Tcpdump - The Basics

Basic Packet Capture

Specify the Network Interface

The first thing to decide is which network interface to listen to using `-i INTERFACE`. You can choose to listen on all available interfaces using `-i any`; alternatively, you can specify an interface you want to listen on, such as `-i eth0`

A command such as `ip address show` (or merely `ip a s`) would list the available network interfaces

Save the Captured Packets

saving to a file using `-w FILE`

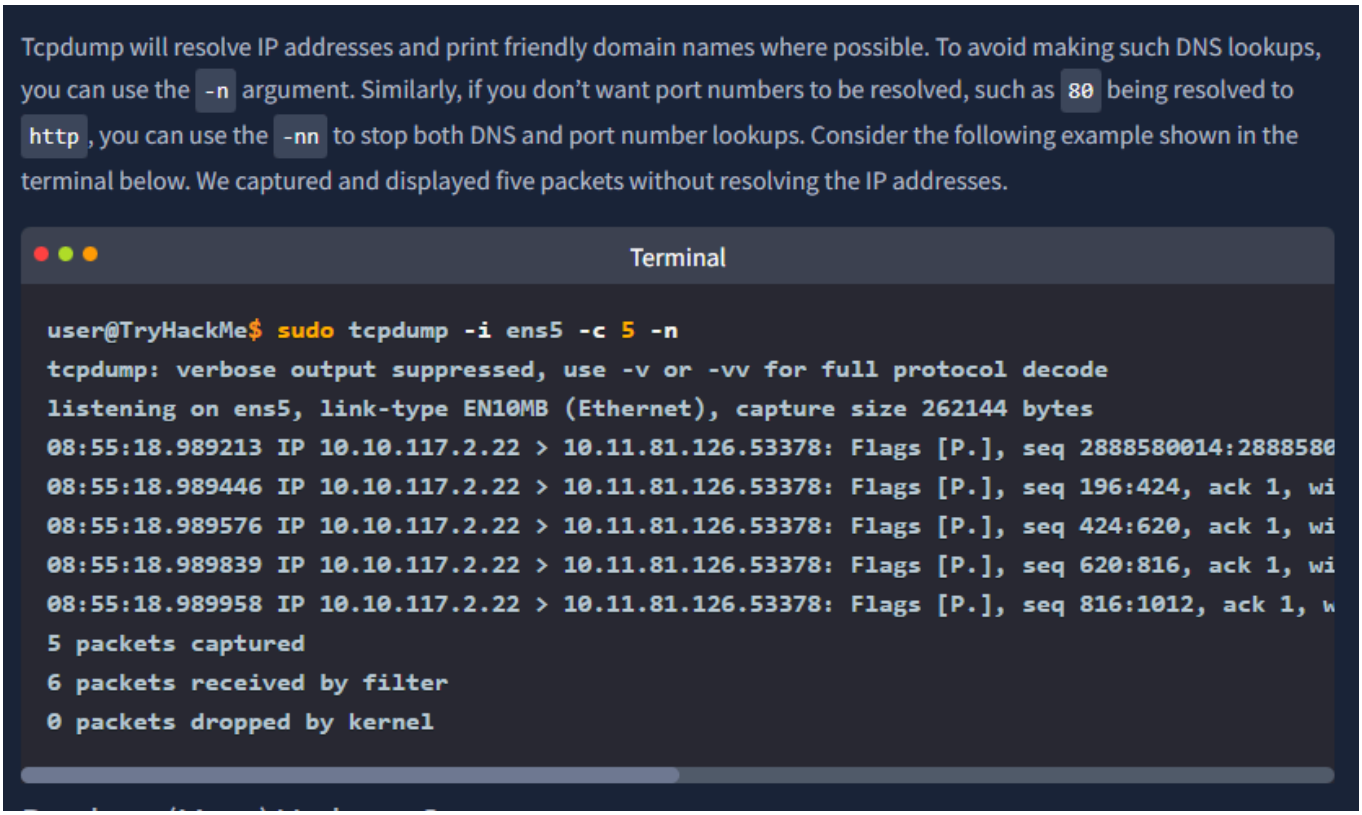
Read Captured Packets from a File

can use Tcpdump to read packets from a file by using `-r FILE`

Limit the Number of Captured Packets

can specify the number of packets to capture by specifying the count using `-c COUNT`

Don’t Resolve IP Addresses and Port Numbers



Produce (More) Verbose Output

According to the Tcpdump manual page (`man tcpdump`), the addition of `-v` will print "the time to live, identification, total length and options in an IP packet" among other checks

Summary and Examples

The table below provides a summary of the command line options.

