

PROFESSIONAL TRAINING REPORT
AT
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in INFORMATION TECHNOLOGY

By
M.Kalaivani
REG. NUMBER :- 39120050



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

NOVEMBER 2021



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE
UNIVERSITY)**

Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Kalaivani,M (39120150)** who carried out the project entitled "**Http Packet Sniffers**" under my supervision November 2021.

Internal Guide

Dr.J.Jabez

Head of the Department

Dr.R.SUBHASHINI,M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

InternalExaminer

ExternalExaminer

DECLARATION

I, **Kalaivani.M** hereby declare that the project report entitled **Http Packet Sniffers** done by me under the guidance of **Dr.J,Jabez** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in **INFORMATION TECHNOLOGY**

DATE:

PLACE:CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of INFORMATION TECHNOLOGY** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G.MATHIVANAN M.E.,Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.



CERTIFICATE OF COMPLETION

CSW-I-18-081121 is given to

KALAIVANI M

For completing the Cyber Security training that includes basics of Cyber Security, Pentesting, Networking, and WebApp.

Aaron V Sandeen

AARON SANDEEN

CEO - CSW

N. Vengatesh

VENGATESH NAGARAJAN

MENTOR

ABSTRACT

“Http Package sniffers” A Packet Sniffer also known as packet analyzer ,Protocol analyzer is a piece of hardware or software used to monitor network traffic . Sniffers work by examining Streams of data Packets that flow between computer on a network as well as between networked as well as between networked Computer and the larger internet . These packets are intended for and addressed to specific machines, but using a packet sniffer in "promiscuous mode" allows IT professionals, end users or malicious intruders to examine any packet, regardless of destination. It's possible to configure sniffers in two ways. The first is "unfiltered" meaning they will capture all packets possible and write them to a local hard drive for later examination. Next is "filtered" mode, meaning analyzers will only capture packets that contain specific data elements.

Packet sniffers can be used on both wired and wireless networks — their efficacy depends on how much they are able to "see" as a result of network security protocols.

LIST OF FIGURES:

page no:

- | | |
|---------------|----|
| ○ Screenshots | 7 |
| ○ Screenshots | 8 |
| ○ Screenshots | 9 |
| ○ Screenshots | 10 |

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 WHAT IS CYBER SECURITY?	1
	1.2WHAT IS INFORMATION GATHERING?	2
2.	INFORMATION GATHERING OF THE PRESENT INVESTIGATION	3
	2.1 TCP	3
	2.2 UDP	3
	2.3 ICMP	3
3.	EXPERIMENTAL OR MATERIALS AND METHODS, REQUIREMENTS USED	5
	3.1 VIRTUAL BOX	5
	3.2 KALI LINUX	6
	3.3 python3	7
	3.4 BEAUTIFULSOUP	8
4	SUMMARY	9
	4.1 RESULT	10
	4.2 CONCLUSION	11

REFERENCES	12
APPENDIX	13
A. SCREENSHOTS	14
B. SOURCE CODE	15

INTRODUCTION

1.1 What is Cyber Security?

Cyber security is the application of technologies, processes and controls to protect systems, networks, programs, devices and data from cyber attacks. It aims to reduce the risk of cyber attacks and protect against the unauthorised exploitation of systems, networks and technologies

BENIFITS OF CYBER SECURITY :

- Protects system against viruses, worms, spyware and other unwanted programs
- Protection against data from theft
- Protects the computer from being hacked
- Minimizes computer freezing and crashes
- Gives privacy to users

HOW DO CYBER SECURITY WORKS :

cyber security is the practice of defending your electronic systems, networks, computers, mobile devices, programs and data from malicious digital attacks. ... This type of security uses hardware and software to protect any data that's sent through your computer and other devices to the network.

