Exercise: Capturing UDP Packets

We'll now look at a command-line tool that allows us to capture UDP packets.

WE'LL COVER THE FOLLOWING

Let's get into viewing real packets.

What is `tcpdump`?

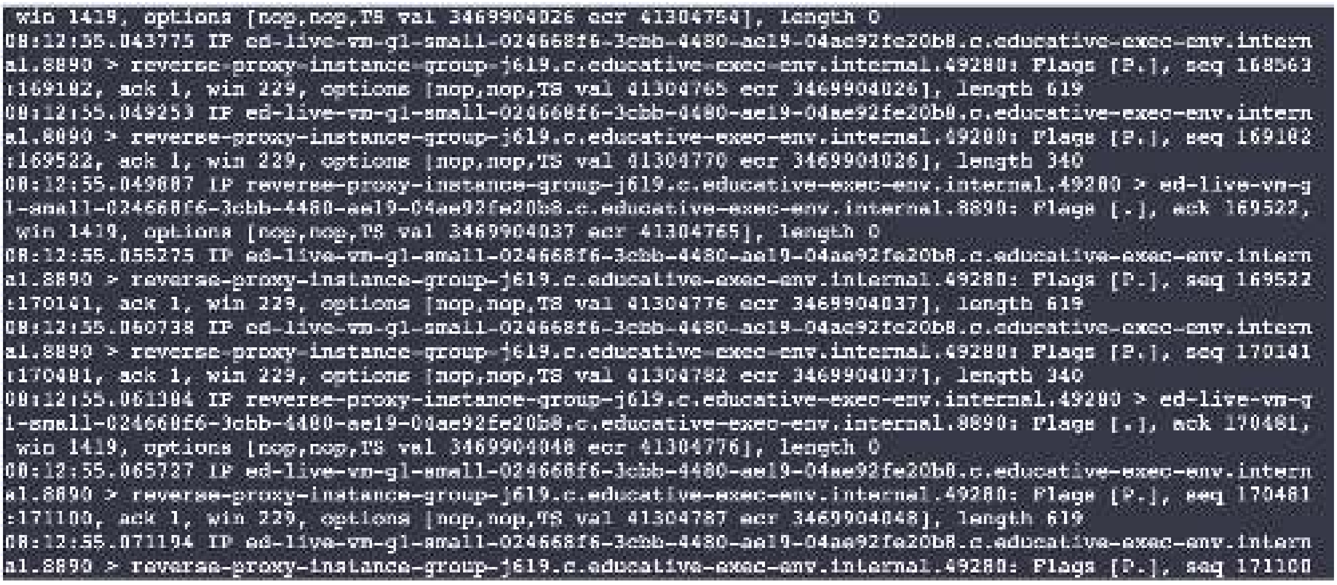
`tcpdump` is a command-line tool that can be used to view packets being sent and received on a computer. The simplest way to run it is to simply type the following command into a terminal and hit enter. You can try this on the terminal provided at the end of this lesson!

```
tcpdump
```

Packets will start getting printed rapidly to give a comprehensive view of the traffic.

