Assignment 4
CSC 4220/6220
Due Date: April 7, 2017 at 11:59 pm
Late deadline: April 14, 2017 at 11:59 pm

Scapy Lab

The purpose of this lab is to further your understanding of the concepts of packet structure, functional layering, and the behavior of real networks. In order to do this, you will craft your own packets, send them out to real destinations, and observe the results. You will do this using the Python programming language and the Scapy library. If you aren't familiar with the Python language, a good tutorial is located here: https://docs.python.org/2/tutorial/index.html. Note that Scapy uses Python 2.x whereas Python 3.x is the current version of the Python language.

To begin, install Scapy on your system. The exact requirements and process are documented at the Scapy website: http://www.secdev.org/projects/scapy/. Students running the Windows operating system might want to consult this website:

http://www.secdev.org/projects/scapy/doc/installation.html#windows.

You will need "root" or "administrative" access on the system you use to complete this lab since the process of creating packets is an inherently privileged process. To test that Scapy is successfully installed, open a Python interpreter and type the following: `from scapy import *`. If you do not see any errors, Scapy was successfully installed.

Open the scapy interpreter. Linux/Unix/Mac users should run the following from a terminal: sudo scapy. Windows users should open a command prompt and run the following: scapy. If you see a message similar to "Welcome to Scapy (2.3.1)" then you can proceed. Now, answer the following questions:

1. Create an IP packet and assign it to a variable name of your choice (for example, p or mypacket). Give it a source address of 0.0.0.0 and a destination address of 127.0.0.1. Now, show the details of the packet using p.show() (or whatever you named your packet).

   a. What is the default TTL? Can you modify the default value? What are the implications of setting a small TTL? What are the implications of setting a large TTL?
   b. What flags are shown? What are the possible values for this field?
   c. We gave the packet a source address of 0.0.0.0. Is this a valid IP address? What is the significance of this IP address?
   d. We gave the packet a destination address of 127.0.0.1. Where will the packet go if we send it out? What is the significance of this IP address?
   e. What value is displayed in the 'proto' field? Why is this? (hint: keep this in mind as we add layers to the packet)

2. Add a UDP layer to your packet and show the details of the packet using p.show() (or whatever you named your packet).

   a. What changed in the IP layer?
   b. What default values are in the UDP layer?

3. Change the destination address to 8.8.8.8. Now, show the details of the packet using p.show() (or whatever you named your packet).

   a. Did anything other than the destination address change?
   b. 8.8.8.8 is not local to your computer. Briefly describe the process required for your packet to reach this address.

4. Add a DNS layer to your packet and show the details of the packet using p.show() (or whatever you named your packet).

   a. The DNS layer is not at the same level as your UDP layer. At which layer of the Internet protocol stack is DNS found?
   b. Why is UDP used in our example?
   c. Could we replace UDP with TCP and still use DNS? (hint: consider how the DNS servers for a given domain [e.g. google.com] update each other.

Additional questions for Graduate Students (Answer earlier questions 1 through 4 also)

5. Recall that DNS serves two purposes to its clients: to translate a given name to an IP address and to translate a given IP address to a name. With this in mind, add the following parameters to the DNS message (here it's assumed your packet variable name is p; change this as needed): `p.rd=1, p.qd=DNSQR(qname="www.facebook.com")`. Now, show the details of the packet using p.show() (or whatever you named your packet).

   a. What is the significance of the rd field?
   b. Given the destination address in the IP layer of our packet, could the rd field be left with a value of zero? Why or why not?
   c. What is the value of the qtype field?
   d. What would the value be if we instead wanted to look up an IP address to determine the name associated with it? (hint: look up RFC 1035 or do a search for "list of DNS record types")
   e. What is the numeric value for the query type in question (c)?
   f. What is the numeric value for the query type in question (d)?

6. Before continuing, change the source address of your packet to the address of your wired or wireless interface (typically 10.x.x.x or 192.168.x.x). Now, send your DNS packet out using the sr1() function and save the result in a new variable (r is the variable name for the result in this example). Show the result using r.show()

   a. Why couldn't you just leave the source address as 0.0.0.0?
   b. Under the Question Record field, you should have one or more Resource Record fields. List the values of each Resource Record as follows: rrname, type, rdata
   c. Each Resource Record field has a TTL listed. Is this the same as the TTL in the packet you created? Why or why not?

7. Create a new IP packet using either the same variable name or a different one. Add a TCP layer to this packet. Now, show the details of the packet using p.show() (or whatever you named your packet).

   a. The default value for the fragment field in the IP layer is zero. Will this packet be fragmented? Why or why not?
   b. In the TCP header, the dataofs (data offset) field has a value of None. Why do you think this is?

8. Change the source address for the packet to your local wired or wireless adapter's address and change the destination address to 216.58.192.68 (google.com). Now, send the packet using the sr1() function, saving the result in a variable (r is the variable name for the result in this example). Show the result using r.show().

   a. The sequence number in the result will likely be a large number. Recall that both ends of the connection choose initial sequence numbers. Which side chose the sequence number displayed in the result?

   b. Why does the ACK in the result have a value of 1?

   c. How have the source and destination ports changed from the initial TCP packet you created in step 7?

   d. What data is in the options field of the TCP header? What does it mean?

   e. How many hops did this reply packet have to go through to reach you (this value will vary depending on your location – for the purposes of answering this question show the computation you used to get your answer)

   f. What are the TCP flags set to?