

# Lab 2 (Week 2)

## Wireshark and Tcpdump

CAN201

Dr. Fei Cheng

Dr. Gordon Boateng

# Outline

- Packet sniffer
- Wireshark
  - Getting Wireshark
  - Running Wireshark
  - Taking Wireshark for a test run
- Tcpdump
- Hands-on Practice

# Introduction

- One's understanding of network protocols can often be greatly deepened by
  - “seeing protocols in action”.
  - “playing around with protocols”.
- Wireshark (Packet Sniffer) can help us in
  - observing the sequence of messages exchanged between two protocol entities.
  - delving down into the details of protocol operation.

Network testbed facilitating certain scenario can help us in

- causing protocols to perform certain actions.
- observing these actions and their consequences.

# Packet Sniffer

- Packet sniffer: captures (“sniffs”) messages being sent/received from/by the sniffing target (e.g., your computer); also, it typically stores and displays the contents of the various protocol fields.
  - Wireshark
  - Tshark
  - Tcpdump

# Packet Sniffer

- Packet sniffer: captures (“sniffs”) messages being sent/received from/by the sniffing target (e.g., your computer); also, it typically stores and displays the contents of the various protocol fields.
  - Wireshark
  - Tshark
  - Tcpdump
- Traffic control framework: control (or even manipulate) the original messages instead of a copy
  - NetfilterQueue

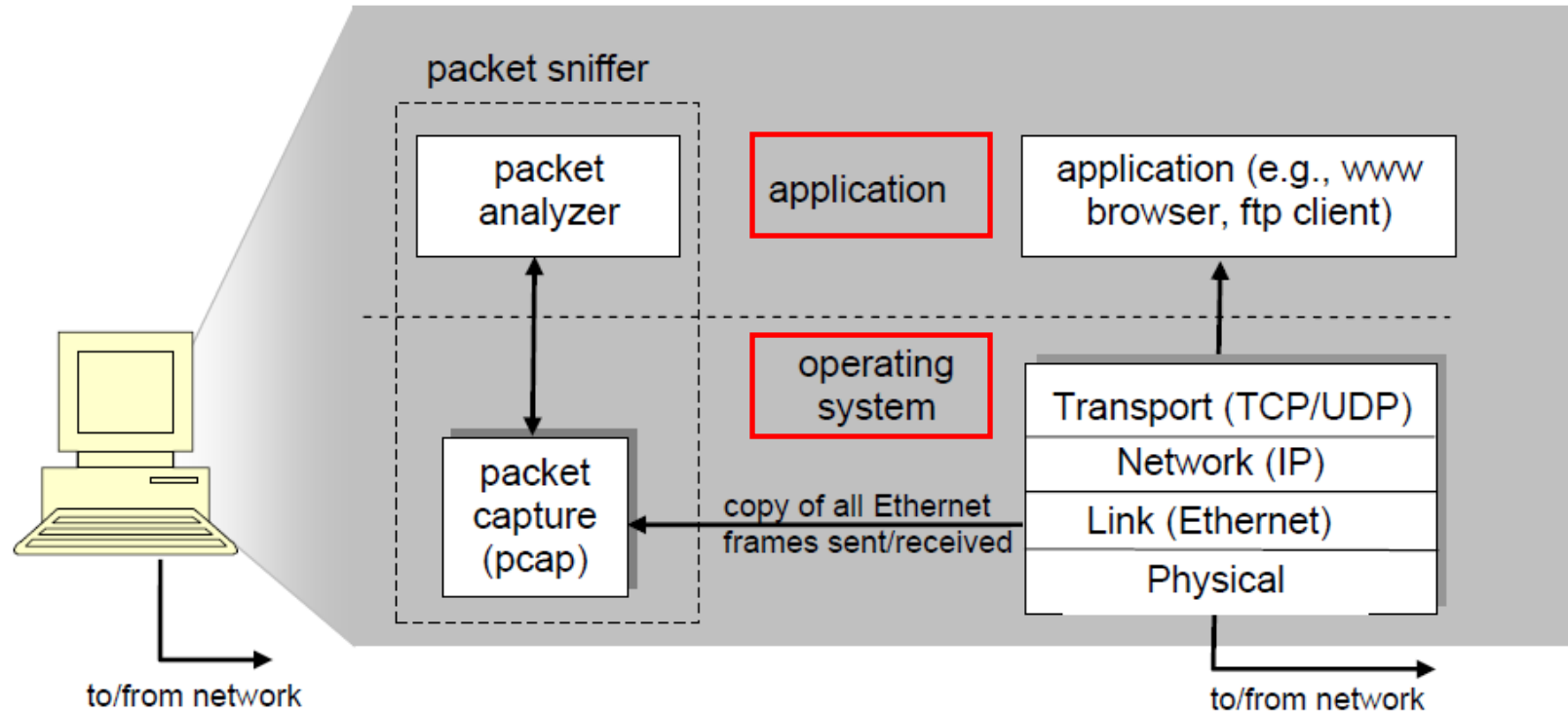
# Packet Sniffer

- Packet sniffer: captures (“sniffs”) messages being sent/received from/by the sniffing target (e.g., your computer); also, it typically stores and displays the contents of the various protocol fields.
  - Wireshark
  - Tshark
  - Tcpdump
- Traffic control framework: control (or even manipulate) the original messages instead of a copy
  - NetfilterQueue
- Network scanner: scan the target (system or network) via sending probing packets
  - Nmap

# Tools' Links

- Wirshark: <https://www.wireshark.org/>
- Tshark: <https://www.wireshark.org/docs/man-pages/tshark.html>
- Tcpdump: <https://www.tcpdump.org/>
- NetfilterQueue: <https://pypi.org/project/NetfilterQueue/>
- Nmap: <https://nmap.org/>

# Packet Sniffer Structure





# Wireshark

Wireshark is a free network protocol analyzer and so an ideal packet analyzer for our labs:

- runs on Windows, Mac, and Linux/Unix computers.
- includes the capability to analyze hundreds of protocols
- has a well-designed user interface
- operates in computers using Ethernet, serial (PPP and SLIP), 802.11 wireless LANs, and many other link-layer technologies.
- it is stable, has a large user base and well-documented support:
  - User guide ([http://www.wireshark.org/docs/wsug\\_html\\_chunked/](http://www.wireshark.org/docs/wsug_html_chunked/))
  - Man pages (<http://www.wireshark.org/docs/man-pages/>)
  - Detailed FAQ (<http://www.wireshark.org/faq.html>)

# Getting Wireshark

Download and install the Wireshark software:

- Go to (<http://www.wireshark.org/download.html>) and download and install the Wireshark binary for your computer.
- For Ubuntu (Linux), look at this:
  - <https://cloudcone.com/docs/article/how-to-install-wireshark-on-ubuntu-18-04-lts/>

