"Network Applications and Design" Homework Assignment #1

"Sniff HTTP traffic using Wireshark" (3 points)

Due date: Wednesday, April 5th (Fri) 2019, 9AM.
- Submit hardcopy during class and softcopy on the server.

In this homework assignment, you'll install "Wireshark" packet sniffer program, use it to sniff your own HTTP traffic, and answer some simple questions. The key objectives are to get experience in using Wireshark and learn how to sniff network traffic.

Here are the steps and requirements on what you'll need to do.

Please download and install 'Wireshark' program from https://www.wireshark.org/. They support Windows, MacOS, and Linux. Please install on your own personal computer; do NOT do this on the server.

- Wireshark is a packet sniffer which can read all packets going through your computer.
- When you install Wireshark, it will automatically ask you to install libpcap or WinPcap packet capture library. Just click on yes.

Before you run Wireshark, it's good to know which 'network interface' you are using (ethernet? WiFi?), and what your own IP address is.

- On Windows, open 'cmd', and do 'ipconfig'
- On Linux or MacOS, open a 'terminal', and do 'ifconfig'
- Jot down your interface and IP address

There are many buttons and features in Wireshark, but all you need for now are three functions

- start capture (select the correct interface), stop capture, and filter

Here is an example steps of what you will be doing in general;

- (better to close all other programs and activities on your computer)
- open a web browser
- open Wireshark
- start capture (on Wireshark)
- While Wireshark is running, enter the URL:
  - http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html
  - and have that page displayed in your browser.
- stop capture
- type "http" in the filter
  - you may also filter by ip.addr, ip.src, ip.dst, tcp.port, etc., and use conditions ==, !=, >= etc.
- see the messages… see in detail… we will do this in chapter 1, 2, and 3.
  - By clicking on '+' and '-' right-pointing and down-pointing arrowheads to the left side of the packet details window, you'll be able to see protocol information details.

Try doing the same for other URLs also

- http://nsl.cau.ac.kr
- http://cau.ac.kr/~jpaek

Please see the separate document "**1.Wireshark_Intro_v7.0.pdf**" for more details.

Please understand that, although we'll only be doing some very basic simple things in this homework assignment, Wireshark will be a very useful and powerful too when you study networking!

Now, let's begin our main lab.

[A]. The Basic HTTP GET/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short and contains no embedded objects. Do the following:

- Start up your web browser.
- Start up the Wireshark packet sniffer (but don't yet begin packet capture).
- Enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here).
- Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
- Enter the following to your browser;
    - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
    - Your browser should display the very simple, one-line HTML file.
- Stop Wireshark packet capture.

Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. Please focus on the HTTP message only for now, but you may also need to look at other packet information, and you're more than welcome to look at other protocol fields for your own interest (and study).

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages and indicate where in the message you've found the information that answers the following questions. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).

Answer the following questions:

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What is the IP address of your computer? Of the gaia.cs.umass.edu server?
3. What is the status code returned from the server to your browser?
4. When was the HTML file that you are retrieving last modified at the server?
5. How many bytes of content are being returned to your browser?

[B]. The HTTP CONDITIONAL GET/response interaction

Most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty. (To do this under Internet Explorer, select Tools->Internet Options->Delete File; these actions will remove cached files from your browser's cache.) Now do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
    - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
    - Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Answer the following questions:

6. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

7. Inspect the contents of the server response. Did the server explicitly return the contents of the file?
8. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
9. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

[C]. Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we download a long HTML file. Do the following:

- Start up your web browser, and make sure your browser's cache is cleared
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
    - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html
- Stop Wireshark packet capture, and filter "http"

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request.

Answer the following questions:

10. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the 'Bill of Rights'?
11. What is the status code and phrase in the response?
12. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

[D]. HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s). Do the following:

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
    - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
    - Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. Your browser will have to retrieve these logos from the indicated web sites, one from the gaia.cs.umass.edu web site, and another from caite.cs.umass.edu server.
- Stop Wireshark packet capture, and filter "http".

Answer the following questions:

13. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
14. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel?

What and how to submit:

- Answer all the questions (1~14) listed above in a .hwp or .docx document file.
  - Please make your answers short and concise, while exact and clear.
- You must submit both a hardcopy and softcopy of your report.
- Hardcopy report should be submitted during class.
- Here is the instruction on how to submit the softcopy files :

① Login to your server account at nsl2.cau.ac.kr.

② In your home directory, create a directory "submit_net/submit_<student ID>_hw1" (ex> "/student/20149999/submit_net/submit_20149999_hw1")

③ Put your report file in that directory.

Other requirements:

- You must include your name and student ID at the beginning of the report.

Grading criteria:

- You get 3 points
  - ✓ if all of your answers are correct, AND
  - ✓ if you meet all of above requirements, AND
  - ✓ if your hardcopy report is well written.
- Otherwise, partial deduction may apply.
- No delayed submissions are accepted.
- Copying other student's work will result in negative points.

Extra, for fun and learning!!!!

- Try '**ping**' to any famous server (e.g. www.google.com)
  - ✓ and see from Wireshark what messages are being sent and received, which protocol it is using, how many/often they are sent/received, and what values in the header changes.
- Try '**traceroute**' to any famous server (e.g. www.google.com)
  - ✓ and see from Wireshark what messages are being sent and received, which protocol it is using, how many/often they are sent/received, and what values in the header changes.
  - ✓ Note: on Windows, the command is 'tracert'.