COSC 4P14
Laboratory 2

Due date: October 4th, 2020 at 23:55 (11:55pm)
Delivery method: the student needs to delivery the assignment only through Sakai.
Delivery contents: a document with a description and codes (see Submission instructions).
Attention: check the Late Assignment Policy.

Laboratory Overview
This laboratory is intended to introduce to Network programming at the Application layer. Thus, following a hands-on practice, the lab is composed of a series of exercises where students will analyze and implement programs using the socket API.
The TA will provide students with codes and instructions about how to complete the exercises. Also, the TA will inform about the expected layout of answers handled back.

Laboratory Details
The laboratory consists of answering/completing all the following list of items:

1. In this question, we will deal with the resolution of names through the Domain Name System.

- **1.a.** What is a whois database?
- **1.b.** Use various whois databases on the Internet to obtain the names of two DNS servers. Indicate which whois databases you used.
- **1.c.** Use nslookup on your local host to send DNS queries to three DNS servers: your local DNS server and the two DNS servers you found in part (b). Try querying for Type A, NS, and MX reports. Summarize your findings.
- **1.d.** Use nslookup to find a Web server that has multiple IP addresses. Does the Web server of your institution (school or company) have multiple IP addresses?
- **1.e.** Use the ARIN whois database to determine the IP address range used by your university.
- **1.f.** Describe how an attacker can use whois databases and the nslookup tool to perform reconnaissance on an institution before launching an attack.
- **1.g.** Discuss why whois databases should be publicly available.

2. In this question, we use the useful dig tool available on Unix and Linux hosts to explore the hierarchy of DNS servers. Recall that a DNS server in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server. First read the man page for dig, and then answer the following questions.

- **2.a.** Starting with a root DNS server (from one of the root servers [a-m].root-servers.net), initiate a sequence of queries for the IP address for your department's Web server by using dig. Show the list of the names of DNS servers in the delegation chain in answering your query.
- **2.b.** Repeat part (a) for several popular Web sites, such as google.com, yahoo.com, or amazon.com.

3. In this question, we will observer/analyze application-layer packets using a sniffer - Wireshark.

3.1. HTTP.

3.1.a. TASK: **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 steps:

- Start up your web browser.
- Start up the Wireshark packet sniffer, as described in the Introductory lab (but don't yet begin packet capture). Enter "http" (just the letters, not the quotation marks) 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, and don't want to see the clutter of all captured packets).
- 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.

QUESTIONS:

(i) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

(ii) What languages (if any) does your browser indicate that it can accept to the server?

(iii) What is the IP address of your computer? Of the gaia.cs.umass.edu server?

(iv) What is the status code returned from the server to your browser?

(v) When was the HTML file that you are retrieving last modified at the server?

(vi) How many bytes of content are being returned to your browser?

(vii) By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

3.1.b. TASK: **Retrieving Long Documents.**

In the 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, 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-file3.html`. Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

QUESTIONS:

(i) How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

(ii) Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

(iii) What is the status code and phrase in the response?

(iv) How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

3.2. DNS.

3.2.a. TASK: **Tracing DNS (Web) with Wireshark.**

Let's first capture the DNS packets that are generated by ordinary Web-surfing activity:

- Empty the DNS cache in your host. (In Linux Fedora= "sudo systemd-resolve –flush-caches")
- Open your browser and empty your browser cache. (With Internet Explorer, go to Tools menu and select Internet Options; then in the General tab select Delete Files.)
- Open Wireshark and enter "ip.addr == your_IP_address" into the filter, where you obtain your_IP_address with ipconfig. This filter removes all packets that neither originate nor are destined to your host.
- Start packet capture in Wireshark.
- With your browser, visit the Web page: `http://www.ietf.org`
- Stop packet capture.

QUESTIONS:

(i) Locate the DNS query and response messages. Are they sent over UDP or TCP?

(ii) What is the destination port for the DNS query message? What is the source port of DNS response message?

(iii) To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?

(iv) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

(v) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

3.2.b. TASK: **Tracing DNS (nslookup) with Wireshark.**

Using *nslookup*, do the following steps:

- Start packet capture.
- Do an nslookup on `www.mit.edu`.
- Stop packet capture.

QUESTIONS (focus on the last DNS query and response messages):

(i) What is the destination port for the DNS query message? What is the source port of DNS response message?

(ii) To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

(iii) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

(iv) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

(v) Provide a screenshot.

4. In this programming question, you will write a client ping program in Java. Your client will send a simple ping message to a server, receive a corresponding pong message back from the server, and determine the delay between when the client sent the ping message and received the pong message. This delay is called the Round Trip Time (RTT). The functionality provided by the client and server is similar to the functionality provided by standard ping program available in modern operating systems. However, standard ping programs use the Internet Control Message Protocol (ICMP), which we will study later in this course. Here we will create a nonstandard (but simple!) UDP-based ping program.

   Your ping program is to send 10 ping messages to the target server over UDP. For each message, your client is to determine and print the RTT when the corresponding pong message is returned. Because UDP is an unreliable protocol, a packet sent by the client or server may be lost. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and print a message accordingly.

   In this question, your job is to write the client and server codes; both codes (client and server) will be very similar to each other. It is recommended that you first design the server code. You can then write your client code based on the server code, liberally cutting and pasting some lines from the server code.

Submission
Students will submit their answers through Sakai in the respective tab created for the laboratory. The deadline for the submission will be informed by the TA. Otherwise, the submission deadline is at the end of the respective laboratory session.
The submission for this assignment will consist of two parts:

1. A description file with answers/explanations to each exercise item. Include a note explaining your code: where it is, how to compile it, and how to run it.

2. The respective code addressing the answers (in a compressed/zip file).

Note that it is not the fault of the marker if they are unable to mark your assignment due to lack of clarity or detail in the descriptions. Also, make sure you submit a reasonable or common file format (pdf for the description file).

Marking Scheme
Marks will be awarded for completeness and demonstration of understanding of the material. It is important that you fully show your knowledge when providing solutions in a concise manner. Quality and conciseness of solutions are considered when awarding marks. Every code added to the originals should be well commented and explicitly indicated in the code files; lack of clarity may lead you to loose marks, so keep it simple and clear.

Late Assignment Policy
No late submission is allowed for laboratory submissions.

Plagiarism

Students are expected respect academic integrity and deliver evaluation materials that are only produced by themselves. Any copy of content, text or code, from other students, books, web, or any other source is not tolerated. If there is any indication that an activity contains any part copied from any source, a case will be open and brought to a plagiarism committee's attention. In case plagiarism is determined, the activity will be cancelled, and the author(s) will be subject to the university regulations.

For further information on this sensitive subject, please refer to the document below:

`https://brocku.ca/academic-integrity/`