CS 7473 Network Security
Rujit Raval
Project 1

Objective: Building a packet parser and a packet generator capable of reading packets from a real network and placing packets in a real network (Ethernet).

Difficulties: The SimplePacketDriver was very useful for the project. Main difficulty was to configure the laptop's adapter to make program working. For using the USB to Ethernet adapter finding the right driver to install was a big task. On other hand, even after finding proper driver, packetparser was able to capture packets from the network but packetgenerator was not sending packets on the network. It took me two days to find the solution for this problem. What I found that, NPF service can use Npcap Packet Driver or Nmap Packet Driver.
"Sc qc npf" command displays the driver being used. Packetgenerator was not working with Nmap packet driver. After uninstalling the nmap and enabling npcap driver, My packetgenerator started workin perfectly.

Usage:

To use Packet generator:
Compile: javac PacketGenerator.java
Run: java PacketGenerator *filename*

To use Packet parser:
Compile: javac PacketParser.java
Run: java PacketParser [-c count] [-r filename] [-o filename] [-t type] [-h] [-src saddress] [-dst daddress] [-sord saddress daddress] [-sandd saddress daddress] [-sport port1 port2] [-dport port1 port2]

-c count	Exit after receiving count packets
-r filename	Read packets from file (your program should read packets from the network by default)
-o filename	Save output to filename
-t type	Print only packets of the specified type where type is one of: eth, arp, ip, icmp, tcp or udp
-h	Print header info only as specified by -t
-src saddress	Print only packets with source address equal to saddress
-dst daddress	Print only packets with destination address equal to daddress
-sord saddress daddress	Print only packets where the source address matches saddress or the destination address matches daddress
-sandd saddress daddress	Print only packets where the source address matches saddress and the destination address matches daddress
-sport port1 port2	Print only packets where the source port is in the range port1-port2
-dport port1 port2	Print only packets where the destination port is in the range port1-port2

Description:

Packet Generator:

Packet generator can read a packet from the file and put that traffic onto the network.

public PacketGenerator() {}:

Constructor for opening the adapter. For the better use, it asks for the adapter number to open and use to send the packets.

public static void main(String[] args) {}

Main method of the class which takes an argument for the filename and calls the init() method.

private void SetFile(String file) {}

Method to assign the filename to the variable.

private void init() {}

Method to read data from the file and call PacketSender.

private class PacketSender implements Runnable{}

Method to run the thread and call sendpacket().

public static int hexPairToByte(String hexPair){}

Converts raw data read from file to the int hex pairs.

Packet parser:

Packet Parser works by using following files:

1. **Packet.java**

This class provides two variables to indicate type of the packet and variable to capture validity of the packet.

2. **PacketParser.java**

This is the one with main class.

public PacketParser(){}:

Constructor for opening the adapter. For the better use, it asks for the adapter number to open and use to send the packets.

public static void main(String[] args){}:

Reads the user arguments and calls various methods given below.

private boolean HasAddressArgument(){}:

Returns true if any of the address argument is given either by '-src saddress' , '-dst daddress', '-sord saddress daddress' or '-sandd saddress daddress'.

private boolean HasPortArgument(){}:
Returns true if any of the port argument is given either by '-sport port1 port2' or '-dport port1 port2'.

private void SetSrcPort(String portStart, String portEnd){}:
Method to handle '-sport port1 port2' argument.

private void SetDestPort(String portStart, String portEnd){}:
Method to handle '-dport port1 port2' argument.

private void SetORAddress(String srcAdd, String dstAdd){}:
Reads the '-sord saddress daddress' arguments and set Source and destination addresses for OR.

private void SetANDAddress(String srcAdd, String dstAdd){}:
Reads the '-sandd saddress daddress' arguments and set Source and destination addresses for OR.

private void SetSrcAddress(String srcAdd){}:
Reads the '-src saddress' argument and set Source addresses for filtering the output.

private void SetDestAddress(String dstAdd){}:
Reads the '-dst daddress' argument and set Source addresses for filtering the output.

private void SetPacketType(String Type){}:
Method to set the packet type used in other classes.

private void SetHeaderOnly(){}:
Method to set HeaderOnly variable true if -h argument is given by the user.

private void SetPacketCount(String count){}:
Method to read the -c argument for filtering the number of packets.

private void SetInputFile(String in){}:
Method to read the '-r filename' argument for getting the filename to read from.

private void SetOutputFile(String out){}:
Method to read the '-o filename' argument for save output into the filename.

private void output(Object obj){}:
Method to print the output on console or save into the file.

private void init(){}:
Method to read the packets from live traffic if no '-r filename' argument given or read from the file if -r argument is set.

public static int HextoBytes(String Hex){}:
Method that converts hex pairs into bytes.

private class PacketHandler implements Runnable{}:
Class that implements the thread which handles the package. This class reads the type of packet from the EthernetPacket class and depending upon the packet types sets argument for the various methods to be called.