PACKET SNIFFERS :

Packet sniffing is the practice of gathering, collecting, and logging some or all packets that pass through a computer network, regardless of how the packet is addressed. In this way, every packet, or a defined subset of packets, may be gathered for further analysis. You as a network administrators can use the collected data for a wide variety of purposes like monitoring bandwidth and traffic. A packet sniffer, sometimes called a packet analyzer, is composed of two main parts. First, a network adapter that connects the sniffer to the existing network. Second, software that provides a way to log, see, or analyze the data collected by the device.

IMPORTANCE OF PACKET SNIFFERS :

Packet sniffing plays a very important role in network management. Network managers and technicians use packet sniffers to diagnose underlying problems on their networks. So a packet sniffer is essentially a tool that helps monitor network traffic and troubleshoot a network. In network management, packet sniffing plays a very crucial role. Network managers and technicians use packet sniffers, also called packet analyzers, to diagnose underlying problems in their networks. So, a packet sniffer is essentially a tool that aids in monitoring network traffic and troubleshooting a network.

EXAMPLE OF PACKET SNIFFERS :

Tcpdump and Wireshark are examples of packet sniffers. Tcpdump provides a CLI packet sniffer, and Wireshark provides a feature-rich GUI for sniffing and analyzing packets. By default, tcpdump operates in promiscuous mode. This simply means that all packets reaching a host will be sent to tcpdump for inspection.

PACKET SNIFFER USED FOR :

Packet sniffing is a technique whereby packet data flowing across the network is detected and observed. Network administrators use packet sniffing tools to monitor and validate network traffic, while hackers may use similar tools for nefarious purposes

SNIFFING TOOLS :

A sniffer is a software or hardware tool that allows the user to “sniff” or monitor your internet traffic in real time, capturing all the data flowing to and from your computer

INFORMATION GATHERING OF THE PRESENT INVESTIGATION

2.1 TCP :

Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation by which applications can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other..

2.2 UDP:

User Datagram Protocol (UDP) is a communications protocol that is primarily used to establish low-latency and loss-tolerating connections between applications on the internet. UDP speeds up transmissions by enabling the transfer of data before an agreement is provided by the receiving party.

2.3 ICMP:

The Internet Control Message Protocol (ICMP) is a protocol that devices within a network use to communicate problems with data transmission. In this ICMP definition, one of the primary ways in which ICMP is used is to determine if data is getting to its destination and at the right time.

EXPERIMENTAL OR MATERIALS AND METHODS, REQUIREMENTS USED

3.1 VIRTUAL BOX

Oracle **VM VirtualBox** is a free and open-source hosted hypervisor for x86 virtualization, developed by Oracle Corporation. **IT is a virtualization software.**

Operating system: Windows, macOS (only Intel-based Macs), Linux and Solaris.

USES OF VIRTUAL BOX

Oracle VM VirtualBox is cross-platform virtualization software. It **allows users to extend their existing computer to run multiple operating systems** including Microsoft Windows, MacOS X, Linux, and Oracle Solaris, at the same time.

ADVANTAGES:

- Lower hardware costs. Many organizations don't fully utilize their hardware resources.
- Quicker Desktop Provisioning and Deployment. Deploying a new physical server often takes numerous time-consuming steps.
- Smaller Footprint.
- Enhanced Data Security.
- Portability.
- Improved IT Efficiency.

3.2 KALI LINUX

Kali Linux is a Debian-derived Linux distribution designed for digital forensics and penetration testing. It is maintained and funded by Offensive Security.

USES OF KALI LINUX

Kali Linux is mainly used for **advanced Penetration Testing and Security Auditing**. Kali contains several hundred tools which are geared towards various information security tasks such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering.