Sample Output

However, some might not find it to be very helpful because it does not allow for a more **zoomed-in and fine-grained dissection of the packets**, which is the main purpose of `tcpdump` (it's technically a packet *analyzer*). So you might want to consider using some flags to filter relevant packets out.



```
 win 1419, options [nop,nop,TS val 3469904026 ecr 41304754], length 0
08:12:55.043775 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 168563
:169182, ack 1, win 229, options [nop,nop,TS val 41304765 ecr 3469904026], length 619
08:12:55.049253 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 169182
:169522, ack 1, win 229, options [nop,nop,TS val 41304770 ecr 3469904026], length 340
08:12:55.049887 IP reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280 > ed-live-vm-g
1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.internal.8890: Flags [.], ack 169522,
 win 1419, options [nop,nop,TS val 3469904037 ecr 41304765], length 0
08:12:55.055275 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 169522
:170141, ack 1, win 229, options [nop,nop,TS val 41304776 ecr 3469904037], length 619
08:12:55.060738 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 170141
:170481, ack 1, win 229, options [nop,nop,TS val 41304782 ecr 3469904037], length 340
08:12:55.061384 IP reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280 > ed-live-vm-g
1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.internal.8890: Flags [.], ack 170481,
 win 1419, options [nop,nop,TS val 3469904048 ecr 41304776], length 0
08:12:55.065727 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 170481
:171100, ack 1, win 229, options [nop,nop,TS val 41304787 ecr 3469904048], length 619
08:12:55.071194 IP ed-live-vm-g1-small-024668f6-3cbb-4480-ae19-04ae92fe20b8.c.educative-exec-env.intern
al.8890 > reverse-proxy-instance-group-j619.c.educative-exec-env.internal.49280: Flags [P.], seq 171100
```

... what??

Useful `tcpdump` Flags

Here are some flags that you might find useful in your exploration of this tool. You can find more details about each on tcpdump's Manpage

Let's zoom into the traffic a bit

Saving `tcpdump` Output to a File with `-w`

Instead of having all the output print to the console, we can save it to view at a later date or to feed into another program to analyze.

```
tcpdump -w filename.ext
```

Try using this tool in the following code executable.

```
tcpdump -w output.pcap # Saving output to a file called 'output.pcap'
```

The file `output.pcap` will have all the packets saved to it. Try running this command in the terminal below. Note that the process does not exit without a keyboard interrupt. The next flag will help us stop packet capture in a predetermined fashion.

> **Note .pcap** files are used to store the packet data of a network. Packet analysis programs such as Wireshark (think of it like tcpdump with a GUI) export and import packet captures in pcap files.

Counting Packets with `-c`

This flag makes `tcpdump` capture a defined number of packets. Here's how it's used.

```
tcpdump -w output.pcap -c 10 # Capturing 10 packets
```

You can't view the file just yet. Let's do it next.

Printing PCAP Files With `-r`

Great! Let's actually **read `.pcap` files** now. Here's how to do it.

```
tcpdump -w output.pcap -c 10 # Capturing 10 packets
tcpdump -r output.pcap # Printing the captured packets in a PCAP file
```

We've gotten pretty far with this. There are plenty of other flags and arguments you could give to `tcpdump` to make it capture packets precisely as per your requirements.

Looking at Real UDP Packet Headers

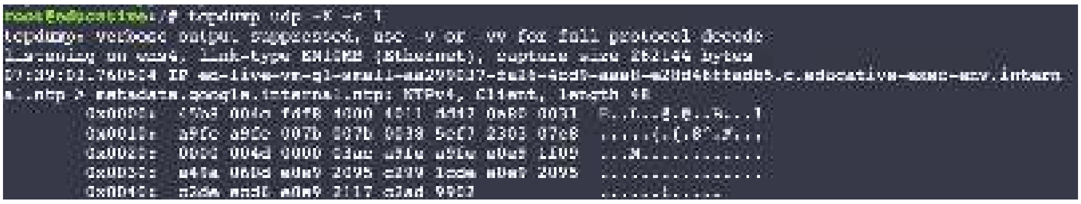
Here's a script to capture and print one UDP packet.

> Note that the code *may* time out before it actually captures a packet. We would suggest running this one on the terminal.

```
tcpdump udp -X -c 1 # Capturing 1 UDP packet
```

The `-X` flag just prints the payload of the packet (the data) in both hex and ASCII.