3. EthernetPacket.java

public EthernetPacket(byte[] packet){}:
Sets the packet type as Ethernet.

public String toString(){}:
Adds the value of Source MAC, Destination MAC, Ethertype and data into the output.

private void Parse(){}:
Get the values of Source MAC, Destination MAC, Ethertype. Method sets the value of EtherType as either IPv4 or ARP.

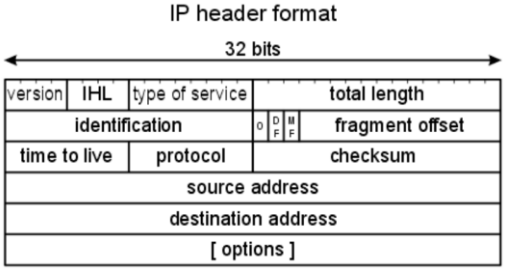
public String ToHex(){}:
Method converts object to the hex data. This method is called by PacketParser class.

public static String HexString(byte[] b){}:
Method converts hex data into string. This is mainly used to convert type into string.

public static int BytesToInt(byte[] Data){}:
Method converts byte data into integer. Used to convert port values into integer in TCPPacket.

public static int UBytesToInt(byte b){}:
Method converts unsigned bytes to integer. Used in ICMPPacket class to convert code and type into integer and ARPPacket class to convert hardware address length and protocol address length into integer.

4. IPPacket.java



Class IPPacket extends the class EthernetPacket.

public IPPacket(byte[] packet){}:
Sets the packet type as IP.

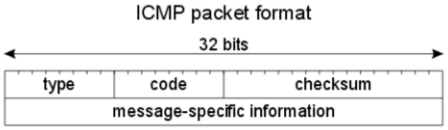
private void Parse(){}:
Get the values of Header length, protocol, Source IP, Destination IP. It also checks the IP protocol number and sets the protocol values such as TCP if '06', UDP if '11' and ICMP if '01'.

private int GetHeaderLen(char HLenChar){}:
Take a character of Header length as an argument and convert it into integer.

public String toString(){}:
Adds the value of header length, Payload start, Protocol, Source address and destination address into the output.

public IPPacket AddressFilter(InetAddress Source, InetAddress Destination, InetAddress SourceOR, InetAddress DestinationOR, InetAddress SourceAND, InetAddress DestinationAND) {}:
Method for filtering the addresses for arguments such as -src, -dst, -sord, -sandd which is called through PacketParser class.

5. ICMPPacket.java



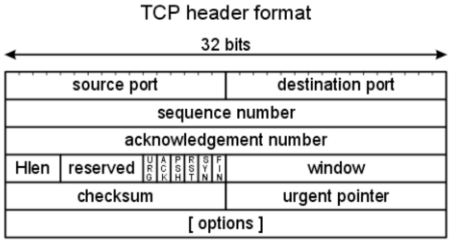
Class checks for 2 fields if the packet type is ICMP, ICMPType and ICMPCode and prints to the output.

public ICMPPacket(byte[] packet){}:
Sets the packet type as ICMP.

private void Parse(){}:
Get the value of type and code.

public String toString(){}:
Adds the value of type and code into the output.

6. TCPPacket.java



Class checks for 2 fields if the packet type is TCP, source port and destination port, and prints to the output.

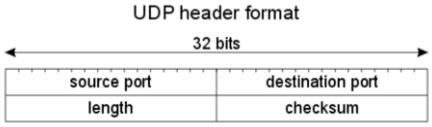
public TCPPacket(byte[] packet){}:
Sets the packet type as TCP.

private void Parse(){}:
Gets the value of source and destination ports.

public String toString(){}:
Adds the value of source and destination ports into the output.

public TCPPacket PortFilter(Integer SourcePortStart, Integer SourcePortEnd, Integer DestinationPortStart, Integer DestinationPortEnd)
Method for filtering the ports. Called through PacketParser class.

7. UDPPacket.java



Class checks for 2 fields if the packet type is UDP, source port and destination port, and prints to the output.

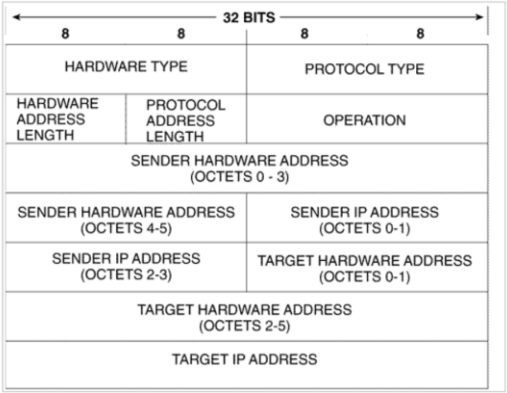
public UDPPacket(byte[] packet){}:
Sets the packet type as UDP.

private void Parse(){}:
Gets the value of source and destination ports.

public String toString(){}:
Adds the value of source and destination ports into the output.

public UDPPacket PortFilter(Integer SourcePortStart, Integer SourcePortEnd, Integer DestinationPortStart, Integer DestinationPortEnd)
Method for filtering the ports. Called through PacketParser class.