ADVANTAGES OF KALI LINUX:

1. More than 600 penetration tools included
2. Adherence of file-system hierarchy standard

3. Open development tree
4. Multiple-language support
5. Completely customizable
6. Free (as in beer) and always will be
7. Wide-ranging wireless device support

3.3 PYTHON3

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

3.4 METASPLOITABLE :

The Metasploitable virtual machine is an intentionally vulnerable version of Ubuntu Linux designed for testing security tools and demonstrating common vulnerabilities. This virtual machine is compatible with VMWare, VirtualBox, and other common virtualization platforms

ADVANTAGES OF METASPLOITABLE :

- Open source. One of the top reasons why one should go with Metasploit is because it is open source and actively developed. ...
- Support for testing large networks and easy naming conventions. ...
- Smart payload generation and switching mechanism. ...
- Cleaner exits. ...
- The GUI environment

SYSTEM REQUIREMENTS

- 1TB HDD
- 8 GB RAM

TOOLS USED

- VISUAL STUDIO CODE (text editor)
- KALI LINUX (operating system)
- VIRTUAL BOX (virtualization)
- PYTHON3

REFERENCE :

- BLACK HAT PYTHON 2ND EDITION
- GITHUP.COM
- GOOGLE.COM

RESULT:

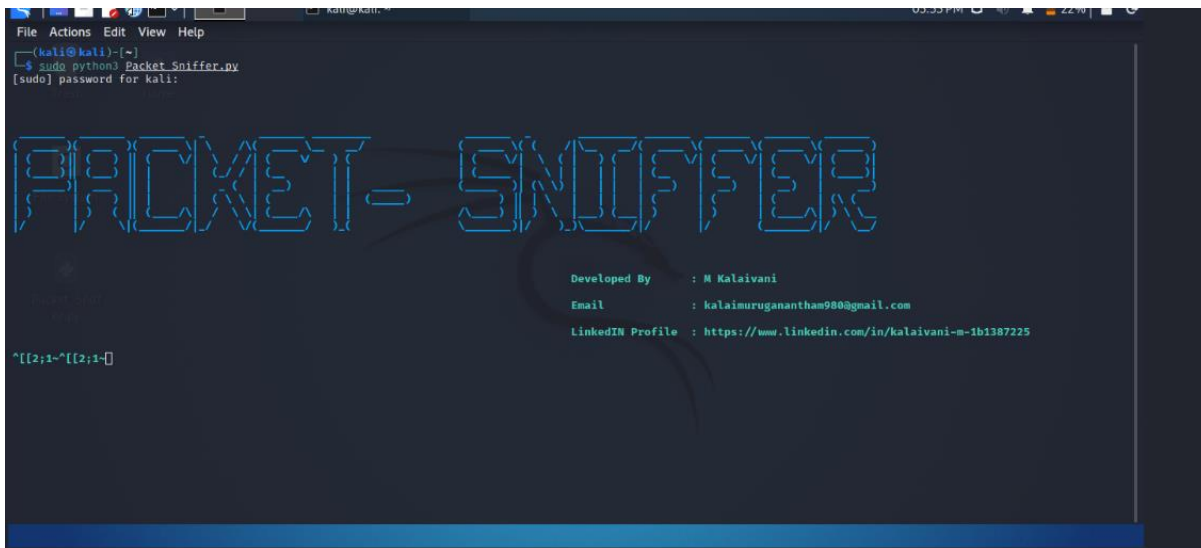
THUS the tools I have been implemented has been executed successfully.

CONCLUSION:

A high-quality network analyzer bundled with useful advanced features as the above will help any engineer or administrator diagnose and deal with network problems quickly and efficiently, but also capture suspicious network traffic patterns often associated with hacking attempts.

