example tcp dump activity

Use `ifconfig` to identify the interfaces that are available:

```
sudo ifconfig
```

example output

```
analyst@b4aade4b3e15:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 172.18.0.2  netmask 255.255.0.0  broadcast 172.18.255.255
        ether 02:42:ac:12:00:02  txqueuelen 0  (Ethernet)
        RX packets 760  bytes 13683706 (13.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 523  bytes 44695 (43.6 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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 53  bytes 8173 (7.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53  bytes 8173 (7.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The Ethernet network interface is identified by the entry with the `eth` prefix.

So, in this lab, you'll use `eth0` as the interface that you will capture network packet data from in the following tasks.

Use `tcpdump` to identify the interface options available for packet capture:

```
sudo tcpdump -D
```

example output

```
analyst@b4aade4b3e15:~$ sudo tcpdump -D
1.eth0 [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)
```

This command will also allow you to identify which network interfaces are available. This may be useful on systems that do not include the `ifconfig` command.

Inspect the network traffic of a network interface with tcpdump

In this task, you must use `tcpdump` to filter live network packet traffic on an interface.

- Filter live network packet data from the `eth0` interface with `tcpdump`:

```
sudo tcpdump -i eth0 -v -c5
```

- This command will run `tcpdump` with the following options:
  - `-i eth0`: Capture data specifically from the `eth0` interface.
  - `-v`: Display detailed packet data.
  - `-c5`: Capture 5 packets of data.

  Now, let's take a detailed look at the packet information that this command has returned. Some of your packet traffic data will be similar to the following example output:

```
analyst@b4aade4b3e15:~$ sudo tcpdump -i eth0 -v -c5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:34:41.849715 IP (tos 0x0, ttl 64, id 5124, offset 0, flags [DF], proto TCP (6), length 113)
    b4aade4b3e15.5000 > nginx-us-central1-b.c.qwiklabs-terminal-vms-prod-00.internal.35886:
Flags [P.], cksum 0x588b (incorrect -> 0x2db4), seq 824003473:824003534, ack 494355080, win
501, options [nop,nop,TS val 3565226228 ecr 381006698], length 61
13:34:41.849984 IP (tos 0x0, ttl 63, id 46029, offset 0, flags [DF], proto TCP (6), length 52)
    nginx-us-central1-b.c.qwiklabs-terminal-vms-prod-00.internal.35886 > b4aade4b3e15.5000:
Flags [.], cksum 0xf838 (correct), ack 61, win 507, options [nop,nop,TS val 381006741 ecr
3565226228], length 0
13:34:41.850797 IP (tos 0x0, ttl 64, id 56996, offset 0, flags [DF], proto UDP (17), length
69)
    b4aade4b3e15.32961 > metadata.google.internal.domain: 15232+ PTR? 2.0.17.172.in-addr.arpa.
```

```
(41)
13:34:41.853851 IP (tos 0x0, ttl 63, id 0, offset 0, flags [none], proto UDP (17), length 143)
    metadata.google.internal.domain > b4aade4b3e15.32961: 15232 1/0/0 2.0.17.172.in-addr.arpa.
PTR nginx-us-central1-b.c.qwiklabs-terminal-vms-prod-00.internal. (115)
13:34:41.854922 IP (tos 0x0, ttl 64, id 9599, offset 0, flags [DF], proto UDP (17), length 74)
    b4aade4b3e15.56670 > metadata.google.internal.domain: 49479+ PTR? 254.169.254.169.in-
addr.arpa. (46)
5 packets captured
6 packets received by filter
0 packets dropped by kerne
```

The specific packet data in your lab may be in a different order and may even be for entirely different types of network traffic. The specific details, such as system names, ports, and checksums, will definitely be different. You can run this command again to get different snapshots to outline how data changes between packets.

Exploring network packet details

In this example, you'll identify some of the properties that `tcpdump` outputs for the packet capture data you've just seen.

1. In the example data at the start of the packet output, `tcpdump` reported that it was listening on the `eth0` interface, and it provided information on the link type and the capture size in bytes:

   ```
   tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
   ```

2. On the next line, the first field is the packet's timestamp, followed by the protocol type, IP:

   ```
   22:24:18.910372 IP
   ```

3. The verbose option, `-v`, has provided more details about the IP packet fields, such as TOS, TTL, offset, flags, internal protocol type (in this case, TCP (6)), and the length of the outer IP packet in bytes:

   ```
   (tos 0x0, ttl 64, id 5802, offset 0, flags [DF], proto TCP (6), length 134)
   ```

   The specific details about these fields are beyond the scope of this lab. But you should know that these are properties that relate to the IP network packet.

4. In the next section, the data shows the systems that are communicating with each other:

