ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES
Autonomous status accorded by UGC and Andhra University
Approved by AICTE, Permanently Affiliated to Andhra University
Accredited by NBA (IT,CSE,EEE,ECE, and Mech) & accredited by NAAC with "A" Grade
Sangivalasa - 531162, Bheemunipatnam (Mandal), Visakhapatnam (Dist.)
Phone: 08933 - 225083, 225084, 226131, Fax: 08933-226395
Email: principal@anits.edu.in

PRAGNANAM BRAHMA
ANITS
COLLEGE CODE - ANIL

A
Laboratory Manual
For

CRYPTOGRAPHY AND NETWORK SECURITY LAB

Faculty In-Charge:
1. Dr. V.Usha Bala, Asst. Professor
2. Dr.G.Jagadish, Asst. Professor
3. Mrs.S.S.N.L.Priyanka, Asst. Professor
4. Mr.Sk.A.Manoj, Asst. Professor

HOD,CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VISION:

Our vision is to emerge as a world class Computer Science and Engineering department through excellent teaching and strong research environment that responds swiftly to the challenges of changing computer science technology and addresses technological needs of the stakeholders.

MISSION:

To enable our students to master the fundamental principles of computing and to develop in them the skills needed to solve practical problems using contemporary computer-based technologies and practices to cultivate a community of professionals who will serve the public as resources on state-of-the-art computing science and information technology.

Course outcomes:

1.	Understand the process of capturing Network traffic using tools(Ethereal,Wireshark, Tcpdump)
2.	Implement Cryptographic algorithms in C/C++/Java
3.	Understand Buffer Over Flow attacks, Intrusion Detection Systems and Honeypots.
4	Create applications in Client Server architecture.
5.	Set up secure mail and web communication channels.

PROGRAM OUTCOMES (POs):

PROGRAM SPECIFIC OUTCOMES (PSOs):

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

A Laboratory Manual For CRYPTOGRAPHY AND NETWORK SECURITY (CSE 416)

Semester – 1

PRAGNANAM BRAHMA
ANITS

Prepared by

1. Dr. V.Usha Bala, Asst. Professor
2. Dr.G.Jagadish, Asst. Professor
3. Mrs.S.S.N.L.Priyanka,Asst.Professor
4. Mr.Sk.A.Manoj, Asst. Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

S.NO	LIST OF EXPERIMENTS	CO
1	Working with Sniffers for monitoring network communication using a)Ethereal b)Wire shark c) Snort d) tcp dump.	1
2	Implementation and Performance evaluation of various cryptographic algorithms in C/C++ a)DES b)RSA.	2
3	Using IP TABLES on Linux and setting the filtering rules.	4
4	Using open SSL for web server - browser communication.	3
5	Configuring S/MIME for e-mail communication.	3
6	Understanding the buffer overflow and format string attacks.	3
7	Using NMAP for ports monitoring.	4
8	Secure Socket programming.	3
	CASE STUDIES	
9	Study of GNU PGP.	3
10	Study Intrusion Detection Systems and Honey pots.	4

LIST OF INDUSTRY RELEVANT SKILLS:

1. **Security Incident Handling and Response**
2. **Intrusion Detection**
3. **Firewall IDS/IPS/Skills**
4. **Malware Analysis(Using Wireshark and IPTables)**

GUIDELINES TO TEACHERS

1. **The teachers should train the students on various features of Cyber Security, its advantages and disadvantages.**
2. **The teachers should also train the students to ethically use the internet.**
3. **To prepare students for real world security challenges.**

INSTRUCTIONS TO STUDENTS:

1. **The students need to learn how to safeguard their information on the internet.**
2. **The students should be able to overcome various threats and attacks on the internet.**

GUIDELINES TO LAB PROGRAMMERS:

- **Installation of ETHREAL,WIRESHARK and TCPDump packet capturing tools.**
- **Availability of full time INTERNET with good network configuration during lab sessions.**

LAB RUBRICS

CRYPTOGRAPHY & NETWORK SECURITY LAB	
Course Code: CSE416	**Credits : 2**
Instruction : 3 Periods/Week	**Sessional Marks : 50**
End Exam : 3 Hours	**End Exam Marks : 50**

Key Performance Criteria(KPC) (25 pts)	4-Very Good	3-Good	2-Fair	1-Need to improve
Problem Statement (2)	The thorough knowledge of the problem statement.(2)	The better knowledge of the problem statement(2)	The basic knowledge of the problem statement(2)	The partial knowledge of the problem statement.(1)
Experimental procedure (Algorithm/Flowchart/Tools Description) (4)	The experimental procedure is explained with the relevant implementation of the Tool.(4)	The experimental procedure is explained clearly and the details are covered.(3)	The experimental procedure is explained and few details are covered.(2)	The experimental procedure is explained, some minor implementation details are missing.(1)
Working with Tools and Simulation(4)	Simulation of the tools w.r.t the given Problem Statement is executed using the respective commands/ Source code effectively. (4)	Simulation of the tools w.r.t the given Problem Statement is executed using the respective commands/ Source code.(3)	Simulation of the tools w.r.t the given Problem Statement is executed using the respective commands/ Source code With tool.(3)	Simulation of the tools w.r.t the given Problem Statement is executed partially using the respective commands/ Source code.(2)
Test Case Verification (3)	Produces correct output for all mentioned test cases correctly in implementation /simulation(3)	Produces correct output for majority of test cases correctly in implementation /simulation(3)	Produces correct output for few important possibl e test cases correctly in implementation /simulation(3)	Produces Wrong output for most of the test cases in the implementation /simulation (1)
Oral Presentation/ Viva(5)	In depth knowledge on the concept and answered all the questions(5)	Good knowledge on the concept and answered all the questions(4)	Basic knowledge on the concept and answered some of the questions(3)	With basic knowledge on the concept and answered few questions(2)

Presentation / Documentation based on Observation (4)	Presented accurately all the prescribed documentation on time (4)	Presented all the required documentation on time as per the prescribed format (3)	Presented documents in a readable manner but not so neatly. Submitted documents on time.(2)	Submitted documents in ambiguity and not on time.(2)
Code of Conduct- (Courtesy , safety and ethics based on physical observation) (3)	While conducting the procedure, the student is in proper dress code, always respectful of others and leaves the area clean.(3)	While conducting the procedure, the student is in proper dress code, many times respectful of others and leaves the area clean only after being reminded.(2)	While conducting the procedure, the student is in partial dress code, sometimes respectful of others and leaves the area clean only after being reminded.(2)	While conducting the procedure, the student is not in proper dress code , not respectful of others and leaves the area messy even after being reminded.(1)

PRACTICAL 1:

Working with Sniffers for monitoring network communication using a)Ethereal b)Wire shark c) Snort d) TCP dump.

1a. Practical significance of Ethereal:

(i) One of the best security and network troubleshooting tools available is a protocol analyzer (or packet sniffer) named Ethereal. Ethereal runs on both Windows and Linux; it captures all packets promiscuously. Ethereal uses WinPcap to pull packets off the network.

1b. Practical significance of WIRESHARK:

(i) Wireshark is a network analyzer. It can read and process capture files from a number of different products, including other sniffers, routers, and network utilities. It uses the popular Promiscuous Capture Library (libpcap)-based capture format and can easily interfere with other products that use libpcap. Wireshark possesses an easy-to-read and configurable graphical user interface (GUI) along with rich display filter capabilities.

2. Relevant Program Outcomes:

PO-1, 2, 3, 5, 9, 11

3. Competency and practical skills:

This practical is expected to develop the following skills:

a. Practical skill:

(i) Ability to work with ETHREAL and capture the network packets.

(ii) Provides an option to select the interface across which one wants to capture the packets.

b. Competency skill:

(i) Familiarity with the Ethereal packet capture and analyzing the network packets.

(ii) Troubleshoot network problems.

4.Pre-requisites :

- **Knowledge of Networking and packet tracing.**
- **You may need special privileges to start a live capture.**
- **• You need to choose the right network interface to capture packet data from.**
- **• You need to capture at the right place in the network to see the traffic you want to see.**

5. Resources required:

ETHEREAL/WIRESHARK, 40GB hard disk with minimum of 4 G HZ RAM

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 40GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with) 5.Wireshark runs on most UNIX and UNIX-like platforms including Linux and most BSD variants. 6.A supported network card for capturing-Ethernet, IEEE 802.11
2	Installation of ETHEREAL , WIRESHARK packet capturing tools.

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard; mouse and monitor are properly working with
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions will help in preventing injuries to people and damage to computer equipment.
5. A safe work place should be clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities at the workplace. Power supplies and monitors contain high voltage, handle with care.
7. Ensure that packet capturing tool ETHEREAL/WIRESHARK is installed in the system before usage.
8. Ensure the availability of the Internet for the successful execution of these experiments.

7. Algorithm/circuit/Diagram/Description with Sample outputs:

1a. Ethereal:

- **Capturing packets using Ethereal:**
  - Begin the capture process by selecting “Start” from the Ethereal “Capture” menu , do not stop the process.
  - Depending on the type of network and the network adapter, you may immediately see packets being saved to your machine. On other configurations, you will not see anything until you create some network traffic.

▪ Open a web page, when the web page finished loading, go back to Ethereal and push the stop button. Ethereal will process and load summaries of all of the packets sent and received by your machine.

● Looking at Packets Captured by Ethereal:

● Once you have captured a set of packets, Ethereal should present you with a colorful window.

1b. Wireshark:

1. A good approach to network troubleshooting involves several steps such as:
a. recognizing the symptoms;
b. defining the problem,
c. analyzing the problem,
d. isolating the problem,
e. identifying and testing the cause of the problem,
f. solving the problem, and
g. verifying that the problem has been solved.

● Capturing packets using Wireshark:

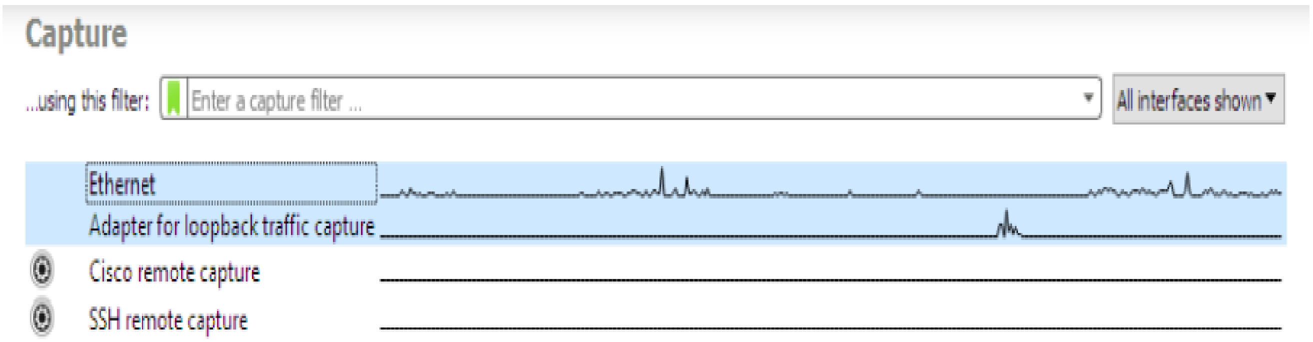
▪ The following methods can be used to start capturing packets with Wireshark:
● You can double-click on an interface in the welcome screen.
● You can select an interface in the welcome screen, then select Capture › Start or click the first 57 toolbar button.
● You can get more detailed information about available interfaces using The "Capture Options" Dialog Box (Capture › Options…).

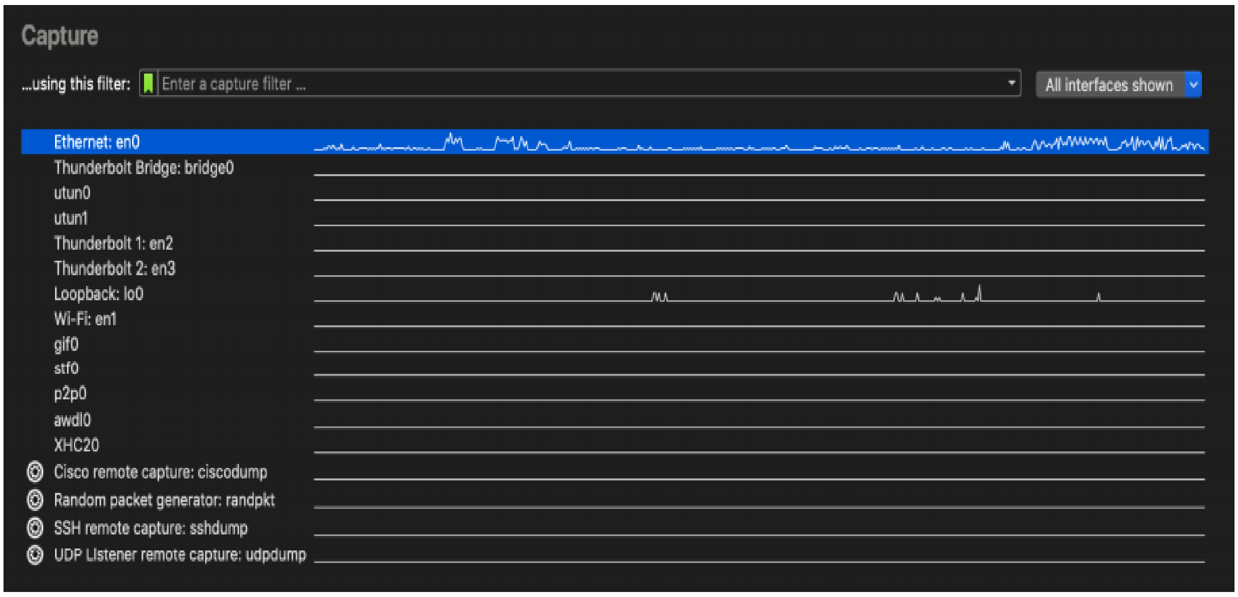
● If you already know the name of the capture interface you can start Wireshark from the command line:
● $ wireshark -i eth0 -k
● This will start Wireshark capturing on interface eth0.
● More details can be found at Start Wireshark from the command line.

● Sample Outputs while Viewing:

The "Capture" Section Of The Welcome Screen

When you open Wireshark without starting a capture or opening a capture file it will display the "Welcome Screen," which lists any recently opened capture files and available capture interfaces. Network activity for each interface will be shown in a sparkline next to the interface name. It is possible to select more than one interface and capture from them simultaneously.




Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Ethernet: en0
Thunderbolt Bridge: bridge0
utun0
utun1
Thunderbolt 1: en2
Thunderbolt 2: en3
Loopback: lo0
Wi-Fi: en1
gif0
stf0
p2p0
awdl0
XHC20
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

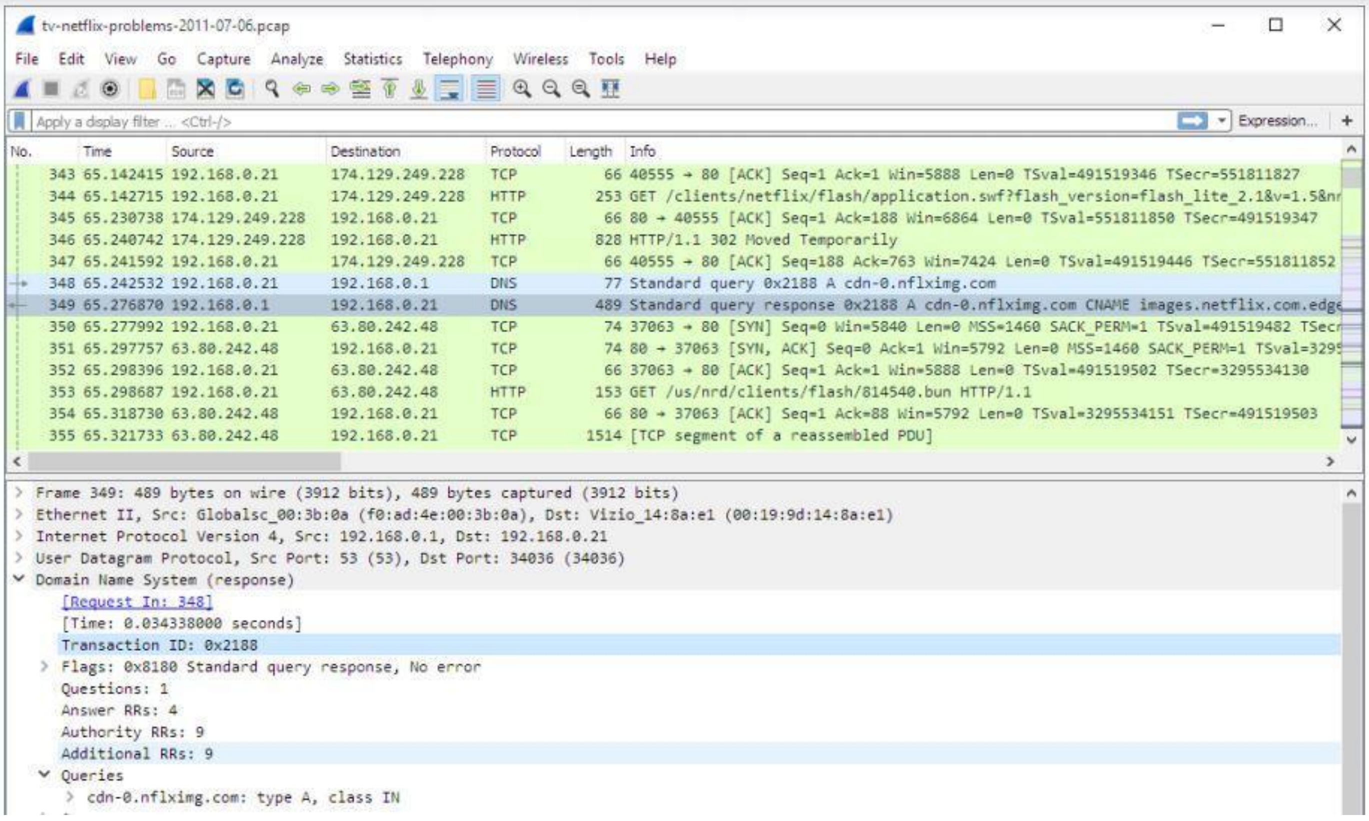
▪ Begin the capture process by selecting “Start” from the Ethereal “Capture” menu , do not stop the process.

▪ Depending on the type of network and the network adapter, you may immediately see packets being saved to your machine. On other configurations, you will not see anything until you create some network traffic.

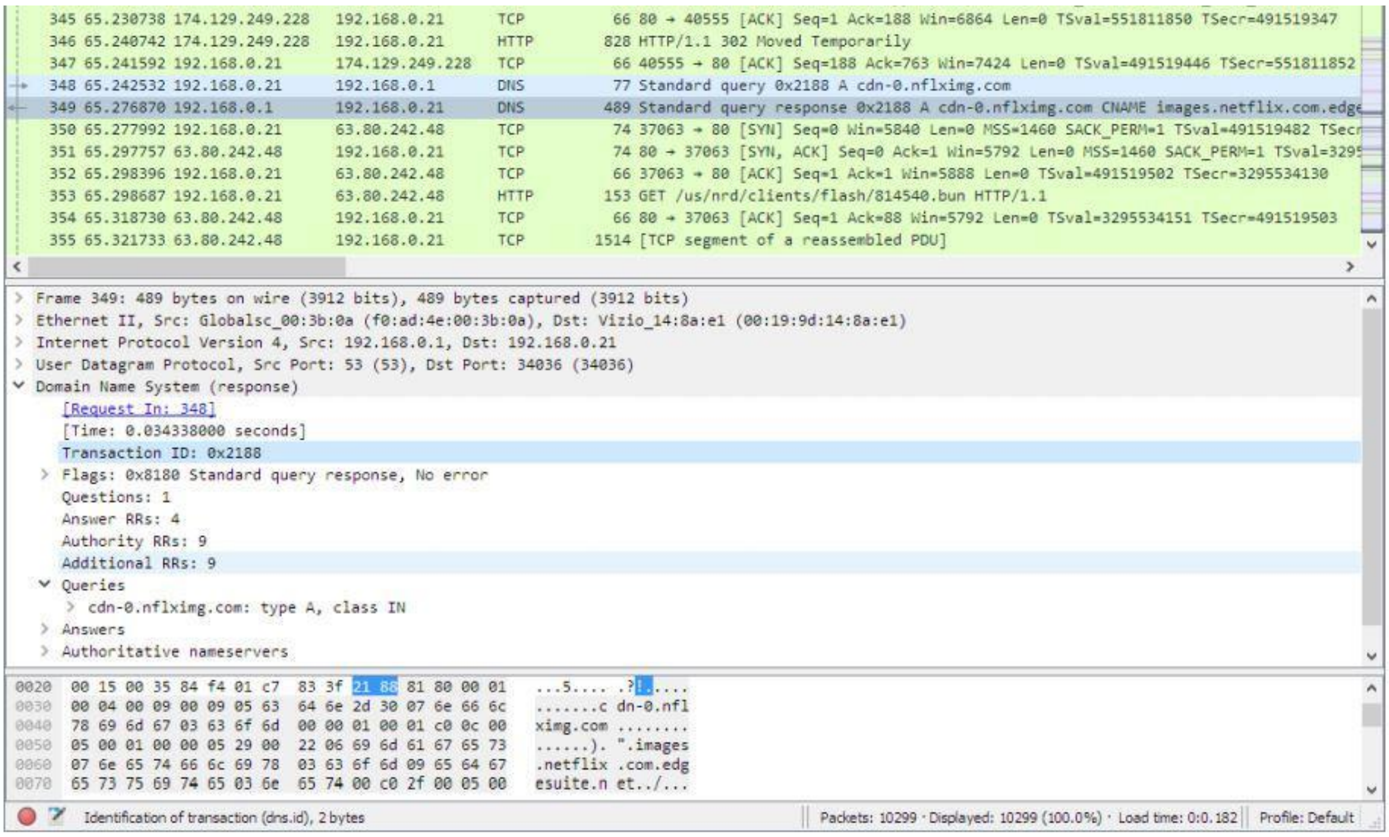
▪ Open a web page, when the web page finished loading, go back to Ethereal and push the stop button. Ethereal will process and load summaries of all of the packets sent and received by your machine.

● Looking at Packets Captured by Wireshark:

● Once you have captured a set of packets, Wireshark should present you with a colorful window as shown below.


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time      Source          Destination     Protocol Length Info
 343 65.142415 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
 344 65.142715 192.168.0.21    174.129.249.228 HTTP      253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
 345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
 346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
 347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
 348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
 349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
 350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
 351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
 352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
 353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
 354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
 355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]

> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
```



```
 345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
 346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
 347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
 348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
 349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
 350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
 351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
 352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
 353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
 354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
 355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]

> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
```

```
  > Answers
  > Authoritative nameservers

0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...