Command	Explanation
`tcpdump -i INTERFACE`	Captures packets on a specific network interface
`tcpdump -w FILE`	Writes captured packets to a file
`tcpdump -r FILE`	Reads captured packets from a file
`tcpdump -c COUNT`	Captures a specific number of packets
`tcpdump -n`	Don't resolve IP addresses
`tcpdump -nn`	Don't resolve IP addresses and don't resolve protocol numbers
`tcpdump -v`	Verbose display; verbosity can be increased with `-vv` and `-vvv`

Consider the following examples:

- `tcpdump -i eth0 -c 50 -v` captures and displays 50 packets by listening on the `eth0` interface, which is a wired Ethernet, and displays them verbosely.

- `tcpdump -i wlo1 -w data.pcap` captures packets by listening on the `wlo1` interface (the WiFi interface) and writes the packets to `data.pcap`. It will continue till the user interrupts the capture by pressing CTRL-C.
- `tcpdump -i any -nn` captures packets on all interfaces and displays them on screen without domain name or protocol resolution.

Filtering Expressions

Filtering by Host

Can limit captured packets to host using host IP or host HOSTNAME.

this example has packets exchanged with `example.com` and save them to `http.pcap`
capturing packets requires you to be logged-in as `root` or to use `sudo`

e.g:
sudo tcpdump host example.com -w http.pcap

limit the packets to those from a particular source IP address or hostname must use `src host IP` or `src host HOSTNAME`

Can limit packets to those sent to a specific destination using `dst host IP` or `dst host HOSTNAME`

Filtering by Port

want to capture all DNS traffic, you can limit the captured packets to those on `port 53`,
DNS uses UDP and TCP ports 53 by default.

can limit the packets to those from a particular source port number or to a particular destination port number using `src port PORT_NUMBER` and `dst port PORT_NUMBER`, respectively

Filtering by Protocol

You can limit your packet capture to a specific protocol; examples
include: `ip`, `ip6`, `udp`, `tcp`, and `icmp`

Logical Operators

Three logical operators that can be handy:

- `and`: Captures packets where both conditions are true. For example, `tcpdump host 1.1.1.1 and tcp` captures `tcp` traffic with `host 1.1.1.1`.
- `or`: Captures packets when either one of the conditions is true. For instance, `tcpdump udp or icmp` captures UDP or ICMP traffic.

- `not`: Captures packets when the condition is not true. For example, `tcpdump not tcp` captures all packets except TCP segments; we expect to find UDP, ICMP, and ARP packets among the results.

Command	Explanation
`tcpdump host IP` or `tcpdump host HOSTNAME`	Filters packets by IP address or hostname
`tcpdump src host IP` or	Filters packets by a specific source host
`tcpdump dst host IP`	Filters packets by a specific destination host
`tcpdump port PORT_NUMBER`	Filters packets by port number
`tcpdump src port PORT_NUMBER`	Filters packets by the specified source port number
`tcpdump dst port PORT_NUMBER`	Filters packets by the specified destination port number
`tcpdump PROTOCOL`	Filters packets by protocol; examples include `ip`, `ip6`, and `icmp`

examples:

- `tcpdump -i any tcp port 22` listens on all interfaces and captures `tcp` packets to or from `port 22`, i.e., SSH traffic.
- `tcpdump -i wlo1 udp port 123` listens on the WiFi network card and filters `udp` traffic to `port 123`, the Network Time Protocol (NTP).
- `tcpdump -i eth0 host example.com and tcp port 443 -w https.pcap` will listen on `eth0`, the wired Ethernet interface and filter traffic exchanged with `example.com` that uses `tcp` and `port 443`. In other words, this command is filtering HTTPS traffic related to `example.com`.

Advanced Filtering

we can limit the displayed packets to those smaller or larger than a certain length:

- `greater LENGTH`: Filters packets that have a length greater than or equal to the specified length
- `less LENGTH`: Filters packets that have a length less than or equal to the specified length

man pcap-filter - filters manual page

Binary Operations

`&` (And) takes two bits and returns 0 unless both inputs are 1
`|` (Or) takes two bits and returns 1 unless both inputs are 0

`!` (Not) takes one bit and inverts it; an input of 1 gives 0, and an input of 0 gives 1

Header Bytes

filter packets based on the contents of a header byte

Using pcap-filter, Tcpdump allows you to refer to the contents of any byte in the header using the following syntax `proto[expr:size]`, where:

- `proto` refers to the protocol. For example, `arp`, `ether`, `icmp`, `ip`, `ip6`, `tcp`, and `udp` refer to ARP, Ethernet, ICMP, IPv4, IPv6, TCP, and UDP respectively.
- `expr` indicates the byte offset, where `0` refers to the first byte.
- `size` indicates the number of bytes that interest us, which can be one, two, or four. It is optional and is one by default.

filtering TCP packets based on the set TCP flags.

