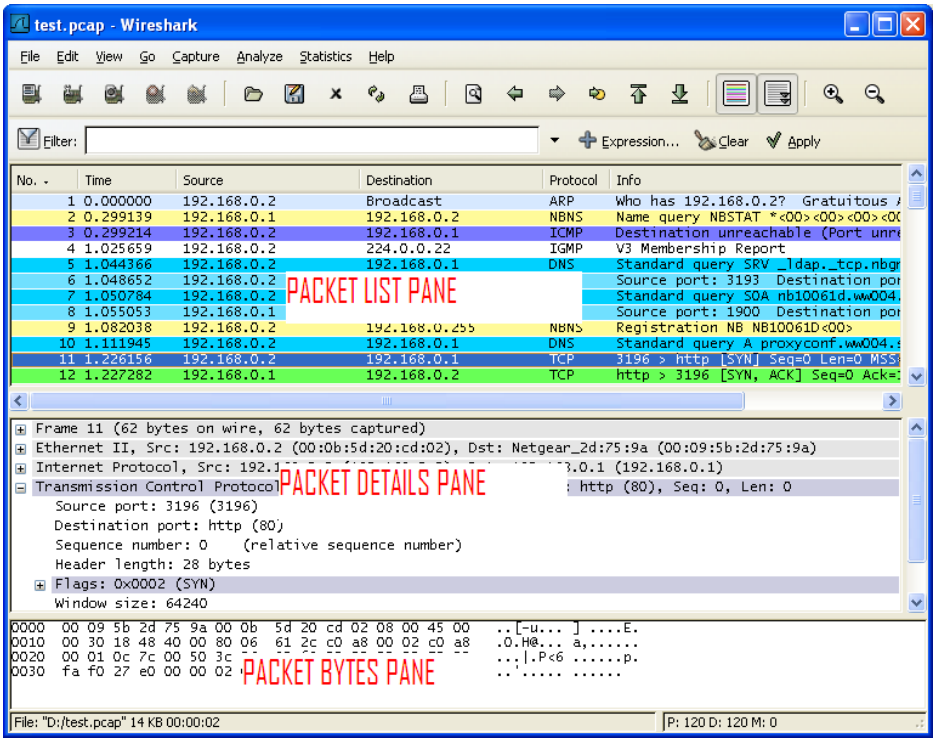
Due: January 12, 2009, beginning of tutorial.

A *packet sniffer* is a utility that can be used inspect and record all traffic passing through a machine on a network. For this part of the assignment, you will install a popular packet sniffer called **Wireshark** and use that to analyze your activity on the Internet.

a) Installation and Simple Packet Sniffing Walkthrough:

Wireshark can be downloaded from *http://www.wireshark.org* and versions are available for all major operating systems. The installation should be straight-forward, and all the default installation options will suffice for this assignment. If you are using Windows, Wireshark will also install WinPCap, which is a driver that is needed to actually capture the packets.

Once you have it running, you will be presented with an interface that looks like this:



As you can see, there are three panes in the main window. When you first start Wireshark, all of the panes should be initially empty because the packet sniffer is not running yet.

Once you have it running…

The *Packet List Pane* will list every packet that is either sent or received from your computer. Each packet has a timestamp based upon when it is sent or received. Each packet has a source (sender) and a destination (recipient). **From this information you can figure out whether the packet is incoming or outgoing. How? Also you will notice that some packets do not have a specific destination address and instead have "Broadcast" as their destination. What do you think this means?** For each packet, the Packet List Pane also displays a brief summary of the data in the packet and the kind of protocol the

data corresponds to. Wireshark is smart enough to automatically recognize many different standard Internet protocols, which makes it very useful.

If you select a packet in the *Packet List Pane*, details about it will appear in the Packet Details Pane and the *Packet Bytes Pane*. The Packet Bytes Pane displays the raw data contained in the packet in both hexadecimal (bytes written in base 16) and ASCII (standard encoding for plain text) formats. This is how the data looks exactly as it is sent or received over the network. The Packet Details Pane displays a parsed version of the packet data that is easier for a human to read and is organized hierarchically (you can use the + or – buttons to expand each field in the hierarchy).

But first we need to start the packet sniffer. Get ready by having your web browser already open, and close any applications besides your web browser that use the Internet (i.e. instant messaging, files sharing, games, etc.) because these applications often send and receive a lot of packets that we will not be concerned with.

