Computer Engineering 4DN4
Laboratory #1
Network Scanning and Packet Sniffing

Tcpdump, Windump, WireShark and Nmap are free open source network analyzers and essential tools used for network analysis, troubleshooting, and protocol and app development. Data can be captured and displayed "live" from active network interfaces. WireShark, in particular, includes a wide variety of features including display/capture filters, and has built-in decoding for most standard networking protocols. In this lab, we introduce these tools and use them to do some network scanning and protocol tracing. You must install them on your own PC or laptop and use them to do the experiments. The final experiments require that you have Python 3 installed.

1 Preparation

1. It is important that you attend the lectures or review the recorded lectures that discuss tcpdump, windump and WireShark.

2. Directions for installing tcpdump (or windump) are available in the lecture notes.

3. Go to `https://WWW.wireShark.org/download.html`. Choose and click on the stable release for your OS. There are versions for MacOS and Windows. For Linux, install it using your distribution's package manager.

4. Run the installer and confirm the installation. If you have already installed the packet capture (pcap) library (e.g., WinpCap when you installed WinDump), do not have the WireShark installer reinstall it.

5. Once the installer has finished, try running WireShark to make sure that the install finished properly. You should see a startup screen, something like that shown in Figure 1. It may look different depending on the installed version and your OS.

6. The WireShark startup screen contains links to their website and user guide:

   `https://www.wireSHARK.org`
   `https://www.wireshark.org/docs/wsug_html_chunked/`

7. Try some preliminary packet captures. First make sure that the proper interface is selected from the start screen. Alternately, you can select it by pulling down the `Capture > Interfaces` menu. Then click on the start capture button (green or blue shark fin) or use `Capture > Start`. You should see some packets being captured but you may have to initiate some network activity, e.g., using a web browser. Hit the stop capture (Red) button or use `Capture > Stop`. You can then inspect some of the captured packets.