6

Screenshots for output

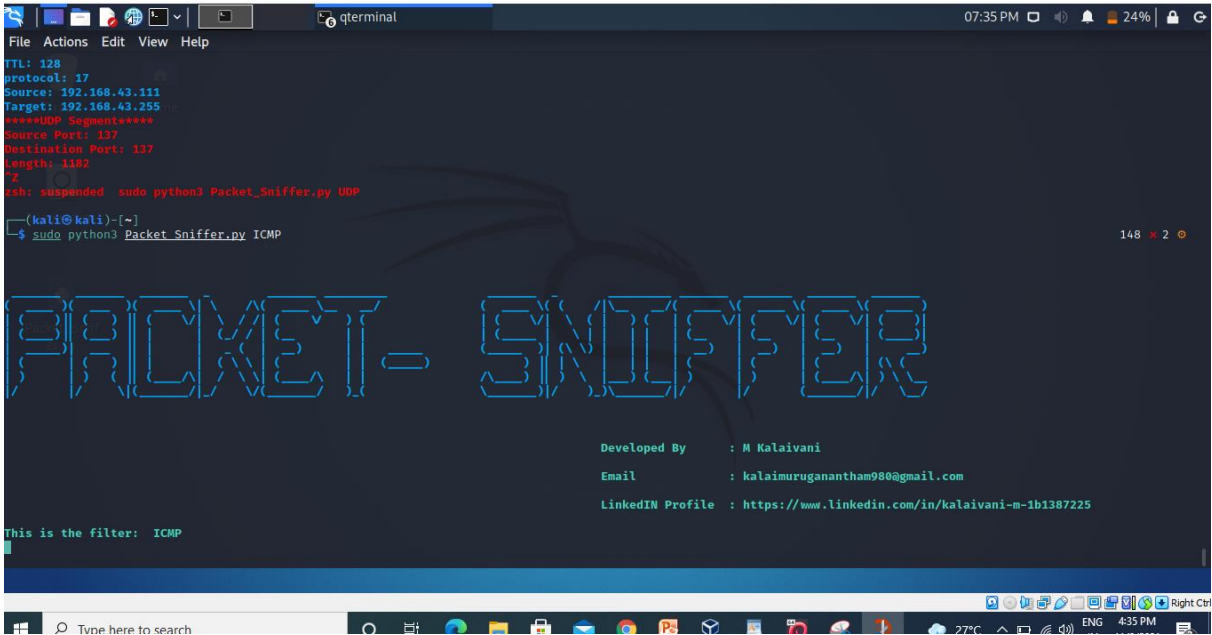


```
File Actions Edit View Help
(kali@kali)~$ sudo python3 Packet_Sniffer.py
[sudo] password for kali:

PACKET-SNIFFER

Developed By : M Kalaivani
Email : kalaimuruganatham980@gmail.com
LinkedIn Profile : https://www.linkedin.com/in/kalaivani-m-1b1387225

*[[2;1-~[[2;1-
```



```

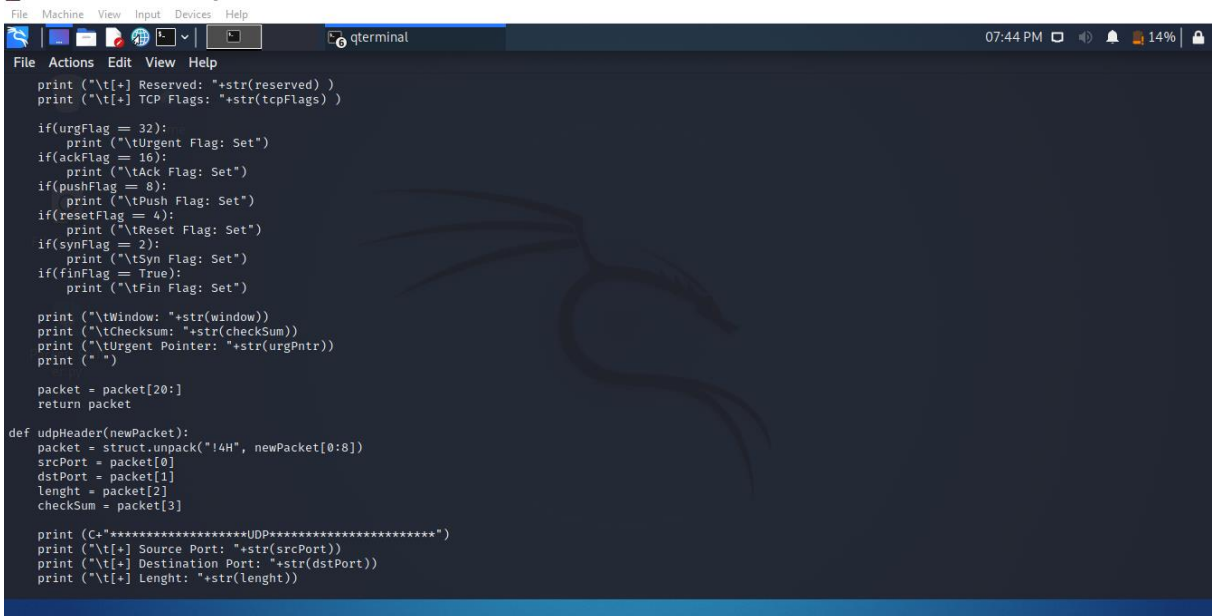
File Actions Edit View Help
TTL: 128
protocol: 17
Source: 192.168.43.111
Target: 192.168.43.255
*****UDP Segment*****
Source Port: 137
Destination Port: 137
Length: 1182
^Z
zsh: suspended sudo python3 Packet_Sniffer.py UDP
(kali@kali)-[~]
$ sudo python3 Packet_Sniffer.py ICMP

PACKET-SNOFFER

Developed By      : M Kalaivani
Email             : kalaimuruganatham980@gmail.com
LinkedIN Profile  : https://www.linkedin.com/in/kalaivani-m-1b1387225

This is the filter: ICMP

