Experiment No: 4

Network Data Analysis using tcpdump

Installation
If tcpdump is not installed, you can install it using :

For Ubuntu/Debian OS

apt install tcpdump ***or***
sudo apt install tcpdump

```
root@mahe-HP-Z1-G9-Tower-Desktop-PC:~# apt install tcpdump
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tcpdump is already the newest version (4.99.4-3ubuntu4).
The following package was automatically installed and is no longer required:
  nvidia-firmware-535-535.171.04
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 166 not upgraded.
root@mahe-HP-Z1-G9-Tower-Desktop-PC:~#
```

```
mahe@mahe-HP-Z1-G9-Tower-Desktop-PC:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.20.32.50  netmask 255.255.255.0  broadcast 172.20.32.255
        inet6 fe80::5e60:baff:fe31:c16e  prefixlen 64  scopeid 0x20<link>
        ether 5c:60:ba:31:c1:6e  txqueuelen 1000  (Ethernet)
        RX packets 17078  bytes 6055384 (6.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2373  bytes 349895 (349.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  memory 0x82280000-822a0000
```

Capturing packets with tcpdump
To capture packets for troubleshooting or analysis, tcpdump requires elevated permissions, so in the following examples most commands are prefixed with sudo.

To begin, use the command tcpdump --list-interfaces (or -D for short) to see which interfaces are available for capture:

```
root@mahe-HP-Z1-G9-Tower-Desktop-PC:~# tcpdump --list-interfaces
1.eno1 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
root@mahe-HP-Z1-G9-Tower-Desktop-PC:~#
```

In the example above, you can see all the interfaces available in my machine. The special interface any allows capturing in any active interface.
Let's use it to start capturing some packets. Capture all packets in any interface by running this command:

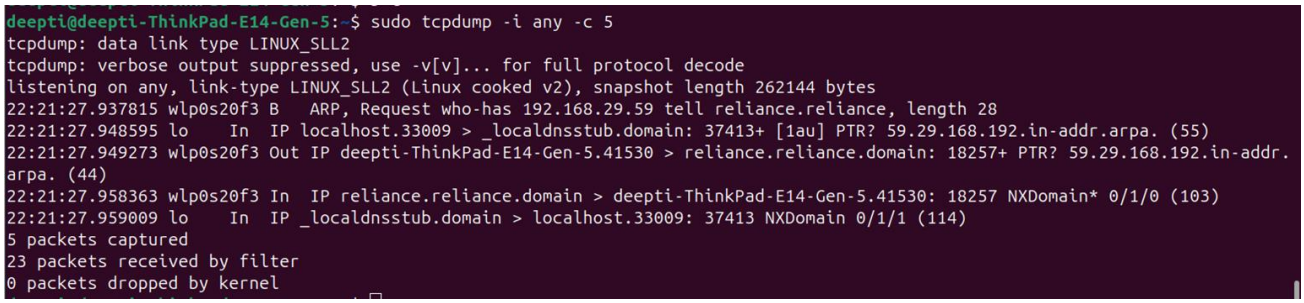
```
deepti@deepti-ThinkPad-E14-Gen-5:~$ sudo tcpdump --interface any
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
22:10:30.480896 wlp0s20f3 Out IP deepti-ThinkPad-E14-Gen-5.52450 > 93.243.107.34.bc.googleusercontent.com.https: Flags [P.], seq 2135179813:2135179852, ack 72492986, win 246,
options [nop,nop,TS val 2962263852 ecr 1801340511], length 39
22:10:30.481222 wlp0s20f3 Out IP deepti-ThinkPad-E14-Gen-5.52450 > 93.243.107.34.bc.googleusercontent.com.https: Flags [FP.], seq 39:63, ack 1, win 246, options [nop,nop,TS va
l 2962263852 ecr 1801340511], length 24
22:10:30.497679 wlp0s20f3 In  IP 93.243.107.34.bc.googleusercontent.com.https > deepti-ThinkPad-E14-Gen-5.52450: Flags [.], ack 64, win 277, options [nop,nop,TS val 1801391510
 ecr 2962263852], length 0
22:10:30.497679 wlp0s20f3 In  IP 93.243.107.34.bc.googleusercontent.com.https > deepti-ThinkPad-E14-Gen-5.52450: Flags [F.], seq 1, ack 64, win 277, options [nop,nop,TS val 18
01391510 ecr 2962263852], length 0
22:10:30.497770 wlp0s20f3 Out IP deepti-ThinkPad-E14-Gen-5.52450 > 93.243.107.34.bc.googleusercontent.com.https: Flags [.], ack 2, win 246, options [nop,nop,TS val 2962263869
ecr 1801391510], length 0
```