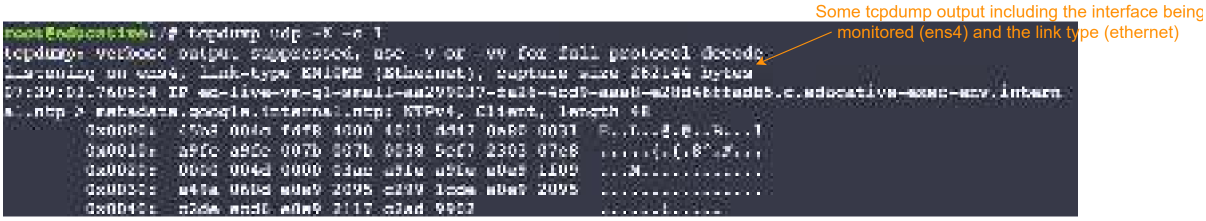
Here’s what the output is depicting.



```
root@[illegible]:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:01.760504 IP ed-live-vm-g1-small-aa29903[illegible].c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fdf8 4000 4011 d447 0a80 0031  E..L..@.@..G...1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8\.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e5 1109  ...M............
        0x0030:  e45a 060d e0e9 2095 c299 1cda e0e9 2095  ................
        0x0040:  c2de ee30 e0e9 2117 c3ad 9902            ......!.....
```

The command that starts tcpdump is on the first line

1 of 11

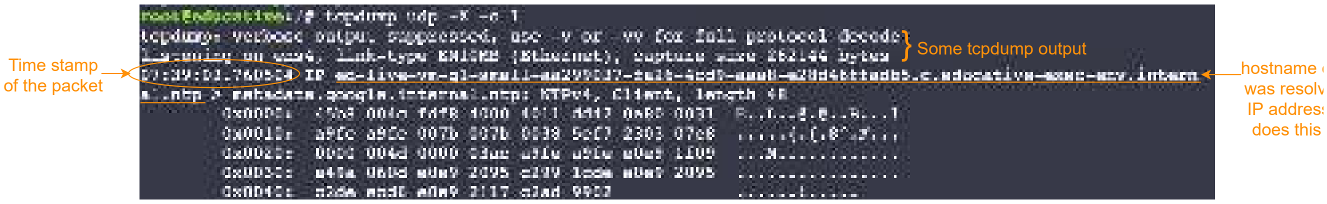


```
root@[illegible]:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:01.760504 IP ed-live-vm-g1-small-aa29903[illegible].c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fdf8 4000 4011 d447 0a80 0031  E..L..@.@..G...1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8\.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e5 1109  ...M............
        0x0030:  e45a 060d e0e9 2095 c299 1cda e0e9 2095  ................
        0x0040:  c2de ee30 e0e9 2117 c3ad 9902            ......!.....
```

Some tcpdump output including the interface being monitored (ens4) and the link type (ethernet)

Here's some general output from tcpdump itself

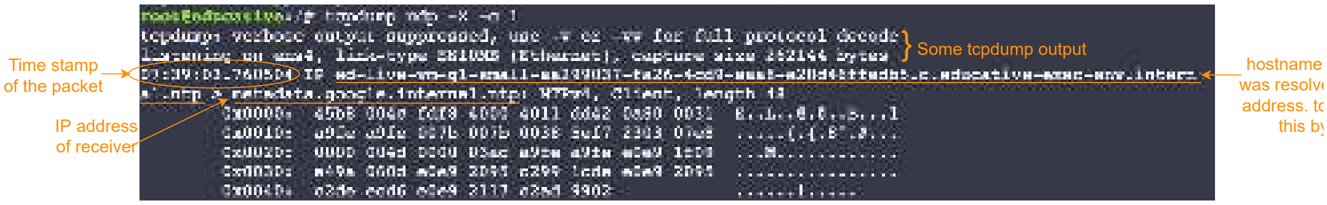
2 of 11



```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fa16-4cd9-aaa8-a28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  4500 004c fdf8 4000 4011 d417 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8\.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e49a 060d e0e9 2095 c299 1cda e0e9 2095  ................
        0x0040:  e2de ecd6 e0e9 2117 e2ad 9902            ......!.....
```

The format of the next line is like so 'IP address of sender > IP address of receiver'. Notice that the IP addresses have been resolved into hostnames. tcpdump does this by default. If you wish to see the actual IP address, pass in the '-n' flag. Also notice the time stamp.