## Download Wireshark

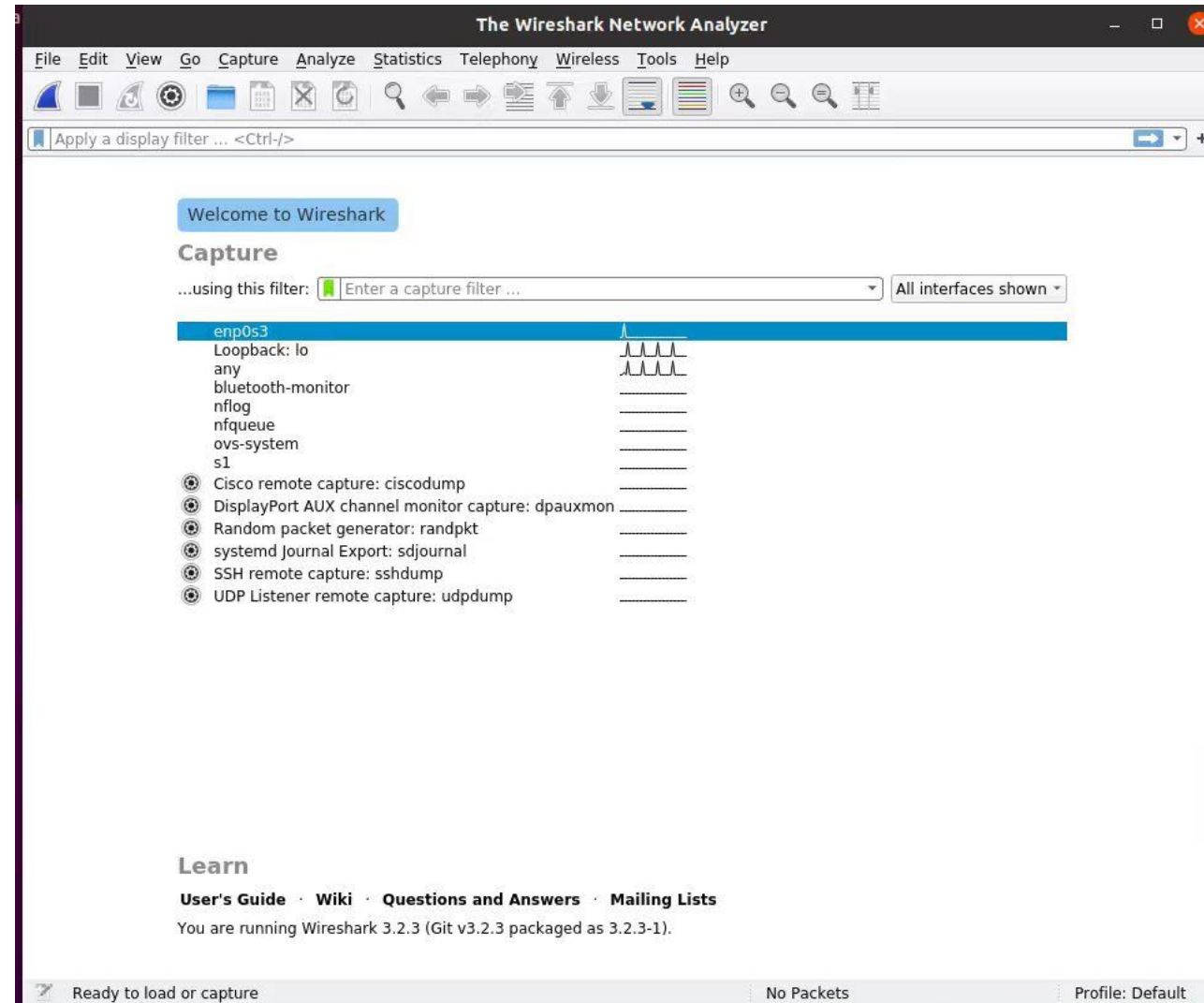
The current stable release of Wireshark is 3.4.9. It supersedes all previous releases. You can also download the latest development release (3.6.0rc1) and documentation.

### Stable Release (3.4.9)

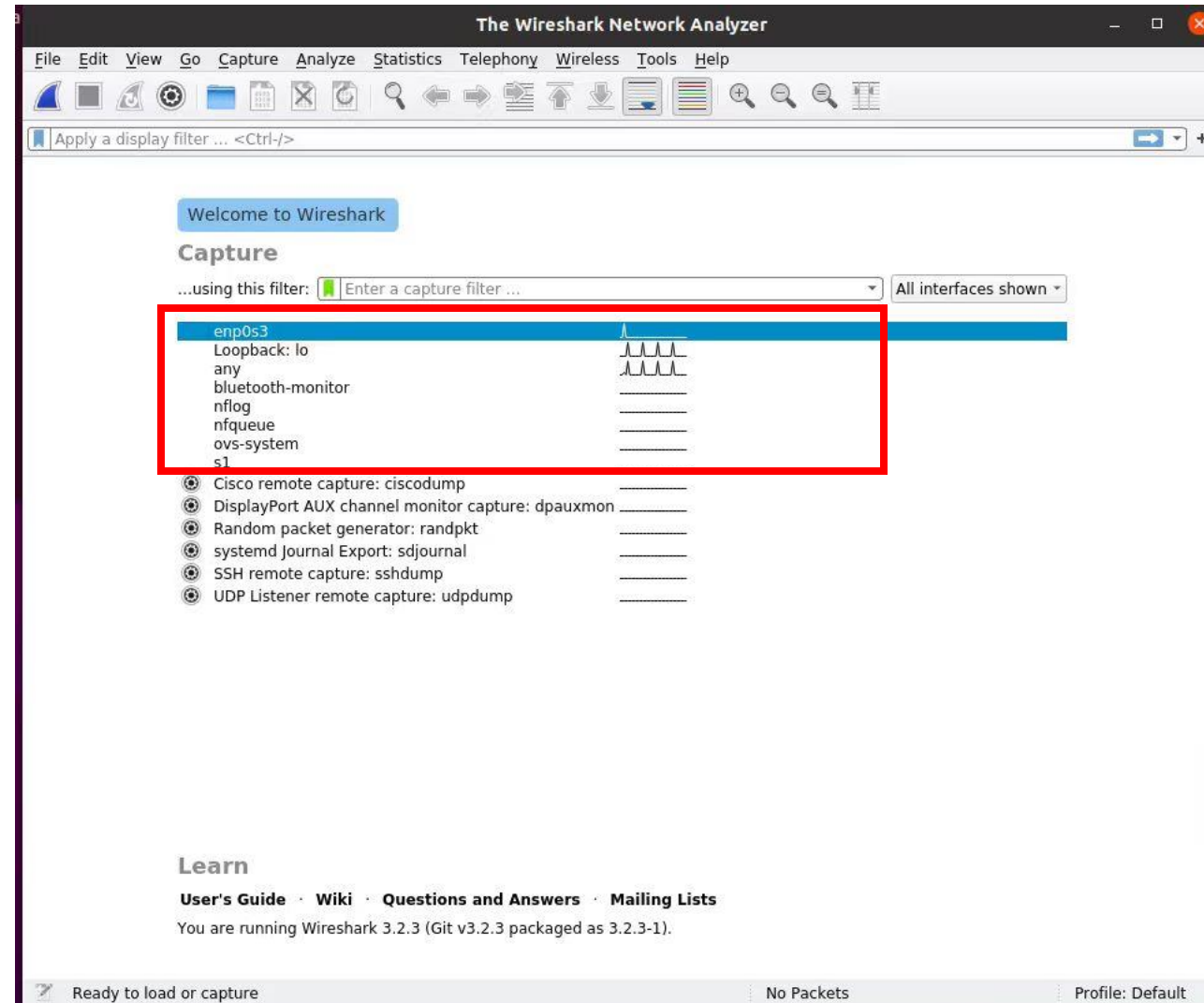
- Windows Installer (64-bit)
- Windows Installer (32-bit)
- Windows PortableApps® (32-bit)
- macOS Intel 64-bit .dmg
- Source Code