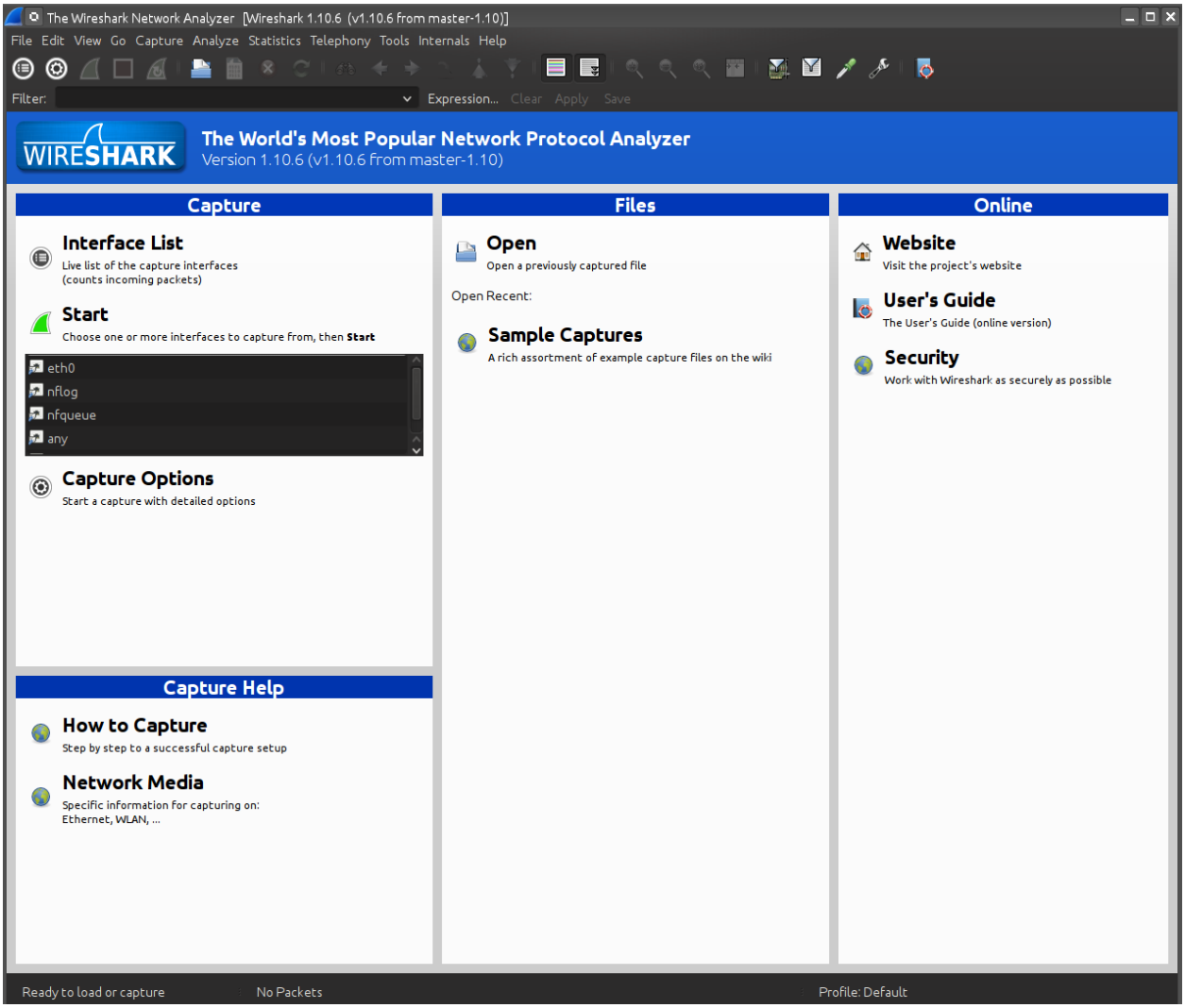

Figure 1: WireShark Startup Screen

There are various sections of the WireShark capture screen. Below the menu section there are three panes:

1. Captured packet listing. This gives a list of the all packets that have been captured. You can scroll through then and use your mouse to select packets.

2. Packet details. This shows the protocol headers and their content for the selected packet.

3. Packet content. This a canonical hex+ASCII display of the selected packet content.

Display Filter: Below the menu bar on the left, there is a "Filter:" text box that can be used to limit what is displayed in the packet capture panes. This is useful when there is a lot of network activity that would otherwise clutter up the listing. An example of using the display filter is as follows. First, figure out the IP address associated with the host that you are planning

