Programming Assignment 2

Deadline Feb. 25 11:59PM, Late deadline Feb. 28 11:59pm.

This assignment has multiple questions. Please put all your answer in one word or pdf file with the name CampusID_PA2.doc or CampusID_PA2.pdf. All the question in Section 2, 3, 4 need screenshots to prove your answers. Section 2, 3, 4 is proceed in your installed virtual machine.

Section 1: *Virtual Machine*

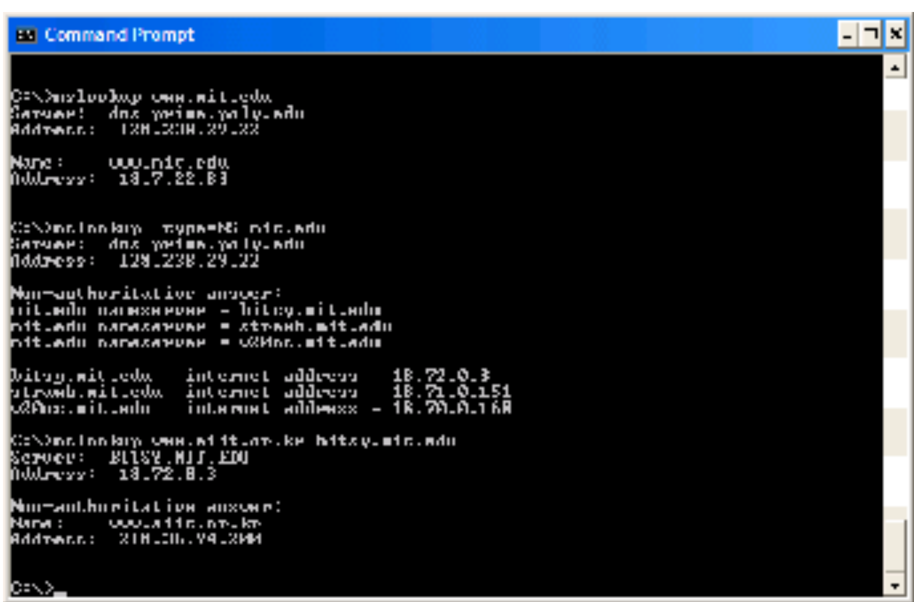
Install a virtual machine on your computer with Linux OS. You can use VMware, VirtualBox (free), and Parallels Desktop. The do the following tasks in the virtual machine and answer the following open questions:

a. Why a virtual machine is important for a cyber security engineer?
b. Why a virtual machine is important for a hacker?

Section 2: *nslookup*

In this assignment, we'll make extensive use of the *nslookup* tool, which is available in most Linux/Unix and Microsoft platforms today. To run *nslookup* in Linux/Unix, you just type the *nslookup* command on the command line. To run it in Windows, open the Command Prompt and run *nslookup* on the command line.

In its most basic operation, *nslookup* tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms). To accomplish this task, *nslookup* sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result.



The above screenshot shows the results of three independent *nslookup* commands (displayed in the Windows Command Prompt). In this example, the client host is located on the campus of Polytechnic University in Brooklyn, where the default local DNS server is dns-prime.poly.edu. When running *nslookup*, if no DNS server is specified, then *nslookup* sends the query to the default DNS server, which in this case is dns-prime.poly.edu. Consider the first command:

```
nslookup www.mit.edu
```

In words, this command is saying "please send me the IP address for the host www.mit.edu". As shown in the screenshot, the response from this command provides two pieces of information: (1) the name and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the host name and IP address of www.mit.edu. Although the response came from the local DNS server at Polytechnic University, it is quite possible that this local DNS server iteratively contacted several other DNS servers to get the answer.

Now consider the second command:

```
nslookup –type=NS mit.edu
```

In this example, we have provided the option "-type=NS" and the domain "mit.edu". This causes *nslookup* to send a query for a type-NS record to the default local DNS server. In words, the query is saying, "please send me the host names of the authoritative DNS for mit.edu". (When the –type option is not used, *nslookup* uses the default, which is to query for type A records.) The answer, displayed in the above screenshot, first indicates the DNS server that is providing the answer (which is the default local DNS server) along with three MIT name servers. Each of these servers is indeed an authoritative DNS server for the hosts on the MIT campus. However, *nslookup* also indicates that the answer is "non-authoritative," meaning that this answer came from the cache of some server rather than from an authoritative MIT DNS server. Finally, the answer also includes the IP addresses of the authoritative DNS servers at MIT. (Even though the type-NS query generated by *nslookup* did not explicitly ask for the IP addresses, the local DNS server returned these "for free" and *nslookup* displays the result.)

