EE487: Applications of Cyber Engineering Name(s):
SX-11: Sockets, Clients, and Servers

Discussion: At this point we've explored the network "stack" from the physical layer up to the transport layer, and we've learned how to think of network communications in a layered model. Each layer takes care of its job and knows how to interface with the layers directly above and below. The transport layer is focused on end-to-end communications, meaning that it is the first layer on the stack that cares about the whole "flow" of data from one computer to the computer on the other end of the conversation. Remember that IP focuses on routing individual packets from one hop to another; it treats every packet as an individual and is not aware at all of the larger data flow comprised of hundreds or thousands of packets. TCP and UDP (most widely used transport layer protocols), however, are aware of the overall dataflow.

So now we need to focus on the application again, which after all is what the user actually cares about. Let's say our application is email, or web, or iTunes, or any number of thousands of applications that are out there. Almost all of them want some way to send data from one place to an end point. This is the service that TCP and UDP provide – think of them like a postal service that applications get to use without worrying about the details. But, even the postal service has rules: the letters or packages need to be of certain sizes and weights, they need stamps, and they need to be put into special boxes. Similarly, TCP and UDP have rules that applications must follow to use them. The application uses a special programming interface called a "socket" to accomplish this. Technically, sockets are at layer 5 of the OSI model, but often folks forget this and just consider them to be the interface between the application and the network transport. The good news for the application developer is that he can use one common programming interface to request network transport and then not worry about any of the other details.

We also need to understand what the terms "client" and "server" mean in the network context. Simply stated, a client initiates connections to a server, and a server responds with requested information. If you remember how TCP starts a connection, it always starts with a three-way handshake, initiated by the client. This is how firewalls work, incidentally. They allow conversations to be initiated in one direction, but not the other, and they can tell the difference because of the "syn" TCP flag.

Setup: Open your personal Ubuntu Virtual Machine (VM) using VSphere Client and be sure you are at a CLI with the "midshipman" user prompt. You will have sudo NOPASSWD privileges on your machine, so be careful!

Reading: This lab follows very closely with the Erickson text from page 203 to page 213 (sections 0x425 and 0x426). Sections 0x400 through 0x424 should be read prior to starting the lab.

EE487: Applications of Cyber Engineering Name(s):
SX-11: Sockets, Clients, and Servers

Section 1: TCP and UDP socket review

1. Which protocol does a "stream socket" use?

    a. Why is it called a stream socket?

    b. What are examples of applications that require a stream socket?

2. Which protocol does a "datagram" socket use?

    a. Why is it called a datagram socket?

    b. What are examples of applications that require a datagram socket?

3. What C library contains the functions necessary for an application to use sockets?

    a. Can an application communicate on the Internet without using a socket?

    b. Sockets behave a lot like _____ in terms of how they are accessed.

    c. **T / F** – Sockets can only be used with TCP and UDP.

    d. When an application requests a socket, it needs to pass some information so that the socket can be created. What information is necessary in the case of a TCP or UDP socket (not including the actual data to be sent)?

4. What does it mean for a computer to "listen" on a port?

   a. Does that make it a client or a server?

   b. What is it listening for?

   c. What port would the incoming messages come from?

   d. **`T / F`** – A given port on a computer can be bound to by multiple applications at the same time.

Section 2: A Simple Server (Hacking section 0x425)

5. Determine your IP address and give it to your lab partner. Similarly get his IP address. Write both below. Then compile and execute the program simple_server.c (reference page 207). **Note**: You may also need the "hacking.h" header files in your working directory.

   Me -
   Neighbor -

   a. In a separate terminal, type the command

      `ps –A | tail`

      to see if you can find your server running. Is it there?

   b. Now try the command "**`ss -ln`**" and note what you see.

6. Now, connect to your partner's server using the instructions on page 207 and note what you see. (Note: Type **`Ctrl+]`** when you are ready to disconnect)

    a. What port did your computer connect FROM?

    b. What port did your computer connect TO?

    c. Was it a TCP or UDP port?

    d. What IP address did your computer connect to? How is that different than a port?

    e. What if you had tried connecting to port 7891 instead? Try it, then describe what happens and why.

Section 3: A Web Client (Hacking section 0x426)

7. Back in the day, telnet was how we accessed networks to do useful things, like setting up a barbershop appointment or registering for the next semester's classes (but I show my age now). Now we typically use the "web." Both of these, however, are just applications. In section 2 we set up a simple "telnet" server. We can do the exact same thing for a web server. Web servers happen to speak a special language called "hyper-text transfer protocol" or http that you are familiar with. They also typically operate on port 80.

    a. Type "**`grep 80 /etc/services`**". What service runs on port 80? What can you infer about the additional output?

    b. Does a program HAVE to use these ports, or are they just the agreed to standard?

    c. How might a hacker take advantage of this?

d. In a web browser, navigate to a site of your choice. Now type the same address, but add a ":80" at the end of the domain name (right after the .com part). What happened?

e. Now try the same thing with ":81" on the end. What happened now? Why?

8. Now compile and run the program *host_lookup.c*. Use it to look up some of your favorite web servers. Do the same on Linux with the command "**`dig`**" and on Windows using "**`nslookup`**". Do this for at least three web sites and note what you see below.

9. Now compile the program *webserver_id*. Use it to profile at least those same three web sites and note what you find. Why might this be useful information for a hacker?

10. What webserver software and version are being run at www.ebay.com?

    Are there any known vulnerabilities/exploits for this version of the server? (**Disclaimer:** DO NOT HACK EBAY!)

EE487: Applications of Cyber Engineering Name(s):
SX-11: Sockets, Clients, and Servers

Conclusion and Results:

Your typed lab report will consist of two paragraphs, in the first paragraph:
• Briefly describe what you did in the lab in your own words.
• Discuss something new that you learned.

In the second paragraph, answer the questions:
• How could an adversary use this knowledge or these tools for malicious purposes?
• How could you use your new understanding to protect your systems and personnel from attack?

Staple the completed report to the back of your original lab and turn it in to your instructor at the beginning of the next class.