Networking Lab Assignment 17

Packet capturing and filtering application.

Albin Antony

3 April 2019

1 Packet capturing and filtering application

1.1 Aim

To develop a packet capturing and filtering application using raw sockets.

1.2 Theory

1.2.1 Network packets and packet sniffers

When an application sends data into the network, it is processed by various network layers. Before sending data, it is wrapped in various headers of the network layer. The wrapped form of data, which contains all the information like the source and destination address, is called a network packet

1.3 Algorithm

Algorithm 1 Algorithm for creating N threads

```
START
CREATE SOCKET
RECEIVE all packets
UNPACK the packets
FORMAT the packets
PRINT the filtered data
STOP
```

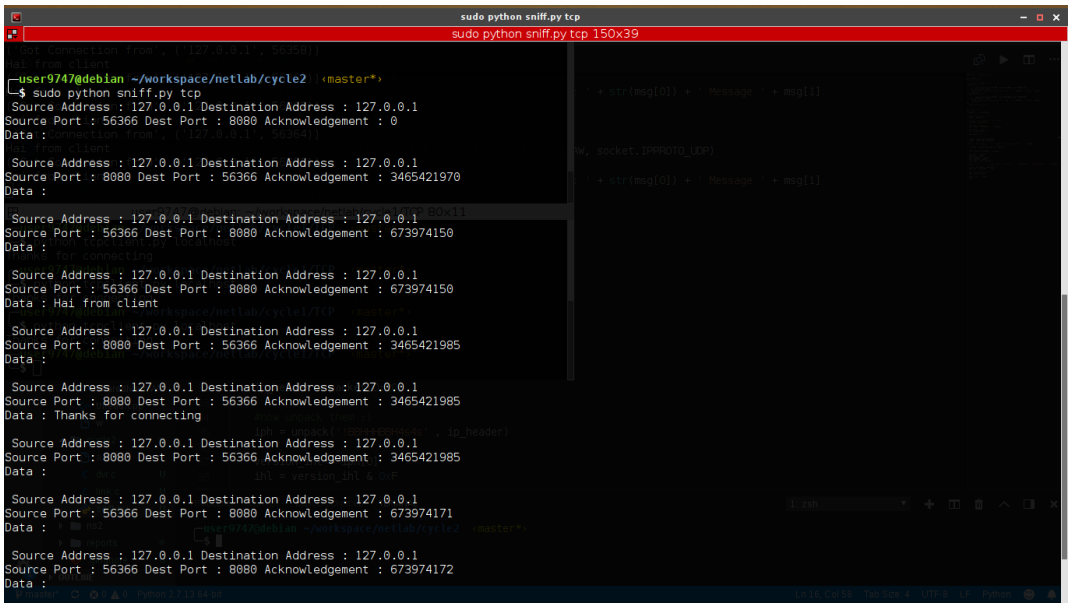
1.4 Program

```
import socket, sys
from struct import *

if(sys.argv[1]=="tcp"):
        try:
                s = socket.socket(socket.AF_INET, socket.
                    SOCK_RAW, socket.IPPROTO_TCP)
        except socket.error , msg:
                print 'Socket could not be created. Error
                    Code : ' + str(msg[0]) + ' Message ' +
                    msg[1]
                sys.exit()
elif(sys.argv[1]=="udp"):
        try:
                s = socket.socket(socket.AF_INET, socket.
                    SOCK_RAW, socket.IPPROTO_UDP)
        except socket.error , msg:
                print 'Socket could not be created. Error
                    Code : ' + str(msg[0]) + ' Message ' +
                    msg[1]
                sys.exit()
else:
        print "Specify protocol"


# receive a packet
while True:
        packet = s.recvfrom(65565)

        packet = packet[0]

        ip_header = packet[0:20]

        iph = unpack('!BBHHHBBH4s4s' , ip_header)

        version_ihl = iph[0]
        ihl = version_ihl & 0xF

        iph_length = ihl * 4


        s_addr = socket.inet_ntoa(iph[8]);
```

```
        d_addr = socket.inet_ntoa(iph[9]);

        print  ' Source Address : ' + str(s_addr) + '
            Destination Address : ' + str(d_addr)

        tcp_header = packet[iph_length:iph_length+20]

        tcph = unpack('!HHLLBBHHH' , tcp_header)

        source_port = tcph[0]
        dest_port = tcph[1]
        acknowledgement = tcph[3]
        doff_reserved = tcph[4]
        tcph_length = doff_reserved >> 4

        print 'Source Port : ' + str(source_port) + ' Dest
            Port : ' + str(dest_port) + ' Acknowledgement : '
             + str(acknowledgement)

        h_size = iph_length + tcph_length * 4
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data
        print
```

1.5 Output

All TCP packets were captured and their headers were displayed to the terminal.



1.6 Result

Implemented a program to capture and filter packets in python 2.7 and executed on Debian 9.4 Kernel 4.9 and outputs were verified.