Now finally consider the third command:

```
nslookup www.aiit.or.kr bitsy.mit.edu
```

In this example, we indicate that we want to the query sent to the DNS server bitsy.mit.edu rather than to the default DNS server (dns-prime.poly.edu). Thus, the query and reply transaction takes place directly between our querying host and bitsy.mit.edu. In this example, the DNS server bitsy.mit.edu provides the IP address of the host www.aiit.or.kr, which is a web server at the Advanced Institute of Information Technology (in Korea).

Now that we have gone through a few illustrative examples, you are perhaps wondering about the general syntax of *nslookup* commands. The syntax is:

```
nslookup –option1 –option2 host-to-find dns-server
```

In general, *nslookup* can be run with zero, one, two or more options. And as we have seen in the above examples, the dns-server is optional as well; if it is not supplied, the query is sent to the default local DNS server.

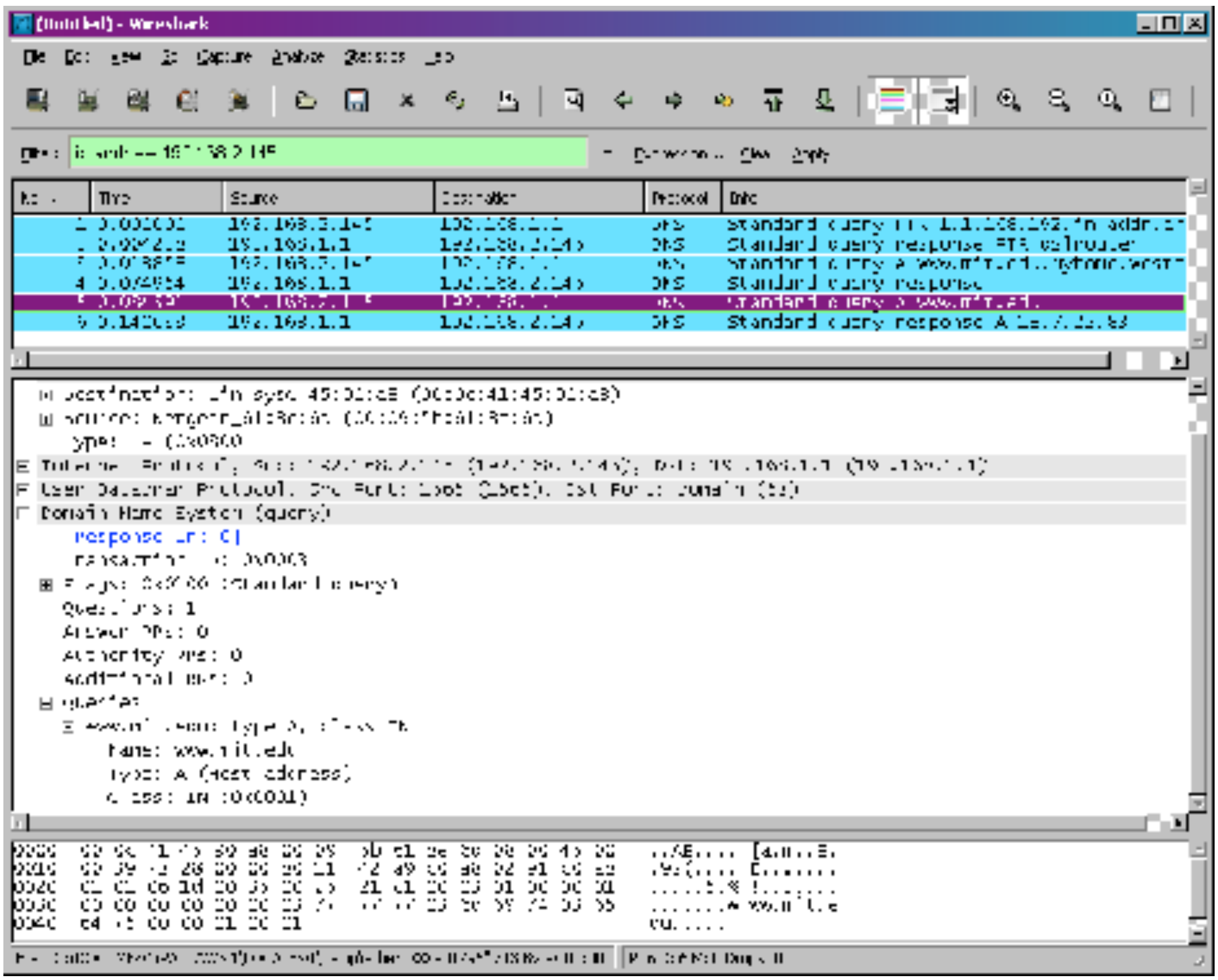
Now that we have provided an overview of *nslookup*, it is time for you to test drive it yourself. Do the following (and write down the results):

1. Run *nslookup* to obtain the IP address of a Web server in Asia (e.g. http://www.baidu.com). What is the IP address of that server?