```
7acb26dc1f44.5000 > nginx-us-east1-c.c.qwiklabs-terminal-vms-prod-00.internal.59788:
```

By default, `tcpdump` will convert IP addresses into names, as in the screenshot. The name of your Linux virtual machine, also included in the command prompt, appears here as the source for one packet and the destination for the second packet. In your live data, the name will be a different set of letters and numbers.
The direction of the arrow (>) indicates the direction of the traffic flow in this packet. Each system name includes a suffix with the port number (.5000 in the screenshot), which is used by the source and the destination systems for this packet.

5. The remaining data filters the header data for the inner TCP packet:

```
Flags [P.], cksum 0x5851 (incorrect > 0x30d3), seq 1080713945:1080714027, ack
62760789, win 501, options [nop,nop,TS val 1017464119 ecr 3001513453], length 82
```

The flags field identifies TCP flags. In this case, the P represents the push flag and the period indicates it's an ACK flag. This means the packet is pushing out data.
The next field is the TCP checksum value, which is used for detecting errors in the data. This section also includes the sequence and acknowledgment numbers, the window size, and the length of the inner TCP packet in bytes.

Capture network traffic with tcpdump

In this task, you will use `tcpdump` to save the captured network data to a packet capture file.

In the previous command, you used `tcpdump` to stream all network traffic. Here, you will use a filter and other `tcpdump` configuration options to save a small sample that contains only web (TCP port 80) network packet data.

1. Capture packet data into a file called `capture.pcap`:

```
sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &
```

```
analyst@b4aade4b3e15:~$ sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &
[1] 12811
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

You may need to press the **ENTER** key to get your command prompt back after running this command.
This command will run `tcpdump` in the background with the following options:

- `-i eth0`: Capture data from the `eth0` interface.
- `-nn`: Do not attempt to resolve IP addresses or ports to names.This is best practice from a security perspective, as the lookup data may not be valid. It also prevents malicious actors from being alerted to an investigation.

- `-c9`: Capture 9 packets of data and then exit.
- `port 80`: Filter only port 80 traffic. This is the default HTTP port.
- `-w capture.pcap`: Save the captured data to the named file.
- `&`: This is an instruction to the Bash shell to run the command in the background.

This command runs in the background, but some output text will appear in your terminal. The text will not affect the commands when you follow the steps for the rest of the lab.

2. Use `curl` to generate some HTTP (port 80) traffic:

```
curl opensource.google.com
```

When the `curl` command is used like this to open a website, it generates some HTTP (TCP port 80) traffic that can be captured.

```
analyst@b4aade4b3e15:~$ curl opensource.google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://opensource.google/">here</A>.
</BODY></HTML>
analyst@b4aade4b3e15:~$ 9 packets captured
10 packets received by filter
0 packets dropped by kernel

[1]+  Done                    sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap
```

3. Verify that packet data has been captured:

```
ls -l capture.pcap
```

Note: *The "Done" in the output indicates that the packet was captured.*

```
analyst@b4aade4b3e15:~$ ls -l capture.pcap
-rw-r--r-- 1 root root 1445 Sep 13 13:51 capture.pcap
```

Filter the captured packet data

In this task, use `tcpdump` to filter data from the packet capture file you saved previously.