8. ARPPacket.java



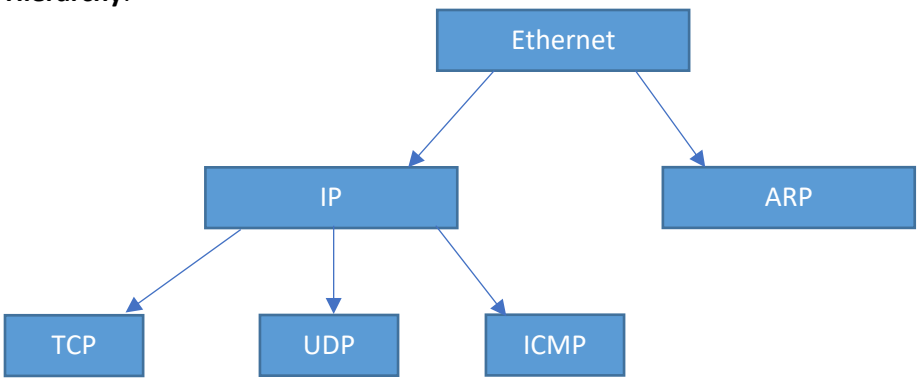
Class checks for fields such as Hardware address length, Protocol address length, Sender hardware address, sender IP address, Target hardware address and Target IP address if the packet type is ARP and prints it to the output.

public ARPPacket(byte[] packet){}:
Sets the packet type as ARP.

private void Parse(){}:
Gets the value of Hardware address length, Protocol address length, Sender hardware address, sender IP address, Target hardware address and Target IP address.

public String toString(){}:
Adds the value of ARP fields into the output.

Hierarchy:



These classes are used to handle packets for different protocols.

Output Files: I have captured the traffic through my packetparser and stored in various files such as output.dat, output1.dat, output2.dat, output3.dat, data.dat.
On this dat files I have used following queries with arguments to get answers for the questions.

Questions:

1. Client IP, username and password of the telnet session to 192.168.1.22.

java PacketParser -r data.dat -dport 22 24

Client IP: 192.168.1.66
Username: igroup15
Password: 192.168.1.66

2. Client IP of the failed ftp logon session to 192.168.1.66.

java PacketParser -r data.dat -dst 192.168.1.66 -dport 20 22

Client IP: 192.168.1.62

3. Client IP, username, password and name of the file transferred to 192.168.1.42.

java PacketParser -r data.dat -dst 192.168.1.42 -dport 20 22

Client IP: 192.168.1.62
Username: group14
Password: 192.168.1.62
Filename: STOR FTP-GROUP14.NFO

4. Client IP, name and content of the html file transferred by 192.168.1.10.

java PacketParser -r data.dat -sport 79 81

Content: hey, Congrats, you' ve found our web page!
Client IP: 192.168.1.22

java PacketParser -r data.dat -dst 192.168.1.10 -dport 79 81

FileName: CS7493

5. IP address of hosts iodine and hydrogen as returned by DNS servers at 192.168.1.14 and 192.168.1.46.

java PacketParser -r data.dat -dport 52 54 -dst 192.168.1.14

Name: iodine.ssac.utulsa.edu
IP: 192.168.1.66

java PacketParser -r data.dat -dport 52 54 -dst 192.168.1.46

Name: hydrogen. group1.ssac.utulsa.edu
IP: 192.168.1.22

6. Send an ARP request to 192.168.1.200, capture the ARP reply with your parser and report the MAC address in your report.

Request:
FF FF FF FF FF FF 00 00 00 04 6F 16 08 06 00 01
08 00 06 04 00 01 00 00 00 04 6F 16 C0 A8 01 15
00 00 00 00 00 00 C0 A8 01 C8 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00

File: ARP_REPLY.dat

```
Source MAC: 00-22-15-61-E3-F4
Destination MAC: 00-00-00-04-6F-16
EtherType: ARP
Data:
 00 00 00 04 6F 16 00 22   15 61 E3 F4 08 06 00 01    ....o.."a......
08 00 06 04 00 02 00 22   15 61 E3 F4 C0 A8 01 C8    ......."a......
00 00 00 04 6F 16 C0 A8   01 15 00 00 00 00 00 00    ....o...........
00 00 00 00 00 00 00 00   00 00 00 00                ............

Hardware Address Length: 6
IP Address Length: 4
Sender Hardware Address: 00-22-15-61-E3-F4
Sender IP Address: /192.168.1.200
Target Hardware Address: 00-00-00-04-6F-16
Target IP Address: /192.168.1.21
```