Unix for Telecommunications

Portfolio Task – P-Lab-06-PCAP-Programming
Pass Level Task

I. Introduction

In this lab you will learn how to use the **libpcap** (or similar) library with either the C or Python programming languages to parse packets. You will develop a tool to parse pre-captured PCAP files.

II. Purpose

To gain and/or enhance the following practical skills:

- Build a customised packet parsing tool/program instead of relying on existing tools (e.g tcpdump, tshark)
- Develop a better understanding of packet capture/sniffing tools
- Able to understand software documentation/examples and apply them
- Develop a deeper understanding of network traffic behaviour

III. Preparation

You can prepare for this lab by reading some of the documentation and examples available at:

- http://www.tcpdump.org/manpages/pcap.3pcap.html
- http://www.tcpdump.org/pcap.html
- https://eecs.wsu.edu/~sshaikot/docs/lbpcap/libpcap-tutorial.pdf
- http://recursos.aldabaknocking.com/libpcapHakin9LuisMartinGarcia.pdf

And for Python

- https://dpkt.readthedocs.io/en/latest/
- https://dpkt.readthedocs.io/en/latest/api/index.html

You should also review the basic concepts of:

- TCP/IP packets
- Packet sizes / lengths
- Existing packet analysers, such as tcpdump and tshark

IV. Requirements

You are to develop a software tool to parse a PCAP file and display a summary of all the packets within the file, as well as a brief summary of the overall contents of the file. The output format is standardised to allow for automated assessment of your work.

A. Program Requirements

Your program must be placed in the following directory:

```
/home/student/pcap_lab
```

Your program must be able to be executed directly from the command line and take one command line parameter. The command line parameter is the name of the PCAP file to parse. Execution should be as per:

```
./mtcpdump filename.pcap
```

Your program should open the PCAP file specified, process the contents of the file, and then print the required information.

B. Extracting and Displaying Individual Packet Information

For each packet in the PCAP file, you are required to extract and print the following information:

- Packet timestamp
- Source IP address and port number
- Destination IP address and port number
- Protocol (TCP / UDP)
- Packet length
- Packet Time-to-live (TTL)

All packet information is to be displayed to the screen (**stdout**) in the following format:

```
[2018-07-25 01:27:13.400464] - 172.16.10.62:5001 -> 172.16.11.72:32903 (tcp, len=52, ttl=63)
[2018-07-25 01:27:13.402465] - 172.16.10.62:5001 -> 172.16.11.72:32903 (tcp, len=52, ttl=63)
[2018-07-25 01:27:13.408465] - 172.16.10.62:5001 -> 172.16.11.72:32903 (tcp, len=52, ttl=63)
[2018-07-25 01:27:13.417466] - 172.16.10.62:5001 -> 172.16.11.72:32903 (tcp, len=52, ttl=63)
[2018-07-25 01:27:13.417486] - 172.16.11.72:32903 -> 172.16.10.62:5001 (tcp, len=52, ttl=64)
```

To enable automated assessment of your work, it is imperative that your output matches the format exactly.

In order to complete this task you will need to consider:

- When processing the PCAP files what tasks do you need to perform to extract individual fields for display?
- How might you modify mtcpdump such that it can perform a live capture rather than processing a PCAP file?

C. Displaying Summary Information from the PCAP File

Once all packets in the PCAP file have been parsed, you are required to output a brief summary to the screen (**stdout**) of the PCAP file including a count of the number of packets, the total number of bytes captured, and the packet size. The format of this output must be as per the following example:

```
------------------- SUMMARY -------------------

Total packets: 158141
Total bytes (Kbytes): 151375
Average packet size (bytes): 980

-----------------------------------------------
```

To enable automated assessment of your work, it is imperative that your output matches the format exactly.

In order to complete this task you will need to consider what mechanisms will you employ to allow you to extract this data from the PCAP file?

D. PCAP files

Two PCAP files are provided with the task resources:

- **`test1.pcap`** - A simple PCAP file that containing seventeen packets from two distinct flows. This file is provided to make it easy for you to test and debug your program. You can easily open this file in Wireshark and manually review the correctness of your program
- **`test2.pcap`** - A more complex PCAP file that contains multiple flows. Once you are confident that your program works properly, you can test it with this file.

When you execute the marking script to verify your program, it will execute your program against both these files as well as a third – hidden – PCAP file to confirm that your program works as intended.

V. Choosing your Programming Language

You are free to choose whichever programming language you like to develop your software. Before making your selection, it is strongly recommended you consider the task and read the associated links for information.

A. C or C++

The gcc compiler has already been installed on your rule host along with the libpcap library. This means that you will not have to install any further software to complete the lab using C or C++

Within the task resources available on Doubtfire, you will also find a sample C program with two functions (**`print_packet_info()`** and **`print_summary()`**). These functions will print parameterised variables to exactly match the format required to pass the lab assessment. It is strongly recommended you use these functions to help ensure your output meets the requirements for assessment.

B. Python

Python has not been installed on your rule host. If you would like to complete your lab using Python, you will need to research how to install Python and any extra libraries required on your rule host before you begin.

Within the task resources available on Doubtfire, you will also find a sample Python program with two functions (**`print_packet_info()`** and **`print_summary()`**). These functions will print parameterised variables to exactly match the format required to pass the lab assessment. It is strongly recommended you use these functions to help ensure your output meets the requirements for assessment.

Note: *Developing your software in Python will be easier than using C, however the task of installing Python effectively makes the complexity of either approach equal*

Note: *If you are intending to also complete the Distinction Level task, it is recommended to use Python to significantly make your overall task easier*

C. Other Languages

No other programming language has been installed on your rule host. If you would like to use an alternate programming language, you will need to research how to install it and any required libraries before you begin. It is also your responsibility to develop code to correctly output information in the required format. Only C and Python example code will be provided.

D. Report

If you are preparing a lab report for this lab, please answer the question as to how and why you chose your ultimate programming language to complete the task.

VI. Assessment

The due date for completion of practical work is **11:00pm**, exactly **six** days after your scheduled class.

Note: *The nominated submission day/time holds regardless of whether that day is a non-teaching day or public holiday*

A. Self Assessment

You can self-assess your progress at any time via the marking script available at
http://ruleprimary1.caia.swin.edu.au

B. Completion of task in Doubtfire

Download the PDF output of the marking script from http://ruleprimary1.caia.swin.edu.au and submit it to Doubtfire. Your tutor will confirm completion of the lab by examing the rule marking log files on the rule server.

If you complete the task during class beforehand, you may demonstrate completion in class to your tutor.

Note: *The downloaded PDF is not evidence of successful completion of the lab, it is a document to demonstrate completion within your portfolio. Your tutor will assess the evidence via either direct confirmation via the marking script or via the log files generated when you run the marking script*

C. Tutor Discussion

In order for the submission to be marked as complete, you must discuss your work with the tutor