1. Use the `tcpdump` command to filter the packet header data from the `capture.pcap` capture file:

```
sudo tcpdump -nn -r capture.pcap -v
```

example output:

```
analyst@b4aade4b3e15:~$ sudo tcpdump -nn -r capture.pcap -v
reading from file capture.pcap, link-type EN10MB (Ethernet)
13:51:00.530684 IP (tos 0x0, ttl 64, id 58151, offset 0, flags [DF], proto TCP (6), length 60)
    172.18.0.2.48328 > 64.233.182.139.80: Flags [S], cksum 0xa3b7 (incorrect -> 0xfeb3), seq 3977332362, win 65320, options [mss 1420,sackOK,TS val 109490055 ecr 0,nop,wscale 7], length 0
13:51:00.531500 IP (tos 0x60, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    64.233.182.139.80 > 172.18.0.2.48328: Flags [S.], cksum 0xddb4 (correct), seq 1521600211, ack 3977332363, win 65535, options [mss 1420,sackOK,TS val 4261937288 ecr 109490055,nop,wscale 8], length 0
13:51:00.531518 IP (tos 0x0, ttl 64, id 58152, offset 0, flags [DF], proto TCP (6), length 52)
    172.18.0.2.48328 > 64.233.182.139.80: Flags [.], cksum 0xa3af (incorrect -> 0x0a5a), ack 1, win 511, options [nop,nop,TS val 109490056 ecr 4261937288], length 0
13:51:00.531593 IP (tos 0x0, ttl 64, id 58153, offset 0, flags [DF], proto TCP (6), length 137)
    172.18.0.2.48328 > 64.233.182.139.80: Flags [P.], cksum 0xa404 (incorrect -> 0x790d), seq 1:86, ack 1, win 511, options [nop,nop,TS val 109490056 ecr 4261937288], length 85: HTTP, length: 85
        GET / HTTP/1.1
        Host: opensource.google.com
        User-Agent: curl/7.64.0
        Accept: */*

13:51:00.531808 IP (tos 0x60, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    64.233.182.139.80 > 172.18.0.2.48328: Flags [.], cksum 0x0b03 (correct), ack 86, win 256, options [nop,nop,TS val 4261937289 ecr 109490056], length 0
13:51:00.533901 IP (tos 0x80, ttl 126, id 0, offset 0, flags [DF], proto TCP (6), length 634)
    64.233.182.139.80 > 172.18.0.2.48328: Flags [P.], cksum 0xf3cd (correct), seq 1:583, ack 86, win 256, options [nop,nop,TS val 4261937291 ecr 109490056], length 582: HTTP, length: 582
        HTTP/1.1 301 Moved Permanently
```

```
        Location: https://opensource.google/
        Cross-Origin-Resource-Policy: cross-origin
        Content-Type: text/html; charset=UTF-8
        X-Content-Type-Options: nosniff
        Date: Wed, 13 Sep 2023 13:51:00 GMT
        Expires: Wed, 13 Sep 2023 14:21:00 GMT
        Cache-Control: public, max-age=1800
        Server: sffe
        Content-Length: 223
        X-XSS-Protection: 0

        <HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-
8">
        <TITLE>301 Moved</TITLE></HEAD><BODY>
        <H1>301 Moved</H1>
        The document has moved
        <A HREF="https://opensource.google/">here</A>.
        </BODY></HTML>
13:51:00.533907 IP (tos 0x0, ttl 64, id 58154, offset 0, flags [DF], proto TCP (6),
length 52)
    172.18.0.2.48328 > 64.233.182.139.80: Flags [.], cksum 0xa3af (incorrect ->
0x07be), ack 583, win 507, options [nop,nop,TS val 109490058 ecr 4261937291], length
0
13:51:00.535313 IP (tos 0x0, ttl 64, id 58155, offset 0, flags [DF], proto TCP (6),
length 52)
    172.18.0.2.48328 > 64.233.182.139.80: Flags [F.], cksum 0xa3af (incorrect ->
0x07bb), seq 86, ack 583, win 507, options [nop,nop,TS val 109490060 ecr 4261937291],
length 0
13:51:00.535704 IP (tos 0x80, ttl 126, id 0, offset 0, flags [DF], proto TCP (6),
length 52)
    64.233.182.139.80 > 172.18.0.2.48328: Flags [F.], cksum 0x08b4 (correct), seq
583, ack 87, win 256, options [nop,nop,TS val 4261937292 ecr 109490060], length 0
```

This command will run `tcpdump` with the following options:

- `-nn`: Disable port and protocol name lookup.
- `-r`: Read capture data from the named file.
- `-v`: Display detailed packet data.