From the Wireshark menu, go to **Capture -> Options**. This will give you a bunch of options that you can learn more about at *http://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureOptions.html*

But the only option you really have to make a selection for is the "Interface". This corresponds to the network adapter on your computer that you want Wireshark to capture data from. Depending upon your hardware, you may see a few of these adapters, corresponding to your wireless card, your wired Ethernet jack, etc. **You need to select the one that corresponds to how you connect to the Internet.** If you can't figure it out, you can always try each one individually.

Under "Capture Filter", you should type: not broadcast and not multicast

We're not particularly interested in those kinds of packets, but if you want to see what they look like, you can leave the "Capture Filter" blank.

Press the Start button. The packet sniffer is now running.

Now, switch to your web browser and load the Google homepage (http://*www.google.com*). Try searching for something.

After the search results have loaded, go back to Wireshark, and from the menu select **Capture -> Stop**. The packet sniffer is no longer running. Always remember to stop the sniffer when you are done using it, since it can slow down the performance of your computer.

In the Packet List, look at the HTTP packets. Recall that HTTP is the main protocol that web browsers and web servers use to communicate with each other. Also recall from tutorials that the web browsers use the *GET command* to make *requests* to the web server.

Find all the packets that correspond to requests made from your web browser to the web server, and write down what those requests are, and what you think they correspond to. Also, are there any non-HTTP packets you see listed and if so, what do you think is their purpose?

b) What does Google know about me?

Privacy is a very big issue on the web. By default, your web browser reveals a lot of information about you to the web server.

Carefully analyze each of the HTTP request packets that your browser sent to Google in part (a). **What information about you and your computer does your browser reveal to Google in these packets?** Try not to overlook anything.

Obviously, you have to tell Google what your IP address is or it will not know how to deliver a response back to you, but you may be surprised to learn how much other information your browser willingly reveals to Google.

Are there some websites that you wouldn't want to give this information to? Google prides itself on having a "Do No Evil" motto, but are there some malicious purposes that an Evil website could use this information for?

On the other hand, what do you think are the benefits of your browser sending this information to the server? In other words, why would the people who made the HTTP protocol include this information?

You may have noticed in your past web surfing habits that certain websites are able to identify the geographical location that you are located in. They are almost always able to identify what country you are from, and sometimes they can even correctly identify the city. **How is this possible?** HINT: For this question, you will not find the answer directly by analyzing the packets. If you don't already know, try to search on Google for the answer.

c) Security

The HTTP protocol is not a secure protocol, but the HTTPS protocol is. As you will see, it is not good to send passwords and other information you wouldn't want falling into the wrong hands using an insecure protocol…

When you're surfing the web using HTTP, the URL in the Address Bar will begin with http://

When you're surfing with HTTPS, the URL will begin with https://

First, find a website of your choice that you can login to with a username and password that has a login page that uses the insecure HTTP protocol, not HTTPS. HINT: While online commerce websites will almost all be secure in handling login information, you will find that sites like online forums, review sites, etc. still do not pass login credentials securely. If you can, try to find a website that you think should be using HTTPS but is not using it.

Run the packet sniffer while you attempt to login to this website. Always remember to stop the sniffer when you are done using it, since it can slow down the performance of your computer.

Find the packet that contains the password you used to login to this website. Print out the contents of this packet as evidence that you found it, but replace the actual embedded password with XXXX, because we don't need to know what password you typed.

That packet, containing your password, passed through several machines before it reached its final destination. You can use the *traceroute* utility demonstrated in tutorial to find out what machines your packet most likely passed through.

From a command line, type the following:

tracert *hostname* (if you are using Windows)

traceroute *hostname* (if you are using MacOS, Linux or any variant of Unix)

where *hostname* is the domain name or IP address of the web server where you sent the packet.

Print out the entire route given to you by traceroute. NOTE: If it takes more than 30 hops (number of nodes passed through), then you can abort traceroute and give me just the first 30 hops. Sometimes firewalls will block traceroute requests. If this happens you will see something like **** for some of the hops. This is okay, because you should still be able to identify some of the machines that the packet is passing through.

As you may have already deduced, if someone put a packet sniffer on any of the machines in this route, they would also be able to read the packet you sent and therefore know the password you typed.

For comparison, try logging into a website that does use HTTPS and running the packet sniffer while you do so. What differences do you notice in the kinds of packets that get sent and what they contain? Can you identify your password in any of these packets? Why should you only use secure protocols when sending confidential information over the Internet?

d) Fun (*Optional*)

Try sniffing packets from any other non-web-based Internet application of your choice (e.g. instant messaging, VOIP, Skype, Bittorent, computer games, etc.) **By analyzing the packets, can you learn anything about how these programs work that you didn't already know?**