### Old Stable Release (3.2.17)

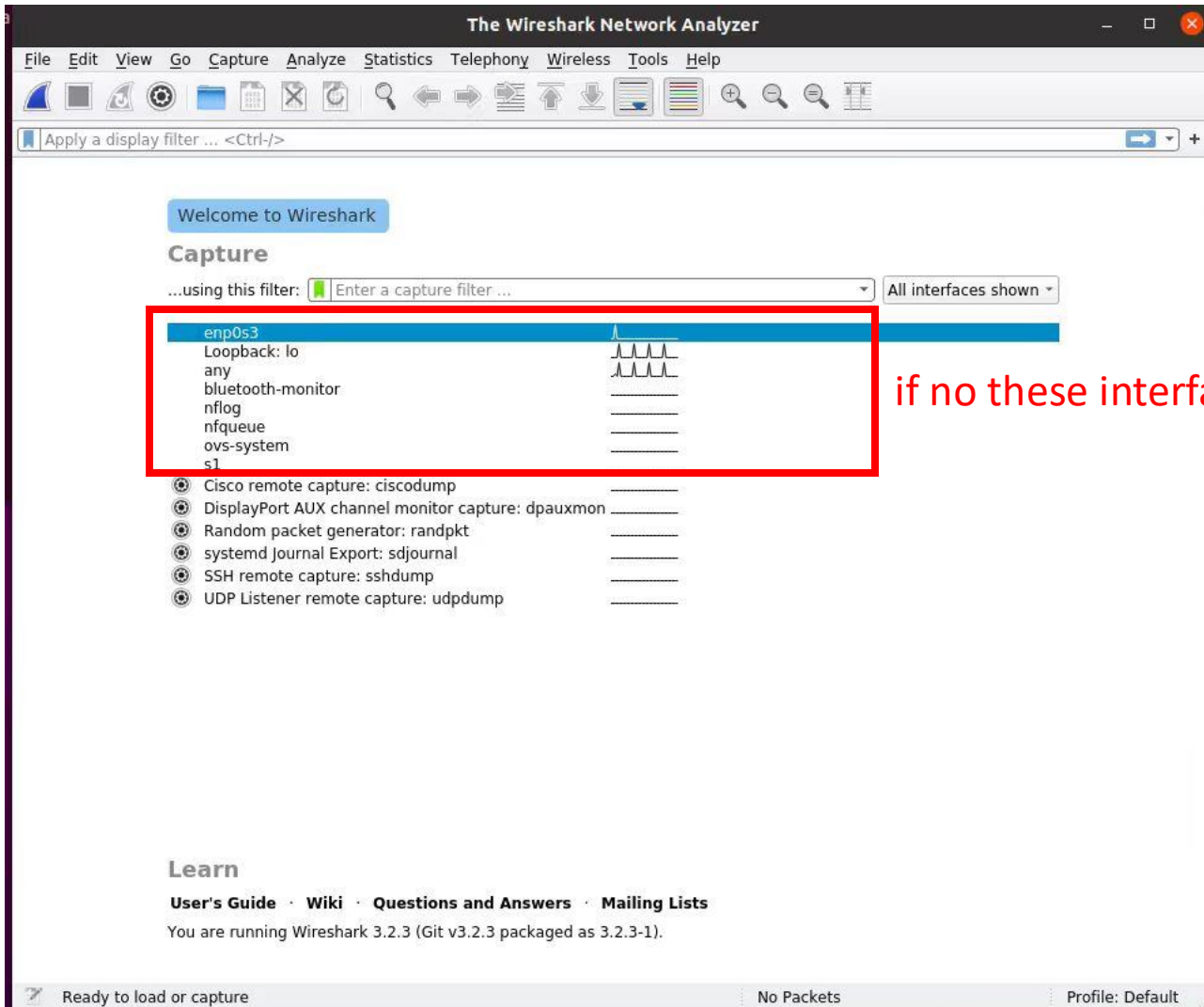
# First time running Wireshark



# First time running Wireshark



# First time running Wireshark



Linux/macOS

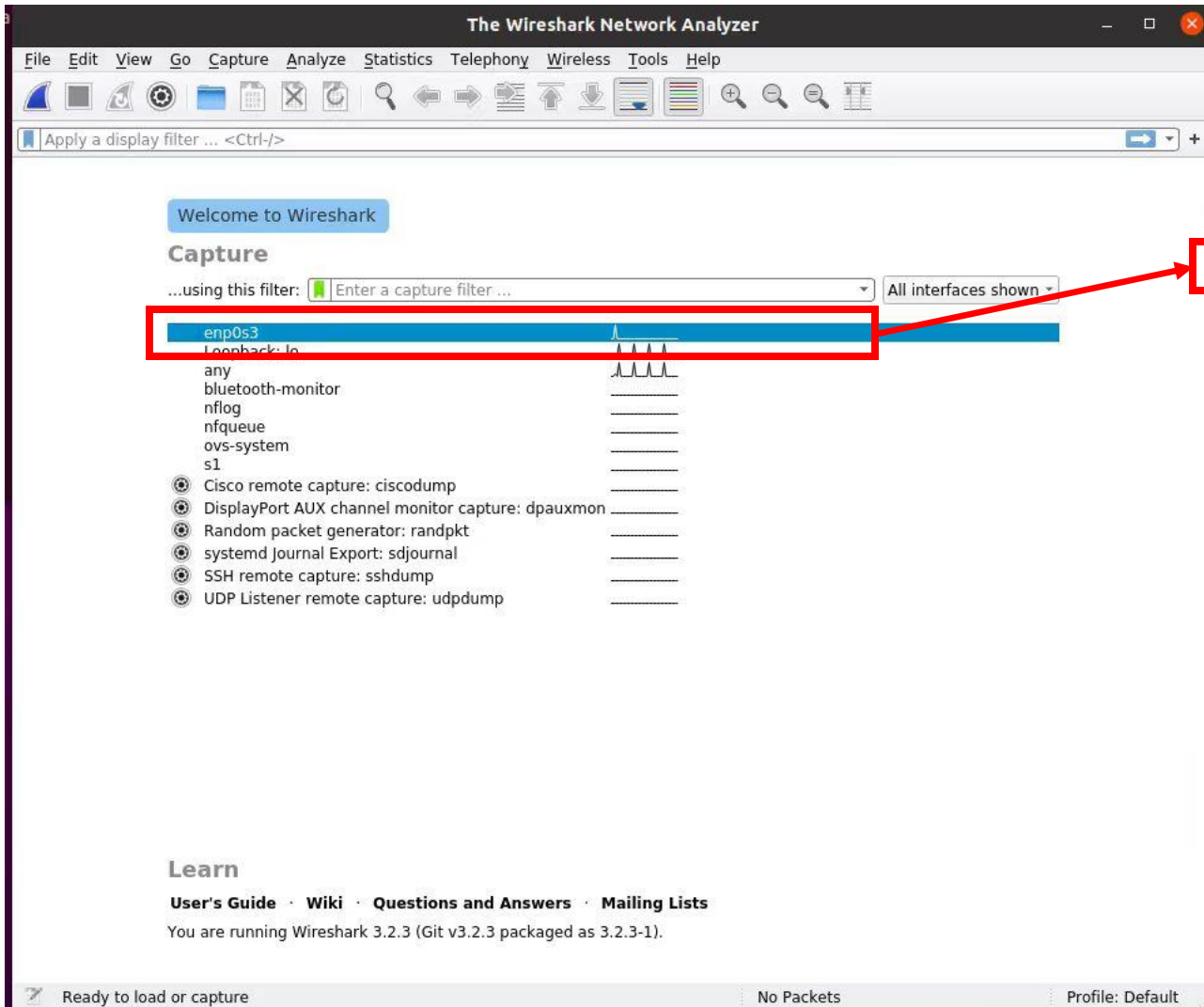
Open terminal:  
sudo wireshark

if no these interfaces:

Windows

Run as administrator!

# Select an interface for sniffing!

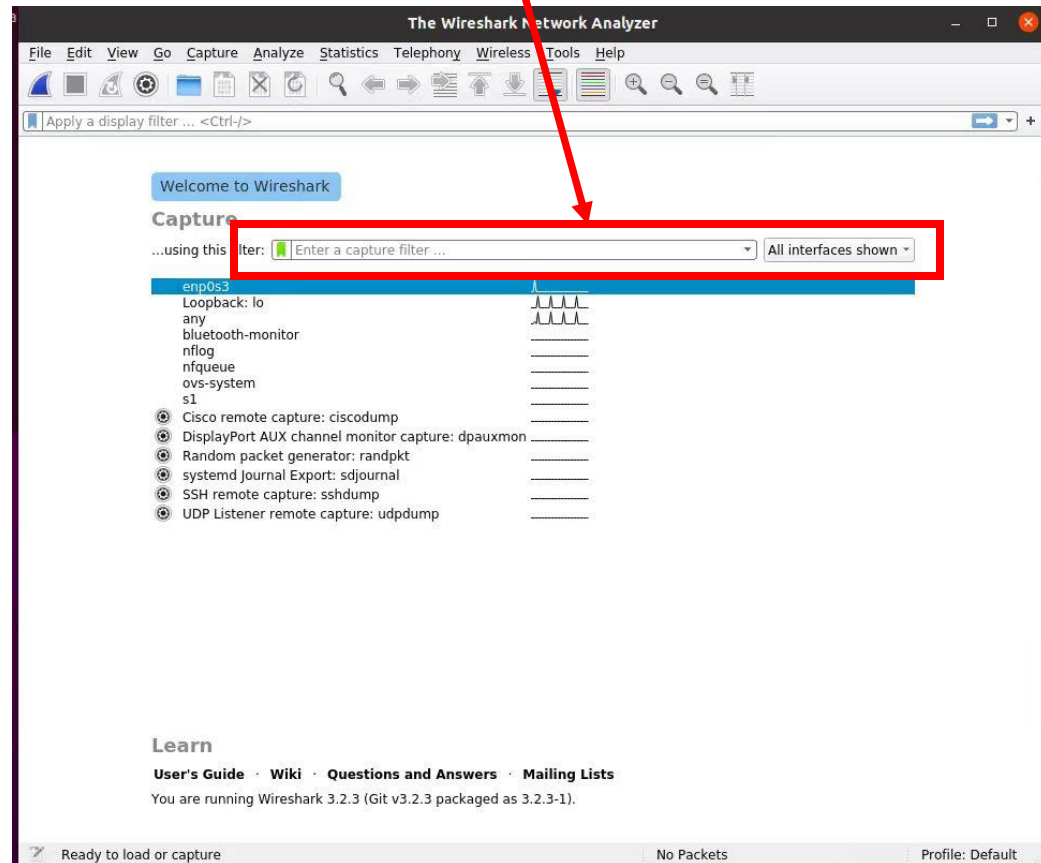


```
can201@can201-VirtualBox: ~  
can201@can201-VirtualBox:~$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::5ab2:4698:c697:2fd2 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:2b:1a:15 txqueuelen 1000 (Ethernet)  
    RX packets 689294 bytes 1036621965 (1.0 GB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 148231 bytes 9101685 (9.1 MB)  
    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  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 1654 bytes 121846 (121.8 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 1654 bytes 121846 (121.8 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
can201@can201-VirtualBox:~$
```