- **Timestamp**: 22:10:30.480896 - This indicates the exact time when the packet was captured.
- **Interface**: wlp0s20f3 - This is the network interface through which the packet was captured. In this case, it appears to be a wireless network interface.
- **Protocol**: IP - The packet is using the IPv4 protocol.
- **Source Address**: deepti-ThinkPad-E14-Gen-5.52450 - This seems to be the hostname or a descriptive label for the source device. The actual IP address is not shown here but would typically be included in a detailed packet log.
- **Destination Address**: 93.243.107.34.bc.googleusercontent.com - This is the hostname of the destination IP address. It appears to be associated with a Google service, as indicated by the googleusercontent.com domain. The IP address is 93.243.107.34.
- **Destination Port**: https - This indicates that the destination port is 443, which is commonly used for HTTPS (secure HTTP) traffic.
- **Flags**: [P.] - This indicates that the TCP packet has the PSH (Push) and ACK (Acknowledgment) flags set. The PSH flag tells the receiver to push the data to the application immediately, and the ACK flag acknowledges the receipt of data from the other side.
- **Sequence Number**: 2135179813:2135179852 - This specifies the range of the sequence numbers for the TCP segment. The sequence number indicates where in the data stream this segment belongs, and the range shows the number of bytes included in this packet.
- **Acknowledgment Number**: 72492986 - This is the acknowledgment number indicating the next expected byte from the other side of the connection.

- **Window Size**: 246 - This is the size of the TCP window, which indicates the amount of data the receiver is willing to accept. It is part of the flow control mechanism in TCP.
- **Options**: [nop,nop,TS val 2962263852 ecr 1801340511] - These are TCP options. In this case, it includes timestamp options. TS val is the timestamp value of the sender, and ecr is the timestamp echo reply value, which helps in measuring round-trip time and synchronization.
- **Length**: 39 - This indicates the length of the TCP payload in bytes.

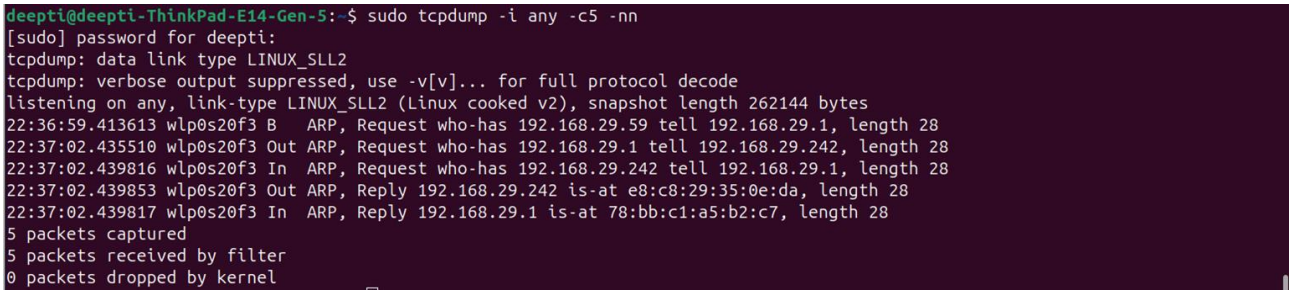
This packet capture shows an outgoing TCP segment from a device with a hostname deepti-ThinkPad-E14-Gen-5.52450 to an IP address 93.243.107.34, which resolves to bc.googleusercontent.com, on port 443 (HTTPS). The packet includes data (as indicated by the PSH flag) and acknowledges previous data from the other side. The TCP options include timestamps for performance measurement.

Tcpdump continues to capture packets until it receives an interrupt signal. You can interrupt capturing by pressing Ctrl+C. As you can see in this example, tcpdump captured 8 packets. To limit the number of packets captured and stop tcpdump, use the -c (for *count*) option:



In this case, tcpdump stopped capturing automatically after capturing five packets.

When troubleshooting network issues, it is often easier to use the IP addresses and port numbers; disable name resolution by using the option -n and port resolution with -nn:



- **Timestamp**: 22:36:59.413613 - This indicates the exact time when the ARP request was captured.
- **Interface**: wlp0s20f3 - This is the network interface through which the packet was captured. In this case, it appears to be a wireless interface.
- **Protocol**: ARP - This denotes that the packet is an ARP request.
- **Type**: Request - This indicates that the ARP packet is a request. An ARP request is used to inquire about the MAC address associated with a specific IP address.

