Wireshark Lab: Introduction

Objectives

- Use Wireshark to capture network traffic
- Observe the encapsulation of protocol layers
- Observe background activity on the network

In this lab, you will learn to use the Wireshark packet sniffer. Wireshark (formerly called Ethereal) is a widely-used tool for monitoring networks and tracking down network problems. It observes all network traffic that reaches your computer and lets you analyze it. It understands most common network protocols, so it can pick apart the details of a network exchange rather than just presenting you with raw data.

Capturing a trace

If using Windows, click on capture packets from the local area network.

To start Wireshark from the lab machines running Ubuntu Linux, select Applications->Internet->Wireshark (capture trace). A text window will open indicating that packets are being captured, and telling you how many have been captured so far. To stop capturing packets and move on analyzing them, type <ctrl>-c, i.e., hold down the “Ctrl” key and press “c”. After you type <ctrl>-c, the main Wireshark application will load up and display everything you captured. If you want to save what you have captured to a USB flash drive or somewhere else, you can do so from within Wireshark. If you are using Wireshark on your own PC, you will need to learn to capture packets on your own.

What to trace

Close all browser windows if you have any running. Open Firefox with only a single tab open. Start capturing packets using the procedure described above. Note that you should only start capturing; do not hit <ctrl>-c yet. From your Firefox window, load the page http://cit.cs.dixie.edu/it/it2400/ while letting Wireshark capture packets in the background. After the page has loaded, close the browser. Let the capture window continue to capture packets for 30 seconds or so, then type <ctrl>-c in that window to stop capturing and start analyzing.

Analyzing the trace

The main Wireshark window is divided into three sections. At the top, you will see a colorful list of packets that were captured. Each different color indicates a different protocol. In the middle you will find the protocol-level details about the currently selected packet. In the bottom pane, you will see the raw data from the current packet. Start by looking at the web request you generated using Firefox. Near the top of the window, there will be a box labeled “Filter”. Type “http” into this box and press [enter]. This will cause Wireshark to only display packets that are part of an HTTP (web) request or response. You should see a much smaller number of packets. From the list of HTTP packets, find the one that initiated the request for the page indicated above. In the list of packets, look for one where the “Info” column says “GET /it/it2400/HTTP/1.1” and select it. The middle pane will be updated to display details of the network packet that carried the request from your browser to the web site. It should display five lines, labeled:

<table>
<thead>
<tr>
<th>Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet II</td>
</tr>
<tr>
<td>Internet Protocol</td>
</tr>
<tr>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>Hypertext Transfer Protocol</td>
</tr>
</tbody>
</table>

Meanwhile, the bottom pane shows the raw bytes that made up the packet. Click on each of the five protocol lines in the middle pane, and see what is highlighted in the bottom pane for each one. You should observe that the entire packet is highlighted when you click on “Frame”, while only parts of the packet are highlighted when you click on the other four lines. Furthermore, if you click on the remaining four lines from top to bottom, you will see that they divide the packet into four non-overlapping sections. The status bar at the very bottom will also tell you how many bytes are currently highlighted, and what protocol they represent.
“Frame” shows the entire packet, but the other four lines each represent a single protocol, and each one comes from a different protocol layer. The application layer protocol is HTTP, and if you click on it you will notice that it comprises the end of the packet. Clicking on TCP (Transmission Control Protocol) highlights the bytes immediately before the HTTP section, and so forth. This is the process of encapsulation. As each layer receives data from the layer above, it adds its own headers to the beginning and leaves the data it is given alone. For example, IP (Internet Protocol) adds a header of 20 bytes to the beginning of the data it is given from the transport layer. Confirm this by clicking on the different protocol layers and observing their relationship.

You can also look into a specific protocol and see the pieces that make it up. Try expanding the IP section and find the “Time to live:” header. What is its value? How many bytes does it take up (click on it and look in the bottom pane to answer this last part).

**What to pass off**

Keep your trace window open to pass off the lab. You should perform a screen capture(s) that will appropriately identify the following: (You may want to be creative and circle or otherwise identify the information within your screen capture)

- The size of the entire packet, and the size of the four layers.
- The total number of bytes used up in headers that are not part of the application layer. This is the amount of overhead that goes in each packet in addition to actual HTTP data.
- Where “GET /it/it2400/” appears in the raw packet data (the bottom of the three panes).
- The “Time to live” field in the IP header, and the amount of space it takes in the header.
- At least 5 other protocols that you observed during your trace. Click on the “Clear” Button next to the filter to display everything again, instead of just the HTTP packets.

**What to submit**

Screenshot(s) that shows the answers to the above. Upload to the correct spot on Canvas for this lab.