Terminal:  
ifconfig

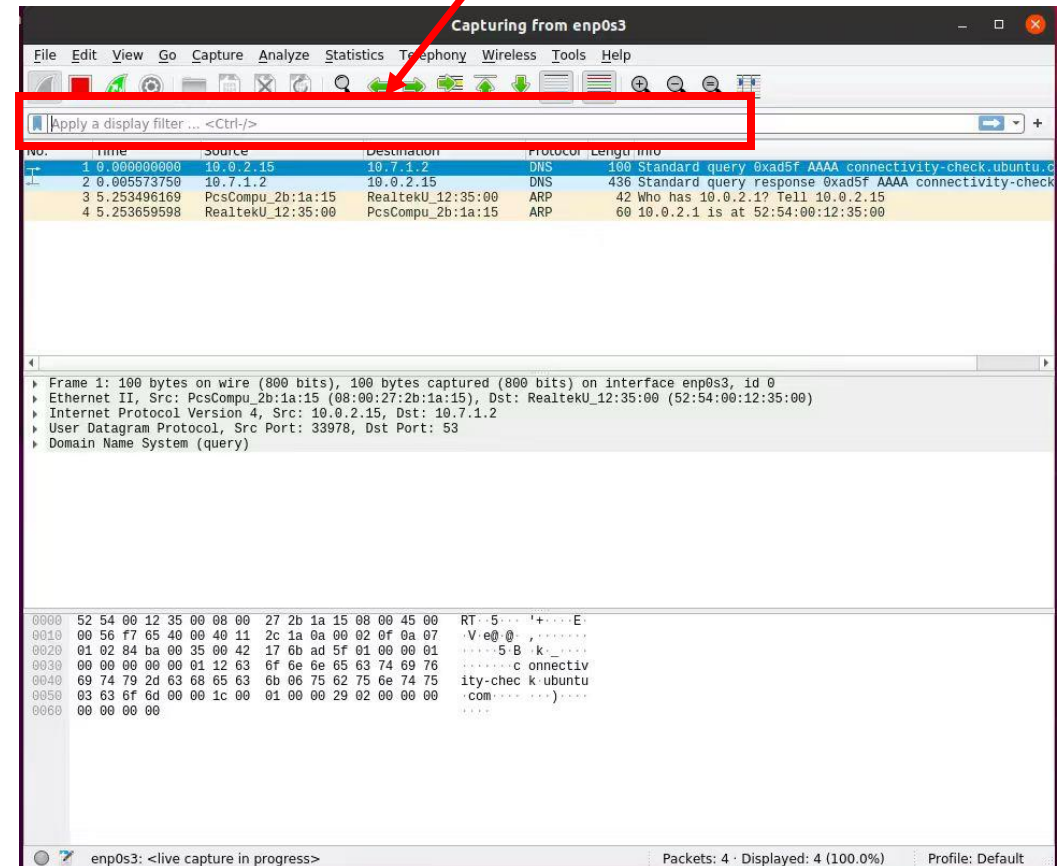
# Filtering!