- **ARP Request Details**:
- **Who-has**: 192.168.29.59 - This specifies the IP address for which the sender is seeking the corresponding MAC address.
- **Tell**: 192.168.29.1 - This specifies the IP address of the sender, which is making the request.
- **Length**: 28 - This is the length of the ARP packet in bytes.

This log entry shows that a device on the network with the IP address 192.168.29.1 is sending an ARP request to find out the MAC address associated with the IP address 192.168.29.59. The request is broadcast to all devices on the local network segment, and the device with the IP address 192.168.29.59 should reply with its MAC address so that the sender can communicate with it directly. The length of the ARP request packet is 28 bytes.

Filtering Packets:

One of tcpdump's most powerful features is its ability to filter the captured packets using a variety of parameters, such as source and destination IP addresses, ports, protocols, etc. Let's look at some of the most common ones.

Protocol

To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:

```
deepti@deepti-ThinkPad-E14-Gen-5:~$ sudo tcpdump -i any -c5 icmp
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
23:01:41.809063 wlp0s20f3 Out IP deepti-ThinkPad-E14-Gen-5 > reliance.reliance: ICMP deepti-ThinkPad-E14-Gen-5 udp port echo un
reachable, length 37
23:03:14.878644 wlp0s20f3 Out IP deepti-ThinkPad-E14-Gen-5 > reliance.reliance: ICMP deepti-ThinkPad-E14-Gen-5 udp port echo un
reachable, length 37
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

- **Timestamp**: 23:01:41.809063 - This indicates the exact time when the packet was captured.
- **Interface**: wlp0s20f3 - This is the network interface through which the packet was captured, in this case, a wireless interface.
- **Direction**: Out - This shows that the packet is being sent out from the interface.
- **Protocol**: IP - Indicates that the packet is using the IPv4 protocol.
- **Source Address**: deepti-ThinkPad-E14-Gen-5 - This represents the source of the packet. In a detailed packet log, this would be replaced with the actual IP address. Here it’s a descriptive label.
- **Destination Address**: reliance.reliance - This is the hostname of the destination. The actual IP address would be shown in a more detailed log.
- **ICMP Type**: ICMP - Indicates that the packet is using the ICMP protocol, which is often used for error messages and operational information.

- **ICMP Message**: deepti-ThinkPad-E14-Gen-5 udp port echo unreachable - This is an ICMP message indicating that a UDP packet could not be delivered. The specific type of ICMP message is "Destination Unreachable," with a code specifying "Port Unreachable." This indicates that the destination port on the remote host is not available or not reachable.
- **Length**: 37 - This indicates the length of the ICMP message in bytes.

This log entry shows that an ICMP message was sent out from the device with the label deepti-ThinkPad-E14-Gen-5 to a destination with the hostname reliance.reliance. The ICMP message is indicating that a UDP packet sent to a certain port could not be delivered because the port is unreachable. This is commonly used for diagnostic purposes to inform the sender that the destination port is not available or that there was an issue with the communication.

Host

Limit capture to only packets related to a specific host by using the host filter:

```
deepti@deepti-ThinkPad-E14-Gen-5:~$ sudo tcpdump -i any -c5 -nn host 192.168.29.1
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
23:32:11.482899 wlp0s20f3 In  ARP, Request who-has 192.168.29.242 tell 192.168.29.1, length 28
23:32:11.482923 wlp0s20f3 Out ARP, Reply 192.168.29.242 is-at e8:c8:29:35:0e:da, length 28
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

ARP (Address Resolution Protocol), is used in networking to map IP addresses to MAC (Media Access Control) addresses.

- **23:32:11.482899**: This is the timestamp indicating when the ARP request was made.
- **wlp0s20f3**: This is the name of the network interface (likely a Wi-Fi interface) on your device that is sending or receiving the ARP packet.
- **In ARP, Request who-has 192.168.29.242 tell 192.168.29.1**: This means that the device at IP address 192.168.29.1 (likely your router or gateway) is asking which device on the network has the IP address 192.168.29.242. It wants to know the MAC address associated with that IP.
- **length 28**: This indicates the size of the ARP packet.