You must specify the `-nn` switch again here, as you want to make sure `tcpdump` does not perform name lookups of either IP addresses or ports, since this can alert threat actors. This returns output data similar to the following:

```
reading from file capture.pcap, link-type EN10MB (Ethernet)
20:53:27.669101 IP (tos 0x0, ttl 64, id 50874, offset 0, flags [DF], proto TCP (6), length 60)
    172.17.0.2:46498 > 146.75.38.132:80: Flags [S], cksum 0x5445 (incorrect), seq 4197622953, win 65320, options [mss 1420,sackOK,TS val 610940466 ecr 0, nop,wscale 7], length 0
20:53:27.669422 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    146.75.38.132:80: > 172.17.0.2:46498: Flags [S.], cksum 0xc272 (correct), seq 2026312556, ack 4197622953, win 65535, options [mss 1420,sackOK,TS val 155704241 ecr 610940466, nop,wscale 9], length 0
```

As in the previous example, you can see the IP packet information along with information about the data that the packet contains.

2. Use the `tcpdump` command to filter the extended packet data from the `capture.pcap` capture file:

```
sudo tcpdump -nn -r capture.pcap -X
```

example output:
example output

```
analyst@b4aade4b3e15:~$ sudo tcpdump -nn -r capture.pcap -X
reading from file capture.pcap, link-type EN10MB (Ethernet)
13:51:00.530684 IP 172.18.0.2.48328 > 64.233.182.139.80: Flags [S], seq 3977332362, win 65320, options [mss 1420,sackOK,TS val 109490055 ecr 0,nop,wscale 7], length 0
        0x0000:  4500 003c e327 4000 4006 b40b ac12 0002  E..<.'@.@.......
        0x0010:  40e9 b68b bcc8 0050 ed11 468a 0000 0000  @......P..F.....
        0x0020:  a002 ff28 a3b7 0000 0204 058c 0402 080a  ...(............
        0x0030:  0686 af87 0000 0000 0103 0307            ............
13:51:00.531500 IP 64.233.182.139.80 > 172.18.0.2.48328: Flags [S.], seq 1521600211, ack 3977332363, win 65535, options [mss 1420,sackOK,TS val 4261937288 ecr 109490055,nop,wscale 8], length 0
        0x0000:  4560 003c 0000 4000 7e06 58d3 40e9 b68b  E`.<..@.~.X.@...
        0x0010:  ac12 0002 0050 bcc8 5ab1 c6d3 ed11 468b  .....P..Z.....F.
        0x0020:  a012 ffff ddb4 0000 0204 058c 0402 080a  ................
        0x0030:  fe08 0088 0686 af87 0103 0308            ............
13:51:00.531518 IP 172.18.0.2.48328 > 64.233.182.139.80: Flags [.], ack 1, win 511, options [nop,nop,TS val 109490056 ecr 4261937288], length 0
        0x0000:  4500 0034 e328 4000 4006 b412 ac12 0002  E..4.(@.@.......
        0x0010:  40e9 b68b bcc8 0050 ed11 468b 5ab1 c6d4  @......P..F.Z...
        0x0020:  8010 01ff a3af 0000 0101 080a 0686 af88  ................
```

```
        0x0030:  fe08 0088                                ....
13:51:00.531593 IP 172.18.0.2.48328 > 64.233.182.139.80: Flags [P.], seq 1:86, ack 1,
win 511, options [nop,nop,TS val 109490056 ecr 4261937288], length 85: HTTP: GET /
HTTP/1.1
        0x0000:  4500 0089 e329 4000 4006 b3bc ac12 0002  E....)@.@.......
        0x0010:  40e9 b68b bcc8 0050 ed11 468b 5ab1 c6d4  @......P..F.Z...
        0x0020:  8018 01ff a404 0000 0101 080a 0686 af88  ................