Capture Filter



Welcome Page

Display Filter

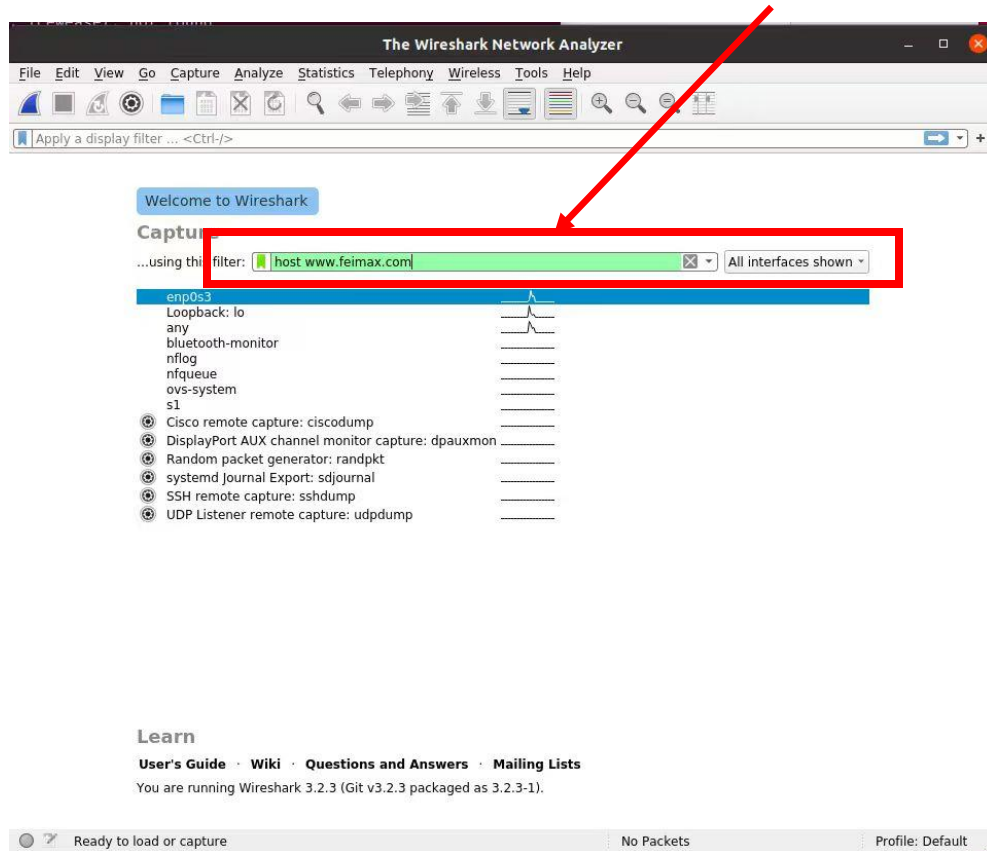


Capturing Page

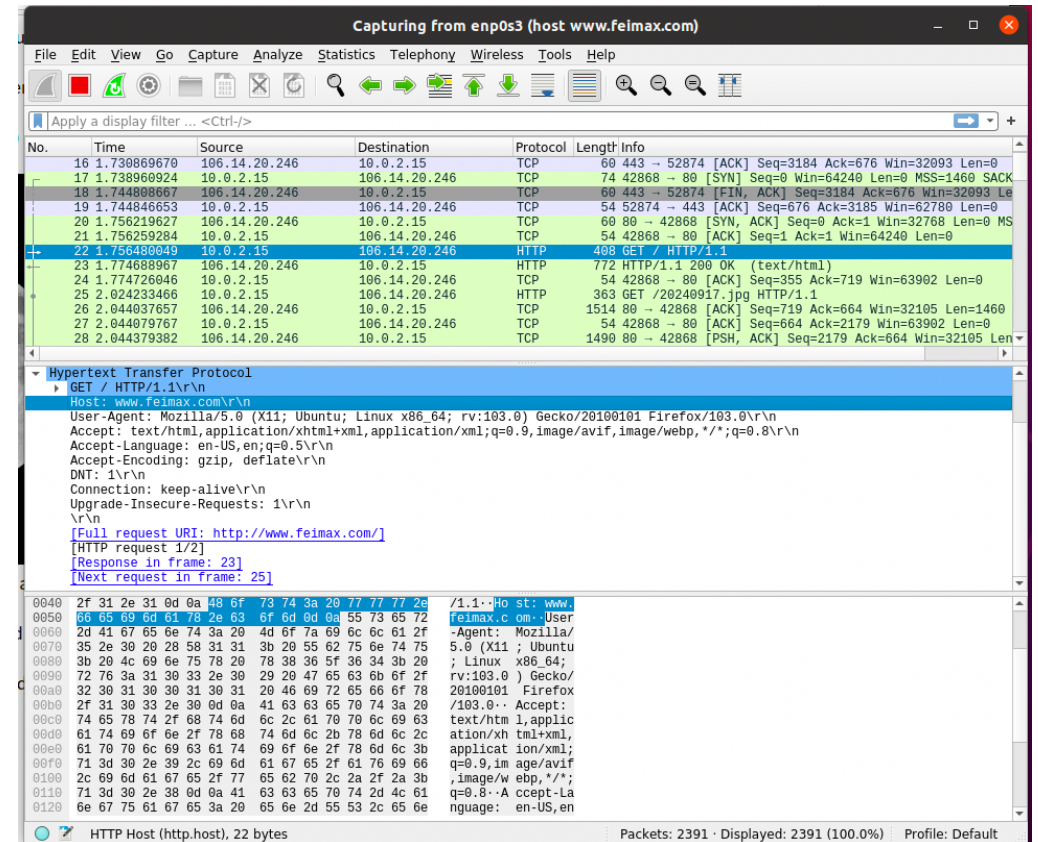


# Capturing Page

Use this capturing filter... and open <http://www.feimax.com> in the browser ...



=>





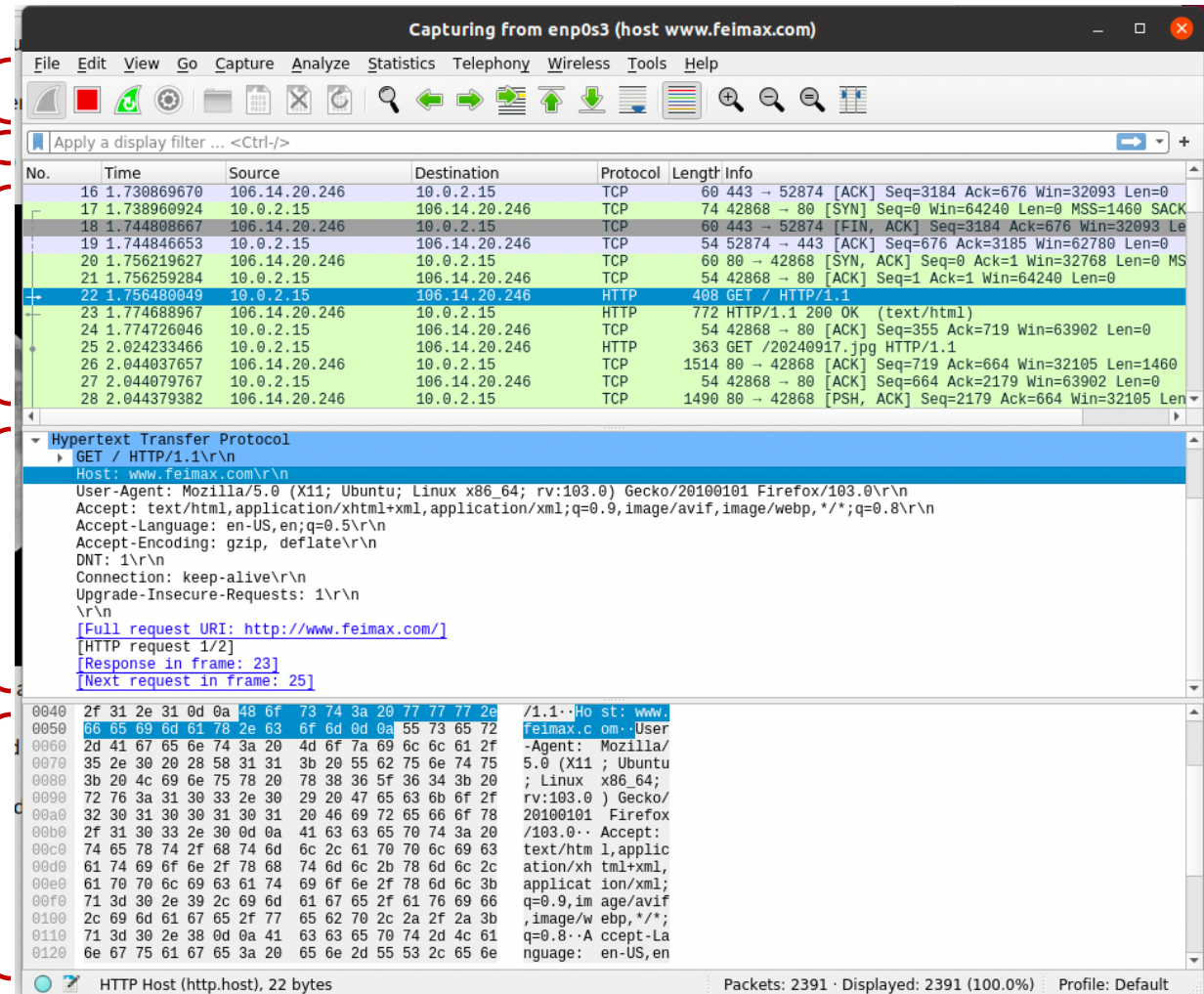
# Capturing Page

Command Menus  
Display Filter Specification

Listing of captured packets

Details of selected  
packet header

Packet content in  
HEX and ASCII



First try with Wireshark!

# More filters

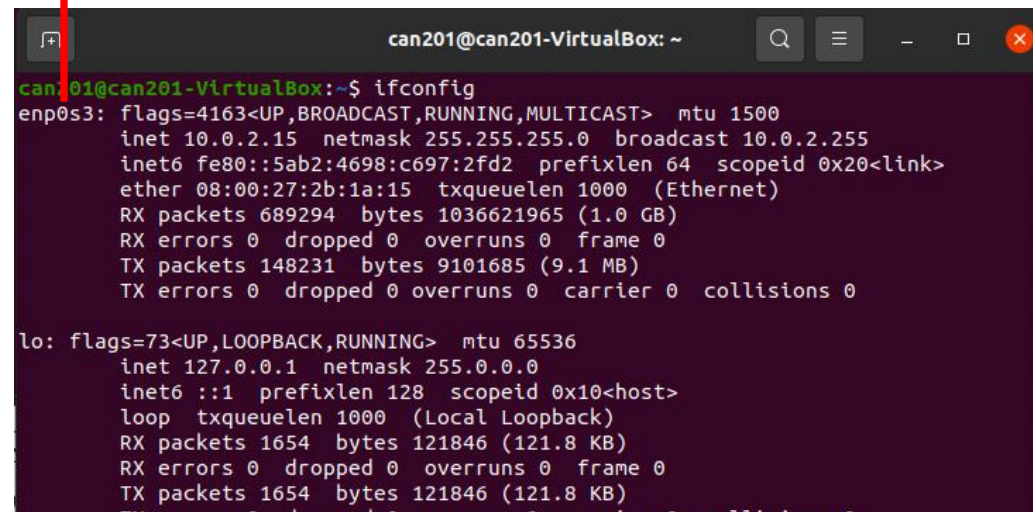
Scenario	Capture Filter (before capture)	Display Filter (after capture)
Traffic from a specific IP	host 192.168.1.10	ip.addr == 192.168.1.10
Traffic from a subnet	net 192.168.1.0/24	ip.addr == 192.168.1.0/24
HTTP traffic	tcp port 80	http
HTTPS (TLS) traffic	tcp port 443	tls
DNS traffic	udp port 53	dns
ICMP (Ping) traffic	icmp	icmp
SSH traffic (port 22)	tcp port 22	tcp.port == 22
Multiple ports (20–25)	tcp portrange 20-25	tcp.port >= 20 && tcp.port <= 25
Traffic from a source IP	src host 192.168.1.10	ip.src == 192.168.1.10
Traffic to a destination IP	dst host 192.168.1.20	ip.dst == 192.168.1.20
HTTP traffic from a specific IP	tcp port 80 and src host 192.168.1.10	ip.src == 192.168.1.10 && http
DNS queries for a specific domain	(not supported)	dns.qry.name == "www.google.com"
TCP retransmissions	(not supported)	tcp.analysis.retransmission

