Lab 1: Ethereal and Network Programming

Burke Libbey

January 24, 2008

1: Host Identification

Table 1: Host Identification Table

IP Address	130.179.130.190
IP Name	dh130190.eng.umanitoba.ca
Domain Name	eng.umanitoba.ca
Physical Address	00:0a:95:69:db:3c

2: Sample Capture Analysis

Table 2: Protocol Statistics

Protocol	Percentage	Packets
UDP	2.14	100
TCP	97.64	4553
HTTP	2.59	121
DHCP	0	0
DNS	1.03	48

Table 3: HTTP Traffic Statistics

IP Address	Sent	Received
130.179.133.35	74	70
205.200.78.70	34	34
130.179.128.15	10	11
64.233.167.104	10	10

3: Generated Capture Analysis

Table 4: Protocol Statistics

Protocol	Percentage	Packets
UDP	1.43	640
TCP	93.90	42029
FTP	0.27	120
HTTP	0.15	69
DHCP	0	0
DNS	0.12	54

The addresses involved in the largest transfer were 156.56.247.193 and 130.179.130.190 (localhost). These hosts exchanged 44761 packets.

4: Client/Server Programming

'localhost' is almost always assigned as a reference to 127.0.0.1, the local address of a given machine.

Network traffic is, of course, generated when the program is modified to run on two seperate hosts, as verified with WireShark.

Port numbers are almost as important in networking applications as IP addresses. Without a port number, an application has no idea how to contact a specific server process on the target machine. If an application were to guess all possible ports when trying to contact a server, it would not only take a very long time, but it could potentially wreak havoc with other server processes running on other ports.

A server process (in Java) instantiates the ServerSocket class, and listens on a port (usually) for incoming connections, which it will then act on. A client process instantiates the Socket class, then connects to a remote host:port. Clients do not open and monitor ports for connections.

5: Program Modifications

Find code and capture file at:
`http://burkelibbey.org/classes/winter2008/ece3700/lab1`

There's really not much to discuss here. Some packets are incoming text data; some are outgoing.

6: Intro to Telnet (omitted)

7: More Telnet

Differentiating this from a 'normal telnet session' really depends on what one defines a normal telnet session as. Assuming a login shell is implied, then the difference is that this doesn't produce a login shell or mask the password when you enter it.

The `ClientServer.cap` and `Telnet.cap` data are nearly identical, since telnet is a very simple protocol. It's fairly obvious that telnet has almost no overhead.

Telnet can communicate with any service, provided the service supports character input. Binary protocols such as nfs would either be exceptionally awkward or completely non-functional with only character input.

Using telnet to connect to other services can be a handy tool for stepping through each stage of connection negotiation. For example, if an SMTP server is rejecting connections for no apparent reason, you could connect to the host on port 25 with telnet and step through the whole process from there to figure out when the rejection happens, which would probably provide some sort of clue as to what step to take next.