```



```

File Machine View Input Devices Help
print ("\t[+] Reserved: "+str(reserved) )
print ("\t[+] TCP Flags: "+str(tcpFlags) )

if(urgFlag == 32):
    print ("\tUrgent Flag: Set")
if(ackFlag == 16):
    print ("\tAck Flag: Set")
if(pushFlag == 8):
    print ("\tPush Flag: Set")
if(resetFlag == 4):
    print ("\tReset Flag: Set")
if(synFlag == 2):
    print ("\tSyn Flag: Set")
if(finFlag == True):
    print ("\tFin Flag: Set")

print ("\tWindow: "+str(window))
print ("\tChecksum: "+str(checkSum))
print ("\tUrgent Pointer: "+str(urgPtr))
print (" ")

packet = packet[20:]
return packet

def udpHeader(newPacket):
    packet = struct.unpack("!4H", newPacket[0:8])
    srcPort = packet[0]
    dstPort = packet[1]
    lenght = packet[2]
    checkSum = packet[3]

    print (C+"*****UDP*****")
    print ("\t[+] Source Port: "+str(srcPort))
    print ("\t[+] Destination Port: "+str(dstPort))
    print ("\t[+] Lenght: "+str(lenght))

```

```
File Actions Edit View Help
+ ] Source Port: 2048
+ ] Destination Port: 10051
sequence: 1941700719
acknowledgment: 1682054486
****Flags****
URG: 2048
ACK: 17704
PSH: 52
EST: 44062
SYN: 0
FIN: 10502
****TCP Data****
None
*****TCPv4*****
Version: 4
Header Length: 20
TL: 64
Protocol: 6
Source: 192.168.43.110
Target: 142.250.185.163
****TCP Segment****
+ ] Source Port: 111
+ ] Destination Port: 25666
sequence: 492177408
acknowledgment: 658731964
****Flags****
URG: 2048
ACK: 17664
PSH: 52
EST: 3975
SYN: 16384
FIN: 16390
****TCP Data****
None
|
```