# Tcpdump

Refer to <https://opensource.com/article/18/10/introduction-tcpdump>

Command-line:

\$ sudo tcpdump -i enp0s3 -w data.pcap



```
can201@can201-VirtualBox: ~  
can201@can201-VirtualBox:~$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::5ab2:4698:c697:2fd2 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:2b:1a:15 txqueuelen 1000 (Ethernet)  
    RX packets 689294 bytes 1036621965 (1.0 GB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 148231 bytes 9101685 (9.1 MB)  
    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  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 1654 bytes 121846 (121.8 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 1654 bytes 121846 (121.8 KB)
```

# Hands-on Practice

Based on last week's two VMs, i.e., VM1 and VM2, do the following steps:

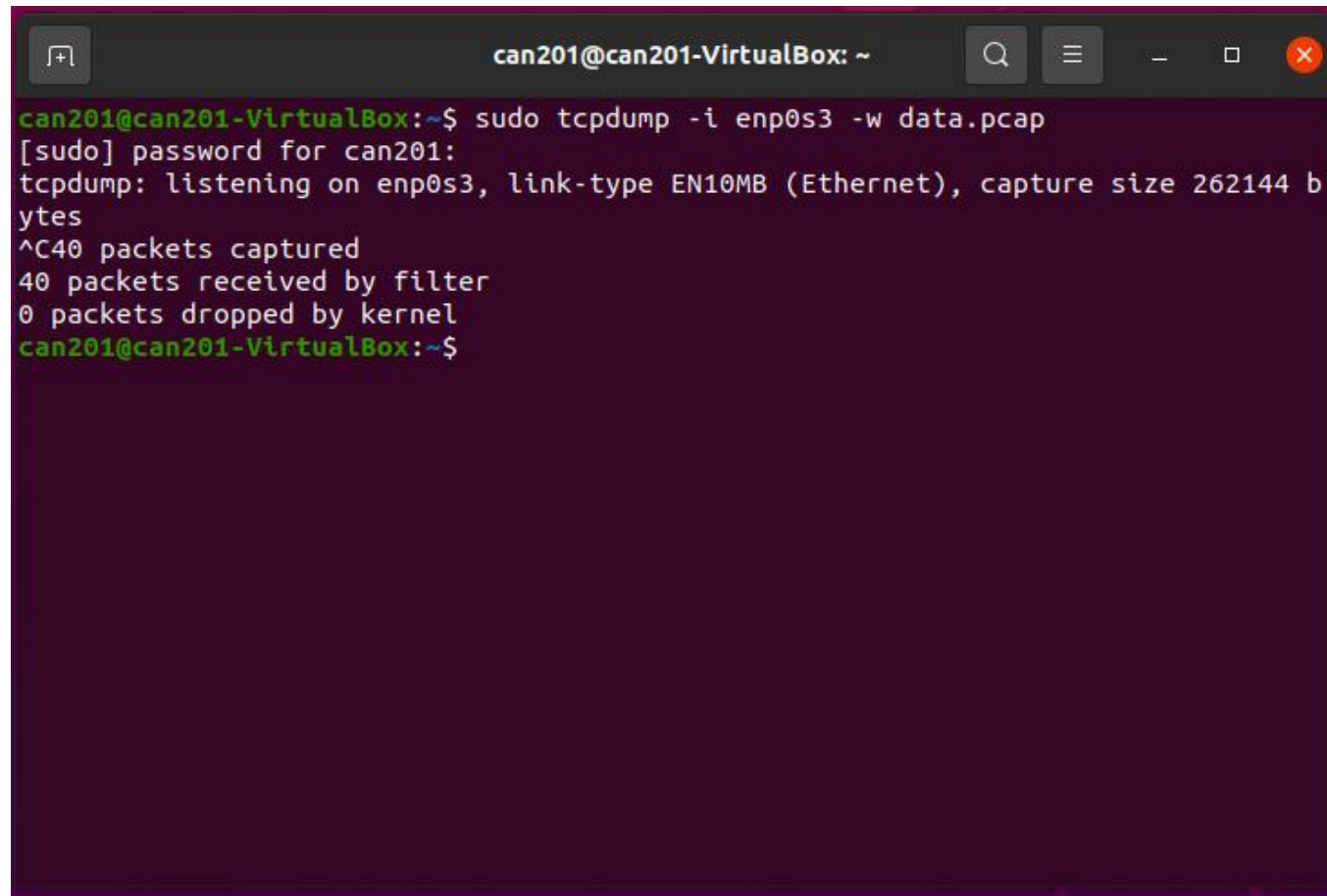
1. Run Tcpdump on VM2 for listening on the network interface and save the captured traffic into data.pcap file.
2. Use VM1 to ping VM2, no more than 10 ICMP packets (using 'Ctrl + C' to stop 'ping' command).
3. On VM2, use 'Ctrl + C' to stop 'tcpdump' command, and then use Wireshark to open the data.pcap file and display the captured ICMP packets.
4. In the display window of Wireshark, select and highlight one entry/line to indicate one of the captured ICMP ping packets.
5. Show the result to TA to manifest your understanding.

```
can201@can201-VirtualBox: ~  
can201@can201-VirtualBox:~$ sudo tcpdump -i enp0s3 -w data.pcap  
[sudo] password for can201:  
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 b  
ytes  
█
```

Use VM1 to ping VM2!!!

```
can201@can201-VirtualBox: ~  
can201@can201-VirtualBox:~$ ping 10.0.2.15  
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.  
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.026 ms  
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.059 ms  
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.044 ms  
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.046 ms  
64 bytes from 10.0.2.15: icmp_seq=5 ttl=64 time=0.049 ms  
^C  
--- 10.0.2.15 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4085ms  
rtt min/avg/max/mdev = 0.026/0.044/0.059/0.010 ms  
can201@can201-VirtualBox:~$ █
```