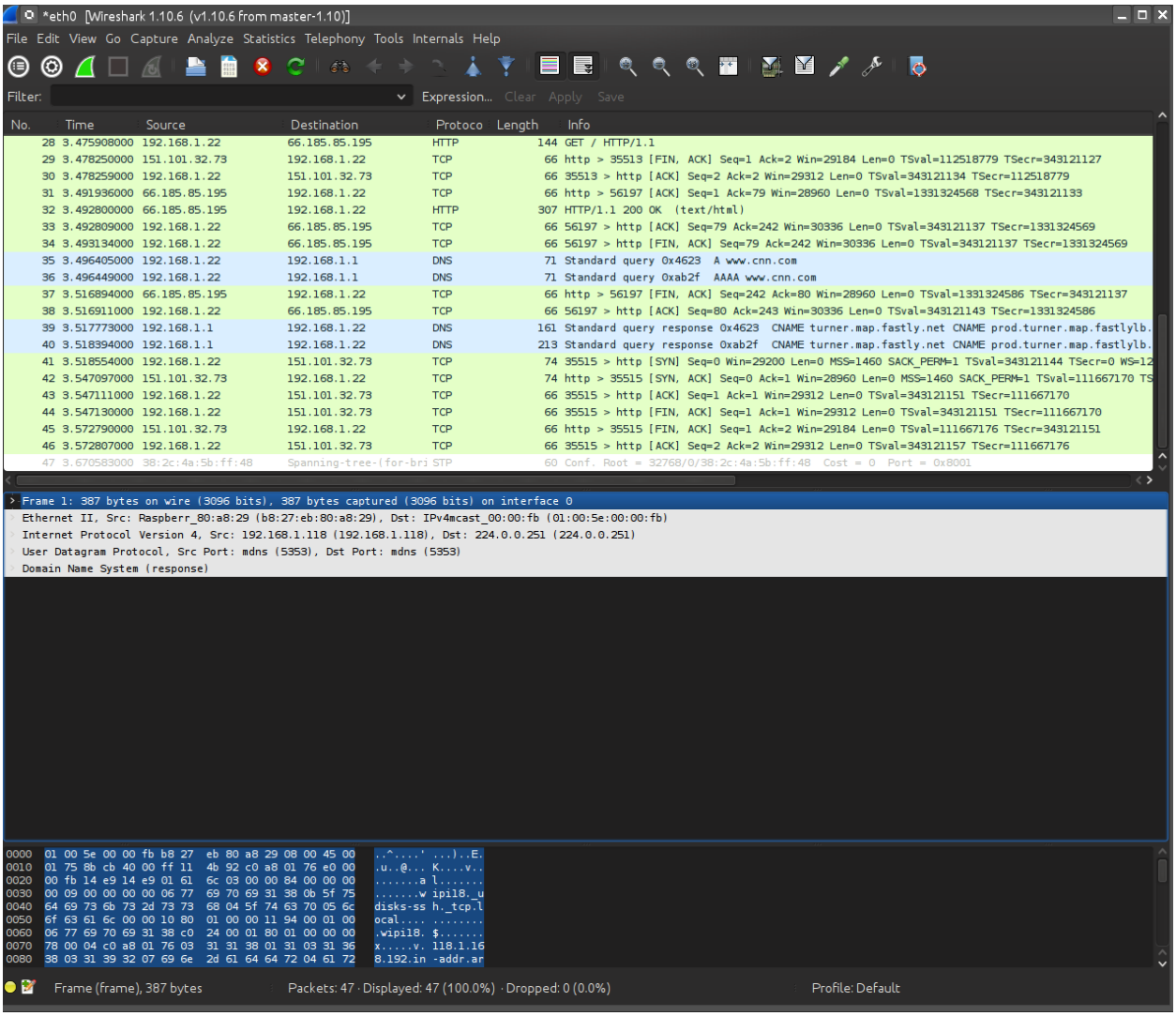

Figure 2: WireShark Capture Screen

to interact with, i.e., `<address>` in dotted quad format. In some of the experiments below, the host is `www.ece.mcmaster.ca`, whose IP address is `130.113.10.149`. Then type `ip.addr == 130.113.10.149` into the Filter box. This will limit the packets displayed to those with that source or destination address.

There are many WireShark display filter examples online. There is also a cheat sheet at

`https://wiki.wireshark.org/DisplayFilters`

Capture Filter: An alternative is to use a capture filter. To enter a capture filter, pull down the Capture > Options menu (or click on the circular capture options icon). Enter the filter in the text area. You can also save capture filters by using the Capture > Capture Filters menu. Just click on the "Create a new filter" button and enter a name and the filter. See the lecture notes for some examples of capture filters, e.g., to capture packet interactions with the host `www.ece.mcmaster.ca`, just enter `host www.ece.mcmaster.ca` as the capture filter.

Once you are familiar with tcpdump or windump, and WireShark, proceed to the experiments below.

2 Experiments

In all of the experiments below, describe the protocol interactions (including packet details) that have occurred. With these descriptions, include text output of the packet capture to illustrate your descriptions. These can be obtained from WireShark as follows.

