CS3640 Written Assignment-2

Due: Mar 1, 2022 midnight
Submit as a single PDF on ICON

Lecture 4 introduced the idea of packet sniffing in networks. The goal of this assignment is to put that theory into practice. You will learn how to capture and analyze network packets with the help of Wireshark, an open-source packet sniffer and analyzer.

Q1. Understanding packet analyzers **25 points**

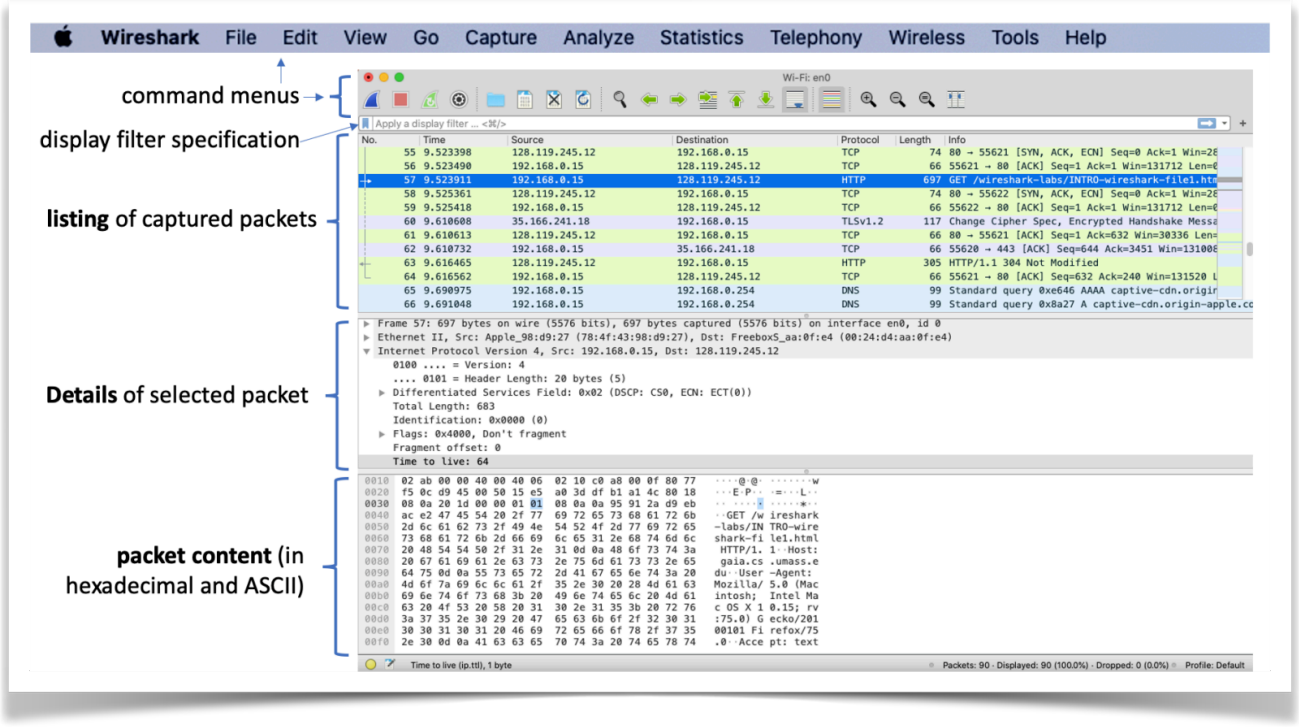
Wireshark maintains two useful references, a user guide (https://www.wireshark.org/docs/wsug_html/) and a developer guide (https://www.wireshark.org/docs/wsdg_html/). While these guides are extensive and encyclopedic, they help us understand the design and operation of packet analyzers. Using them as references, answer the following questions: (i) Identify the core functional blocks of Wireshark. Describe each of them in a sentence or two of your own. *Refer to Chapter-6 of the developer guide.* (ii) Is Wireshark an intrusion detection system? Could it be used to spoof packets? Explain why or why not. *Refer to Chapter-1 of the user guide.*

(i) Describe seven main functional blocks of Figure-6.1 ***using your own words***. (ii) No, Wireshark is not an IDS since it does not actively warn its users of any network attacks. Wireshark cannot be used to manipulate packets on the network either, it is simply a monitoring tool.

Q2. Setting up Wireshark **25 points**

Download and install Wireshark (https://www.wireshark.org/) on your computer/laptop. Explore the tool and familiarize yourself with its basic operations. Initiate a packet capture on your connected interface (WiFi or Ethernet), wait until a few packets are captured (they show up on the display window), and then stop the live capture. Now, provide a screenshot of Wireshark that shows captured packets, and point to three areas that display listing of packets, headers of the selected packet, and contents of the selected packet. Finally, identify at least three different protocols that appear in the protocol column.

Attached below is an example screenshot (courtesy: Jim Kurose). In there, we can see TCP, HTTP, DNS, and TLS. Other commonly encountered ones include ARP, UDP, ICMP, etc.



Q3. HTTP GET **25 points**

This question explores web client-server interaction by downloading a simple HTML file. First, start up Wireshark and set it to display only HTTP packets (one way to do this is by entering "http" in the display filter). Before you begin your packet capture, make sure that your web browser is not actively running any heavy web transactions as it will clutter the Wireshark display with unwanted http packets. Next, start the packet capture in Wireshark and enter the web address http://apache.org/apache-name/. Once your browser displays the contents of the page, stop the Wireshark capture.

By looking at the information in the HTTP GET request and response, answer the following questions: (i) what version of HTTP is your browser running? (ii) identify the source and destination IP addresses and port numbers (iii) how many bytes of http content was returned to your browser? (iv) when was the requested file last modified at the web server? (v) provide screenshot(s) that clearly show portions of Wireshark display where you got your answers from.

(i) Most likely value is ***HTTP/1.1*** (and you will find it at the end of the GET request line), but other values are acceptable as long as you show it in the screenshot (ii) IP addresses could be found in the IP header, and port numbers will be in the TCP header (iii) ***Content-Length*** header of HTTP will have this value in bytes (iv) ***Last-Modified*** header of HTTP will have this in human readable date-time format.

Q4. HTTP Conditional GET **25 points**

Recall from lecture 7 that web browsers employ object caching and perform conditional GET when retrieving an object that has already been cached. Before we begin the experiment, make sure your browser's cache is empty. For example, in Chrome, you will need to select Menu -> Clear browsing data. Once the cache is cleared, we initiate a Wireshark packet capture and set http in display filter. Enter the same web address that you used in the last question: http://apache.org/apache-name/ and verify that the page is displayed on the browser. Then, quickly hit refresh button in your browser to generate a second http request for the same file. Once the page refreshes, stop the Wireshark capture.

Answer the following questions after inspecting the captured packets: (i) Are there any differences between the first and the second HTTP requests? (ii) Did the web server respond with same or different response code for the two HTTP requests? (iii) Did the second HTTP request return any data? Explain your observations. Please remember to mark the portions of Wireshark display that justify your answers.

(i) the first HTTP would not contain the ***If-Modified-Since*** header but the second one will have it. (ii) the first time request gets a ***200 OK*** response and any repeated requests within the timeframe specified in the ***Cache-Control: max-age*** header, should result in ***304 Not Modified*** response. However, this answer could vary depending on the browser. For instance, my version of Chrome inserts ***max-age=0*** in its request, which forces the apache server to send 200 OK every time. (iii) if the second HTTP response was 304 Not Modified, then it will come with ***null payload***. Otherwise, the ***Content-length*** header values should match in both responses.