Identification of transaction (dns.id), 2 bytes    Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182    Profile: Default
```

Wireshark's main window consists of parts that are commonly known from many other GUI programs.
1. The menu (see The Menu) is used to start actions.

2. The main toolbar (see The “Main” Toolbar) provides quick access to frequently used items from the menu.
3. The filter toolbar (see The “Filter” Toolbar) allows users to set display filters to filter which packets are displayed (see Filtering Packets While Viewing).
4. The packet list pane (see The “Packet List” Pane) displays a summary of each packet captured. By clicking on packets in this pane you control what is displayed in the other two panes.
5. The packet details pane (see The “Packet Details” Pane) displays the packet selected in the packet list pane in more detail.
6. The packet bytes pane (see The “Packet Bytes” Pane) displays the data from the packet selected in the packet list pane, and highlights the field selected in the packet details pane.
7. The status-bar shows some detailed information about the current program state and the captured data.

Wireshark’s MENU:

The main menu contains the following items:

◆ File This menu contains items to open and merge capture files, save, print, or export capture files in whole or in part, and to quit the Wireshark application.

◆ Edit This menu contains items to find a packet, time reference or mark one or more packets, handle configuration profiles, and set your preferences; (cut, copy, and paste are not presently implemented).

◆ View This menu controls the display of the captured data, including colorization of packets, zooming the font, showing a packet in a separate window, expanding and collapsing trees in packet details, ….

◆ Go This menu contains items to go to a specific packet.

◆ Capture This menu allows you to start and stop captures and to edit capture filters.

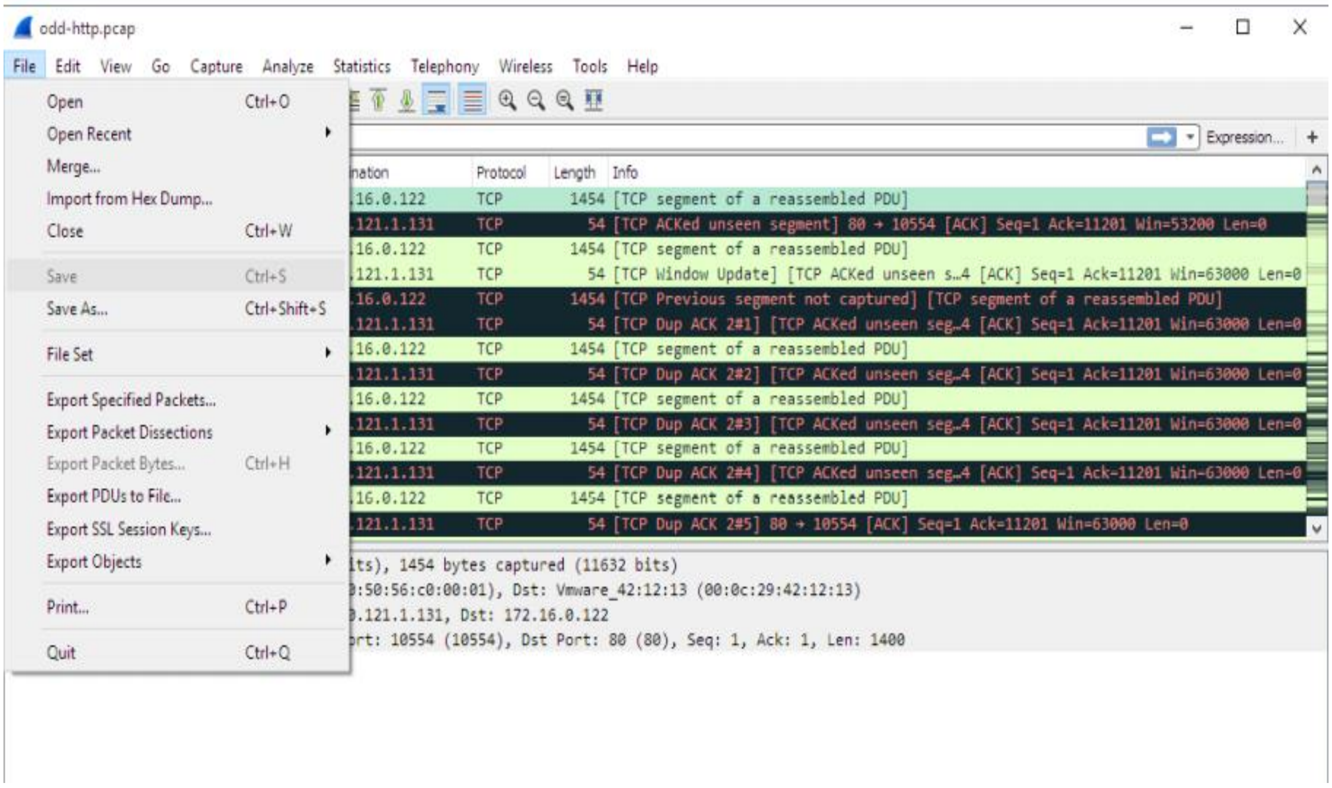
◆ Analyze This menu contains items to manipulate display filters, enable or disable the dissection of protocols, configure user specified decodes and follow a TCP stream.

◆ Statistics This menu contains items to display various statistic windows, including a summary of the packets that have been captured, display protocol hierarchy statistics and much more.

◆ Telephony This menu contains items to display various telephony related statistic windows, including a media analysis, flow diagrams, display protocol hierarchy statistics and much more.

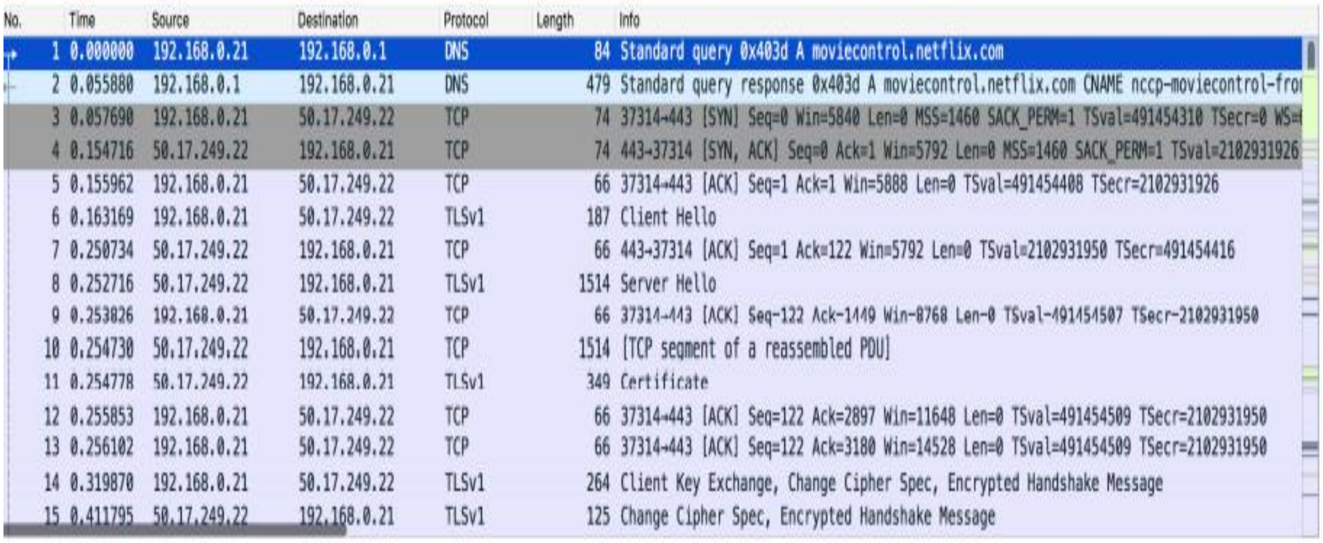
◆ Wireless This menu contains items to display Bluetooth and IEEE 802.11 wireless statistics. Tools This menu contains various tools available in Wireshark, such as creating Firewall ACL Rules.

◆ Help This menu contains items to help the user, e.g. access to some basic help, manual pages of the various command line tools, online access to some of the webpages, and the usual about dialog.



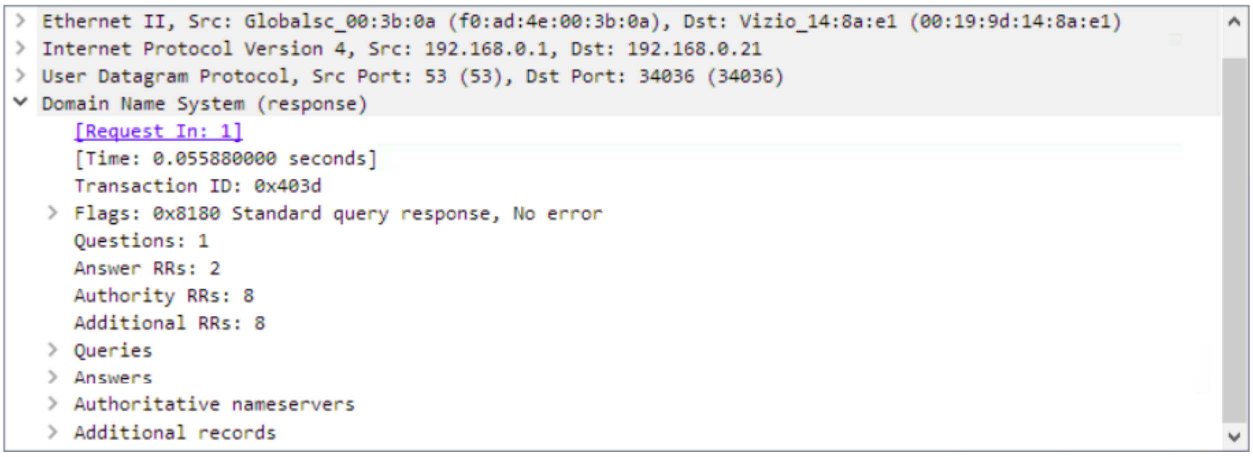
```
nation       Protocol  Length Info
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
.16.0.122    TCP       1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
.16.0.122    TCP       1454 [TCP segment of a reassembled PDU]
.121.1.131   TCP         54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0

its), 1454 bytes captured (11632 bits)
:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
.121.1.131, Dst: 172.16.0.122
rt: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
```

Packet-list pane:



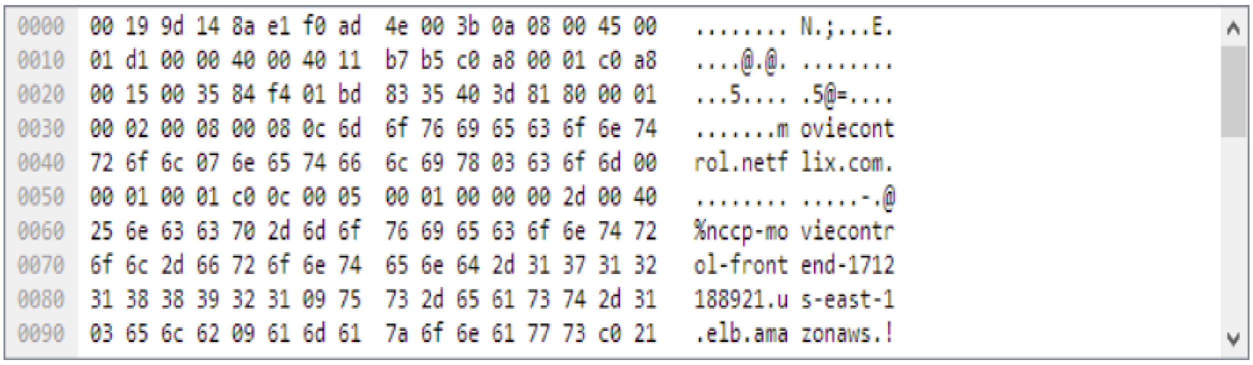
```
No.  Time      Source         Destination    Protocol  Length Info
  1 0.000000  192.168.0.21   192.168.0.1    DNS         84 Standard query 0x403d A moviecontrol.netflix.com
  2 0.055880  192.168.0.1    192.168.0.21   DNS        479 Standard query response 0x403d A moviecontrol.netflix.com CNAME nccp-moviecontrol-fro
  3 0.057690  192.168.0.21   50.17.249.22   TCP         74 37314→443 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491454310 TSecr=0 WS=
  4 0.154716  50.17.249.22   192.168.0.21   TCP         74 443→37314 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2102931926
  5 0.155962  192.168.0.21   50.17.249.22   TCP         66 37314→443 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491454408 TSecr=2102931926
  6 0.163169  192.168.0.21   50.17.249.22   TLSv1      187 Client Hello
  7 0.250734  50.17.249.22   192.168.0.21   TCP         66 443→37314 [ACK] Seq=1 Ack=122 Win=5792 Len=0 TSval=2102931950 TSecr=491454416
  8 0.252716  50.17.249.22   192.168.0.21   TLSv1     1514 Server Hello
  9 0.253826  192.168.0.21   50.17.249.22   TCP         66 37314→443 [ACK] Seq=122 Ack=1449 Win=8768 Len=0 TSval=491454507 TSecr=2102931950
 10 0.254730  50.17.249.22   192.168.0.21   TCP       1514 [TCP segment of a reassembled PDU]
 11 0.254778  50.17.249.22   192.168.0.21   TLSv1      349 Certificate
 12 0.255853  192.168.0.21   50.17.249.22   TCP         66 37314→443 [ACK] Seq=122 Ack=2897 Win=11648 Len=0 TSval=491454509 TSecr=2102931950
 13 0.256102  192.168.0.21   50.17.249.22   TCP         66 37314→443 [ACK] Seq=122 Ack=3180 Win=14528 Len=0 TSval=491454509 TSecr=2102931950
 14 0.319870  192.168.0.21   50.17.249.22   TLSv1      264 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
 15 0.411795  50.17.249.22   192.168.0.21   TLSv1      125 Change Cipher Spec, Encrypted Handshake Message
```

Packet-details pane:



```
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 1]
    [Time: 0.055880000 seconds]
    Transaction ID: 0x403d
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 8
    Additional RRs: 8
  > Queries
  > Answers
  > Authoritative nameservers
  > Additional records
```

Packet bytes pane:



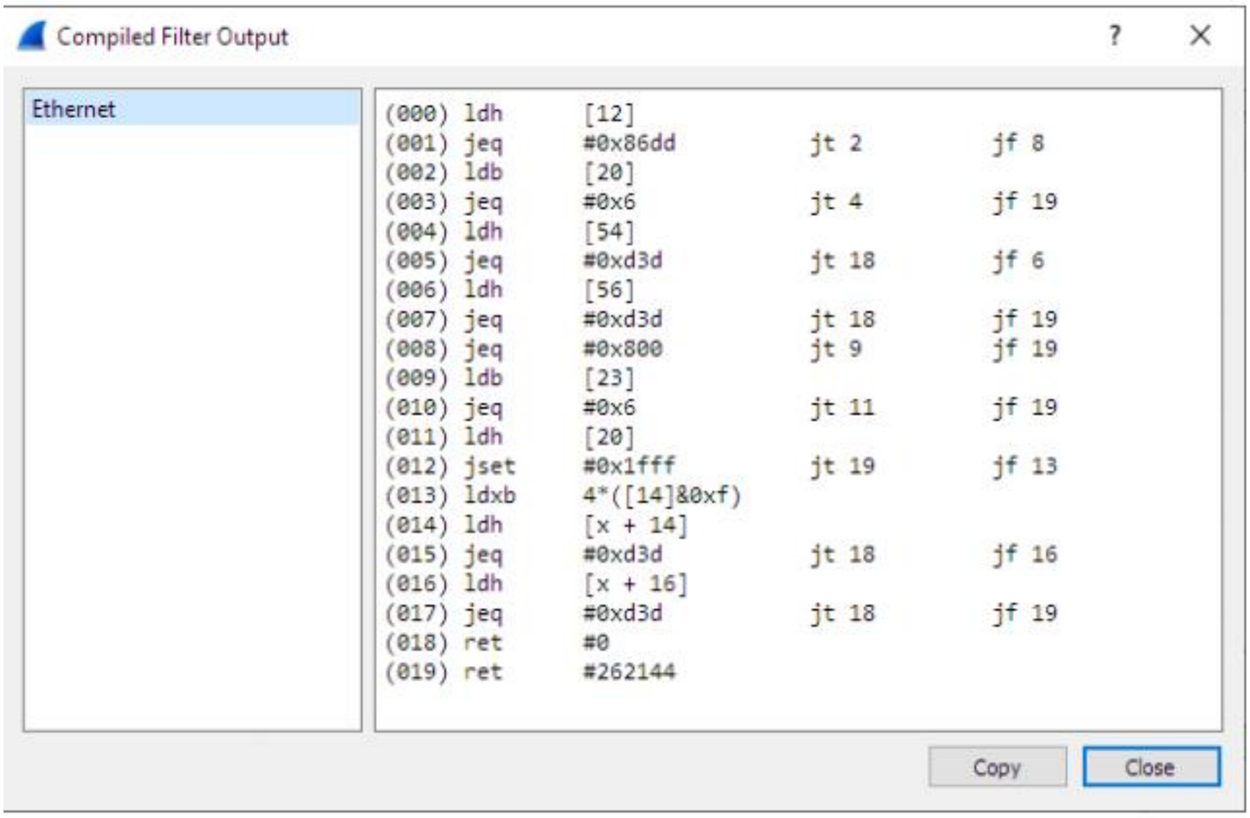
```
0000  00 19 9d 14 8a e1 f0 ad  4e 00 3b 0a 08 00 45 00   ........ N.;...E.
0010  01 d1 00 00 40 00 40 11  b7 b5 c0 a8 00 01 c0 a8   ....@.@. ........
0020  00 15 00 35 84 f4 01 bd  83 35 40 3d 81 80 00 01   ...5.... .5@=....
0030  00 02 00 08 00 08 0c 6d  6f 76 69 65 63 6f 6e 74   .......m oviecont
0040  72 6f 6c 07 6e 65 74 66  6c 69 78 03 63 6f 6d 00   rol.netf lix.com.
0050  00 01 00 01 c0 0c 00 05  00 01 00 00 00 2d 00 40   ........ .....-.@
0060  25 6e 63 63 70 2d 6d 6f  76 69 65 63 6f 6e 74 72   %nccp-mo viecontr
0070  6f 6c 2d 66 72 6f 6e 74  65 6e 64 2d 31 37 31 32   ol-front end-1712
0080  31 38 38 39 32 31 09 75  73 2d 65 61 73 74 2d 31   188921.u s-east-1
0090  03 65 6c 62 09 61 6d 61  7a 6f 6e 61 77 73 c0 21   .elb.ama zonaws.!
```

Capturing LIVE Network data:
Capturing live network data is one of the major features of Wireshark.

The Wireshark capture engine provides the following features:
• Capture from different kinds of network hardware such as Ethernet or 802.11.
• Simultaneously capture from multiple network interfaces.
• Stop the capture on different triggers such as the amount of captured data, elapsed time, or the number of packets. • Simultaneously show decoded packets while Wireshark is capturing.
• Filter packets, reducing the amount of data to be captured.
• Save packets in multiple files while doing a long term capture, optionally rotating through a fixed number of files (a “ringbuffer”).

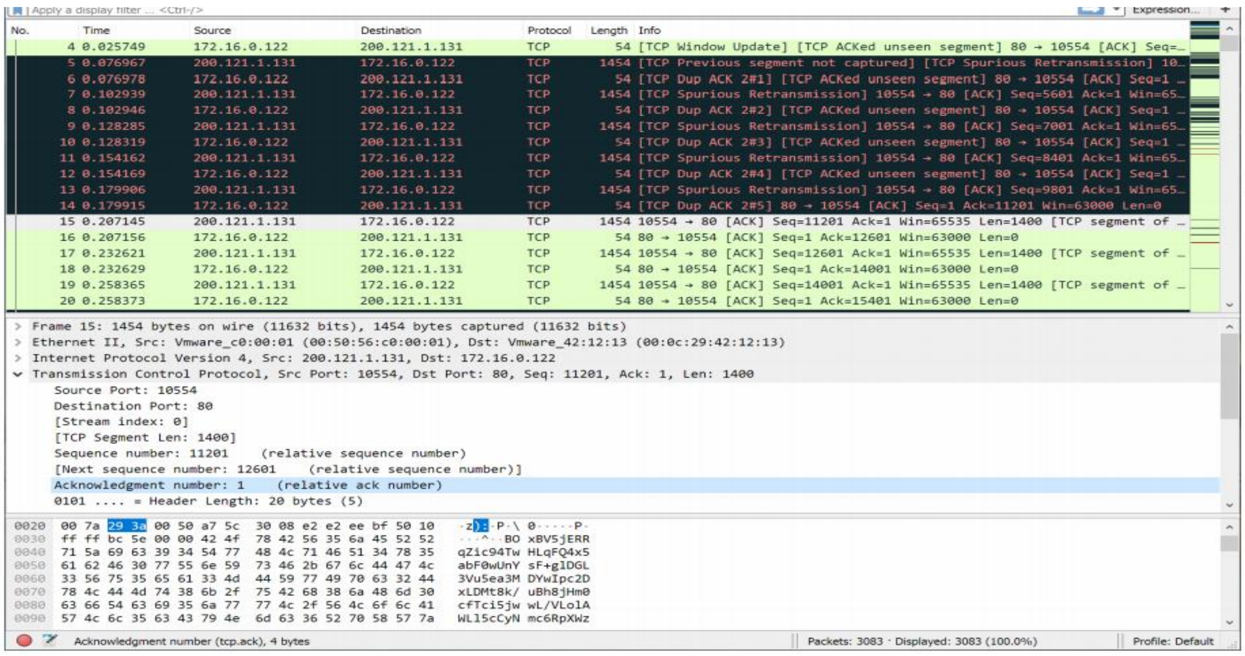
The capture engine still lacks the following features:
• Stop capturing (or perform some other action) depending on the captured data

Compiled Filter Output



```
(000) ldh      [12]
(001) jeq      #0x86dd          jt 2     jf 8
(002) ldb      [20]
(003) jeq      #0x6             jt 4     jf 19
(004) ldh      [54]
(005) jeq      #0xd3d           jt 18    jf 6
(006) ldh      [56]
(007) jeq      #0xd3d           jt 18    jf 19
(008) jeq      #0x800           jt 9     jf 19
(009) ldb      [23]
(010) jeq      #0x6             jt 11    jf 19
(011) ldh      [20]
(012) jset     #0x1fff          jt 19    jf 13
(013) ldxb     4*([14]&0xf)
(014) ldh      [x + 14]
(015) jeq      #0xd3d           jt 18    jf 16
(016) ldh      [x + 16]
(017) jeq      #0xd3d           jt 18    jf 19
(018) ret      #0
(019) ret      #262144
```

Viewing the Captured Packets:

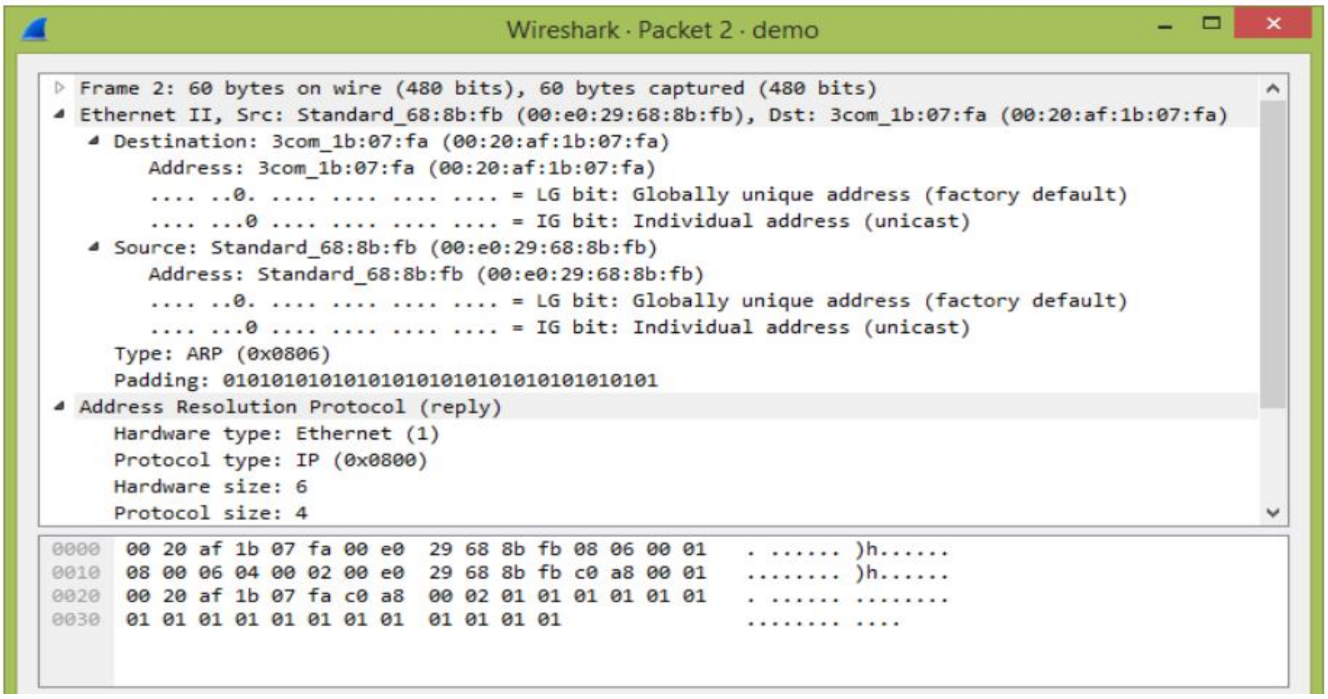