Port

To filter packets based on the desired service or port, use the port filter. For example, capture packets related to a web (HTTP) service by using this command:

```
deepti@deepti-ThinkPad-E14-Gen-5:~$ sudo tcpdump -i any -c5 -nn port 80
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
23:37:33.095655 wlp0s20f3 Out IP 192.168.29.242.53008 > 91.189.91.96.80: Flags [S], seq 827465360, win 32120, options [mss 1460
,sackOK,TS val 2633572687 ecr 0,nop,wscale 7], length 0
23:37:33.347456 wlp0s20f3 In  IP 91.189.91.96.80 > 192.168.29.242.53008: Flags [S.], seq 2432622692, ack 827465361, win 65160,
options [mss 1460,sackOK,TS val 3706575337 ecr 2633572687,nop,wscale 7], length 0
23:37:33.347546 wlp0s20f3 Out IP 192.168.29.242.53008 > 91.189.91.96.80: Flags [.], ack 1, win 251, options [nop,nop,TS val 263
3572939 ecr 3706575337], length 0
23:37:33.347739 wlp0s20f3 Out IP 192.168.29.242.53008 > 91.189.91.96.80: Flags [P.], seq 1:89, ack 1, win 251, options [nop,nop
,TS val 2633572939 ecr 3706575337], length 88: HTTP: GET / HTTP/1.1
23:37:33.710370 wlp0s20f3 In  IP 91.189.91.96.80 > 192.168.29.242.53008: Flags [P.], seq 1:186, ack 89, win 509, options [nop,n
op,TS val 3706575590 ecr 2633572939], length 185: HTTP: HTTP/1.1 204 No Content
5 packets captured
8 packets received by filter
0 packets dropped by kernel
```

- **23:37:33.347739**: The timestamp showing when this network packet was sent.
- **wlp0s20f3**: The network interface (probably a Wi-Fi interface) on your device that sent this packet.
- **Out IP 192.168.29.242.53008 > 91.189.91.96.80**: This indicates that an IP packet is being sent from your device's local IP address (192.168.29.242) using port 53008 to the external IP address 91.189.91.96 on port 80 (which is typically used for HTTP traffic).
- **Flags [P.]**: The flags in the TCP header. The "P" flag means "Push," indicating that the data should be sent immediately. The "." indicates that this packet is part of a sequence of packets.
- **seq 1:89, ack 1, win 251**: This part refers to the sequence number, acknowledgment number, and window size, which are used to manage the flow of data between the devices.
- **options [nop,nop,TS val 2633572939 ecr 3706575337]**: These are TCP options, where "nop" stands for "No Operation," and "TS val" and "ecr" refer to the timestamp value and echo reply, which help with timing and ordering of packets.
- **length 88**: The length of the packet, indicating that it carries 88 bytes of data.
- **HTTP: GET / HTTP/1.1**: This is an HTTP GET request, which is asking the server at 91.189.91.96 for the root resource (/) over HTTP 1.1.

In simpler terms, the device made a web request to the server at IP address 91.189.91.96. The server likely hosts a website, and this packet is part of the process of fetching that webpage.

Source IP/hostname

You can also filter packets based on the source or destination IP Address or hostname. For example, to capture packets from host 192.168.29.242:

```
deepti@deepti-ThinkPad-E14-Gen-5:~$ sudo tcpdump -i any -c5 -nn src 192.168.29.242
[sudo] password for deepti:
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
00:01:40.408344 wlp0s20f3 Out ARP, Reply 192.168.29.242 is-at e8:c8:29:35:0e:da, length 28
00:01:41.298485 wlp0s20f3 Out ARP, Reply 192.168.29.242 is-at e8:c8:29:35:0e:da, length 28
00:01:47.683532 wlp0s20f3 Out IP 192.168.29.242 > 224.0.0.22: igmp v3 report, 1 group record(s)
^C
3 packets captured
3 packets received by filter
0 packets dropped by kernel
```

This message is another ARP (Address Resolution Protocol) packet, but this time it's an ARP Reply. Here's what it means:

- **00:01:40.408344**: The timestamp indicating when the ARP reply was sent.
- **wlp0s20f3**: The network interface on your device that sent the ARP reply (likely a Wi-Fi interface).
- **Out ARP, Reply 192.168.29.242 is-at e8:c8:29:35:0e**