Section 3: Tracing DNS with Wireshark

- Start packet capture by using Wireshark.
- Do an *nslookup* on www.mit.edu
- Stop packet capture.

You should get a trace that looks something like the following:



We see from the above screenshot that *nslookup* actually sent three DNS queries and received three DNS responses. For the purpose of this assignment, in answering the following questions, ignore the first two sets of queries/responses, as they are specific to *nslookup* and are not normally generated by standard Internet applications. You should instead focus on the last query and response messages.

4. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
5. Provide a screenshot.

Section 4: Capturing packets from an execution of *traceroute*

In order to generate a trace of IP datagrams for this lab, we'll use the *traceroute* program to send datagrams of different sizes towards some destination, X. Recall that *traceroute* operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by at least one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing *traceroute*) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing *traceroute* can learn the identities of the routers between itself and destination *X* by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

We'll want to run *traceroute* and have it send datagrams of various lengths.

- With the Unix/MacOS *traceroute* command, the size of the UDP datagram sent towards the destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the *traceroute* command line immediately after the name or address of the destination. For example, to send *traceroute* datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

```
%traceroute gaia.cs.umass.edu 2000
```

Do the following:

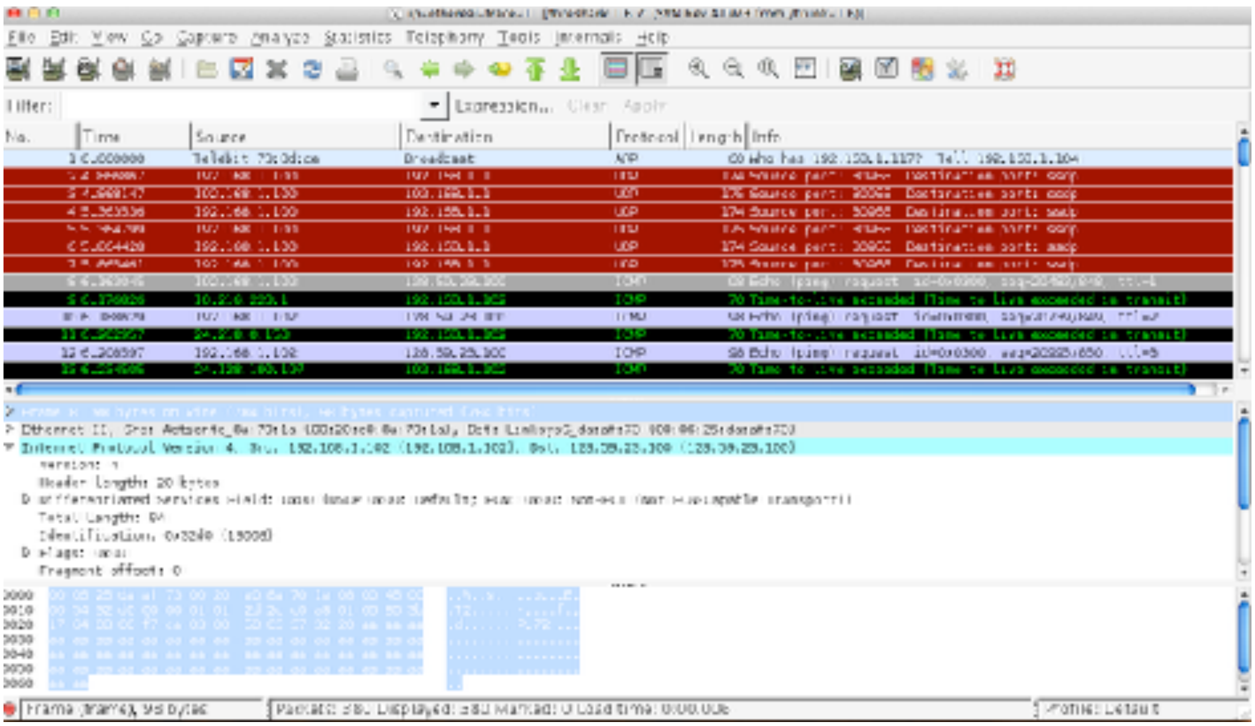
- Start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen (we'll not need to select any options here).

- Enter three *traceroute* commands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

Stop Wireshark tracing.

In your trace, you should be able to see the series of UDP segment sent by your computer Whenever possible, when answering a question below you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. 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. To print a packet, use *File->Print*, choose *Selected packet only*, choose *Packet summary line,* and select the minimum amount of packet detail that you need to answer the question.

6. Select the first UDP segment sent by your computer, and expand the Internet Protocol part of the packet in the packet details window.



What is the IP address of your computer?

7. Within the IP packet header, what is the value in the upper layer protocol field?

8. How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

9. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.