        0x0030:  fe08 0088 4745 5420 2f20 4854 5450 2f31  ....GET./.HTTP/1
        0x0040:  2e31 0d0a 486f 7374 3a20 6f70 656e 736f  .1..Host:.openso
        0x0050:  7572 6365 2e67 6f6f 676c 652e 636f 6d0d  urce.google.com.
        0x0060:  0a55 7365 722d 4167 656e 743a 2063 7572  .User-Agent:.cur
        0x0070:  6c2f 372e 3634 2e30 0d0a 4163 6365 7074  l/7.64.0..Accept
        0x0080:  3a20 2a2f 2a0d 0a0d 0a                   :.*/*....
13:51:00.531808 IP 64.233.182.139.80 > 172.18.0.2.48328: Flags [.], ack 86, win 256,
options [nop,nop,TS val 4261937289 ecr 109490056], length 0
        0x0000:  4560 0034 0000 4000 7e06 58db 40e9 b68b  E`.4..@.~.X.@...
        0x0010:  ac12 0002 0050 bcc8 5ab1 c6d4 ed11 46e0  .....P..Z.....F.
        0x0020:  8010 0100 0b03 0000 0101 080a fe08 0089  ................
        0x0030:  0686 af88                                ....
13:51:00.533901 IP 64.233.182.139.80 > 172.18.0.2.48328: Flags [P.], seq 1:583, ack
86, win 256, options [nop,nop,TS val 4261937291 ecr 109490056], length 582: HTTP:
HTTP/1.1 301 Moved Permanently
        0x0000:  4580 027a 0000 4000 7e06 5675 40e9 b68b  E..z..@.~.Vu@...
        0x0010:  ac12 0002 0050 bcc8 5ab1 c6d4 ed11 46e0  .....P..Z.....F.
        0x0020:  8018 0100 f3cd 0000 0101 080a fe08 008b  ................
        0x0030:  0686 af88 4854 5450 2f31 2e31 2033 3031  ....HTTP/1.1.301
        0x0040:  204d 6f76 6564 2050 6572 6d61 6e65 6e74  .Moved.Permanent
        0x0050:  6c79 0d0a 4c6f 6361 7469 6f6e 3a20 6874  ly..Location:.ht
        0x0060:  7470 733a 2f2f 6f70 656e 736f 7572 6365  tps://opensource
        0x0070:  2e67 6f6f 676c 652f 0d0a 4372 6f73 732d  .google/..Cross-
        0x0080:  4f72 6967 696e 2d52 6573 6f75 7263 652d  Origin-Resource-
        0x0090:  506f 6c69 6379 3a20 6372 6f73 732d 6f72  Policy:.cross-or
        0x00a0:  6967 696e 0d0a 436f 6e74 656e 742d 5479  igin..Content-Ty
        0x00b0:  7065 3a20 7465 7874 2f68 746d 6c3b 2063  pe:.text/html;.c
        0x00c0:  6861 7273 6574 3d55 5446 2d38 0d0a 582d  harset=UTF-8..X-
        0x00d0:  436f 6e74 656e 742d 5479 7065 2d4f 7074  Content-Type-Opt
        0x00e0:  696f 6e73 3a20 6e6f 736e 6966 660d 0a44  ions:.nosniff..D
        0x00f0:  6174 653a 2057 6564 2c20 3133 2053 6570  ate:.Wed,.13.Sep
        0x0100:  2032 3032 3320 3133 3a35 313a 3030 2047  .2023.13:51:00.G
        0x0110:  4d54 0d0a 4578 7069 7265 733a 2057 6564  MT..Expires:.Wed
```

```
        0x0120:  2c20 3133 2053 6570 2032 3032 3320 3134  ,.13.Sep.2023.14
        0x0130:  3a32 313a 3030 2047 4d54 0d0a 4361 6368  :21:00.GMT..Cach
        0x0140:  652d 436f 6e74 726f 6c3a 2070 7562 6c69  e-Control:.publi
        0x0150:  632c 206d 6178 2d61 6765 3d31 3830 300d  c,.max-age=1800.
        0x0160:  0a53 6572 7665 723a 2073 6666 650d 0a43  .Server:.sffe..C
        0x0170:  6f6e 7465 6e74 2d4c 656e 6774 683a 2032  ontent-Length:.2
        0x0180:  3233 0d0a 582d 5853 532d 5072 6f74 6563  23..X-XSS-Protec
        0x0190:  7469 6f6e 3a20 300d 0a0d 0a3c 4854 4d4c  tion:.0....<HTML
        0x01a0:  3e3c 4845 4144 3e3c 6d65 7461 2068 7474  ><HEAD><meta.htt