```
No.  Time       Source         Destination    Protocol Length Info
   4 0.025749   172.16.0.122   200.121.1.131  TCP        54 [TCP Window Update] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=…
   5 0.076967   200.121.1.131  172.16.0.122   TCP      1454 [TCP Previous segment not captured] [TCP Spurious Retransmission] 10…
   6 0.076978   172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#1] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
   7 0.102939   200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=5601 Ack=1 Win=65…
   8 0.102946   172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#2] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
   9 0.128285   200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=7001 Ack=1 Win=65…
  10 0.128319   172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#3] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
  11 0.154162   200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=8401 Ack=1 Win=65…
  12 0.154169   172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#4] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
  13 0.179906   200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=9801 Ack=1 Win=65…
  14 0.179915   172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
  15 0.207145   200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=11201 Ack=1 Win=65535 Len=1400 [TCP segment of …
  16 0.207156   172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=12601 Win=63000 Len=0
  17 0.232621   200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=12601 Ack=1 Win=65535 Len=1400 [TCP segment of …
  18 0.232629   172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=14001 Win=63000 Len=0
  19 0.258365   200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=14001 Ack=1 Win=65535 Len=1400 [TCP segment of …
  20 0.258373   172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=15401 Win=63000 Len=0
```

```
> Frame 15: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
> Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
> Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
v Transmission Control Protocol, Src Port: 10554, Dst Port: 80, Seq: 11201, Ack: 1, Len: 1400
    Source Port: 10554
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 1400]
    Sequence number: 11201    (relative sequence number)
    [Next sequence number: 12601    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
```

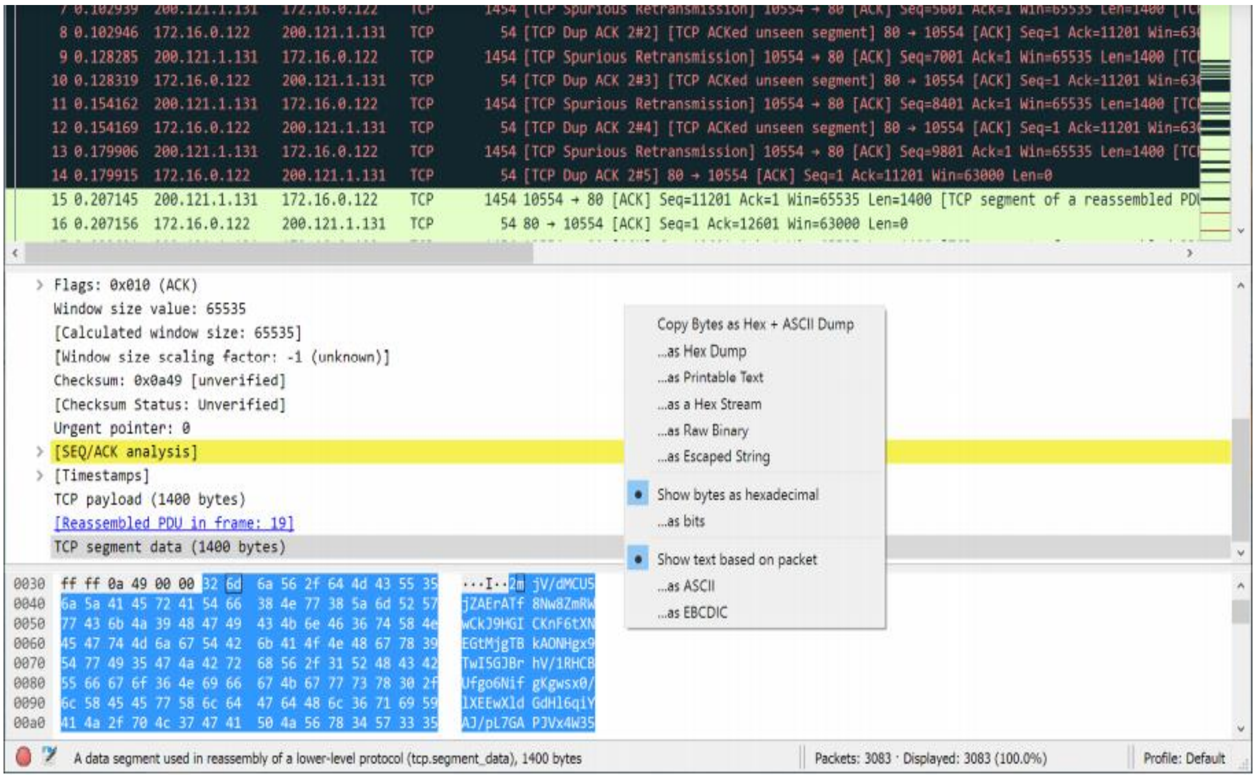
```
0020  00 7a 29 3a 00 50 a7 5c  30 08 e2 e2 ee bf 50 10   ·z):·P·\ 0·····P·
0030  ff ff bc 5e 00 00 42 4f  78 42 56 35 6a 45 52 52   ···^··BO xBV5jERR
0040  71 5a 69 63 39 34 54 77  48 4c 71 46 51 34 78 35   qZic94Tw HLqFQ4x5
0050  61 62 46 30 77 55 6e 59  73 46 2b 67 6c 44 47 4c   abF0wUnY sF+glDGL
0060  33 56 75 35 65 61 33 4d  44 59 77 49 70 63 32 44   3Vu5ea3M DYwIpc2D
0070  78 4c 44 4d 74 38 6b 2f  75 42 68 38 6a 48 6d 30   xLDMt8k/ uBh8jHm0
0080  63 66 54 63 69 35 6a 77  77 4c 2f 56 4c 6f 6c 41   cfTci5jw wL/VLolA
0090  57 4c 6c 35 63 43 79 4e  6d 63 36 52 70 58 57 7a   WLl5cCyN mc6RpXWz
```

Acknowledgment number (tcp.ack), 4 bytes | Packets: 3083 · Displayed: 3083 (100.0%) | Profile: Default



```
Wireshark · Packet 2 · demo

▷ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
◢ Ethernet II, Src: Standard_68:8b:fb (00:e0:29:68:8b:fb), Dst: 3com_1b:07:fa (00:20:af:1b:07:fa)
   ◢ Destination: 3com_1b:07:fa (00:20:af:1b:07:fa)
        Address: 3com_1b:07:fa (00:20:af:1b:07:fa)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
   ◢ Source: Standard_68:8b:fb (00:e0:29:68:8b:fb)
        Address: Standard_68:8b:fb (00:e0:29:68:8b:fb)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
     Type: ARP (0x0806)
     Padding: 010101010101010101010101010101010101
◢ Address Resolution Protocol (reply)
     Hardware type: Ethernet (1)
     Protocol type: IP (0x0800)
     Hardware size: 6
     Protocol size: 4

0000  00 20 af 1b 07 fa 00 e0  29 68 8b fb 08 06 00 01   . ...... )h......
0010  08 00 06 04 00 02 00 e0  29 68 8b fb c0 a8 00 01   ........ )h......
0020  00 20 af 1b 07 fa c0 a8  00 02 01 01 01 01 01 01   . ...... ........
0030  01 01 01 01 01 01 01 01  01 01 01 01               ........ ....
```

Applying Filters:



```
  7 0.102939  200.121.1.131  172.16.0.122   TCP   1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=5601 Ack=1 Win=65535 Len=1400 [TC
  8 0.102946  172.16.0.122   200.121.1.131  TCP     54 [TCP Dup ACK 2#2] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63
  9 0.128285  200.121.1.131  172.16.0.122   TCP   1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=7001 Ack=1 Win=65535 Len=1400 [TC
 10 0.128319  172.16.0.122   200.121.1.131  TCP     54 [TCP Dup ACK 2#3] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63
 11 0.154162  200.121.1.131  172.16.0.122   TCP   1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=8401 Ack=1 Win=65535 Len=1400 [TC
 12 0.154169  172.16.0.122   200.121.1.131  TCP     54 [TCP Dup ACK 2#4] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63
 13 0.179906  200.121.1.131  172.16.0.122   TCP   1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=9801 Ack=1 Win=65535 Len=1400 [TC
 14 0.179915  172.16.0.122   200.121.1.131  TCP     54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
 15 0.207145  200.121.1.131  172.16.0.122   TCP   1454 10554 → 80 [ACK] Seq=11201 Ack=1 Win=65535 Len=1400 [TCP segment of a reassembled PD
 16 0.207156  172.16.0.122   200.121.1.131  TCP     54 80 → 10554 [ACK] Seq=1 Ack=12601 Win=63000 Len=0

  > Flags: 0x010 (ACK)
    Window size value: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x0a49 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (1400 bytes)
    [Reassembled PDU in frame: 19]
    TCP segment data (1400 bytes)

0030  ff ff 0a 49 00 00 32 6d  6a 56 2f 64 4d 43 55 35   ···I··2m jV/dMCU5
0040  6a 5a 41 45 72 41 54 66  38 4e 77 38 5a 6d 52 57   jZAErATf 8Nw8ZmRW
0050  77 43 6b 4a 39 48 47 49  43 4b 6e 46 36 74 58 4e   wCkJ9HGI CKnF6tXN
0060  45 47 74 4d 6a 67 54 42  6b 41 4f 4e 48 67 78 39   EGtMjgTB kAONHgx9
0070  54 77 49 35 47 4a 42 72  68 56 2f 31 52 48 43 42   TwI5GJBr hV/1RHCB
0080  55 66 67 6f 36 4e 69 66  67 4b 67 77 73 78 30 2f   Ufgo6Nif gKgwsx0/
0090  6c 58 45 45 77 58 6c 64  47 64 48 6c 36 71 69 59   lXEEwXld GdHl6qiY
00a0  41 4a 2f 70 4c 37 47 41  50 4a 56 78 34 57 33 35   AJ/pL7GA PJVx4W35

A data segment used in reassembly of a lower-level protocol (tcp.segment_data), 1400 bytes   Packets: 3083 · Displayed: 3083 (100.0%)   Profile: Default
```

Display Filter Fields:

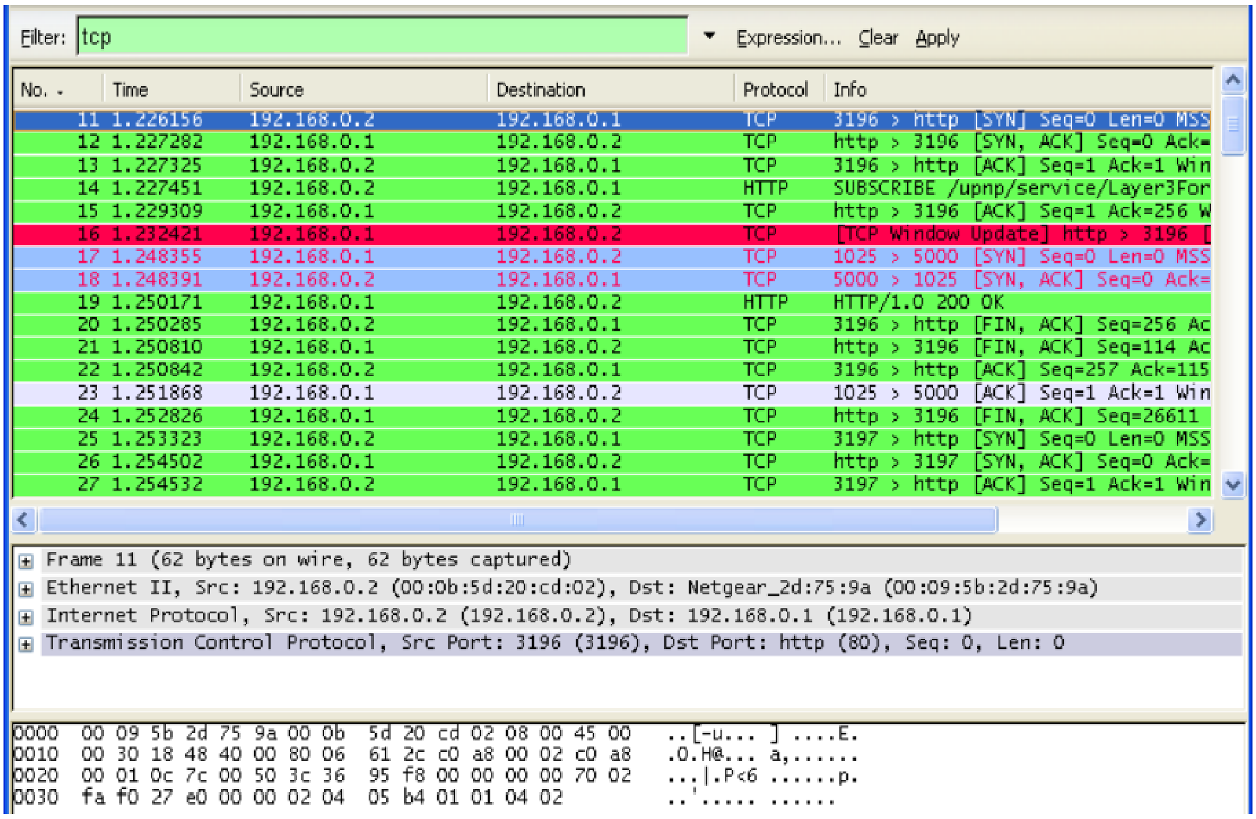
The simplest display filter is one that displays a single protocol.

- ◆ o only display packets containing a particular protocol, type the protocol into Wireshark's display filter toolbar.
- ◆ or example, to only display TCP packets, type tcp into Wireshark's display filter toolbar. Similarly, to only display packets containing a particular field, type the field into Wireshark's display filter toolbar.
- ◆ or example, to only display HTTP requests, type http.request into Wireshark's display filter toolbar.
- ◆ ou can filter on any protocol that Wireshark supports. You can also filter on any field that a dissector adds to the tree view, if the dissector has added an abbreviation for that field.
- ◆ full list of the available protocols and fields is available through the menu item View › Internals › Supported Protocols.

Comparing Values:

- ▪ You can build display filters that compare values using a number of different comparison operators.
- ▪ For example, to only display packets to or from the IP address 192.168.0.1, use ip.addr==192.168.0.1.

Filtering TCP packets while Viewing:



```
No. .  Time      Source        Destination   Protocol Info
   11 1.226156  192.168.0.2   192.168.0.1   TCP      3196 > http [SYN] Seq=0 Len=0 MSS
   12 1.227282  192.168.0.1   192.168.0.2   TCP      http > 3196 [SYN, ACK] Seq=0 Ack=
   13 1.227325  192.168.0.2   192.168.0.1   TCP      3196 > http [ACK] Seq=1 Ack=1 Win
   14 1.227451  192.168.0.2   192.168.0.1   HTTP     SUBSCRIBE /upnp/service/Layer3For
   15 1.229309  192.168.0.1   192.168.0.2   TCP      http > 3196 [ACK] Seq=1 Ack=256 W
   16 1.232421  192.168.0.1   192.168.0.2   TCP      [TCP Window Update] http > 3196 [
   17 1.248355  192.168.0.1   192.168.0.2   TCP      1025 > 5000 [SYN] Seq=0 Len=0 MSS
   18 1.248391  192.168.0.2   192.168.0.1   TCP      5000 > 1025 [SYN, ACK] Seq=0 Ack=
   19 1.250171  192.168.0.1   192.168.0.2   HTTP     HTTP/1.0 200 OK
   20 1.250285  192.168.0.2   192.168.0.1   TCP      3196 > http [FIN, ACK] Seq=256 Ac
   21 1.250810  192.168.0.1   192.168.0.2   TCP      http > 3196 [FIN, ACK] Seq=114 Ac
   22 1.250842  192.168.0.2   192.168.0.1   TCP      3196 > http [ACK] Seq=257 Ack=115
   23 1.251868  192.168.0.1   192.168.0.2   TCP      1025 > 5000 [ACK] Seq=1 Ack=1 Win
   24 1.252826  192.168.0.1   192.168.0.2   TCP      http > 3196 [FIN, ACK] Seq=26611
   25 1.253323  192.168.0.2   192.168.0.1   TCP      3197 > http [SYN] Seq=0 Len=0 MSS
   26 1.254502  192.168.0.1   192.168.0.2   TCP      http > 3197 [SYN, ACK] Seq=0 Ack=
   27 1.254532  192.168.0.2   192.168.0.1   TCP      3197 > http [ACK] Seq=1 Ack=1 Win

⊞ Frame 11 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
⊞ Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
⊞ Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0

0000  00 09 5b 2d 75 9a 00 0b  5d 20 cd 02 08 00 45 00   ..[-u... ] ....E.
0010  00 30 18 48 40 00 80 06  61 2c c0 a8 00 02 c0 a8   .0.H@... a,......
0020  00 01 0c 7c 00 50 3c 36  95 f8 00 00 00 00 70 02   ...|.P<6 ......p.
0030  fa f0 27 e0 00 00 02 04  05 b4 01 01 04 02         ..'..... ......
```

8. Test cases:

- Before reporting any problems, please make sure you have installed the latest version of Wireshark and you have good internet connection.
- **Mismatch in comparing values for displaying filter fields.**
- **For example, expression wrong** ip.addr != 1.2.3.4
- **Correct expression** !(ip.addr == 1.2.3.4)

9.Practical Related Questions:

- **Switch off the promiscuous mode from the capture options window and observe whether you are still able to receive packets from other devices or not.**
- **Which Wireshark Filter Can Be Used To Check All Incoming Requests To A Http Web Server?**
  - **HTTP web servers use TCP port 80. Incoming requests to the web server would have the destination port number as 80. So the filter tcp.dstport==80.**

10 .Exercise Questions :

I) **Exercise One :**

II) Open “Wireshark”, then use the “File” menu and the “Open” command to open the file “Exercise One.pcap”. You should see 26 packets listed. This set of packets describes a ‘conversation’ between a user’s client and a central server. This entire conversation happens automatically, after a user types something and hits enter.

III) Look at the packets to answer the following questions in relation to this conversation. In answering the following questions, use brief descriptions.

IV) For example, “In frame X, the client requests a web page, and in frame Y, the server delivers the content of the page.”

a. What is the IP address of the client that initiates the conversation?

b. Use the first two packets to identify the server that is going to be contacted. List the common name, and three IP addresses that can be used for the serve

V) Exercise Two:

VI) Open “Wireshark”, then use the “File” menu and the “Open” command to open the file “Exercise Two.pcap”. You should see 176 packets listed.

a. In the first few packets, the client machine is looking up the common name (cname) of a web site to find its IP address. What is the cname of this web site? Give two IP addresses for this web site.

b. How many packets/frames does it take to receive the web page (the answer to the first http get request only)?

PRACTICAL 1c:

1c. Practical significance of SNORT:

Snort is the foremost **Open Source Intrusion Prevention System (IPS)** in the world. ...

Snort has three primary uses:

- As a packet sniffer like tcpdump,
- as a packet logger — which is useful for network traffic debugging, or
- it can be used as a full-blown network intrusion prevention system.

Snort can be configured to run in three modes:

• Sniffer mode, which simply reads the packets off of the network and displays them for you in a continuous stream on the console (screen).

• Packet Logger mode, which logs the packets to disk.

• Network Intrusion Detection System (NIDS) mode, which performs detection and analysis on network traffic. This is the most complex and configurable mode.

2. Relevant Program Outcomes:

PO-1, 2, 3, 5, 9, 11

3. Competency and practical skills:

This practical is expected to develop the following skills:

c. **Practical skill:**

(i) Ability to work with SNORT to capture the network packets and analyze them.

(ii) To read traces and learn how to write new snort rules.

d. **Competency skill:**

(iii) Configuration of SNORT tool in 3 modes.

(iv) Troubleshoot network problems and setting up of the Firewall rulesets.

4. Pre-requisites :

- ♦ **Knowledge of Networking and packet tracing using Wireshark.**
- ♦ **Technical understanding of TCP/IP networking and network architecture.**
- ♦ **Basic familiarity with firewall and IPS concepts.**
- ♦ **Good understanding of IDS and IPS concepts.**
- ♦ **Good knowledge of SNORT CONFIG file.**

5. Resources required:

SNORT 40GB hard disk with minimum of 4 G HZ RAM

.No	e of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 40GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with) 5. SNORT runs on most UNIX and UNIX-like platforms including Linux and most BSD variants with VM WARE setup. 6. A supported network card for capturing-Ethernet, IEEE 802.11 7. Access to a machine with Wireshark and Snort installed. You are best to do this on a nal machine, or even on a VM. 8. A wired network jack.
2	Installation of SNORT-IPS tool.

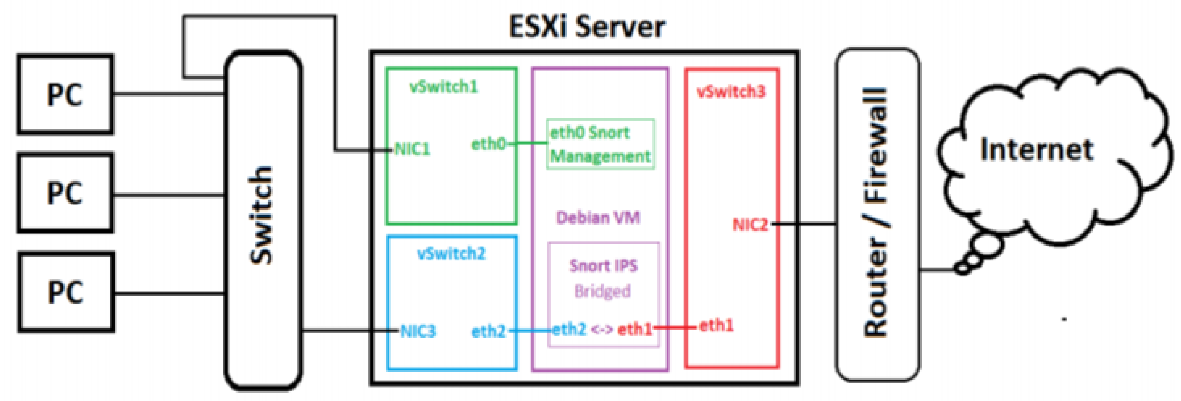
6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard; mouse and monitor are properly working with
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions will help in preventing injuries to people and damage to computer equipment.

5. A safe work place should be clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities at the workplace. Power supplies and monitors contain high voltage, handle with care.
7. Ensure that IPS tool SNORT is installed and Configured in the system before usage.
8. Ensure the availability of the Internet for the successful execution of these experiments.

7. Algorithm/circuit/Diagram/Description with Sample outputs:

1) SNORT-IPS setup overview:



2) Snort Installation

1. Install Snort · cd /usr/src · wget https://www.snort.org/downloads/snort/snort-2.9.7.0.tar.gz · tar -zxf snort-2.9.7.0.tar.gz && cd snort-2.9.7.0 · ./configure --enable-sourcefire && make && make install

2. Create Snort directories: · mkdir /usr/local/etc/snort · mkdir /usr/local/etc/snort/rules · mkdir /var/log/snort · mkdir /usr/local/lib/snort_dynamicrules

3. Create empty rules files: · touch /usr/local/etc/snort/rules/white_list.rules · touch /usr/local/etc/snort/rules/black_list.rules · touch /usr/local/etc/snort/rules/local.rules · touch /usr/local/etc/snort/rules/snort.rules · touch /usr/local/etc/snort/sid-msg.map

4. Create snort user and grant privileges: · groupadd snort && useradd -g snort snort · chown snort:snort /var/log/snort

5. Copy snort configuration files: Chapter: PulledPork Installation 6 · cp /usr/src/snort-2.9.7.0/etc/*.conf* /usr/local/etc/snort · cp /usr/src/snort-2.9.7.0/etc/*.map /usr/local/etc/snort

6. Configure Snort (edit snort.conf)

· vim /usr/local/etc/snort/snort.conf · Line #45 - ipvar HOME_NET 172.26.12.0/22 – make this match your internal network; · Line #48 - ipvar EXTERNAL_NET !$HOME_NET · Line #104 - var RULE_PATH rules · Line #109 - var WHITE_LIST_PATH rules · Line #110 - var BLACK_LIST_PATH rules · Line #293 - add this to the end after "decompress_depth 65535" max_gzip_mem 104857600 · Line #521 - add this line - output unified2: filename snort.log, limit 128 · Line #543 - delete or comment out all of the "include $RULE_PATH" lines except: ¬ include $RULE_PATH/local.rules ¬ include $RULE_PATH/snort.rules – add after local.rules

7. Make sure at line #265 the following rules are uncommented: · preprocessor normalize_ip4 · preprocessor normalize_tcp: ips ecn stream · preprocessor normalize_icmp4 · preprocessor normalize_ip6 · preprocessor normalize_icmp6

8. On line #188 at the end of step #2 of snort.cong add: · config policy_mode:inline

9. Configure daq at line #159 in snort.cong · config daq: afpacket · config daq_dir: /usr/local/lib/daq · config daq_mode: inline · config daq_var: buffer_size_mb=1024

10. Save changes to snort.conf

3) Using Snort

1. Snort in inline mode no database logging (console alerts): · snort –d –A console –u snort –g snort –c /usr/local/etc/snort/snort.conf –i eth1:eth2 –Q

2. Snort in inline mode no database logging (Logged alerts): · snort –d –A full –u snort –g snort –c /usr/local/etc/snort/snort.conf –i eth1:eth2 –Q

3. Snort in inline mode and database logging (console alerts): · snort –d –A console –u snort –g snort –c /usr/local/etc/snort/snort.conf –i eth1:eth2 –Q & /usr/local/bin/barnyard2 –c /usr/local/etc/snort/barnyard2.conf –d /var/log/snort –f snort.log –w /usr/local/etc/snort/bylog.waldo –C /usr/local/etc/snort/classification.config

4. Snort in inline mode and database logging (Logged alerts): · snort –d –A full –u snort –g snort –c /usr/local/etc/snort/snort.conf –i eth1:eth2 –Q & /usr/local/bin/barnyard2 –c /usr/local/etc/snort/barnyard2.conf –d /var/log/snort –f snort.log –w /usr/local/etc/snort/bylog.waldo –C /usr/local/etc/snort/classification.config

5. Snort in inline mode no database logging no snort status (console alerts):

