CS3130
Assignment #2

In class I introduced Wireshark, a graphical network protocol analyzer; However, there is a lesser-known command-line counterpart called tcpdump. Tcpdump does what Wireshark does, but it dumps its packet captures as ASCII text to the screen or to a file.

For this assignment, I want you to write a python program that will analyze tcpdump files. Your program will only accept command arguments; the user will not be asked to provide any input. The arguments that your program will accept are:

1. --IP
   this option will display the sender and receiver for all IP packet captured
2. --UDP
   this option will display the sender /port and receiver/port for all UDP packets
3. --ARP <REPLY | REQ>
   This will take an additional option: REPLY or REQ. if the user provides REPLY, ARP reply packets will be shown if REQ is provided, ARP request packets are displayed. Anything else is an error.
4. --SRC <port#>
   an additional required option, port#, will be needed. This option will display all packets where the source port is port#.
5. --DEST <port#>
   this command is similar to #4 expect that it displays all packets with destination port equal to port#.

Your program will also need the name of the file that contains the captured packets.

Some examples of how to use your program :

```
#./python3 parseDump.py  --IP packet.pcap          # packet.pcap is the name of file
#./python3 parseDump.py –UDP packet.pcap
#./python3 parseDump.py --SRC 23 packet.pcap
#./python3 parseDump --ARP    packet.pcap                  # error missing parameter
```

To show you how to parse command-line arguments, I've included the follow code that uses the argparse module :

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--IP',help='display all IP packets',action='store_true')
parser.add_argument('--ARP',help='display all IP packets',choices=['REQ','REPLY'])
```

```
parser.add_argument('--UDP',help='display all UDP packets',action='store_true')
parser.add_argument('--SRC',help='display all IP packets with source port',nargs =1)
parser.add_argument('--DEST',help='display all IP packets with destination port',nargs =1)
parser.add_argument('FILENAME',help=”tcpdump file”)
args = parser.parse_args()
print(args)
print(args.IP)
print(args.ARP)
print(args.UDP)
print(args.SRC)
print(args.DEST)
print(args.FILENAME)
```

Save this code in a file called argp.py and run it as :

```
python3 argp.py --IP --ARP REQ --UDP  --SRC 23 --DEST 31  packet.pcap
```

you should see :

```
Namespace(ARP='REQ', DEST=['31'], FILENAME='fred.tcp', IP=True, SRC=['23'], UDP=True)
True
REQ
True
['23']
['31']
Packet.pcap
```

I will start the lab by give you a brief introduction to tcpdump – just enough to allow you to capture packets and save them in a file. This will be followed by a quick discussion on the argparse module.