After a packet capture has been obtained, use the

`File > Export Packet Dissections > as Plain Text file`

to save a text file containing your packet capture. If you find that this File option is "greyed out", use the `File > Save` function to save the capture in pcap format first. Then you should be able to export a text file. Remember to use filters to obtain a clean list of captured packets.

For experiments that use a web browser, your web browser may cache the contents locally after visiting a webpage the first time. If you repeat the same experiment, it will fetch the content from your local machine rather from the server. To prevent this, it is best to use your browser in *private browsing mode* (also called *privacy mode* or *incognito mode*). This should prevent local caching. An alternative is to go into your browser settings menu and clear its cache after each download. In Firefox, for example, it is found in
`Preferences > Advanced > Web Content`.
You have to click on the "Clear Now" button. In Chrome, click on
`Chrome > Clear browsing data`,
a pop-up window should appear; choose a time range and then click on the "Clear All" button.

Below, for all TCP-related captures, check if there are any captured packets that have the RST bit set in the TCP header. If there are, explain how this is related to the current TCP connection and future data transmissions between the source and destination hosts, and how this is related to the application layer protocol.

In addition, for all TCP-related captures, check if there are any out of order TCP segments and any duplicated ACKs, and then pay attention to the sequence numbers and acknowledgement numbers related.

TELNET: Telnet is an application protocol that provides a bidirectional interactive text-oriented communication facility using a virtual terminal connection. Because of serious security concerns when using Telnet over an open network such as the Internet, its use for this purpose has waned significantly in favor of SSH. However, Telnet is still an important tool for testing and troubleshooting network services and demonstrating fundamental client-server interactions.

1. Download the Telnet client and server programs provided with this instruction.
2. Design a Telnet client-server communication experiment so that WireShark can capture the TCP 3-way handshake and the connection teardown process, i.e., the `SYN` → `SYN+ACK` → `ACK` sequence and the `FIN` → `ACK` sequence.
3. Design a Telnet client-server communication experiment so that WireShark can capture the TCP segments with the `PSH` flag set.
4. Compare the captured traffic between two cases: when the client terminates the connection normally (by typing `exit` and the server forcibly terminates the connection using `Ctrl-C`.

HTTPS: Use WireShark to capture a packet trace while visiting a webpage

1. Start a packet capture.
2. Use a web browser and access the following URL:
   `http://www.ece.mcmaster.ca/~dzhao/COE4DN4/`
3. Stop packet capture.

How many different protocols are involved in these captured packets. Briefly describe the interactions of these protocols and their functions.

DNS: Use WireShark to capture a DNS query. This probably happened in the previous experiment, but you can do it as follows.

1. Use the `nslookup` utility to do the DNS query. It is available on all OS platforms, i.e., start the packet capture, then type

   `nslookup <server>`

   in a terminal or command prompt (i.e., PowerShell) window, where `<server>` is the desired machine. Choose one web server in Asia or Europe to do the experiment.

   Note that when you repeat the experiment, the local DNS software may cache recent queries; and if the current request is already cached, there may be no network activity. To clear the cache, on Windows you can type `ipconfig /flushdns`. On Linux, there may be caching, depending on the Linux distribution and version. The latest Ubuntu distributions, for example, use `dnsmasq` but disable caching by default. On MacOS Yosemite and later, you can clear the cache using
   `sudo killall -HUP mDNSResponder`.

   i) Locate the DNS query and response messages. ii) Are they sent over TCP or UDP? iii) What is the destination port for the DNS query message? What is the source port of DNS response message? iv) To what IP address is the DNS query message sent? Use ipconfig (or ifconfig) to determine the IP address of your local DNS server. Are these two IP addresses the same? v) Examine the DNS query message. First, select the DNS query message in the listing of captured packets so that the middle window displays details of the packet header, click the triangle before "Domain Name System

(query)", and check "Answers" under it. Do a google search and briefly explain what is a *standard query*. vi) Examine the DNS response message. First, choose the DNS response message in the listing of captured packets so that the middle section of the window displays details of the packet header, click the triangle before "Domain Name System (response)", and you should see "Answers" under it. Briefly explain details included in the "Answers" after doing a google search.