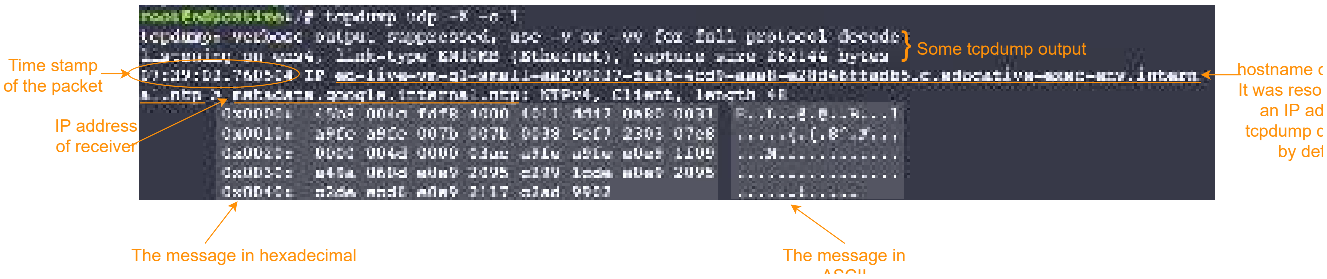
3 of 11



```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fa16-4cd9-aaa8-a28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  4500 004c fdf8 4000 4011 d417 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8\.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e49a 060d e0e9 2095 c299 1cda e0e9 2095  ................
        0x0040:  e2de ecd6 e0e9 2117 e2ad 9902            ......!.....
```

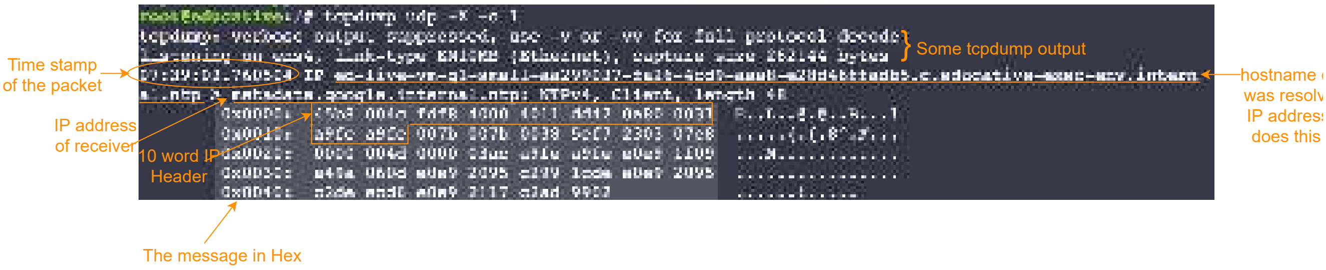
IP address (or hostname really) of the receiver

4 of 11



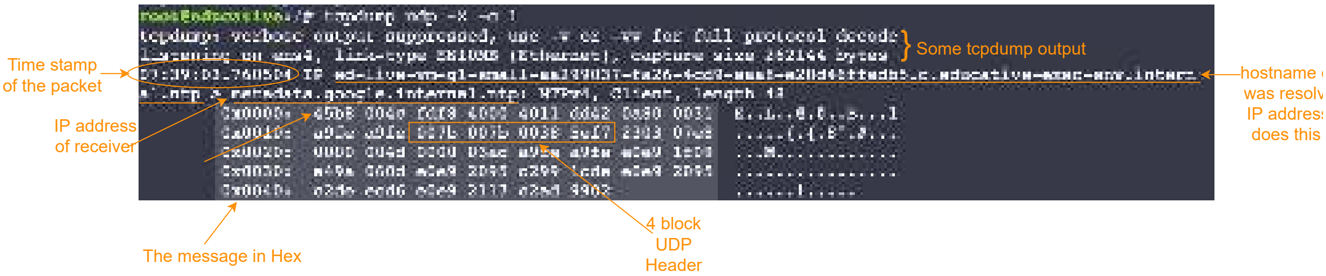
```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fa16-4cd9-aaa8-a28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  4500 004c fdf8 4000 4011 d417 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8\.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e49a 060d e0e9 2095 c299 1cda e0e9 2095  ................
        0x0040:  e2de ecd6 e0e9 2117 e2ad 9902            ......!.....
```