SOURCE CODE :

```
def main():
    conn = socket.socket(socket.PF_PACKET, socket.SOCK_RAW,
                          socket.ntohs(3))

    filters = (["ICMP", 1, "ICMPv6"], ["UDP", 17, "UDP"], ["TCP", 6, "TCP"])
    filter = []

    if len(sys.argv) == 2:
        print("This is the filter: ", sys.argv[1])
        for f in filters:
            if sys.argv[1] == f[0]:
                filter = f

    while True:
        raw_data, addr = conn.recvfrom(65536)
        dest_mac, src_mac, eth_proto, data = ethernet_frame(raw_data)

        if eth_proto == 'IPv6':
            newPacket, nextProto = ipv6Header(data, filter)
            printPacketsV6(filter, nextProto, newPacket)

        elif eth_proto == 'IPv4':
            printPacketsV4(filter, data, raw_data)

    def printPacketsV4(filter, data, raw_data):
        (version, header_length, ttl, proto, src, target, data) = ipv4_Packet(data)

        # ICMP
        if proto == 1 and (len(filter) == 0 or filter[1] == 1):
            icmp_type, code, checksum, data = icmp_packet(data)
```

```

print (C+"*****ICMP*****")
    print ("\tlCMP type: %s" % (icmp_type))
    print ("\tlCMP code: %s" % (code))
    print ("\tlCMP checksum: %s" % (checksum))

# TCP
elif proto == 6 and (len(filter) == 0 or filter[1] == 6):
    print(C+"*****TCPv4*****")
    print('Version: {}\nHeader Length: {}\nTTL: {}'.format(version,
        header_length, ttl))
    print('protocol: {}\nSource: {}\nTarget: {}'.format(proto, src, target))
    src_port, dest_port, sequence, acknowledgment, flag_urg, flag_ack,
    flag_psh, flag_rst, flag_syn, flag_fin = struct.unpack(
        '! H H L L H H H H H H', raw_data[:24])
    print('*****TCP Segment*****')
    print('[+] Source Port: {}\n[+]Destination Port: {}'.format(src_port,
        dest_port))
    print('Sequence: {}\nAcknowledgment: {}'.format(sequence,
        acknowledgment))
    print('*****Flags*****')
    print('URG: {}\nACK: {}\nPSH: {}'.format(flag_urg, flag_ack, flag_psh))
    print('RST: {}\nSYN: {}\nFIN:{}'.format(flag_rst, flag_syn, flag_fin))

    if len(data) > 0:
        # HTTP
        if src_port == 80 or dest_port == 80:
            print('*****HTTP Data*****')
            try:
                http = HTTP(data)
                http_info = str(http.data).split('\n')
                for line in http_info:
                    print(str(line))
            except:
                print(format_output_line("",data))
            else:
                print(R+'*****TCP Data*****')

```

```

        print(format_output_line("",data))
        # UDP
    elif proto == 17 and (len(filter) == 0 or filter[1] == 17):
        print(C+"*****UDPv4*****")
        print('Version: {}\nHeader Length: {}\nTTL: {}'.format(version,
            header_length, ttl))
        print('protocol: {}\nSource: {}\nTarget: {}'.format(proto, src, target))
        src_port, dest_port, length, data = udp_seg(data)
        print(R+"*****UDP Segment*****")
    print('Source Port: {}\nDestination Port: {}\nLength: {}'.format(src_port,
        dest_port, length))

def printPacketsV6(filter, nextProto, newPacket):
    remainingPacket = ""

    if (nextProto == 'ICMPv6' and (len(filter) == 0 or filter[2] == "ICMPv6")):
        remainingPacket = icmpv6Header(newPacket)
    elif (nextProto == 'TCP' and (len(filter) == 0 or filter[2] == "TCP")):
        remainingPacket = tcpHeader(newPacket)
    elif (nextProto == 'UDP' and (len(filter) == 0 or filter[2] == "UDP")):
        remainingPacket = udpHeader(newPacket)

    return remainingPacket

def tcpHeader(newPacket):
    # 2 unsigned short,2unsigned Int,4 unsigned short.
    # 2byt+2byt+4byt+4byt+2byt+2byt+2byt+2byt==20bytes
    packet = struct.unpack("!2H2I4H", newPacket[0:20])
    srcPort = packet[0]
    dstPort = packet[1]
    sqncNum = packet[2]
    acknNum = packet[3]
    dataOffset = packet[4] >> 12
    reserved = (packet[4] >> 6) & 0x003F
    tcpFlags = packet[4] & 0x003F
    urgFlag = tcpFlags & 0x0020