```
· snort -q -d -A console -u snort -g snort -c /usr/local/etc/snort/snort.conf -i eth1:eth2 -Q
```

4) Packet Acquisition

a) Actions, Limits, and Verdicts Action and verdict counts:

show what Snort did with the packets it analyzed. This information is only output in IDS mode (when snort is run with the -c option).

• Alerts is the number of alert, and block actions processed as determined by the rule actions. Here block includes block, drop, and reject actions. Limits arise due to real world constraints on processing time and available memory. These indicate potential actions that did not happen:

• Match Limit counts rule matches were not processed due to the config detection: max queue events setting. The default is 5.

• Queue Limit counts events couldn't be stored in the event queue due to the config event queue: max queue setting. The default is 8.

• Log Limit counts events were not alerted due to the config event queue: log setting. The default is 3. • Event Limit counts events not alerted due to event filter limits.

• Alert Limit counts events were not alerted because they already were triggered on the session. Verdicts are rendered by Snort on each packet: 21

• Allow = packets Snort analyzed and did not take action on.

• Block = packets Snort did not forward, e.g. due to a block rule. "Block" is used instead of "Drop" to avoid confusion between dropped packets (those Snort didn't actually see) and blocked packets (those Snort did not allow to pass).

• Replace = packets Snort modified, for example, due to normalization or replace rules. This can only happen in inline mode with a compatible DAQ.

• Whitelist = packets that caused Snort to allow a flow to pass w/o inspection by any analysis program. Like blacklist, this is done by the DAQ or by Snort on subsequent packets.

• Blacklist = packets that caused Snort to block a flow from passing. This is the case when a block TCP rule fires. If the DAQ supports this in hardware, no further packets will be seen by Snort for that session. If not, snort will block each packet and this count will be higher.

• Ignore = packets that caused Snort to allow a flow to pass w/o inspection by this instance of Snort. Like blacklist, this is done by the DAQ or by Snort on subsequent packets.

• Int Blklst = packets that are GTP, Teredo, 6in4 or 4in6 encapsulated that are being blocked. These packets could get the Blacklist verdict if config tunnel verdicts was set for the given protocol. Note that these counts are output only if non-zero. Also, this count is incremented on the first packet in the flow that alerts. The alerting packet and all following packets on the flow will be counted under Block.

• Int Whtlst = packets that are GTP, Teredo, 6in4 or 4in6 encapsulated that are being allowed. These packets could get the Whitelist verdict if config tunnel verdicts was set for the given protocol. Note that these counts are output only if non-zero. Also, this count is incremented for all packets on the flow starting with the alertingpacket.

Example:
===
===== Action Stats: Alerts: 0 (0.000%) Logged: 0 (0.000%) Passed: 0 (0.000%) Limits: Match: 0 Queue: 0 Log: 0 Event: 0 Alert: 0 Verdicts: Allow: 3716022 (100.000%) Block: 0 (0.000%) Replace: 0 (0.000%) Whitelist: 0 (0.000%) Blacklist: 0 (0.000%) Ignore: 0 (0.000%)

b) Setting up of RULE SETS:

set This keyword sets bits to group for a particular flow. When no group specified, set the default group. This keyword always returns true.

Syntax: flowbits:set,bits[,group]

Usage: flowbits:set,bit1,doc; flowbits:set,bit2&bit3,doc;

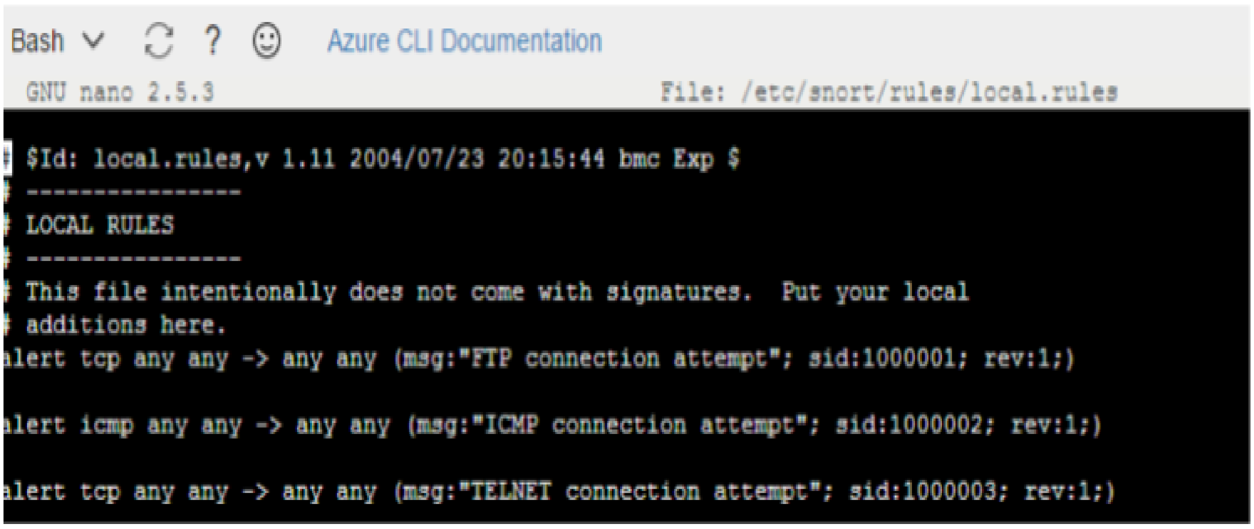
First rule sets bit1 in doc group, second rule sets bit2 and bit3 in doc group. So doc group has bit 1, bit2 and bit3 set

8. Test cases:

a) Test Case-1: Snort can be tested but before that few rules need to be created to test local data packets on the network. Then create some rules to test Snort.

- First, edit the local.rules file:
- nano /etc/snort/rules/local.rules

Creation of sample rules in **local.rules** file



```
GNU nano 2.5.3                          File: /etc/snort/rules/local.rules

# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert tcp any any -> any any (msg:"FTP connection attempt"; sid:1000001; rev:1;)

alert icmp any any -> any any (msg:"ICMP connection attempt"; sid:1000002; rev:1;)

alert tcp any any -> any any (msg:"TELNET connection attempt"; sid:1000003; rev:1;)
```

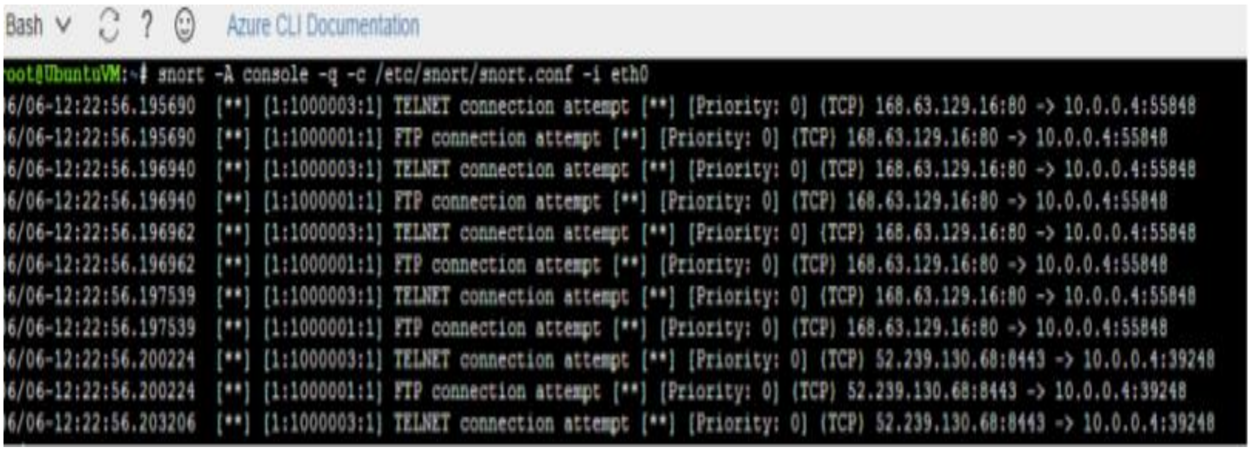
Save and close the file.

- The above rules will generate alerts when someone tries to Ping, FTP or Telnet to the server.

b) **Test Case-2:** Now, run Snort in NIDS mode and send alert output to the console:

- snort -A console -q -c /etc/snort/snort.conf

Generating **Alerts in Snort-NIDS mode**:



```
oot@UbuntuVM:~# snort -A console -q -c /etc/snort/snort.conf -i eth0
6/06-12:22:56.195690  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.195690  [**] [1:1000001:1] FTP connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.196940  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.196940  [**] [1:1000001:1] FTP connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.196962  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.196962  [**] [1:1000001:1] FTP connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.197539  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.197539  [**] [1:1000001:1] FTP connection attempt [**] [Priority: 0] {TCP} 168.63.129.16:80 -> 10.0.0.4:55848
6/06-12:22:56.200224  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 52.239.130.68:8443 -> 10.0.0.4:39248
6/06-12:22:56.200224  [**] [1:1000001:1] FTP connection attempt [**] [Priority: 0] {TCP} 52.239.130.68:8443 -> 10.0.0.4:39248
6/06-12:22:56.203206  [**] [1:1000003:1] TELNET connection attempt [**] [Priority: 0] {TCP} 52.239.130.68:8443 -> 10.0.0.4:39248
```

9.Practical Related Questions:

- Select a website that is not likely to be in your DNS or browser cache. The easiest way to do this is to select a site that you do not often visit. We need to create a tracefile. This tracefile will include possibly the ARP traces for the default gateway lookup, the DNS query and response for the hostname to IP address mapping, and the HTML content transferred over the http connection to the website in question. Select both an HTTP: URL as well as an HTTPS. URL to connect to.
- you also need to determine the IP address of your host.
- Make sure to save the packet capture to a file for later analysis.
- Once you have the packet capture file, you will need to write capture filters for the following.
  - Display all ARP traffic
  - Display all DNS traffic to/from your host
  - Display all HTTP and HTTPS traffic to/from your host

10.Exercise Questions:

Practical Exercises on SNORT:

- log tcp traffic from any port going to ports less than or equal to 6000
  - **log tcp any any -> 192.168.1.0/24 :6000**
- log tcp traffic from priveleged ports less than or equal to 1024 going to ports greater than or equal to 500
  - **log tcp any :1024 -> 192.168.1.0/24 500:**
- To match any IP address except the one indicated by the listed IP address OR give an ALERT message.
  - **alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)**
- For example, it is better to raise an alert on any traffic that originates outside of the local net with the negation operator indicated with a "!"

PRACTICAL 1d:

1d. Practical significance of TCP DUMP:

TCPDump is a very powerful tool because of its strength in capturing packets based on different parameters given. It operates on network layer, so it will be able to capture all packets in and out of the machine. We can use TCPDump to capture and save the packets to a file to analyse it later. TCPDump uses Libcap file for packet capturing.

Tcpdump is a command line utility that allows you to capture and analyze network traffic going through your system. It is often used to help troubleshoot network issues, as well as a security tool.

A powerful and versatile tool that includes many options and filters, tcpdump can be used in a variety of cases. Since it's a command line tool, it is ideal to run in remote servers or devices for which a GUI is not available, to collect data that can be analyzed later. It can also be launched in the background or as a scheduled job using tools like cron.

2. Relevant Program Outcomes:

PO-1, 2, 3, 5, 9, 11

3. Competency and practical skills:

This practical is expected to develop the following skills:

a. Practical skill:

(i) Ability to work with TCPDump to capture the network packets and analyze them.

(ii) Alert on multiple type of attacks in real-time and troubleshooting.

b. Competency skill:

(v) Deployment of TCPDump for looking up hostnames and ports.

(vi) Troubleshoot network problems and reverse map IPs to hostnames in the traffic it collects.
(vii)Alert on multiple type of attacks in real-time and troubleshooting.

4. Pre-requisites :

- ♦ **Knowledge of Networking and packet tracing using Wireshark.**
- ♦ **Technical understanding of TCP/IP networking and network architecture.**
- ♦ **Basic familiarity with firewall and IPS concepts.**
- ♦ **Good knowledge of TCPDump tool.**
- ♦ **Thorough knowledge of TCP/IP protocol stack, state transitions, handshakes, sequence flows, IP Header, UDP Header, TCP Header and the packet details.**

5. Resources required:

TCPDump 40GB hard disk with minimum of 4 G HZ RAM

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 40GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with) 5. TCPDump runs on most UNIX and UNIX-like platforms including Linux. and most BSD variants with VM WARE setup. 6. A supported network card for capturing-Ethernet, IEEE 802.11 7. Access to a machine with Wireshark and TCPDump installed. 8. A wired network jack.
2	Installation of Wireshark, TCPDump tool from RPM/ Installing TCPDump from source files.

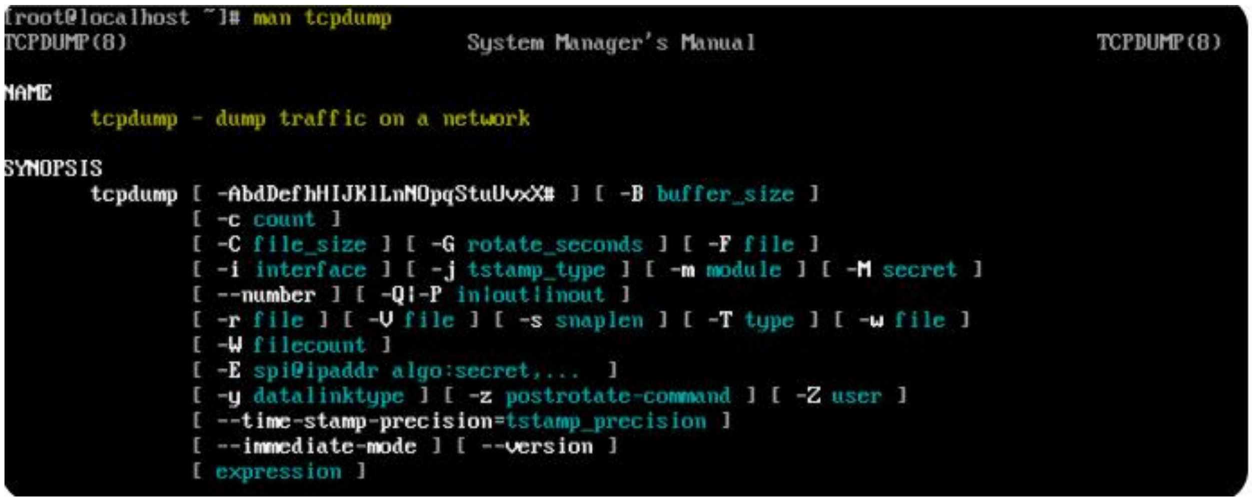
6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard; mouse and monitor are properly working with
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions will help in preventing injuries to people and damage to computer equipment.
5. A safe work place should be clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities at the workplace. Power supplies and monitors contain high voltage, handle with care.
7. Ensure that IPS tool TCPDump is installed and Configured in the system before usage.
8. Ensure the availability of the Internet with right network configuration that has interfaces for the successful execution of these experiments.
9. Make sure that the router that is plugged into is configured to receive the packets and is not in unicast mode.

7. Algorithm/circuit/Diagram/Description with Sample outputs:

- **TCPDump man page**
  - **tcpdump** [**-AbdDefhHIJKlLnNOpqStuUvxX#**] [**-B** *buffer_size*]

    [**-c** *count*] [**--count**] [**-C** *file_size*]
    [**-E** *spi@ipaddr* *algo:secret,...*]
    [**-F** *file*] [**-G** *rotate_seconds*] [**-i** *interface*]
    [**--immediate-mode**] [**-j** *tstamp_type*] [**-m** *module*]
    [**-M** *secret*] [**--number**] [**--print**] [**-Q** *in*|*out*|*inout*]
    [**-r** *file*] [**-s** *snaplen*] [**-T** *type*] [**--version**]
    [**-V** *file*] [**-w** *file*] [**-W** *filecount*] [**-y** *datalinktype*]
    [**-z** *postrotate-command*] [**-Z** *user*]
    [**--time-stamp-precision=***tstamp_precision*]
    [**--micro**] [**--nano**]
    [*expression*]

- 

```
[root@localhost ~]# man tcpdump
TCPDUMP(8)                     System Manager's Manual                    TCPDUMP(8)

NAME
       tcpdump - dump traffic on a network

SYNOPSIS
       tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]
               [ -c count ]
               [ -C file_size ] [ -G rotate_seconds ] [ -F file ]
               [ -i interface ] [ -j tstamp_type ] [ -m module ] [ -M secret ]
               [ --number ] [ -Q|-P in|out|inout ]
               [ -r file ] [ -V file ] [ -s snaplen ] [ -T type ] [ -w file ]
               [ -W filecount ]
               [ -E spi@ipaddr algo:secret,...  ]
               [ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]
               [ --time-stamp-precision=tstamp_precision ]
               [ --immediate-mode ] [ --version ]
               [ expression ]
```

Identifying the interface using TCPDump

- **tcpdump -i eth0**

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

This is an extremely verbose method of getting all types of traffic and can be difficult to process the number of packets that are being transmitted.

- [root@myvm ~]# **tcpdump -i lo**

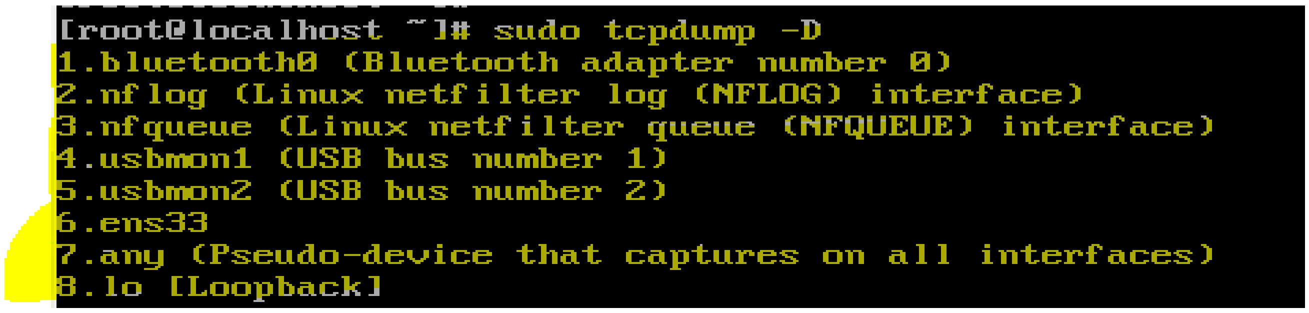
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes

Filtering packets using TCPDump

- To print all packets arriving at or departing from *sundown*:
- **tcpdump host sundown**
- To print traffic between *helios* and either *hot* or *ace*:
- **tcpdump host helios and \(hot or ace \)**
- To print all IP packets between *ace* and any host except *helios*:
- **tcpdump ip host ace and not helios**
- To print all traffic between local hosts and hosts at Berkeley:
- **tcpdump net ucb-ether**
- To print all ftp traffic through internet gateway *snup*: (note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):
- **tcpdump 'gateway snup and (port ftp or ftp-data)'**
- To print traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net).
- **tcpdump ip and not net** *localnet*
- To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.
- **tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net** *localnet*'
- To print the TCP packets with flags RST and ACK both set. (i.e. select only the RST and ACK flags in the flags field, and if the result is "RST and ACK both set", match)

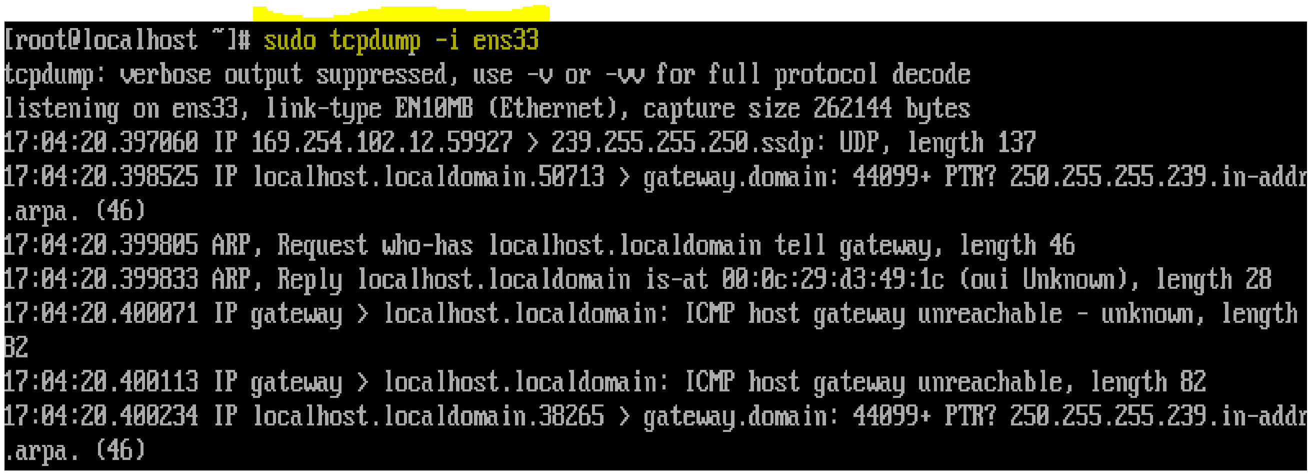
- **tcpdump 'tcp[tcpflags] & (tcp-rst|tcp-ack) == (tcp-rst|tcp-ack)'**
- To print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets. (IPv6 is left as an exercise for the reader.)
- **tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'**
- To print IP packets longer than 576 bytes sent through gateway *snup*:
- **tcpdump 'gateway snup and ip[2:2] > 576'**
- To print IP broadcast or multicast packets that were *not* sent via Ethernet broadcast or multicast:
- **tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'**
- To print all ICMP packets that are not echo requests/replies (i.e., not ping packets):
- **tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'**

- Checking the available interface for tcpdump:
  - **sudo tcpdump -D**



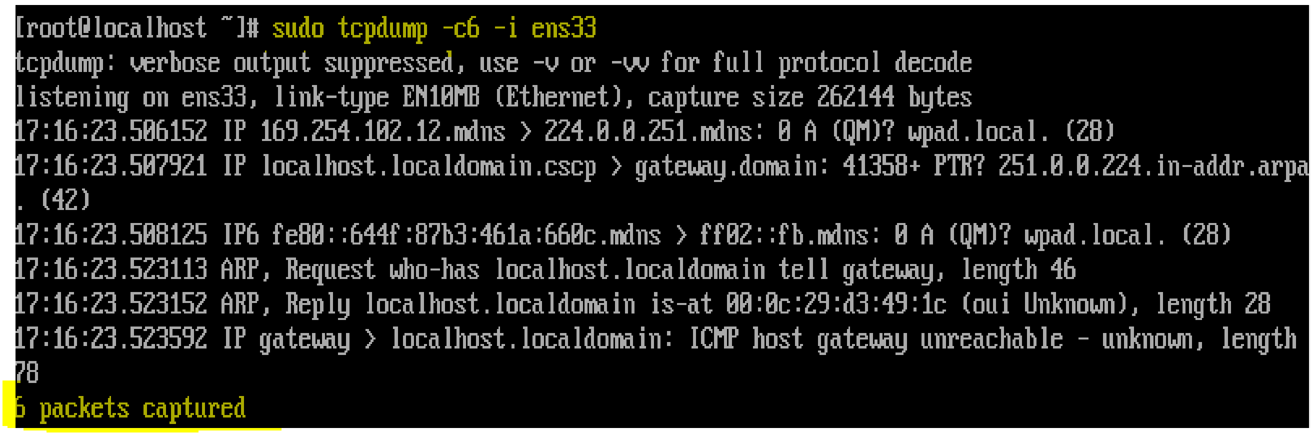
```
[root@localhost ~]# sudo tcpdump -D
1.bluetooth0 (Bluetooth adapter number 0)
2.nflog (Linux netfilter log (NFLOG) interface)
3.nfqueue (Linux netfilter queue (NFQUEUE) interface)
4.usbmon1 (USB bus number 1)
5.usbmon2 (USB bus number 2)
6.ens33
7.any (Pseudo-device that captures on all interfaces)
8.lo [Loopback]
```

- Capturing packets on a specific network interface
  - **sudo tcpdump -i <interface name>** in below image selected interface is ens33
  - **sudo tcpdump -i ens33**



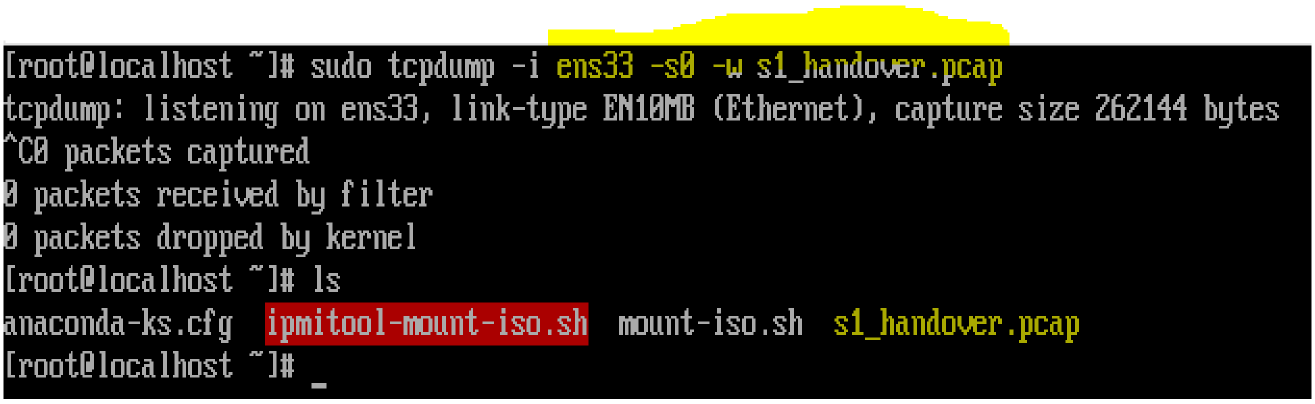
```
[root@localhost ~]# sudo tcpdump -i ens33
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
17:04:20.397060 IP 169.254.102.12.59927 > 239.255.255.250.ssdp: UDP, length 137
17:04:20.398525 IP localhost.localdomain.50713 > gateway.domain: 44099+ PTR? 250.255.255.239.in-addr
.arpa. (46)
17:04:20.399805 ARP, Request who-has localhost.localdomain tell gateway, length 46
17:04:20.399833 ARP, Reply localhost.localdomain is-at 00:0c:29:d3:49:1c (oui Unknown), length 28
17:04:20.400071 IP gateway > localhost.localdomain: ICMP host gateway unreachable - unknown, length
82
17:04:20.400113 IP gateway > localhost.localdomain: ICMP host gateway unreachable, length 82
17:04:20.400234 IP localhost.localdomain.38265 > gateway.domain: 44099+ PTR? 250.255.255.239.in-addr
.arpa. (46)
```

- Capturing limited numbers packets on a specific network interface
  - **sudo tcpdump -c <no. of packets> -i <interface name>**
  - **sudo tcpdump -c 6 -i ens33**