Let's dissect the datagram now



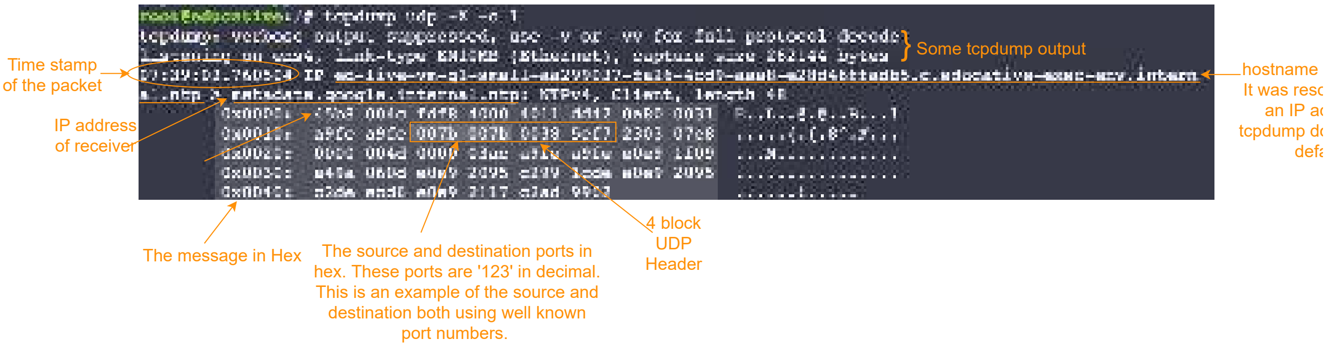
```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fe16-4cd9-aaa8-e28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fdf8 4000 4011 d412 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e46a 060d e0e9 2095 e299 1cde e0e9 2095  ................
        0x0040:  e2de eed6 e0e9 2117 e2ad 9902            ......!.....
```

The first 160 bits are the IP header. Note that a single hex digit is exactly 4 bits so that means the header is of 160/4 = 40 hex digits or 40/4 = 10 blocks. We can safely ignore it for now!



```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fe16-4cd9-aaa8-e28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fdf8 4000 4011 d412 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e46a 060d e0e9 2095 e299 1cde e0e9 2095  ................
        0x0040:  e2de eed6 e0e9 2117 e2ad 9902            ......!.....
```

The UDP header is of 64 bits 4 blocks. Each block represents one UDP field.

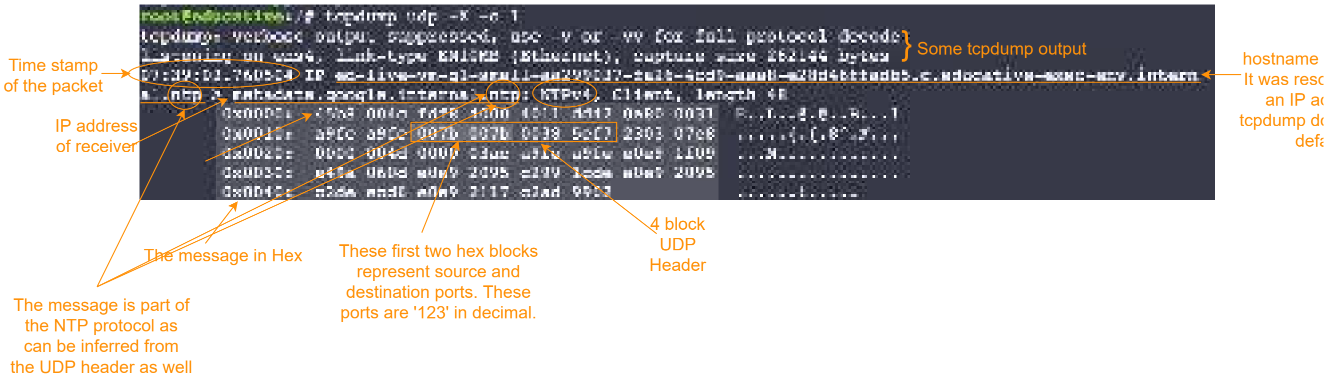


```
root@educative:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-aa299037-fe16-4cd9-aaa8-e28d4bbtadb5.c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fdf8 4000 4011 d412 0a80 0031  E..L..@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5cf7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ac a9fe a9fe e0e9 1109  ...M............
        0x0030:  e46a 060d e0e9 2095 e299 1cde e0e9 2095  ................
        0x0040:  e2de eed6 e0e9 2117 e2ad 9902            ......!.....
```