```

```
ackFlag = tcpFlags & 0x0010
pushFlag = tcpFlags & 0x0008
resetFlag = tcpFlags & 0x0004
synFlag = tcpFlags & 0x0002
finFlag = tcpFlags & 0x0001
window = packet[5]
checksum = packet[6]
urgPntr = packet[7]
```

```
print (C+"*****TCP*****")
print ("\t[+] Source Port: "+str(srcPort) )
print ("\t[+] Destination Port: "+str(dstPort) )
print ("\t[+] Sequence Number: "+str(sqncNum) )
print ("\t[+] Ack. Number: "+str(acknNum) )
print ("\t[+] Data Offset: "+str(dataOffset) )
print ("\t[+] Reserved: "+str(reserved) )
print ("\t[+] TCP Flags: "+str(tcpFlags) )
```

```
if(urgFlag == 32):
print ("\tUrgent Flag: Set")
if(ackFlag == 16):
print ("\tAck Flag: Set")
if(pushFlag == 8):
print ("\tPush Flag: Set")
if(resetFlag == 4):
print ("\tReset Flag: Set")
if(synFlag == 2):
print ("\tSyn Flag: Set")
if(finFlag == True):
print ("\tFin Flag: Set")
```

```
print ("\tWindow: "+str(window))
print ("\tChecksum: "+str(checkSum))
print ("\tUrgent Pointer: "+str(urgPntr))
print (" ")
```

```
packet = packet[20:]  
return packet
```

```
def udpHeader(newPacket):  
packet = struct.unpack("!4H", newPacket[0:8])  
srcPort = packet[0]  
dstPort = packet[1]  
length = packet[2]  
checksum = packet[3]
```

```
print (C+"*****UDP*****")  
print ("\t[+] Source Port: "+str(srcPort))  
print ("\t[+] Destination Port: "+str(dstPort))  
print ("\t[+] Length: "+str(length))  
print ("\t[+] Checksum: "+str(checksum))  
print (" ")
```

```
packet = packet[8:]  
return packet
```

```
def icmpv6Header(data):  
ipv6_icmp_type, ipv6_icmp_code, ipv6_icmp_checksum = struct.unpack(  
">BBH", data[:4])
```

```
print (C+"*****ICMPv6*****")  
print ("\t[+] ICMPv6 type: %s" % (ipv6_icmp_type))  
print ("\t[+] ICMPv6 code: %s" % (ipv6_icmp_code))  
print ("\t[+] ICMPv6 checksum: %s" % (ipv6_icmp_checksum))
```

```
data = data[4:]  
return data
```

```
def nextHeader(ipv6_next_header):  
if (ipv6_next_header == 6):
```

```
    ipv6_next_header = 'TCP'
elif (ipv6_next_header == 17):
    ipv6_next_header = 'UDP'
elif (ipv6_next_header == 43):
    ipv6_next_header = 'Routing'
elif (ipv6_next_header == 1):
    ipv6_next_header = 'ICMP'
elif (ipv6_next_header == 58):
    ipv6_next_header = 'ICMPv6'
elif (ipv6_next_header == 44):
    ipv6_next_header = 'Fragment'
elif (ipv6_next_header == 0):
    ipv6_next_header = 'HOPOPT'
elif (ipv6_next_header == 60):
    ipv6_next_header = 'Destination'
elif (ipv6_next_header == 51):
    ipv6_next_header = 'Authentication'
elif (ipv6_next_header == 50):
```