Traceroute: Use WireShark to capture a traceroute session. This can be easily done by starting WireShark then invoking traceroute in a (MacOS/Linux) shell window, i.e.,
`traceroute <hostname>`
or a (Windows 10) PowerShell window, i.e.,
`tracert <hostname>`.
Include data from part of the capture that illustrates how traceroute works. Discuss what has happened.

Nmap: Use Nmap to do some network scans as described below.

WARNING! Python is serving up the contents of the directory where the server was launched, i.e., it can be read by anyone who can connect to that machine. Make sure that you exit the HTTP server when you are done!

1. Find the **private** IP address of your home router (or a different device).
2. Do a standard TCP connection scan (i.e., -sT) of the above device over the port ranges: 1-100 and 990-1000. Describe what you find and what services seem to be running.
3. Choose a Closed and Open port from the scan above and do a tcpdump or windump capture while scanning the two ports. Describe the packet interactions that are occurring.

The following steps of the experiments can be done at home or in Room BSB 238.

If you will do the experiments at home:

The experiment requires both `Your Computer` and `Device A` to be connected to the same WiFi router. All the scanning work is done at `Your Computer`. If you only have one computer, it can serve as both `Your Computer` and `Device A`.

4. Scan `Device A` from `Your Computer` using a standard nmap TCP connection scan. Explain what you found and discuss the purpose of any services that you found (You will probably have to google search their names.)
5. Scan TCP port 8000 on devices A from `Your Computer` using a standard nmap TCP connection scan. Discuss what result you obtain.
6. On `Device A`, launch the HTTP server that comes as a module with Python 3, i.e., open a terminal shell, and type:
`python -m http.server`
This will create a web server listening on port 8000 that is serving up the contents of

the directory where python was invoked. On `Your Computer`, scan TCP port 8000 of `Device A`, and discuss what result you obtain.

Open a web browser on your own computer and access the web service from `Device A`, i.e., if the IP address of `Device A` is 192.168.1.150, then browse to:
`http://192.168.1.150:8000`
Describe what you see.

If you will do the experiment in BSB 238:

A wireless router is installed and can be accessed using Wi-Fi. The SSID is `COE4DN4` and the Wi-Fi password is `IPv4socket`. You and your lab partner(s) must be all connected to the access point.

Once you are all connected, you can use `ipconfig` (Windows) or `ifconfig` (MacOS/Linux) in a terminal window to find your IP addresses for scanning.

(Note that while you are connected, you will not have Internet access since the router has no backhaul connection.)

If you are working on this lab by yourself, you can scan directly to the IP address that you have been assigned.

In your report, note the IP addresses that you were assigned and the date/time that you did the experiments. Follow the following instructions.

---

4. Scan your lab partner's laptop using a standard nmap TCP connection scan. Explain what you found and discuss the purpose of any services that you found (You will probably have to google search their names.)
5. Scan TCP port 8000 on your partner's laptop using a standard nmap TCP connection scan. Discuss what result you obtain.
6. On your partner's laptop, launch the HTTP server that comes as a module with Python 3, i.e., open a terminal shell, and type:
   `python -m http.server`
   This will create a web server listening on port 8000 that is serving up the contents of the directory where python was invoked. Scan TCP port 8000, as before and discuss what result you obtain.

   Open a web browser and access this web service, i.e., if your partner's IP address is 192.168.1.150, for example, then browse to:
   `http://192.168.1.150:8000`
   Describe what you see.

3 Writeup

Submit a writeup for the lab. Each group (of 3 maximum) may submit a single writeup. Include in your writeup a brief description of everything that you did including an interpretation of the results

that you obtained.