CIS 551 - Computer and Network Security Assignment #2 - Network Analysis

In this group project, your group will write a program `packetparse` that parses and interprets captured Ethernet traffic containing IP datagrams (UDP and TCP), and stores captured email messages sent in that traffic into files.

We will supply a file, produced by `tcpdump`, that contains a packet trace for your system to use as input. All packet input should be from this file; you will not need to actually capture traffic from the computer's Ethernet interface.

Your code (we *strongly* recommend using "C") should be readable and well documented, and it must compile and run on the SEAS Eniac Linux system.

Be sure to include: (1) all your source code, (2) a `Makefile` that compiles your system, and (3) a `README` file included with your code that documents how to compile and use your system (including any bugs, limitations problems of which you were aware. In other words, if you only do part 1, tell us that!).

- Part 1 (33% credit, due before class 2/26/14): *Parsing packets:* In the simplest mode of your program, invoked as `packetparse` with no argument, parse each packet in the standard input and display the following basic header information on standard output:
  - Packet type (TCP, UDP, other)
  - Source and destination MAC address
  - Source and destination IP address (if IP packet)
  - Source and destination ports (if TCP or UDP)
  - Checksum (if TCP) and whether the checksum is valid
  - Payload size

  When the input reaches EOF, print the total number of packets processed and the numbers of TCP, UDP and non-TCP/UDP packets.

- Part 2 (33% credit, due before class 3/05/14): *TCP flows*: In this mode of operation, invoked as `packetparse -t`, your program should record each TCP connection into files that represent the data sent and received on each side of the connection. Ignore packets that aren't TCP in this mode.

  Keep track of the number of connections, such that the first connection you see is connection 1, the second is connection 2, etc. (Note that connections may be going on in parallel, where a new connect starts before the last one ends.) For each connection, create three files in the current directory:

  1. Metadata: basic information about the connection, including:
     - Initiator and responder IP address
     - Initiator and responder port number
     - Number of packets sent, in each direction

- Number of bytes sent, in each direction
- Number of duplicate packets detected, in each direction
- Whether the connection was closed before EOF was reached

For connection 1, this file should be called "1.meta". For connection 2, "2.meta", etc.

2. Data from initiator. This file, called, e.g., "1.initiator" for connection 1, should contain all the TCP payload data in the connection sent from the initiator to the responder, but only if the responder has acknowledged it and it is not a duplicate. Data in each subsequent packet in the connection should be concatenated to the end of the file as it is acknowledged.
3. Data from responder. This file, called, e.g., "1.responder", should contain the corresponding data sent from responder to initiator.

- Part 3 (34% credit, due before class 3/19/14): *Email traffic:* In this mode of operation, invoked as `packetparse -m`, your program should record the SMTP email traffic in the packet trace. Email traffic is sent via TCP to servers that accept connections on port 25, using an application-layer protocol called "SMTP" ("Simple Mail Transport Protocol").

  For each email message sent to an SMTP server, create a file (e.g., "1.mail" for the first message, "2.mail" for the second, etc), that contains:

  - The IP addresses of the sender and receiver
  - Whether the message was accepted or rejected by the server.
  - The message headers and body (if any).

  To do this part of the assignment, you will need to learn about the SMTP protocol, which we did not cover in class in any detail. See Internet standards RFC-821 and RFC-822 for details.