        0x01b0:  702d 6571 7569 763d 2263 6f6e 7465 6e74  p-equiv="content
        0x01c0:  2d74 7970 6522 2063 6f6e 7465 6e74 3d22  -type".content="
        0x01d0:  7465 7874 2f68 746d 6c3b 6368 6172 7365  text/html;charse
        0x01e0:  743d 7574 662d 3822 3e0a 3c54 4954 4c45  t=utf-8">.<TITLE
        0x01f0:  3e33 3031 204d 6f76 6564 3c2f 5449 544c  >301.Moved</TITL
        0x0200:  453e 3c2f 4845 4144 3e3c 424f 4459 3e0a  E></HEAD><BODY>.
        0x0210:  3c48 313e 3330 3120 4d6f 7665 643c 2f48  <H1>301.Moved</H
        0x0220:  313e 0a54 6865 2064 6f63 756d 656e 7420  1>.The.document.
        0x0230:  6861 7320 6d6f 7665 640a 3c41 2048 5245  has.moved.<A.HRE
        0x0240:  463d 2268 7474 7073 3a2f 2f6f 7065 6e73  F="https://opens
        0x0250:  6f75 7263 652e 676f 6f67 6c65 2f22 3e68  ource.google/">h
        0x0260:  6572 653c 2f41 3e2e 0d0a 3c2f 424f 4459  ere</A>...</BODY
        0x0270:  3e3c 2f48 544d 4c3e 0d0a                 ></HTML>..
13:51:00.533907 IP 172.18.0.2.48328 > 64.233.182.139.80: Flags [.], ack 583, win 507,
options [nop,nop,TS val 109490058 ecr 4261937291], length 0
        0x0000:  4500 0034 e32a 4000 4006 b410 ac12 0002  E..4.*@.@.......
        0x0010:  40e9 b68b bcc8 0050 ed11 46e0 5ab1 c91a  @......P..F.Z...
        0x0020:  8010 01fb a3af 0000 0101 080a 0686 af8a  ................
        0x0030:  fe08 008b                                ....
13:51:00.535313 IP 172.18.0.2.48328 > 64.233.182.139.80: Flags [F.], seq 86, ack 583,
win 507, options [nop,nop,TS val 109490060 ecr 4261937291], length 0
        0x0000:  4500 0034 e32b 4000 4006 b40f ac12 0002  E..4.+@.@.......
        0x0010:  40e9 b68b bcc8 0050 ed11 46e0 5ab1 c91a  @......P..F.Z...
        0x0020:  8011 01fb a3af 0000 0101 080a 0686 af8c  ................
        0x0030:  fe08 008b                                ....
13:51:00.535704 IP 64.233.182.139.80 > 172.18.0.2.48328: Flags [F.], seq 583, ack 87,
win 256, options [nop,nop,TS val 4261937292 ecr 109490060], length 0
        0x0000:  4580 0034 0000 4000 7e06 58bb 40e9 b68b  E..4..@.~.X.@...
        0x0010:  ac12 0002 0050 bcc8 5ab1 c91a ed11 46e1  .....P..Z.....F.
        0x0020:  8011 0100 08b4 0000 0101 080a fe08 008c  ................
        0x0030:  0686 af8c                                ....
```

```
analyst@b4aade4b3e15:~$
```

This command will run `tcpdump` with the following options:

- `-nn`: Disable port and protocol name lookup.
- `-r`: Read capture data from the named file.
- `-X`: Display the hexadecimal and ASCII output format packet data. Security analysts can analyze hexadecimal and ASCII output to detect patterns or anomalies during malware analysis or forensic analysis.

Note: *Hexadecimal, also known as hex or base 16, uses 16 symbols to represent values, including the digits 0-9 and letters A, B, C, D, E, and F. American Standard Code for Information Interchange (ASCII) is a character encoding standard that uses a set of characters to represent text in digital form.*

Revision #7
Created 2023-09-13 13:35:06 UTC by naruzkurai
Updated 2023-11-01 01:10:46 UTC by naruzkurai