```
[root@localhost ~]# sudo tcpdump -c6 -i ens33
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
17:16:23.506152 IP 169.254.102.12.mdns > 224.0.0.251.mdns: 0 A (QM)? wpad.local. (28)
17:16:23.507921 IP localhost.localdomain.cscp > gateway.domain: 41358+ PTR? 251.0.0.224.in-addr.arpa
. (42)
17:16:23.508125 IP6 fe80::644f:87b3:461a:660c.mdns > ff02::fb.mdns: 0 A (QM)? wpad.local. (28)
17:16:23.523113 ARP, Request who-has localhost.localdomain tell gateway, length 46
17:16:23.523152 ARP, Reply localhost.localdomain is-at 00:0c:29:d3:49:1c (oui Unknown), length 28
17:16:23.523592 IP gateway > localhost.localdomain: ICMP host gateway unreachable - unknown, length
78
6 packets captured
```

- Capturing packets on a specific network interface and saving to a Wireshark file
  - **sudo tcpdump -i <interface name> -s0 -w < filename.pacp >**
  - **sudo tcpdump -i ens33 -s0 -w s1_handover.pcap**



```
[root@localhost ~]# sudo tcpdump -i ens33 -s0 -w s1_handover.pcap
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
^C0 packets captured
0 packets received by filter
0 packets dropped by kernel
[root@localhost ~]# ls
anaconda-ks.cfg  ipmitool-mount-iso.sh  mount-iso.sh  s1_handover.pcap
[root@localhost ~]# _
```

- Applying Filter with specific protocol
  - **sudo tcpdump -i <interface name> -s0 -w < filename.pacp > <protocol>**
  - **sudo tcpdump -i ens33 -s0 -w s1_setup.pcap sctp**
- Applying multi filter with source IP, specific port and destination IP
  - **sudo tcpdump -i src <ip address> port < port number> dst <ip address> -s0 -w < filename.pacp >**
  - **sudo tcpdump -i src 192.168.3.1 port 36422 dst 192.168.3.100 -s0 -w s1_setup.pacp**
- Displaying run time packets link level header in **HEX** and **ASCII** format
  - **sudo tcpdump -xx -i <interface name>**
  - **sudo tcpdump -xx -i ens33**
- **Filtering packets using TCPDump based on Protocol**
  - To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:
  - **$ sudo tcpdump -i any -c5 icmp**

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes

- In a different terminal, try to ping another machine:
  - **$ ping opensource.com**

```
PING opensource.com (54.204.39.132) 56(84) bytes of data.
64 bytes from ec2-54-204-39-132.compute-
1.amazonaws.com (54.204.39.132): icmp_seq=1 ttl=47 time=39.6 ms
```

Back in the tcpdump capture, notice that tcpdump captures and displays only the ICMP-related packets.

```
09:34:20.136766 IP rhel75 > ec2-54-204-39-132.compute-1.amazonaws.com:
ICMP echo request, id 20361, seq 1, length 64
09:34:20.176402 IP ec2-54-204-39-132.compute-1.amazonaws.com > rhel75:
ICMP echo reply, id 20361, seq 1, length 64
09:34:21.140230 IP rhel75 > ec2-54-204-39-132.compute-1.amazonaws.com:
ICMP echo request, id 20361, seq 2, length 64
09:34:21.180020 IP ec2-54-204-39-132.compute-1.amazonaws.com > rhel75:
ICMP echo reply, id 20361, seq 2, length 64
09:34:22.141777 IP rhel75 > ec2-54-204-39-132.compute-1.amazonaws.com:
ICMP echo request, id 20361, seq 3, length 64
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

- ## Filtering packets using TCPDump based on Port
  - To filter packets based on the desired service or port, use the port filter
  - **$ sudo tcpdump -i any -c5 -nn port 80**

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
09:58:28.790548 IP 192.168.122.98.39330 > 54.204.39.132.80:
Flags [S], seq 1745665159, win 29200, options [mss 1460,sackOK,TS
val 122599140 ecr 0,nop,wscale 7], length 0
09:58:28.834026 IP 54.204.39.132.80 > 192.168.122.98.39330:
Flags [S.], seq 4063583040, ack 1745665160, win 28960,
options [mss 1460,sackOK,TS val 522775728 ecr 122599140,nop,wscale 9], length 0
09:58:28.834093 IP 192.168.122.98.39330 > 54.204.39.132.80: Flags [.], ack 1,
win 229, options [nop,nop,TS val 122599183 ecr 522775728], length 0
09:58:28.834588 IP 192.168.122.98.39330 > 54.204.39.132.80:
Flags [P.], seq 1:113, ack 1, win 229, options [nop,nop,TS
val 122599184 ecr 522775728], length 112: HTTP: GET / HTTP/1.1
09:58:28.878445 IP 54.204.39.132.80 > 192.168.122.98.39330: Flags [.], ack 113,
win 57, options [nop,nop,TS val 522775739 ecr 122599184], length 0
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

- ## Filtering packets using TCPDump based on Source IP/hostname
  - You can also filter packets based on the source or destination IP Address or hostname.

$ sudo tcpdump -i any -c5 -nn src 192.168.122.98

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
```

```
10:02:15.220824 IP 192.168.122.98.39436 > 192.168.122.1.53: 59332+ A?
opensource.com. (32)
10:02:15.220862 IP 192.168.122.98.39436 > 192.168.122.1.53: 20749+ AAAA?
opensource.com. (32)
10:02:15.364062 IP 192.168.122.98.39334 > 54.204.39.132.80:
Flags [S], seq 1108640533, win 29200, options [mss 1460,sackOK,TS
val 122825713 ecr 0,nop,wscale 7], length 0
10:02:15.409229 IP 192.168.122.98.39334 > 54.204.39.132.80: Flags [.],
ack 669337581, win 229, options [nop,nop,TS val 122825758 ecr 522832372],
length 0
10:02:15.409667 IP 192.168.122.98.39334 > 54.204.39.132.80:
Flags [P.], seq 0:112, ack 1, win 229, options [nop,nop,TS
val 122825759 ecr 522832372], length 112: HTTP: GET / HTTP/1.1
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

8. Test cases:

1. **Test case-1**: from specific ip and destined for a specific port
   - find all traffic from 10.5.2.3 going to any host on port 3389.
   - **$ sudo tcpdump -nnvvS src 10.5.2.3 and dst port 3389**
2. **Test case-2**: Find DNS Traffic
   - **$ sudo tcpdump -vvAs0 port 53**

9.Practical Related Questions:

- Which command line parameter would you use to disable name resolution in tcpdump?

- When troubleshooting network issues, it is often easier to **use** the IP addresses and port numbers; **We can disable name resolution** by **using** the **option** -n and port **resolution** with -nn .

- How to find traffic using port ranges

- **$ sudo tcpdump portrange 21-23**

10. Exercise Questions:

Practical Exercises on TCPDump:

- **Exercise-1:** Ignoring packets
  - **$sudo tcpdump -i eth0 -n -c 5 'port !80'**
  - The output will be all the incoming network traffic other than port HTTP:80
- **Exercise-2:** Show only IP6 traffic
  - **$sudo tcpdump ip6**

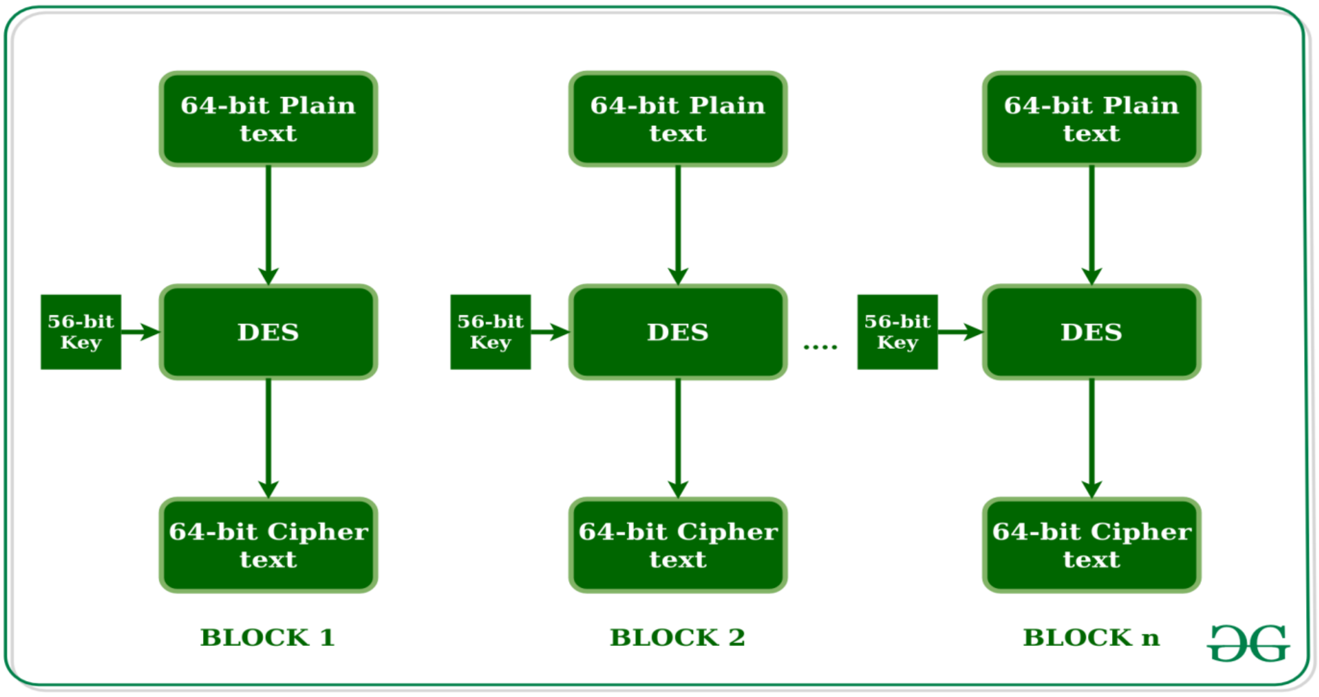
PRACTICAL 2: Implementation and Performance evaluation of

various cryptographic algorithms in C/C++ a)DES b)RSA

2a) Practical significance of DES:

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline.

DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.
We have mentioned that DES uses a 56 bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56 bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.



2. Relevant Program Outcomes: PO 1, PO 2, PO 3, PO 5, PO 9, PO 11

3. Competency and practical skills :

The practical is expected to develop the following skills
1. Ability to encrypt and decrypt the given text using DES algorithm.

4. Prerequisites:

1. Student should have knowledge on Computer Networks

2. Student should have knowledge on C Programming.

5. Resources required:

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 20GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	C compiler

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

7. **Algorithm/circuit/Diagram/Description:**
Algorithm for Encryption & Decryption:

1. In the first step, the 64 bit plain text block is handed over to an initial Permutation (IP) function.
2. The initial permutation performed on plain text.
3. Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
4. Now each LPT and RPT to go through 16 rounds of encryption process.
5. In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block

6. The result of this process produces 64 bit cipher text.

7. **Initial Permutation (IP) –**

As we have noted, the Initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as show in figure.
For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block and so on.
This is nothing but jugglery of bit positions of the original plain text block.

8. As we have noted after IP done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad level steps.

9. **Key transformation –**

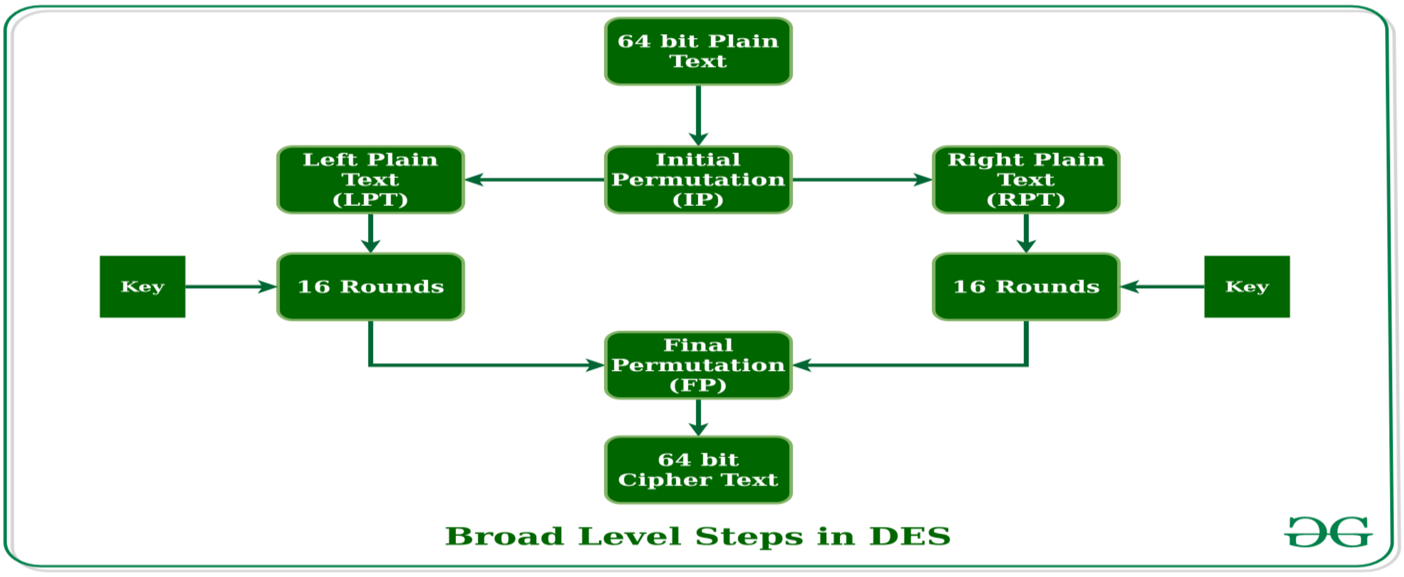
We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called as key transformation. For this the 56 bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.
For example, if the round number 1, 2, 9 or 16 the shift is done by only position for other rounds, the circular shift is done by two positions.

10. After an appropriate shift, 48 of the 56 bit are selected. for selecting 48 of the 56 bits the table show in figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub set of the original 56-bit key it is called Compression Permutation.

11. **Expansion Permutation –**

Recall that after initial permutation, we had two 32-bit plain text areas called as Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called as expansion permutation. This happens as the 32 bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4 bit block of the previous step is then expanded to a corresponding 6 bit block, i.e., per 4 bit block, 2 more bits are added.

12. This process results into expansion as well as permutation of the input bit while creating output.
Key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits. Now the 48-bit key is XOR with 48-bit RPT and resulting output is given to the next step, which is the S-Box substitution.



8. Test cases:

1. Outputs with different inputs must be given and appropriate results must be recorded after execution.

2. Students must also record the errors while executing the program experimenting with different inputs.

9. Sample output:

Encryption:

```
After initial permutation: 14A7D67818CA18AD
After splitting: L0=14A7D678 R0=18CA18AD

Round 1 18CA18AD 5A78E394 194CD072DE8C
Round 2 5A78E394 4A1210F6 4568581ABCCE
Round 3 4A1210F6 B8089591 06EDA4ACF5B5
Round 4 B8089591 236779C2 DA2D032B6EE3
Round 5 236779C2 A15A4B87 69A629FEC913
Round 6 A15A4B87 2E8F9C65 C1948E87475E
Round 7 2E8F9C65 A9FC20A3 708AD2DDB3C0
Round 8 A9FC20A3 308BEE97 34F822F0C66D
Round 9 308BEE97 10AF9D37 84BB4473DCCC
Round 10 10AF9D37 6CA6CB20 02765708B5BF
Round 11 6CA6CB20 FF3C485F 6D5560AF7CA5
Round 12 FF3C485F 22A5963B C2C1E96A4BF3
Round 13 22A5963B 387CCDAA 99C31397C91F
Round 14 387CCDAA BD2DD2AB 251B8BC717D0
Round 15 BD2DD2AB CF26B472 3330C5D9A36D
Round 16 19BA9212 CF26B472 181C5D75C66D

Cipher Text: C0B7A8D05F3A829C
```

Decryption:

```
After initial permutation: 19BA9212CF26B472
After splitting: L0=19BA9212 R0=CF26B472

Round 1 CF26B472 BD2DD2AB 181C5D75C66D
Round 2 BD2DD2AB 387CCDAA 3330C5D9A36D
Round 3 387CCDAA 22A5963B 251B8BC717D0
Round 4 22A5963B FF3C485F 99C31397C91F
Round 5 FF3C485F 6CA6CB20 C2C1E96A4BF3
Round 6 6CA6CB20 10AF9D37 6D5560AF7CA5
Round 7 10AF9D37 308BEE97 02765708B5BF
Round 8 308BEE97 A9FC20A3 84BB4473DCCC
Round 9 A9FC20A3 2E8F9C65 34F822F0C66D
Round 10 2E8F9C65 A15A4B87 708AD2DDB3C0
Round 11 A15A4B87 236779C2 C1948E87475E
Round 12 236779C2 B8089591 69A629FEC913
Round 13 B8089591 4A1210F6 DA2D032B6EE3
Round 14 4A1210F6 5A78E394 06EDA4ACF5B5
Round 15 5A78E394 18CA18AD 4568581ABCCE
Round 16 14A7D678 18CA18AD 194CD072DE8C

Plain Text: 123456ABCD132536
```

10.Practical Related Questions:

1.What is the main purpose of DES?
2. Can we give inputs in the form of Binary also?
3. What is the initial Key length taken?
4. To how many bits the key length of 64- bit key reduced ?
5. What is Block Cipher?
6. What is the best way to extract results either Binary or Hexadecimal?

11. Exercise Questions:

2b) RSA:

1. Practical significance of

RSA Algorithm is used to encrypt and decrypt data in modern computer systems and other electronic devices. RSA algorithm is an asymmetric cryptographic algorithm as it creates 2 different keys for the purpose of encryption and decryption. It is public key cryptography as one of the keys involved is made public. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman who first publicly described it in 1978.

2. Relevant Program Outcomes: PO 1, PO 2, PO 3, PO 5, PO 9, PO 11

3. Competency and practical skills :

The practical is expected to develop the following skills
1. Ability to encrypt and decrypt the given text using RSA algorithm.

4. Prerequisites:

1. Student should have knowledge on Computer Networks

2. Student should have knowledge on C Programming.

5. Resources required:

S.No	Name of the Resource Broad Specification(Approx.)

1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 20GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	C compiler

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

7. **Algorithm/circuit/Diagram/Description:**
Algorithm for Encryption & Decryption:

RSA involves use of public and private key for its operation. The keys are generated using the following steps:-

1. Two prime numbers are selected as p and q
2. n = pq which is the modulus of both the keys.
3. Calculate totient = (p-1)(q-1)
4. Choose e such that e > 1 and co-prime to totient which means gcd (e, totient) must be equal to 1, e is the public key
5. Choose d such that it satisfies the equation de = 1 + k (totient), d is the private key not known to everyone.
6. Cipher text is calculated using the equation c = m^e mod n where m is the message.
7. With the help of c and d we decrypt message using equation m = c^d mod n where d is the private key.