Time stamp of the packet
IP address of receiver
The message in Hex
The source and destination ports in hex. These ports are '123' in decimal. This is an example of the source and destination both using well known port numbers.
4 block UDP Header
Some tcpdump output
hostname
It was reso
an IP ad
tcpdump d
def

The UDP header is of 64 bits i.e., 4 blocks. Each block represents one UDP field. The first two fields are the source and destination ports which are both 007b or port numbers 123 in decimal.

8 of 11

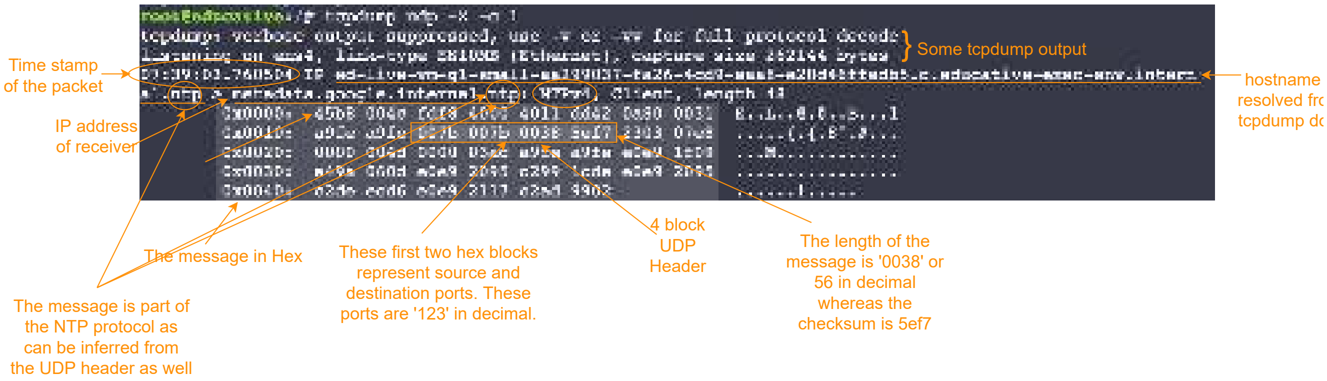


```
root@edpcontainer:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-[illegible].c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fd78 4000 4011 8412 0a80 0031  E..L.x@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5ef7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ae a9fe a9fe e0e9 1109  ...M............
        0x0030:  e45a 060d e0e9 2095 e299 1ede e0e9 2095  ................
        0x0040:  e2de ee8d e0e9 2117 e2ad 9952            ......!...R
```

Time stamp of the packet
IP address of receiver
Some tcpdump output
hostname
It was reso
an IP ad
tcpdump d
defa
The message in Hex
These first two hex blocks represent source and destination ports. These ports are '123' in decimal.
4 block UDP Header
The message is part of the NTP protocol as can be inferred from the UDP header as well

Note that port 123 is reserved for the NTP protocol (which runs on UDP) as shown by the output here.

