyber1.ssh > 10.1.7.39.33880: tcp 188
```

```
16:17:32.695508 ARP, Request who-has 10.1.3.176 tell 10.1.0.1, length 46
16:17:32.695944 IP ehs_cyber1.42004 > 10.1.0.10.domain: UDP, length 51
16:17:32.698541 IP6 fe80::1471:6ab0:5deb:5d4b > ff02::1:fffb:feba: ICMP6, neighbor
solicitation, who has fe80::4a6:4c99:39fb:feba, length 32
16:17:32.704207 ARP, Request who-has ehs_cyber1 tell 10.1.0.10, length 46
16:17:32.704218 ARP, Reply ehs_cyber1 is-at a4:1f:72:8d:44:74 (oui Unknown), length
28
16:17:32.704328 IP 10.1.0.10.domain > ehs_cyber1.42004: UDP, length 51
16:17:32.713980 IP ehs_cyber1.ssh > 10.1.7.39.33880: tcp 100
16:17:32.731142 IP6 fe80::707b:43bd:97a6:38 > ff02::c: HBH ICMP6, multicast listener
reportmax resp delay: 0 addr: ff02::c, length 24
16:17:32.739643 IP ehs_cyber1.ssh > 10.1.7.39.33880: tcp 108
10 packets captured
590 packets received by filter
575 packets dropped by kernel
```

Expanded output

We can use the -v switch to increase the amount of information captured. The -v option means verbose.

tcpdump -c 5 -v

```
$ tcpdump -c 5 -v
tcpdump: listening on enp2s0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:01:02.015140 IP (tos 0x10, ttl 64, id 49578, offset 0, flags [DF], proto TCP (6), length 176)
    ehs_cyber1.ssh > 10.1.7.39.34420: Flags [P.], cksum 0x79c8 (correct), seq
2584250202:2584250326, ack 2539795701, win 501, options [nop,nop,TS val 1521704972 ecr
844429096], length 124
17:01:02.015305 IP (tos 0x10, ttl 64, id 49579, offset 0, flags [DF], proto TCP (6), length 184)
    ehs_cyber1.ssh > 10.1.7.39.34420: Flags [P.], cksum 0x45e8 (correct), seq 124:256, ack 1,
win 501, options [nop,nop,TS val 1521704972 ecr 844429096], length 132
17:01:02.015330 IP (tos 0x0, ttl 250, id 12333, offset 0, flags [none], proto UDP (17), length
145)
    10.1.3.171.61111 > 255.255.255.255.61111: UDP, length 117
17:01:02.015337 IP (tos 0x0, ttl 250, id 12334, offset 0, flags [none], proto UDP (17), length 94)
    10.1.3.171.61111 > 255.255.255.255.61111: UDP, length 66
17:01:02.015846 IP (tos 0x0, ttl 64, id 38199, offset 0, flags [DF], proto UDP (17), length 79)
    ehs_cyber1.39446 > 10.1.0.10.domain: 32839+ [1au] PTR? 39.7.1.10.in-addr.arpa. (51)
```

There are many options available

```
tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]

        [ -c count ] [ --count ] [ -C file_size ]
        [ -E spi@ipaddr algo:secret,... ]
        [ -F file ] [ -G rotate_seconds ] [ -i interface ]
        [ --immediate-mode ] [ -j tstamp_type ] [ -m module ]
        [ -M secret ] [ --number ] [ --print ] [ -Q in|out|inout ]
        [ -r file ] [ -s snaplen ] [ -T type ] [ --version ]
        [ -V file ] [ -w file ] [ -W filecount ] [ -y datalinktype ]
        [ -z postrotate-command ] [ -Z user ]
        [ --time-stamp-precision=tstamp_precision ]
        [ --micro ] [ --nano ]
        [ expression ]
```

Capture data to a file

You can save the data from tcpdump into a file. This will allow you to analyze the information with other programs at a later time.

tcpdump -c 10 -q > capture.txt

In the above example, I have used the same command used earlier. By placing the > character, the computer will send the data from the tcpdump program and create a file called capture.txt instead of displaying it on the screen.

Just use a text editor like Nano or the cat command to read the file.

cat capture.txt

References:
https://www.howtoforge.com/linux-tcpdump-command/

https://www.tcpdump.org/