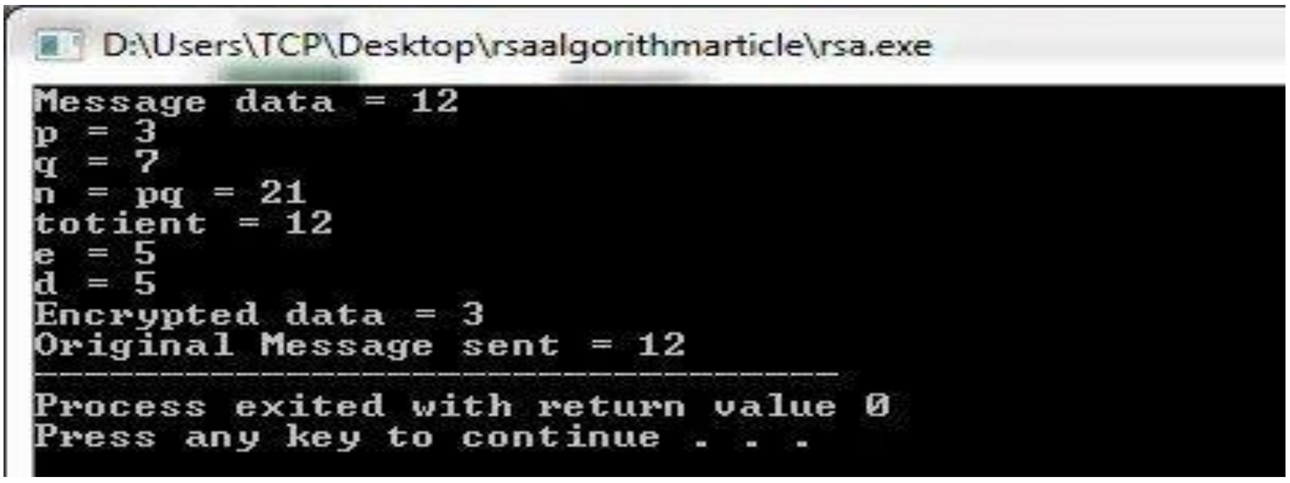
Note: If we take the two prime numbers very large it enhances security but requires implementation of Exponentiation by squaring algorithm and square and multiply algorithm for effective encryption and decryption. For simplicity the program is designed with relatively small prime numbers.

Test cases:

1. Outputs with different inputs must be given and appropriate results must be recorded after execution.

2. Students must also record the errors while executing the program experimenting with different inputs.

9. Sample output:



D:\Users\TCP\Desktop\rsaalgorithmarticle\rsa.exe

```
Message data = 12
p = 3
q = 7
n = pq = 21
totient = 12
e = 5
d = 5
Encrypted data = 3
Original Message sent = 12
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

10.Practical Related Questions:

1. What happens if large prime numbers are taken as input?
2. What happens when too small prime numbers are taken as input?
3. What is the formula for deriving n, e and d?
4. Is RSA block Cipher or stream Cipher?
5. What are the challenges faced by RSA?

11. Exercise Questions:

1. Identify the number of shift bits in key generation rounds of 1,2,6 and 9.

PRACTICAL 3: Using IP TABLES on Linux and setting the filtering rules

1. Practical significance of IP Tables:

IPTables is the name of a firewall system that operates through the command line on Linux. This program is mainly available as a default utility on **Ubuntu**. Administrators often use the IPTables firewall to allow or block traffic into their networks.

Iptables packet filtering mechanism is organized into three different kinds of structures: **tables**, **chains** and **targets**. Network traffic is made up of packets. Iptables identifies the packets received and then uses a set of rules to decide what to do with the packets that matches.

Network traffic is made up of packets. Data is broken up into smaller pieces (called packets), sent over a network, then put back together. Iptables identifies the packets received and then uses a set of rules to decide what to do with them.
IPTables filters packets based on:

- **Tables:** Tables are files that join similar actions. A table consists of several **chains**.
- **Chains:** A chain is a string of **rules**. When a packet is received, iptables finds the appropriate table, then runs it through the chain of **rules** until it finds a match.
- **Rules:** A rule is a statement that tells the system what to do with a packet. Rules can block one type of packet, or forward another type of packet. The outcome, where a packet is sent, is called a **target**.
- **Targets:** A target is a decision of what to do with a packet. Typically, this is to accept it, drop it, or reject it (which sends an error back to the sender).

Tables and Chains

Linux firewall iptables has four default tables. We will list all four along with the chains each table contains.
1. Filter
The **Filter** table is the most frequently used one. It acts as a bouncer, deciding who gets in and out of your network. It has the following default chains:

- **Input** – the rules in this chain control the packets received by the server.
- **Output** – this chain controls the packets for outbound traffic.
- **Forward** – this set of rules controls the packets that are routed through the server.

2. Network Address Translation (NAT)
This table contains NAT (Network Address Translation) rules for routing packets to networks that cannot be accessed directly. When the destination or source of the packet has to be altered, the NAT table is used. It includes the following chains:

- **Prerouting –** this chain assigns packets as soon as the server receives them.
- **Output –** works the same as the output chain we described in the **filter** table.
- **Postrouting –** the rules in this chain allow making changes to packets after they leave the output chain.

3. Mangle
The **Mangle** table adjusts the IP header properties of packets. The table has all the following chains we described above:

- **Prerouting**
- **Postrouting**
- **Output**
- **Input**
- **Forward**

4. Raw
The **Raw** table is used to exempt packets from connection tracking. The raw table has two of the chains we previously mentioned:

- **Prerouting**

- **Output**

5. Security (Optional)
Some versions of Linux also use a **Security** table to manage special access rules. This table includes **input, output,** and **forward** chains, much like the filter table.

Targets

A target is what happens after a packet matches a rule criteria. **Non-terminating** targets keep matching the packets against rules in a chain even when the packet matches a rule.
With **terminating** targets, a packet is evaluated immediately and is not matched against another chain. The terminating targets in Linux iptables are:

- **Accept** – this rule accepts the packets to come through the iptables firewall.
- **Drop** – the dropped package is not matched against any further chain. When Linux iptables drop an incoming connection to your server, the person trying to connect does not receive an error. It appears as if they are trying to connect to a non-existing machine.
- **Return** – this rule sends the packet back to the originating chain so you can match it against other rules.
- **Reject** – the iptables firewall rejects a packet and sends an error to the connecting device.

2. Relevant Program Outcomes:

PO-1, PO-2, PO-3, PO-4, PO-5, PO-9, PO-11

3. Competency and practical skills:

This practical is expected to develop the following skills:

a. Practical skill:
(i) Ability to work with IPTables to capture the network packets and analyze them.
(ii) Filters packets based on Tables, chains, rules and targets.

b. Competency skill:
(i) Deployment of IPTables for managing the network traffic.
(ii) Start/stop/restart IPTable Firewalls to monitor the traffic in real-world.
(iii)Checking and setting new IPTable Firewall rules.
(iv)Block or Allow Specific IP Address in IPTables Firewall.

4. Pre-requisites :

- **Knowledge of Networking and packet tracing using Wireshark.**
- **Technical understanding of TCP/IP networking and network architecture.**
- **Basic familiarity with firewall concepts.**
- **Good knowledge of IPTables tool.**
- **Thorough knowledge of protocols, IPV4, IPV6, NAT Tables and the packet details.**
- **Good knowledge of Root privileges, User and Super user account previliges in Linux.**
- **Good knowledge of firewalld daemon for managing firewall rules.**
- **A useraccount with sudo previliges for managing the firewall rules.**

5. Resources required:

IPTables 40GB hard disk with minimum of 4 G HZ RAM

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 40GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with) 5. IPTables runs on most UNIX and UNIX-like platforms including Linux. 6.A supported network card for capturing-Ethernet, IEEE 802.11 7. Access to a machine with Wireshark and IPTables installed. 8. A wired network jack.
2	Installation of Wireshark, IPTables .

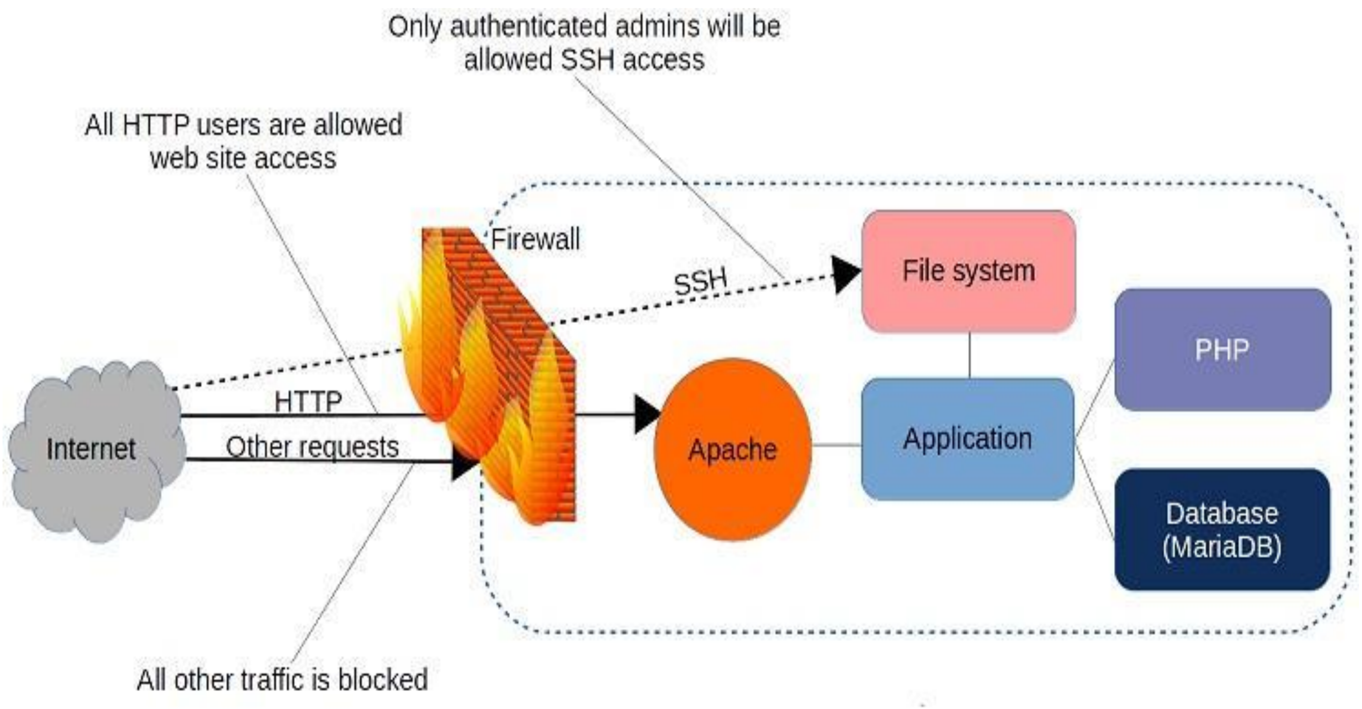
6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard; mouse and monitor are properly working with
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions will help in preventing injuries to people and damage to computer equipment.
5. A safe work place should be clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities at the workplace. Power supplies and monitors contain high voltage, handle with care.
7. Ensure that Wireshark and IPTables is installed and Configured in the system before usage.

8. Ensure the availability of the Internet with right network configuration that has interfaces for the successful execution of these experiments.
9. Make sure that the router that is plugged into is configured to receive the packets and is not in unicast mode.

7. Algorithm/circuit/Diagram/Description with Sample outputs:

A firewall is a set of rules. When a data packet moves into or out of a protected network space, its contents (in particular, information about its origin, target, and the protocol it plans to use) are tested against the firewall rules to see if it should be allowed through.


Only authenticated admins will be allowed SSH access
All HTTP users are allowed web site access
Firewall
SSH
File system
PHP
Internet
HTTP
Other requests
Apache
Application
Database (MariaDB)
All other traffic is blocked

On the one hand, iptables is a tool for managing firewall rules on a Linux machine.
On the other hand, firewalld is also a tool for managing firewall rules on a Linux machine.

Basic Syntax for iptables Commands and Options

```
sudo iptables [option] <em>CHAIN_rule</em> [-j target]
```

Here is a list of some common iptables options:

- **–A —append** – Add a rule to a chain (at the end).

- **–C —check** – Look for a rule that matches the chain's requirements.
- **–D —delete** – Remove specified rules from a chain.
- **–F —flush** – Remove all rules.
- **–I —insert** – Add a rule to a chain at a given position.
- **–L —list** – Show all rules in a chain.
- **–N —new–chain** – Create a new chain.
- **–v —verbose** – Show more information when using a list option.
- **–X —delete–chain** – Delete the provided chain.

Iptables is case-sensitive, so make sure you're using the correct options.

Configure iptables in Linux

By default, these commands affect the **filters** table. If you need to specify a different table, use the **–t** option, followed by the name of the table.

Check Current iptables Status

To view the current set of rules on your server, enter the following in the terminal window:

```
sudo iptables –L
```

```
$  sudo iptables -L

# The output of the above command

Chain    INPUT (policy ACCEPT)
target    prot opt source              destination
Chain    FORWARD (policy DROP)
target    prot opt source              destination
Chain    OUTPUT (policy ACCEPT)
target    prot opt source                 destination
```

The system displays the status of your chains. The output will list three chains:

```
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

Enable Loopback Traffic

It's safe to allow traffic from your own system (the localhost). Append the **Input** chain by entering the following:

```
sudo iptables –A INPUT –i lo –j ACCEPT
```

This command configures the firewall to accept traffic for the localhost (**lo**) interface (**-i**). Now anything originating from your system will pass through your firewall. You need to set this rule to allow applications to talk to the localhost interface.

Allow Traffic on Specific Ports

These rules allow traffic on different **ports** you specify using the commands listed below. A port is a communication endpoint specified for a specific type of data.
To allow HTTP web traffic, enter the following command:

```
sudo iptables –A INPUT –p tcp —dport 80 –j ACCEPT
```

To allow only incoming SSH (Secure Shell) traffic, enter the following:

```
sudo iptables –A INPUT –p tcp —dport 22 –j ACCEPT
```

To allow HTTPS internet traffic, enter the following command:

```
sudo iptables –A INPUT –p tcp —dport 443 –j ACCEPT
```

The options work as follows:

- **–p** – Check for the specified protocol (**tcp**).
- **—dport** – Specify the destination port.
- **–j jump** – Take the specified action.

Control Traffic by IP Address

Use the following command to ACCEPT traffic from a specific IP address.

```
sudo iptables –A INPUT –s 192.168.0.27 –j ACCEPT
```

Replace the IP address in the command with the IP address you want to allow.
You can also DROP traffic from an IP address:

```
sudo iptables –A INPUT –s 192.168.0.27 –j DROP
```

You can REJECT traffic from a range of IP addresses, but the command is more complex:

```
sudo iptables –A INPUT –m iprange —src–range 192.168.0.1–192.168.0.255 -j REJECT
```

The iptables options we used in the examples work as follows:

- **–m** – Match the specified option.
- **–iprange** – Tell the system to expect a range of IP addresses instead of a single one.
- **—src-range** – Identifies the range of IP addresses.

Dropping Unwanted Traffic

If you define **dport** iptables firewall rules, you need to prevent unauthorized access by dropping any traffic that comes via other ports:

```
sudo iptables –A INPUT –j DROP
```

The **–A** option appends a new rule to the chain. If any connection comes through ports other than those you defined, it will be dropped.

Delete a Rule

You can use the **–F** option to clear all iptables firewall rules. A more precise method is to delete the line number of a rule.
First, list all rules by entering the following:

```
sudo iptables –L —line–numbers
```

Locate the line of the firewall rule you want to delete and run this command:

```
sudo iptables –D INPUT <<em>Number</em>>
```

Replace **<*Number*>** with the actual rule line number you want to remove.

Save Your Changes

Iptables does not keep the rules you created when the system reboots. Whenever you configure iptables in Linux, all the changes you make apply only until the first restart.
$sudo apt-get update

$sudo apt-get install iptables-persistent

8. Test cases:

1. **Test case-1**: How to enable IPTables inLinux

o Install the **iptables**-services package (if it is not already installed) by running the following command: $ yum install **iptables**-services.

o **Enable** the service to **start** at boot time by running the following commands: $ systemctl **enable iptables** $ systemctl **enable** ip6tables.

2. **Test case-2**: Block incoming PING requests on IPTables

o **iptables -A INPUT -p icmp -i eth0 -j DROP**

9.Practical Related Questions:

o What are the tables used in IPTables"

o NAT, MANGLE, FILTER and RAW TABLES

o How to add rules to ACCEPT, REJECT, DENY and DROP ssh service running on 22 in iptables.

```
$sudo iptables -A INPUT -s -p tcp --dport 22 -j ACCEPT

$sudo iptables -A INPUT -s -p tcp --dport 22 -j REJECT

$sudo iptables -A INPUT -s -p tcp --dport 22 -j DENY
```

```
$sudo iptables -A INPUT -s -p tcp --dport 22 -j DROP
```

10. Exercise Questions:

Practical Exercises on IPTables:

● **Exercise-1:**

● Please setup the following rules in your network environment and test your firewall settings. State your rules configured with iptables and document your results with excerpts of your terminal output and messages from wireshark.

a) Block ICMP requests from the 192.168.1.10 to the server.

$iptables -A INPUT -p ICMP -s 192.168.1.10 -j DROP

b) Reject ICMP Requests from 192.168.2.20 to the server.

$ iptables -A INPUT -p ICMP -s 192.168.2.20 -j REJECT

c) Reject SSH connections from 192.168.3.30 to the server.

$iptables -A INPUT -p tcp –dport ssh -s 192.168.3.30 -j REJECT

d) Block all traffic from computer 192.168.1.10 to the internet.

$iptables -A FORWARD -s 192.168.1.10 -j DROP

PRACTICAL 4. Using open SSL for web server - browser communication

1. Practical significance: SSL is an open source implementation of the SSL / TLS. It is used to create SSL, TLS and HTTPS connections with. When someone want to run a web application with SSL / HTTPS then their web server needs a private key, public key and a certificate from them. The private key enables their server to identify itself, and the certificate proves that the public key actually belongs to the server.

2. Relevant Program Outcomes:

PO 1, PO 2, PO 3, PO 5, PO 9, PO 11

3. Competency and practical skills :

The practical is expected to make the student to understand the significance of certificates and how to use them for identification purpose.

4. Prerequisites:

1. Student should have knowledge on basic internet usage.
2. Student should have knowledge on e-mail communication and encryption mechanisms.

5. Resources required:

S. No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 20GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet Connection.

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there is uninterrupted internet connection is available.
4. A safe work space is clean, organized, and properly lighted.
5.Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

7. **Algorithm/circuit/Diagram/Description:**

The private key and public key is created by user. The certificate is created by a certificate authority (CA). Here is the process you go through to obtain a private key, public key and certificate from a CA:

1. Generate a private key.
2. Generate a public key matching the private key (if that was not already done in step 1).
3. Generate a certificate signing request.
4. Send the certificate signing request to a certificate authority (CA).
5. Receive certificate from certificate authority.
6. Install private key and certificate in your web server software.

Generating a Private Key

The first step is to generate a private key used for public key / private key (asymmetric) encryption. This OpenSSL command creates a private key which is 2048 bits long, and protected with a triple DES encryption. When executed OpenSSL will prompt you to enter a password to use for the triple DES encryption.

```
openssl genrsa -des3 -out privatekey.pem 2048
```

The private key will be written to a file. The private key file will be named privatekey.pem and will located in the same directed from which you exected the openssl command. If you do not want the private key protected with encryption, leave out the -des3 part. This command creates a private key which is 2048 bits long without password encryption of the key file:

```
openssl genrsa -out privatekey.pem 2048
```

Generating a Certificate Signing Request

The second step is to generate a certificate signing request from the private key. Here is the OpenSSL command that creates a certificate signing request:

```
openssl req -new -key privatekey.pem -out certificate-signing-request.csr
```

The private key is the file named privatekey.pem. That is the output of the private key generation command earlier. The certificate signing request file is the certificate-signing-request.csr. This is the file you will eventually send to a CA.

Sending the Certificate Signing Request to a CA

Once you have a certificate signing request (file) you can send it to a certificate authority (CA). The CA will send you a certificate file back.

Some intermediate CAs may send you several certificates back. One is your certificate, and the rest is the certificate chain needed to trust your certificate. The certificate chain is a chain of certificates from a root CA own to the CA you have used. Web servers typically need all of the certificates in the certificate chain. See your specific web server for more information on how to provide it with the full certificate chain.

Generating a Self-signed Test Certificate

In case you just want to test your web application with HTTPS, you can also generate a self signed certificate with OpenSSL. Here is how that is done:

```
openssl req -new -x509 -key privkey.pem -out self-signed-certificate.pem -days 1095
```

Notice though, that the browsers will show security warnings when you visit a website with a self signed certificate. The browser will allow you to choose to trust the certificate so you can test that everything works with HTTPS.

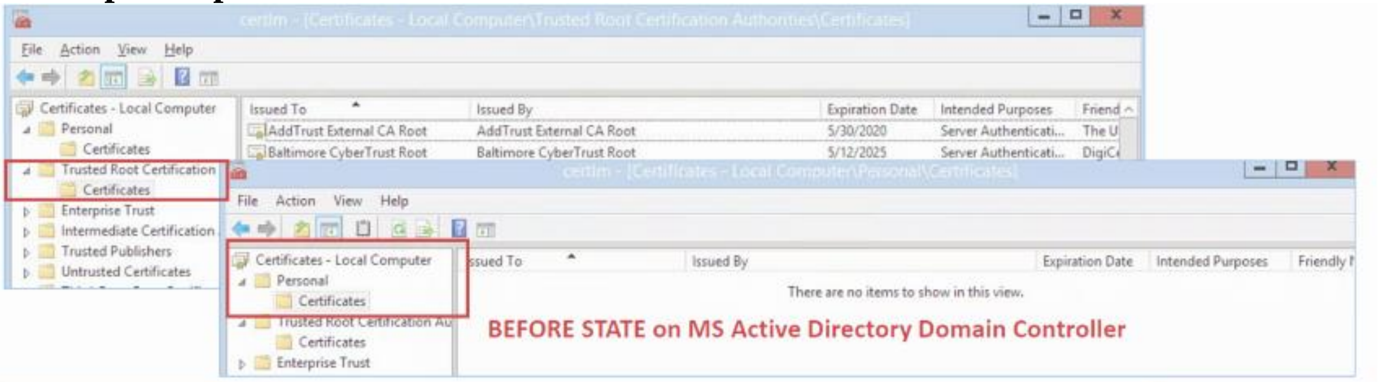
Installing Private Key and Certificate in Web Server

Once you have obtained the certificate (+ optionally a certificate chain) you need to install the private key and certificate (+optionally the certificate chain) in your web server.

8. Test cases:

1. Try to use another private key instead of original one.
2. Students must also record the errors while executing the program experimenting with different inputs.

9. Sample output:


certlm - [Certificates - Local Computer\Trusted Root Certification Authorities\Certificates]

Certificates - Local Computer
- Personal
  - Certificates
- Trusted Root Certification
  - Certificates
- Enterprise Trust
- Intermediate Certification
- Trusted Publishers
- Untrusted Certificates

Issued To	Issued By	Expiration Date	Intended Purposes	Friend
AddTrust External CA Root	AddTrust External CA Root	5/30/2020	Server Authenticati...	The U
Baltimore CyberTrust Root	Baltimore CyberTrust Root	5/12/2025	Server Authenticati...	DigiC

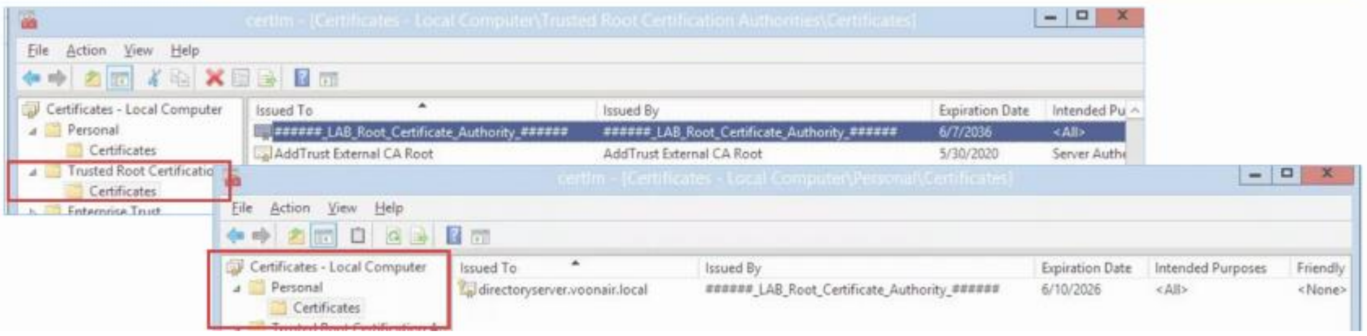
certlm - [Certificates - Local Computer\Personal\Certificates]

Certificates - Local Computer
- Personal
  - Certificates
- Trusted Root Certification Au
  - Certificates
- Enterprise Trust

ssued To	Issued By	Expiration Date	Intended Purposes	Friendly N

There are no items to show in this view.