9 of 11

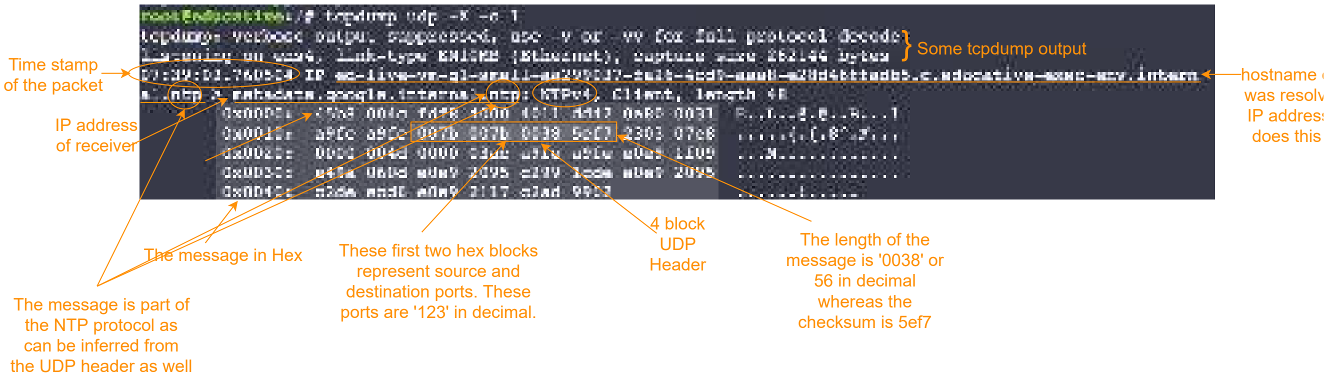


```
root@edpcontainer:/# tcpdump ntp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-[illegible].c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fd78 4000 4011 8412 0a80 0031  E..L.x@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5ef7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ae a9fe a9fe e0e9 1109  ...M............
        0x0030:  e45a 060d e0e9 2095 e299 1ede e0e9 2095  ................
        0x0040:  e2de ee8d e0e9 2117 e2ad 9952            ......!...R
```

Time stamp of the packet
IP address of receiver
Some tcpdump output
hostname
resolved fr
tcpdump d
The message in Hex
These first two hex blocks represent source and destination ports. These ports are '123' in decimal.
4 block UDP Header
The length of the message is '0038' or 56 in decimal whereas the checksum is 5ef7
The message is part of the NTP protocol as can be inferred from the UDP header as well

The next two fields are the length and the checksum!

10 of 11



```
root@edpcontainer:/# tcpdump udp -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:03.760524 IP ed-live-vm-g1-small-[illegible].c.educative-exec-env.intern
al.ntp > metadata.google.internal.ntp: NTPv4, Client, length 48
        0x0000:  45b8 004c fd78 4000 4011 8412 0a80 0031  E..L.x@.@......1
        0x0010:  a9fe a9fe 007b 007b 0038 5ef7 2303 07e8  .....{.{.8^.#...
        0x0020:  0000 004d 0000 03ae a9fe a9fe e0e9 1109  ...M............
        0x0030:  e45a 060d e0e9 2095 e299 1ede e0e9 2095  ................
        0x0040:  e2de ee8d e0e9 2117 e2ad 9952            ......!...R
```

Time stamp of the packet
IP address of receiver
Some tcpdump output
hostname
was resolv
IP addres
does this
The message in Hex
These first two hex blocks represent source and destination ports. These ports are '123' in decimal.
4 block UDP Header
The length of the message is '0038' or 56 in decimal whereas the checksum is 5ef7
The message is part of the NTP protocol as can be inferred from the UDP header as well

That concludes our inspection of a UDP packet. Explore this more! Try capturing a packet on the command line below and try dissecting it!

11 of 11

Try it Yourself!

You can try all the commands in this terminal. Click here to go back

Terminal

In the next lesson, we’ll learn about the transmission control protocol!