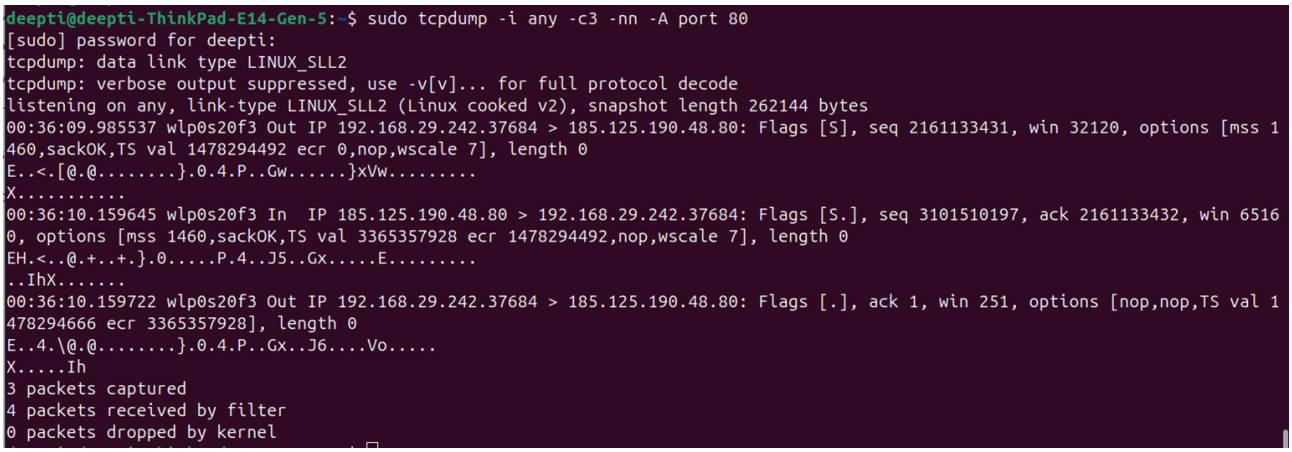
: This indicates that your device is responding to an earlier ARP request. It is informing the network that the IP address 192.168.29.242 is associated with the MAC address e8:c8:29:35:0e:da.

- **length 28**: This is the length of the ARP packet, which is 28 bytes.

When a device on a network wants to communicate with another device, it needs to know the MAC address corresponding to the target device's IP address. If it doesn't have this information, it sends out an ARP request asking "Who has IP address 192.168.29.242?"
The device with that IP address (in this case, your device) replies with an ARP reply, saying "I have IP address 192.168.29.242, and my MAC address is e8:c8:29:35:0e:da."

Checking packet content

In the previous examples, we're checking only the packets' headers for information such as source, destinations, ports, etc. Sometimes this is all we need to troubleshoot network connectivity issues. Sometimes, however, we need to inspect the content of the packet to ensure that the message we're sending contains what we need or that we received the expected response. To see the packet content, tcpdump provides two additional flags: `-X` to print content in hex, and ASCII or `-A` to print the content in ASCII.



- **00:12:33.061840**: The timestamp indicating when the packet was sent.
- **wlp0s20f3**: The network interface on your device that sent the packet (likely a Wi-Fi interface).
- **Out IP 192.168.29.242.43036 > 91.189.91.97.80**:
    - **192.168.29.242**: The local IP address of your device.
    - **43036**: The source port on your device, randomly chosen for this connection.
    - **91.189.91.97**: The destination IP address (likely a server you're connecting to).
    - **80**: The destination port, which is typically used for HTTP (web) traffic.
- **Flags [S]**: The TCP flags indicate the purpose of the packet. "[S]" means it's a **SYN** packet, which is used to initiate a TCP connection.
- **seq 179238273**: The sequence number of the packet, used to keep track of the data being sent and received in the connection.
- **win 32120**: The window size, indicating how much data the sender is willing to receive before needing an acknowledgment.
- **options [mss 1460,sackOK,TS val 1391577869 ecr 0,nop,wscale 7]**: These are TCP options:
    - **mss 1460**: Maximum Segment Size, indicating the maximum amount of data that can be sent in one TCP segment.
    - **sackOK**: Selective Acknowledgment, allowing the receiver to acknowledge received segments individually rather than just cumulatively.
    - **TS val 1391577869 ecr 0**: Timestamp value and echo reply, used for better timing and performance.

    - **nop**: No Operation, used as a placeholder.
    - **wscale 7**: Window scale factor, used to scale the window size for high-speed connections.
- **length 0**: Indicates that this packet contains no data; it's purely for establishing the connection.

This packet is part of the **TCP handshake**, which is a three-step process used to establish a reliable connection between two devices:

1. **SYN**: Your device sends a SYN (synchronize) packet to the server at IP address 91.189.91.97 on port 80, asking to establish a connection.
2. **SYN-ACK**: The server responds with a SYN-ACK packet, acknowledging the request and asking to synchronize its own sequence numbers.
3. **ACK**: Your device sends back an ACK packet, completing the handshake and establishing the connection.