BEFORE STATE on MS Active Directory Domain Controller


certlm - [Certificates - Local Computer\Trusted Root Certification Authorities\Certificates]

Certificates - Local Computer
- Personal
  - Certificates
- Trusted Root Certificatio
  - Certificates
- Enterprise Trust

Issued To	Issued By	Expiration Date	Intended Pu
######_LAB_Root_Certificate_Authority_######	######_LAB_Root_Certificate_Authority_######	6/7/2036	<All>
AddTrust External CA Root	AddTrust External CA Root	5/30/2020	Server Authe

certlm - [Certificates - Local Computer\Personal\Certificates]

Certificates - Local Computer
- Personal
  - Certificates

Issued To	Issued By	Expiration Date	Intended Purposes	Friendly
directoryserver.voonair.local	######_LAB_Root_Certificate_Authority_######	6/10/2026	<All>	<None>

10. Practical Related Questions:

1. What is SSL certificate?
2. How SSL uses both asymmetric and symmetric encryption?
3. What is a Certificate Signing Request (CSR)?
4. Discuss some public-key encryption algorithms used in SSL.
5. What are the authentication levels of SSL/TLS certificates?

11. Exercise Related Questions:

1. Configure openssl x509 extensions for server certificate.
2. Configure Apache with SSL

PRACTICAL-5: Configuring S/MIME for e-mail communication.

1. Practical significance: S/MIME is an acronym for Secure/Multipurpose Internet Mail Extensions. It references a type of public encryption and signing of MIME data to verify a sender's identity. With S/MIME, it is possible to send and receive encrypted emails.

2. Relevant Program Outcomes:
PO 1, PO 2, PO 3, PO 5, PO 9, PO 11

3. Competency and practical skills:

The practical is expected to make the student to understand the significance of certificates and how to use them for identification purpose.

4. Prerequisites:

1. Student should have knowledge on basic internet usage.
2. Student should have knowledge on e-mail communication and encryption mechanisms.

5. Resources required:

S. No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. Hard-Drive Space – 20GB 4. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet Connection.

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there is uninterrupted internet connection is available.
4. A safe work space is clean, organized, and properly lighted.
5. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

7. **Algorithm/circuit/Diagram/Description:**

Encryption process

1. Once the sender clicks on **Send**, the original unencrypted message is captured.
2. The **recipient's public key is used to encrypt** the original message. At the end of the process, an encrypted version of the original message is produced.
3. The encryption message replaces the original message.

4. The email is sent to the recipient.

Decryption process

1. The recipient receives the email.
2. The encrypted message is retrieved.
3. The **recipient's private key is used to decrypt** the encrypted message.
4. The original message is obtained and displayed to the recipient.

Digital signing process

1. Once the sender clicks on **Send**, the original message is captured.
2. The message hash is calculated.
3. The **sender's private key is used to encrypt the hash value.**
4. The encrypted hash value is added to the email.
5. The email is sent to the recipient.

Signature verification process

1. The recipient receives the digitally signed email.
2. The original message is obtained and its hash value is calculated.
3. The encrypted hash is retrieved from the email.
4. The encrypted hash is decrypted using the sender's public key.
5. The decrypted hash and the hash value calculated from the original message obtained are compared. If the values match, the signature is verified.

How to Enable S/MIME Gmail:

1. **Log into an Administrator Account.** Non-admin accounts cannot open the admin console needed to set up a hosted S/MIME encryption solution.
2. **Go to User Settings.** From the Admin console’s Home page, select **Apps > G Suite > Gmail > User Settings**.
3. **Select the Domain or Organization to Configure.** This will be found on the left of the screen, under **Organizations**.
4. **Select the “Enable S/MIME” Box.** There should be a box with the setting that you can enable with a click.
5. **Allow Users to Upload Certificates (Optional).** You can allow users to upload their own S/MIME certificates as an option.

6. **Set up Root Certificate Management (Optional).** You can manage the root certificates used for S/MIME email encryption by:

1. Clicking **Add** next to **Accept these additional Root Certificates for specific domains**.
2. Clicking on **Upload Root Certificate**.
3. Browsing to find the certificate file and selecting **Open**. A verification message should appear. Otherwise, an error message may appear.
4. Under **Encryption level**, choose the encryption level to use with the selected certificate.
5. Under **Address list**, enter at least one domain that will use the uploaded root certificate.
6. Click **Save**.

7. Repeat these steps for each additional certificate chain.

7. **Does Your Domain/Organization NEED to Enable Secure Hash Algorithm 1?** If so, you may need to select the **Allow SHA-1 globally** box. Otherwise, this is not recommended by Google.

8. **Click Save.** Save your settings so they don't get lost.

9. **Have All Users Reload Gmail.** After enabling hosted S/MIME Gmail encryption, users will need to reload their Gmail client to see the change.

10. **Upload S/MIME Certificates.** You can upload personal S/MIME certificates in Gmail if you:

1. Go to **Settings**.
2. Click on the **Accounts** tab.
3. Click on **Edit Info** in the **Send mail as** area. A window should appear with the "enhanced encryption" option—if this was enabled in Step 5 listed above.
4. Click on **Upload a personal certificate**.
5. Select the certificate and click **Open**. A password prompt should appear if this works.
6. Enter the password and click on **Add certificate**.

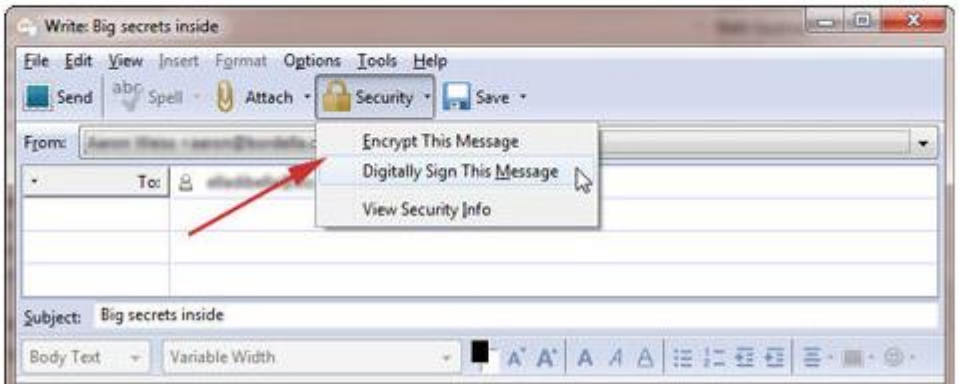
11. **Have Users Exchange S/MIME Keys.** To decrypt encrypted messages, users in the organization will need to exchange S/MIME encryption keys. This can be done by:

1. Sending an S/MIME encrypted message to the recipient with a digital signature that includes the user's public key. This can then be used to send S/MIME-encrypted emails.
2. Asking recipients to send a message. The S/MIME signed message will allow the encryption key to be automatically stored so future messages will be encrypted.

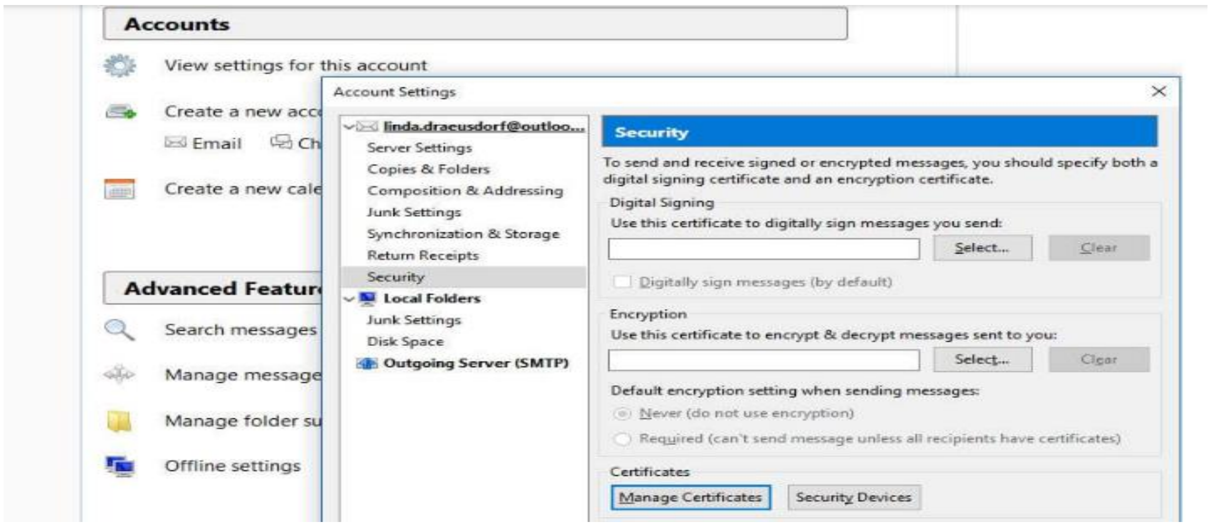
8. Test cases:

1. Try to use another private key instead of original one for decryption process.
2. Use digital signature verification with different public keys.

9. Sample output:



Big secrets inside



Security

To send and receive signed or encrypted messages, you should specify both a digital signing certificate and an encryption certificate.

Digital Signing

Use this certificate to digitally sign messages you send:

Digitally sign messages (by default)

Encryption

Use this certificate to encrypt & decrypt messages sent to you:

Default encryption setting when sending messages:

Never (do not use encryption)

Required (can't send message unless all recipients have certificates)

Certificates

10. Practical Related Questions:

1. Explain the purpose of S/MIME?
2. What are the certificates involved in S/MIME?
3. What is an email digital signature certificate?
4. Discuss functions of S/MIME.
5. What happens if we lose the certificate?

11. Exercise Related Questions:

1. Configure S/MIME in outlook.
2. Configure S/MIME in yahoo mail.

PRACTICAL 06: Understanding the buffer overflow and format string attacks

1. Practical significance:

The program uses an improperly bounded format string, allowing it to write outside the bounds of allocated memory. This behavior could corrupt data, crash the program, or lead to the execution of malicious code.

2. Relevant Program Outcomes: PO1, PO2, PO 3, PO 4, PO 9

3. Competency and practical skills :

The practical is expected to develop the following skills

1. Ability to study and understand Buffer overflow and format string attacks.

4. Prerequisites:

1. Student should have knowledge on Computer Networks

2. Student should have knowledge on C Programming.

5. Resources required:

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2.8 GHz 2. RAM – 4 GB 3. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet

6. Precautions:

1. Check whether the buffer is overflow after attack in RAM
2. Ensure the RAM and ROM are properly working.
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.

7. Algorithm/circuit/Diagram/Description:

1. Reading from the stack.

2. Writing to the stack.

3. Format String Attack Detection & Defenses

8. Program:

```
#include<stdio.h>


int main(int argc, char** argv)
{
    char buffer[100];
    strncpy(buffer, argv[1], 100);


    // We are passing command line
    // argument to printf
    printf(buffer);
    return 0;
}
```

9. Test Cases:

Case1: $ ./a.out "%p %p %p %p %p %p %p %p %p %p %p %p %p %p %p"

```
0xffffdddd 0x64 0xf7ec1289 0xffffdbdf 0xffffdbde (nil) 0xffffdcc4 0xffffdc64 (nil)
0x25207025 0x70252070 0x20702520 0x25207025 0x70252070 0x20702520.
```

Case 2: At about 10 arguments up the stack, we can see a repeating pattern of 0x252070 – those are our %ps on the stack! We start our string with AAAA to see this more explicitly:

Case 3: `$ ./a.out "AAAA%p %p %p %p %p %p %p %p %p %p"`

```
AAAA0xffffdde8 0x64 0xf7ec1289 0xffffdbef 0xffffdbee (nil) 0xffffdcd4 0xffffdc7
```

Case 4: `$ ./a.out 'AAAA%10$p'`

```
AAAA0x41414141
```

10.Practical Related Questions:

1. What happens if buffer size is reachers to 256 Bytes, when RAM size is 1GB ?
2. Explain Buffer overflow attack when sufficient number of memory addresses are allocated?
3. Compare and contrast format string attacks with sketch?

11. Exercise Related Questions:

1. Write a C program to implement Buffer overflow to create attack to victim?
2. Write a C program to implement Format string attack and deploy in LAN?

PRACTICAL 07: Understanding NMAP for ports monitoring

1.Practical significance:

Nmap is now one of the core tools used by network administrators to map their networks. The program can be used to find live hosts on a network, perform port scanning, ping sweeps, OS detection, and version detection

2. Relevant Program Outcomes: PO1, PO2, PO3

3. Competency and practical skills :

The practical is expected to develop the following skills

1. Ability to study and understand NMAP for port monitoring

4. Prerequisites:

1. Student should have knowledge on Computer Networks

2. Student should have knowledge on cyber security.

5. Resources required:

S.No	Name of the Resource Broad Specification(Approx.)
1	Computer System (multi processor) 1. 4 Processors – 2.8 GHz 2. RAM – 4 GB 3. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet

6. Precautions:

1. Check whether the computer is vulnerable or not?
2. Ensure the NMAP tool is properly working.
3. Ensure that there are penetration happened on victim machine while executing the program.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.

7. **Algorithm/circuit/Diagram/Description:**

ALGORITHM:

1. Download the Nmap installer. This can be found for free from the developer's website.

2. Install Nmap. Run the installer once it is finished downloading. ...

3. Run the “Nmap – Zenmap” GUI program. ...

4. Enter in the target for your scan. ...

5. Choose your Profile. ...

6. Click Scan to start scanning. ...

7. Read your results.

8. Program/ Commands:

COMMANDS:

Nmap Port Scanning Commands

The “–open” parameter

In any of the commands below, you can specify the “–open” parameter in your Nmap command to have Nmap only show you ports with an “Open” state.

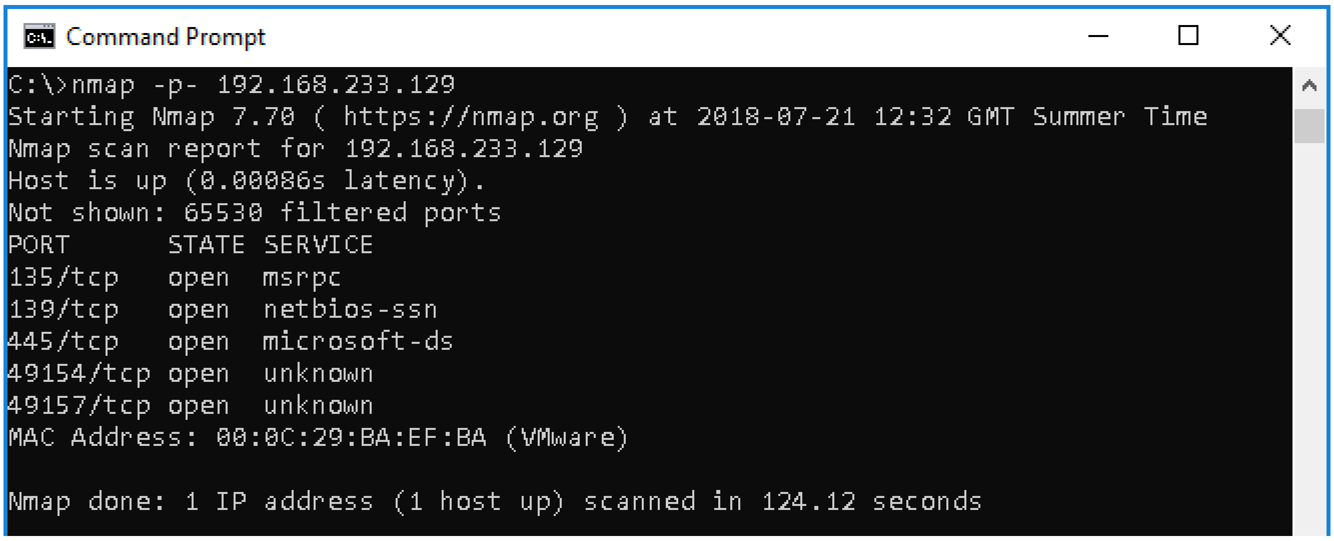
nmap –open [ip_address]

Scanning a single port

nmap -p 80 [ip_address]

This command will initiate a default scan against the target host and look for port 80.

9. Test Cases:



```
C:\>nmap -p- 192.168.233.129
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-21 12:32 GMT Summer Time
Nmap scan report for 192.168.233.129
Host is up (0.00086s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49154/tcp open  unknown
49157/tcp open  unknown
MAC Address: 00:0C:29:BA:EF:BA (VMware)

Nmap done: 1 IP address (1 host up) scanned in 124.12 seconds
```

10.Practical Related Questions:

1. **HOw do you perform Govt. websites port monitoring using nmap**
2. **DIfferentiate between Active ports and passive ports?**

11. Exercise Related Questions:

1. **Deploy NMAP tool in any GMAIL account?**
2. **COnfigure listening port where exploit is present?**

PRACTICAL 08: Secure Socket Programming

1.Practical significance:

Secure Socket Layer (SSL) provides security to the data that is transferred between web browser and server. SSL encrypts the link between a web server and a browser which ensures that all data passed between them remains private and free from attack. Secure Socket Layer Protocols: SSL record protocol. Handshake protocol.

2. Relevant Program Outcomes: PO1, PO2, PO3,PO5

3. Competency and practical skills :

The practical is expected to develop the following skills

1. Ability to study and understand secure sockets at client and server.

4. Prerequisites:

1. Student should have knowledge on Computer Networks

2. Students should have knowledge on client server programming.

5. Resources required:

S.No	Name of the Resource Broad Specification(Approx.)

1	Two Computer Systems (multi processor) 1. 4 Processors – 2.8 GHz 2. RAM – 4 GB 3. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet - Local Area Network

6. Precautions:

1. Check whether the computer is secure or not?
2. Ensure the Socket library is properly working.
3. Ensure that there is a either peer programming is effective or Client -server is effective.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.

7. Algorithm/circuit/Diagram/Description:

ALGORITHM:

Steps involved in SSL handshake:

1.The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.

2.The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL. The server also sends its own digital certificate and, if the client is requesting a server resource that requires client authentication, requests the client's digital certificate.

3.The client uses the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client proceeds.

4.Using all data generated in the handshake so far, the client creates the premaster secret for the session, encrypts it with the server's public key (obtained from the server's digital certificate), and sends the encrypted premaster secret to the server.

5.If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client's own digital certificate to the server along with the encrypted premaster secret.

6.If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster secret, then performs a series of steps which the client also performs, starting from the same premaster secret to generate the master secret.

7.Both the client and the server use the master secret to generate session keys which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity.

8.The client informs the server that future messages from the client will be encrypted with the session key. It then sends a separate encrypted message indicating that the client portion of the handshake is finished.

9.The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate encrypted message indicating that the server portion of the handshake is finished.

10.The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

8. Program/ Commands:

```
/* Server Socket */

#include <errno.h>

#include <unistd.h>

#include <malloc.h>

#include <string.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>
```

```
#include <resolv.h>

#include "openssl/ssl.h"

#include "openssl/err.h"


#define FAIL        -1


int OpenListener(int port)

{   int sd;

        struct sockaddr_in addr;


        sd = socket(PF_INET, SOCK_STREAM, 0);

        bzero(&addr, sizeof(addr));

        addr.sin_family = AF_INET;

        addr.sin_port = htons(port);

        addr.sin_addr.s_addr = INADDR_ANY;

        if ( bind(sd, (struct sockaddr*)&addr, sizeof(addr)) != 0 )

        {

        perror("can't bind port");

        abort();

        }

        if ( listen(sd, 10) != 0 )

        {

        perror("Can't configure listening port");

        abort();

        }

        return sd;
```

```
}


int isRoot()
{
        if (getuid() != 0)
        {
        return 0;
        }
        else
        {
        return 1;
        }


}
SSL_CTX* InitServerCTX(void)
{   SSL_METHOD *method;
        SSL_CTX *ctx;


        OpenSSL_add_all_algorithms();  /* load & register all cryptos, etc. */
        SSL_load_error_strings();   /* load all error messages */
        method = TLSv1_2_server_method();  /* create new server-method instance */
        ctx = SSL_CTX_new(method);   /* create new context from method */
        if ( ctx == NULL )
        {
        ERR_print_errors_fp(stderr);
        abort();
        }
        return ctx;
}

void LoadCertificates(SSL_CTX* ctx, char* CertFile, char* KeyFile)
{
        /* set the local certificate from CertFile */
        if ( SSL_CTX_use_certificate_file(ctx, CertFile, SSL_FILETYPE_PEM) <= 0
)
        {
        ERR_print_errors_fp(stderr);
        abort();
        }
        /* set the private key from KeyFile (may be the same as CertFile) */
if ( SSL_CTX_use_PrivateKey_file(ctx, KeyFile, SSL_FILETYPE_PEM) <= 0 )
        {
        ERR_print_errors_fp(stderr);
        abort();
        }
        /* verify private key */
        if ( !SSL_CTX_check_private_key(ctx) )
        {
        fprintf(stderr, "Private key does not match the public certificate\n");
        abort();
        }
}
```

```
void ShowCerts(SSL* ssl)
{   X509 *cert;
        char *line;

        cert = SSL_get_peer_certificate(ssl); /* Get certificates (if available) */
        if ( cert != NULL )
        {
        printf("Server certificates:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("Subject: %s\n", line);
        free(line);
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);
        X509_free(cert);
        }
        else
        printf("No certificates.\n");
}

void Servlet(SSL* ssl) /* Serve the connection -- threadable */
{   char buf[1024];
        char reply[1024];
        int sd, bytes;
        const                                                            char*
    HTMLecho="<html><body><pre>%s</pre></body></html>\n\n";


        if ( SSL_accept(ssl) == FAIL )  /* do SSL-protocol accept */

        ERR_print_errors_fp(stderr);

        else

        {

        ShowCerts(ssl);     /* get any certificates */

        bytes = SSL_read(ssl, buf, sizeof(buf)); /* get request */

        if ( bytes > 0 )

        {

        buf[bytes] = 0;

        printf("Client msg: \"%s\"\n", buf);

        sprintf(reply, HTMLecho, buf);   /* construct reply */

        SSL_write(ssl, reply, strlen(reply)); /* send reply */

        }

        else

        ERR_print_errors_fp(stderr);

        }

        sd = SSL_get_fd(ssl);      /* get socket connection */

        SSL_free(ssl);       /* release SSL state */

        close(sd);           /* close connection */

    }


    int main(int count, char *strings[])

    {   SSL_CTX *ctx;
```

```
int server;
char *portnum;

if(!isRoot())
{
printf("This program must be run as root/sudo user!!");
exit(0);
}
if ( count != 2 )
{
printf("Usage: %s <portnum>\n", strings[0]);
exit(0);
}
SSL_library_init();

portnum = strings[1];
ctx = InitServerCTX();    /* initialize SSL */
LoadCertificates(ctx, "mycert.pem", "mycert.pem"); /* load certs */
server = OpenListener(atoi(portnum));          /* create server socket */
while (1)
{   struct sockaddr_in addr;
socklen_t len = sizeof(addr);
SSL *ssl;

int client = accept(server, (struct sockaddr*)&addr, &len);   /* accept
connection as usual */
```

```
        printf("Connection:                    %s:%d\n",inet_ntoa(addr.sin_addr),
    ntohs(addr.sin_port));
        ssl = SSL_new(ctx);          /* get new SSL state with context */
        SSL_set_fd(ssl, client);   /* set connection socket to SSL state */
        Servlet(ssl);       /* service connection */
        }
        close(server);      /* close server socket */
        SSL_CTX_free(ctx);      /* release context */
}
```