You can use `tcp[tcpflags]` to refer to the TCP flags field. The following TCP flags are available to compare with:

- `tcp-syn` TCP SYN (Synchronize)
- `tcp-ack` TCP ACK (Acknowledge)
- `tcp-fin` TCP FIN (Finish)
- `tcp-rst` TCP RST (Reset)
- `tcp-push` TCP Push

Based on the above, we can write:

- `tcpdump "tcp[tcpflags] == tcp-syn"` to capture TCP packets with **only** the SYN (Synchronize) flag set, while all the other flags are unset.
- `tcpdump "tcp[tcpflags] & tcp-syn != 0"` to capture TCP packets with **at least** the SYN (Synchronize) flag set.
- `tcpdump "tcp[tcpflags] & (tcp-syn|tcp-ack) != 0"` to capture TCP packets with **at least** the SYN (Synchronize) **or** ACK (Acknowledge) flags set.

How many packets have only the TCP Reset (RST) flag set?

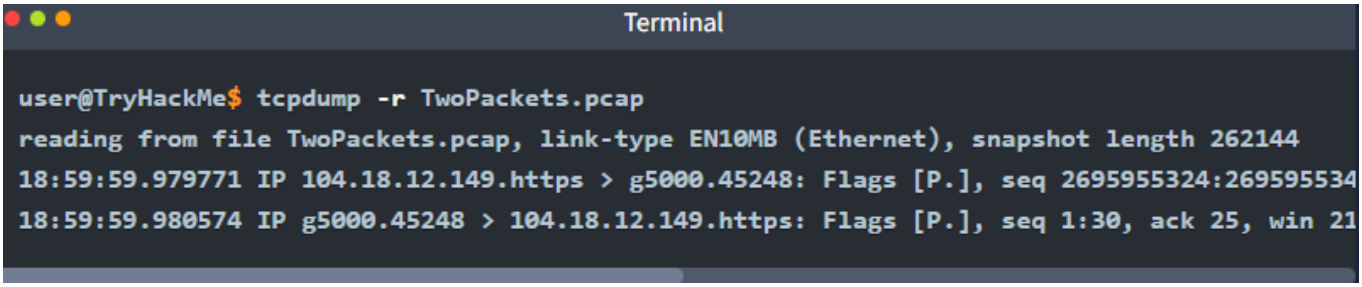
sudo tcpdump -r traffic.pcap 'tcp[tcpflags] == tcp-rst' | wc -l

What is the IP address of the host that sent packets larger than 15000 bytes?
sudo tcpdump -r traffic.pcap 'greater 15000' -n

Displaying Packets

Tcpdump is a rich program with many options to customize how the packets are printed and displayed. We have selected to cover the following five options:

- `-q` : Quick output; print brief packet information
- `-e` : Print the link-level header
- `-A` : Show packet data in ASCII
- `-xx` : Show packet data in hexadecimal format, referred to as hex
- `-X` : Show packet headers and data in hex and ASCII



Brief Packet Information

If you prefer shorter output lines, you can opt for “quick” output with `-q`

```
user@TryHackMe$ tcpdump -r TwoPackets.pcap -q
reading from file TwoPackets.pcap, link-type EN10MB (Ethernet), snapshot length 262144
18:59:59.979771 IP 104.18.12.149.https > g5000.45248: tcp 25
18:59:59.980574 IP g5000.45248 > 104.18.12.149.https: tcp 29
```

Displaying Link-Level Header

you are on an Ethernet or WiFi network and want to include the MAC addresses in Tcpdump output, all you need to do is to add `-e`

```
user@TryHackMe$ tcpdump -r TwoPackets.pcap -e
reading from file TwoPackets.pcap, link-type EN10MB (Ethernet), snapshot length 262144
18:59:59.979771 44:df:65:d8:fe:6c (oui Unknown) > 02:83:1e:40:5d:17 (oui Unknown), etherty
18:59:59.980574 02:83:1e:40:5d:17 (oui Unknown) > 44:df:65:d8:fe:6c (oui Unknown), etherty
```