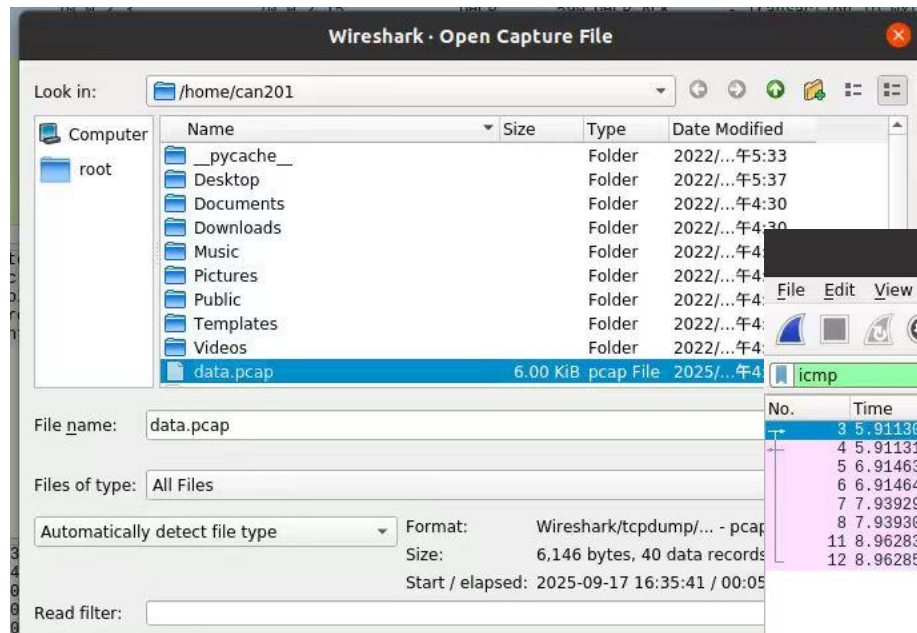
Run Tcpdump on VM2

A terminal window titled 'can201@can201-VirtualBox: ~' with standard window controls. The terminal shows the execution of the 'sudo tcpdump -i enp0s3 -w data.pcap' command. It prompts for a password, then reports that it is listening on enp0s3 with a capture size of 262144 bytes. After pressing Ctrl+C, it shows that 40 packets were captured, 40 were received by the filter, and 0 were dropped by the kernel.

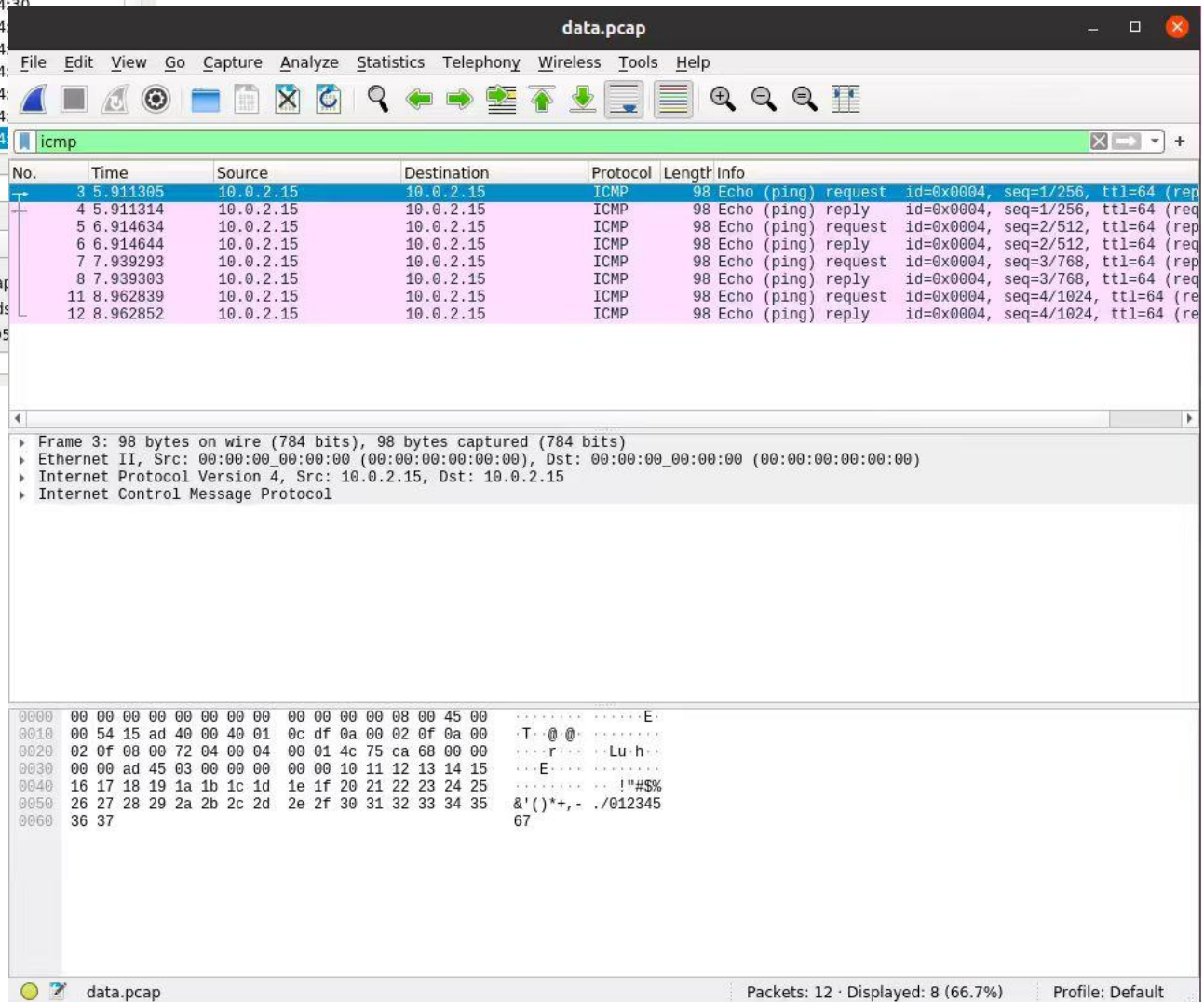
```
can201@can201-VirtualBox:~$ sudo tcpdump -i enp0s3 -w data.pcap
[sudo] password for can201:
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 b
ytes
^C40 packets captured
40 packets received by filter
0 packets dropped by kernel
can201@can201-VirtualBox:~$
```

On VM2, use 'Ctrl + C' to stop 'tcpdump' command





Open “data.pcap” on VM2  
Using Wireshark





- Thanks!