Displaying Packets as ASCII

ASCII stands for American Standard Code for Information Interchange; ASCII codes represent text. In other words, you can expect `-A` to display all the bytes mapped to English letters, numbers, and symbols.

```
user@TryHackMe$ tcpdump -r TwoPackets.pcap -A
reading from file TwoPackets.pcap, link-type EN10MB (Ethernet), snapshot length 262144
18:59:59.979771 IP 104.18.12.149.https > g5000.45248: Flags [P.], seq 2695955324:269595534
E..M..@.5..)h.....BY.......|.;5}...........
..1...k......j.3.2.....&9a.....-L
18:59:59.980574 IP g5000.45248 > 104.18.12.149.https: Flags [P.], seq 1:30, ack 25, win 21
E..Q1.@.@.VV..BYh........;5}..............
..k...1.......1.y.&VC<#._J$..z...D#.`
```

Displaying Packets in Hexadecimal Format

To display the packets in hexadecimal format, we must add `-xx` as shown in the terminal below.

```
user@TryHackMe$ tcpdump -r TwoPackets.pcap -xx
reading from file TwoPackets.pcap, link-type EN10MB (Ethernet), snapshot length 262144
18:59:59.979771 IP 104.18.12.149.https > g5000.45248: Flags [P.], seq 2695955324:269595534
        0x0000:  0283 1e40 5d17 44df 65d8 fe6c 0800 4500
        0x0010:  004d fbd8 4000 3506 d229 6812 0c95 c0a8
        0x0020:  4259 01bb b0c0 a0b1 037c aa3b 357d 8018
        0x0030:  0010 f905 0000 0101 080a 189a 310d ebfa
        0x0040:  6b2e 1703 0300 146a 8f33 1832 e6a2 fb99
        0x0050:  eb26 3961 dad4 1611 152d 4c
18:59:59.980574 IP g5000.45248 > 104.18.12.149.https: Flags [P.], seq 1:30, ack 25, win 21
        0x0000:  44df 65d8 fe6c 0283 1e40 5d17 0800 4500
        0x0010:  0051 6ca8 4000 4006 5656 c0a8 4259 6812
        0x0020:  0c95 b0c0 01bb aa3b 357d a0b1 0395 8018
        0x0030:  087f 17e0 0000 0101 080a ebfa 6be8 189a
        0x0040:  310d 1703 0300 18f4 31fa 798d 2656 433c
        0x0050:  2389 5f4a 24c2 fa7a 1496 8444 238e 60
```

Adding `-xx` lets us see the packet octet by octet. In the example above, we can closely inspect the IP and TCP headers in addition to the packet contents.

Best of Both Worlds

If you would like to display the captured packets in hexadecimal and ASCII formats, Tcpdump makes it easy with the `-X` option.

```
user@TryHackMe$ tcpdump -r TwoPackets.pcap -X
reading from file TwoPackets.pcap, link-type EN10MB (Ethernet), snapshot length 262144
18:59:59.979771 IP 104.18.12.149.https > g5000.45248: Flags [P.], seq 2695955324:26959553
        0x0000:  4500 004d fbd8 4000 3506 d229 6812 0c95  E..M..@.5..)h...
        0x0010:  c0a8 4259 01bb b0c0 a0b1 037c aa3b 357d  ..BY.......|.;5}
        0x0020:  8018 0010 f905 0000 0101 080a 189a 310d  ..............1.
        0x0030:  ebfa 6b2e 1703 0300 146a 8f33 1832 e6a2  ..k......j.3.2..
        0x0040:  fb99 eb26 3961 dad4 1611 152d 4c         ...&9a.....-L
18:59:59.980574 IP g5000.45248 > 104.18.12.149.https: Flags [P.], seq 1:30, ack 25, win 2
        0x0000:  4500 0051 6ca8 4000 4006 5656 c0a8 4259  E..Q1.@.@.VV..BY
        0x0010:  6812 0c95 b0c0 01bb aa3b 357d a0b1 0395  h........;5}....
        0x0020:  8018 087f 17e0 0000 0101 080a ebfa 6be8  ..............k.
        0x0030:  189a 310d 1703 0300 18f4 31fa 798d 2656  ..1.......1.y.&V
        0x0040:  433c 2389 5f4a 24c2 fa7a 1496 8444 238e  C<#._J$..z...D#.
        0x0050:  60
```

a summary of the command line options thats covered.

Command	Explanation
`tcpdump -q`	Quick and quite: brief packet information
`tcpdump -e`	Include MAC addresses
`tcpdump -A`	Print packets as ASCII encoding
`tcpdump -xx`	Display packets in hexadecimal format
`tcpdump -X`	Show packets in both hexadecimal and ASCII formats

What is the MAC address of the host that sent an ARP request?
sudo tcpdump -r traffic.pcap arp -e