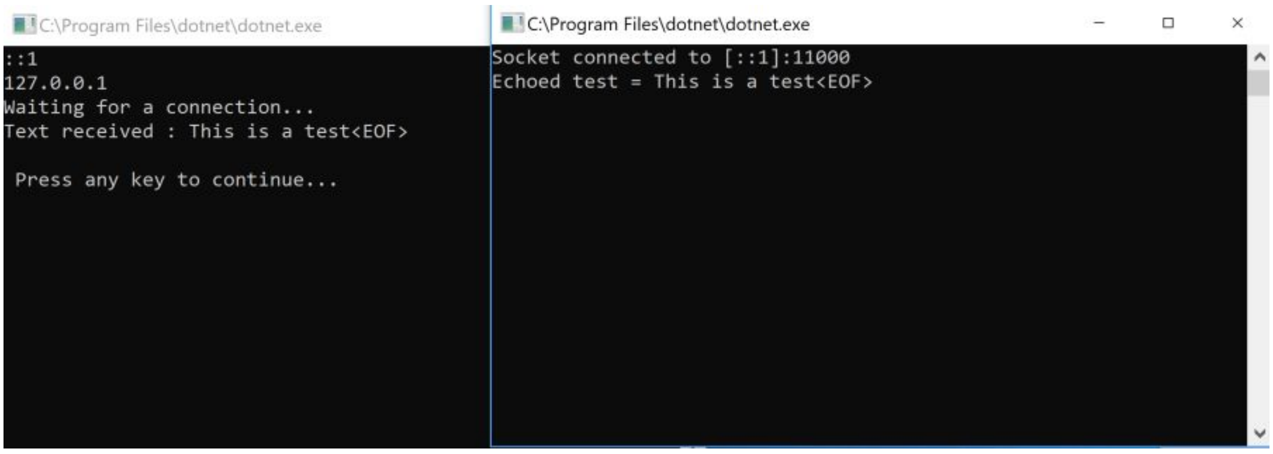
//**SSL-Client.c**

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <malloc.h>
#include <string.h>
#include <sys/socket.h>
#include <resolv.h>
#include <netdb.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define FAIL      -1

int OpenConnection(const char *hostname, int port)
{   int sd;
```

```
        struct hostent *host;
        struct sockaddr_in addr;

        if ( (host = gethostbyname(hostname)) == NULL )
        {
        perror(hostname);
        abort();
        }
        sd = socket(PF_INET, SOCK_STREAM, 0);
        bzero(&addr, sizeof(addr));
        addr.sin_family = AF_INET;
        addr.sin_port = htons(port);
        addr.sin_addr.s_addr = *(long*)(host->h_addr);
        if ( connect(sd, (struct sockaddr*)&addr, sizeof(addr)) != 0 )
        {
        close(sd);
        perror(hostname);
        abort();
        }
        return sd;
}

SSL_CTX* InitCTX(void)
{   SSL_METHOD *method;
        SSL_CTX *ctx;
        OpenSSL_add_all_algorithms();  /* Load cryptos, et.al. */
        SSL_load_error_strings();   /* Bring in and register error messages */
        method = TLSv1_2_client_method();  /* Create new client-method instance */
        ctx = SSL_CTX_new(method);   /* Create new context */
        if ( ctx == NULL )
        {
        ERR_print_errors_fp(stderr);
        abort();
        }
        return ctx;
}

void ShowCerts(SSL* ssl)
{   X509 *cert;
        char *line;

        cert = SSL_get_peer_certificate(ssl); /* get the server's certificate */
        if ( cert != NULL )
        {
        printf("Server certificates:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("Subject: %s\n", line);
        free(line);     /* free the malloc'ed string */
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);     /* free the malloc'ed string */
        X509_free(cert);    /* free the malloc'ed certificate copy */

        }

        else

        printf("Info: No client certificates configured.\n");

}


int main(int count, char *strings[])

{   SSL_CTX *ctx;

        int server;

        SSL *ssl;

        char buf[1024];

        int bytes;

        char *hostname, *portnum;


        if ( count != 3 )

        {

         printf("usage: %s <hostname> <portnum>\n", strings[0]);

        exit(0);

        }

        SSL_library_init();

        hostname=strings[1];

        portnum=strings[2];


        ctx = InitCTX();

        server = OpenConnection(hostname, atoi(portnum));

        ssl = SSL_new(ctx);          /* create new SSL connection state */
        SSL_set_fd(ssl, server);   /* attach the socket descriptor */
        if ( SSL_connect(ssl) == FAIL )   /* perform the connection */
        ERR_print_errors_fp(stderr);
        else
        {   char *msg = "Hello???";

       printf("Connected with %s encryption\n", SSL_get_cipher(ssl));
        ShowCerts(ssl);     /* get any certs */
        SSL_write(ssl, msg, strlen(msg));   /* encrypt & send message */
        bytes = SSL_read(ssl, buf, sizeof(buf)); /* get reply & decrypt */
        buf[bytes] = 0;
        printf("Received: \"%s\"\n", buf);
        SSL_free(ssl);        /* release connection state */
        }
        close(server);         /* close socket */
        SSL_CTX_free(ctx);       /* release context */
        return 0;
    }
```

9. Test Cases:

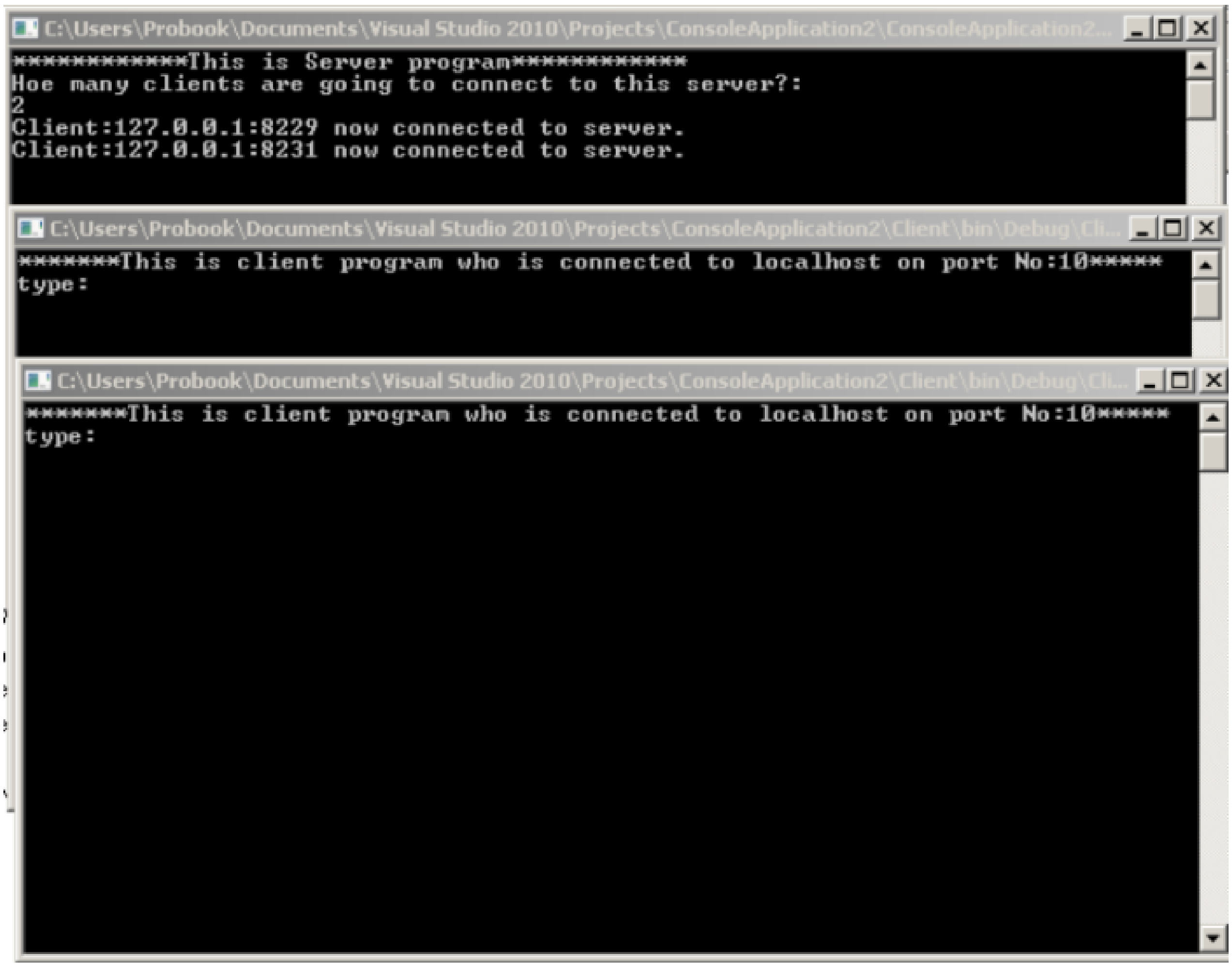
Case 1: Single CLient Socket connected to Server SOcket.


C:\Program Files\dotnet\dotnet.exe

```
::1
127.0.0.1
Waiting for a connection...
Text received : This is a test<EOF>

 Press any key to continue...
```

C:\Program Files\dotnet\dotnet.exe

```
Socket connected to [::1]:11000
Echoed test = This is a test<EOF>
```


C:\Users\Probook\Documents\Visual Studio 2010\Projects\ConsoleApplication2\ConsoleApplication2...

```
*************This is Server program*************
Hoe many clients are going to connect to this server?:
2
Client:127.0.0.1:8229 now connected to server.
Client:127.0.0.1:8231 now connected to server.
```

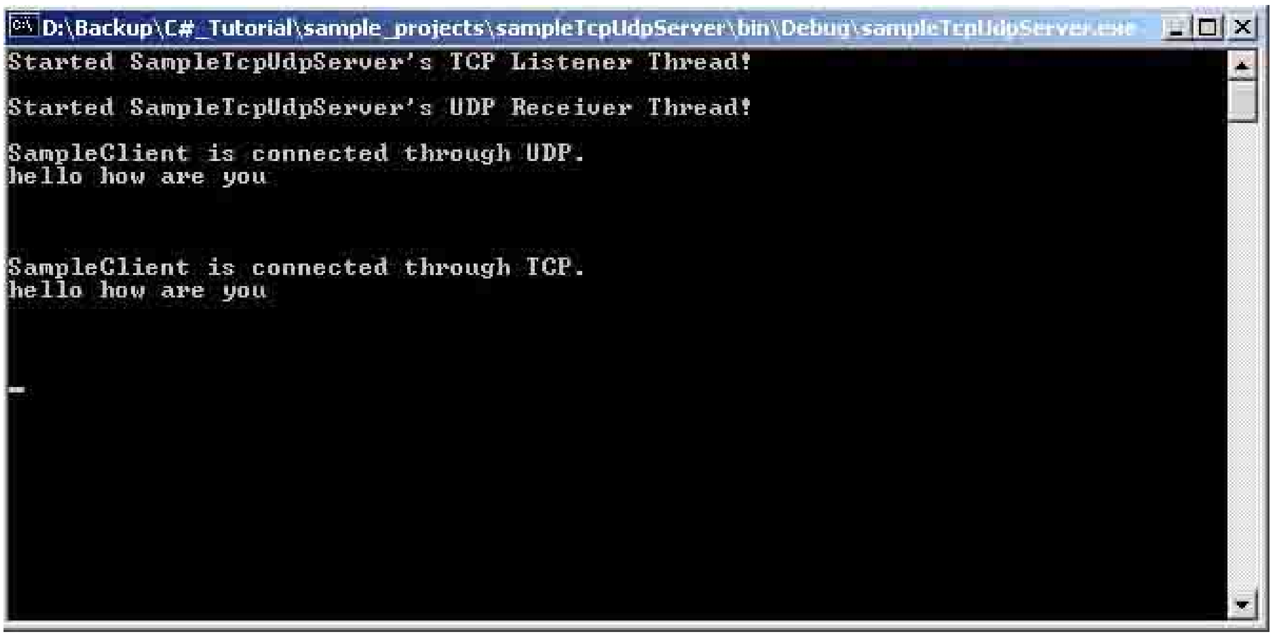
C:\Users\Probook\Documents\Visual Studio 2010\Projects\ConsoleApplication2\Client\bin\Debug\Cli...

```
*******This is client program who is connected to localhost on port No:10*****
type:
```

C:\Users\Probook\Documents\Visual Studio 2010\Projects\ConsoleApplication2\Client\bin\Debug\Cli...

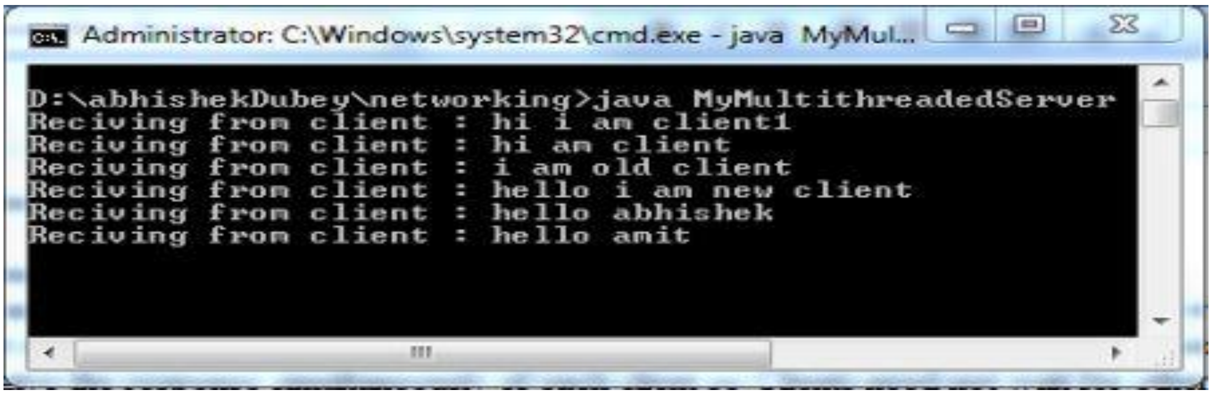
```
*******This is client program who is connected to localhost on port No:10*****
type:
```

CASE 2: MULTIPLE CLIENTS connected:



D:\Backup\C#_Tutorial\sample_projects\sampleTcpUdpServer\bin\Debug\sampleTcpUdpServer.exe

```
Started SampleTcpUdpServer's TCP Listener Thread!

Started SampleTcpUdpServer's UDP Receiver Thread!

SampleClient is connected through UDP.
hello how are you


SampleClient is connected through TCP.
hello how are you
```



```
D:\abhishekDubey\networking>java MyMultithreadedServer
Reciving from client : hi i am client1
Reciving from client : hi am client
Reciving from client : i am old client
Reciving from client : hello i am new client
Reciving from client : hello abhishek
Reciving from client : hello amit
```

10.Practical Questions:

1. What is Socket?
2. Difference between client socket and server socket?
3. Explain peer to peer sockets?

11. Exercise Related Questions:

1. Configure TCP socket in File transfer?
2. COnfigure UDP socket in File transfer?

PRACTICAL 9: Case Study: PGP(Pretty Good Privacy)

1. Practical significance: Pretty Good Privacy or PGP is a popular program used to encrypt and decrypt email over the Internet, as well as authenticate messages with digital signatures and encrypted stored files.

2. Relevant Program Outcomes: PO 1, PO 2, PO 3, PO 5, PO 9, PO 11

3. Competency and practical skills :

The practical is expected to make the student to understand the significance of certificates and how to use them for identification purpose.

4. Prerequisites:

1. Student should have knowledge on basic internet usage.
2. Student should have knowledge on e-mail communication and encryption mechanisms.

5. Resources required:

o	e of the Resource Broad Specification(Approx.)
	puter System ocessor – 2GHz AM – 4GB Iard-Drive Space – 20GB 'GA with 1024×768 screen resolution t hardware requirement will depend upon the bution that we choose to work with)
	net Connection.

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there is uninterrupted internet connection is available.
4. A safe work space is clean, organized, and properly lighted.
5. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

Pretty Good Privacy (PGP)

Step 1 – Create a public/private key pair in PGP

1. Using an ISFTS Student Laptop Computer, power on the computer. During start-up you will be presented with a window to choose an Operating System, select “Windows 2000”. The loading process for Windows 2000 may take several minutes. Once Windows 2000 has completely loaded a desktop login window interface will appear requesting a Username:
“Administrator” and Password: “tartans”.

2. Once log-on is complete, a Windows 2000 Desktop interface will appear. In order to begin utilizing PGP, you must first launch the program. You will need to click on the
“Start” button once; then click once on the “Program Folder”; and click on the “PGP Folder”. At this point you should double click on the “PGP Key” Icon.

3. At this point you should see the Key Generation Wizard. If you do not see this window but rather a “PGP Keys” window, you must click on the “Keys” drop-down menu and select “New Keys” to arrive at the Key Generation Wizard. Start the Key Generation Wizard by clicking the “Next” button.

4. The Key Generation Wizard will now ask for you to select a “Key Pair Type”. You will need to select “Diffie-Hellman”. After selecting the correct “Key Pair Type” proceed to the next window.

5. Now the Key Generation Wizard will ask you to select the “Key Pair Size”. The default setting is 2048 bits. This default setting is fine. All that you need to do is click “Next” and proceed on to the “Key Expiration” window.

6. At this point the Key Generation Wizard will require you to select a “Key Expiration”. The default setting should be “Key pair never expires”. Ensure this setting is correct and then proceed to the next window.

7. Next the Key Generation Wizard requires you to enter a passphrase to protect your private key. Enter a passphrase that you can easily remember in both the “Passphrase” and “Confirmation” boxes. [The passphrase must be at least eight character in length] Once you have provided the passphrase click “Next”. PGP will now create your key pair.

8. Once the key pair is created, the next Key Generation Wizard window will prompt you send your newly created key pair to a root server. Do NOT send the key to the root server. Make sure the box is unchecked then click next.

9. This is the last window in the Key Generation Wizard. Click Finish to complete the Key Generation Wizard.

Step 2 – Create revocation key

A revocation key is used for revoking keys (typically stored on public Internet key servers) when a passphrase has been forgotten or the key pair has been lost. It is recommended that this revoking key (and any backup of your actual key pair) be stored on a floppy disk and kept in an alternate/safe location.

1. To begin using PGP you must right click on the “Pad Lock” icon located in the lower right side of the task bar, and then select the “PGPkeys” option. This will launch a PGPkeys window. If the “Pad Lock” icon is not located in the lower right side of the task bar, you will need to click on the “Start” button on the lower left side of the task bar and select the “Programs” folder.

Now locate the “PGP” folder and click once more.
At this point you will see four choices:
(1)PGPkeys; (2) PGPtools; (3) PGPtray; and (4) Documentation. Select the “PGPkeys” option.
This will launch the PGPkeys window.

2. At the “PGPKeys” window you should see the PGP Key Pair you just created in Step #1. You will need to click on your newly created PGPkey; this will highlight and select your newly created PGPkey. Once the PGPkey is highlighted, click on “Keys” option in the drop- down menu and choose the “Export” function. PGP is now going to ask for a
location to export your PGPkey. In addition to providing a location (Remember this location, because you will need to revisit this later in the Step.) you must be sure to include your “private key(s)”. This is accomplished by checking the small box located in the lower right corner of the Export window.

3. Once you have successfully exported you PGP Key Pair (Including your “private key(s)”) you must now revoke your PGP Key Pair. To do this you will reselect your PGP Key Pair by clicking on your Key Pair, then click on the “Keys” option in the drop-down menu and choose the “Revoke” function. PGP is now going prompt a warning (read the warning and click “yes” to continue) and ask for the passphrase you created in Step #1 while initial creating your PGP Pair Key. After inputting your PGP Passphrase click “Ok” to revoke your PGP Pair Key. You should now see your PGP Pair Key Icon with a red “x” in the PGPkeys window.

4. Now you must repeat Step #1 only this time exporting your “Revoked” PGP Key Pair. Before exporting and saving the “Revoked PGP Key Pair” make sure to rename the Revoked PGP Key Pair so to distinguish it from the Un-Revoked PGP Key.

5. The next step is to delete the revoked key pair from the PGPkeys window. At the PGPKeys window you should see the PGP Key Pair you just revoked. You will need to click on your revoked PGPkey; this will highlight and
select the revoked PGPkey. Once the revoked PGPkey is highlighted, click on the “Edit” option in the drop-down menu and choose the “Delete” function and click “yes” to delete your revoked PGP Key Pair.

6. Next you will need to import your original PGP Key Pair that was saved prior to revoking. From the PGPkeys window, click on the “Key” option in the drop- down menu and choose the “import” function. You will need to select the location where the
original PGP Key Pair was saved; select that file, and click “open”. A second window will appear (Select key(s)) that will list the Key File you just selected. Highlight that file and right click selecting the “key properties” and ensure that the “Implicit Trust” is checked, then click the “import” button. At this point you should see the Imported Key in the PGP Keys window.
You now have a current, un-revoked copy of your PGP Key Pair on your Key Ring and a revoked copy of the PGP Key in a safe place.

Step 3 – Exchanging PGP Keys with other students

This illustrates how pgp key rings can be populated with the public keys of other individuals with whom you wish to securely communicate. This Step will simulate sending and obtaining public keys to a server like MIT’s PGP Public Key Server.

1. Open the PGPKeys window; select your PGP Key Pair from the list highlighting that Key Pair by right clicking. Next choose the “Keys” option from the drop-down menu and select the Export function. This is the location where you will save your Public PGP Key. Make sure when exporting your PGP Key

you DO NOT include your private key. In the lower left hand side of the “Export Key to File” there is a box for including private
keys. Make sure that this box is unchecked; *IF NOT, YOUR “SECRET”PRIVATE KEY WILL BE SENT DEFEATING THE PURPOSE OF PGP*.

2. Next you are going to Import your partner’s PGP Public Key from the pgp-*bin* shared directory on the server. Open your PGP Keys and verify that the new public key you have imported is in place on the key ring.

Step 4 – Signing the new key

PGP requires that trust be established prior to using another person’s public key to communicate securely. This is accomplished by signing their public key with your private key.

1. Right click on the newly imported public key (your partner’s)
2. Select Sign
3. Ensure that the ‘Allow signature to be exported . . . ‘ box is unchecked
4. Have your partner read you the fingerprint for their public key from the display of their system. This should match exactly the fingerprint that is displayed in this window.
5. Once the fingerprint has been verified, click OK
6. Ensure your private key is selected in the drop down box
7. Enter your passphrase and click OK
8. Now the round icon to the right of the key on your key ring should be green
9. Right click on the key again
10. Select Key Properties

11. Slide the bar on the bottom from ‘Untrusted’ to ‘Trusted’
12. Click Close

Step 5 – Encrypting a file using your partner’s public key

PGP can be used to encrypt files so that only specified people can read them. This is accomplished by encrypting the file with the intended recipient’s public key.

1. Create a text file using Notepad. Write a brief secret message (don’t tell your partner what it says) and then save the file as *yourname.txt* to the My Documents folder. Use your real name for this!
2. Open Windows Explorer and browse to the file you just created
3. Right-click on the file and move the mouse over PGP
4. Click on Encrypt
5. Select your partner’s public key from the list by double clicking on it
6. Click OK
7. In the same folder as the .txt file, there should be a file with a similar name which ends in .pgp (i.e. *yourname.txt.pgp*)
8. Copy this file to the *pgp-bin* share

Step 6 – Decrypting the file with your private key

Your intended recipient uses his/her private key to decrypt the file.

1. Open Windows Explorer and browse to the *pgp-bin* share
2. Select the .pgp file that has your partner’s name as part of the filename
3. Double-click on the file
4. In the PGP window, enter your passphrase and click OK
5. In Windows Explorer, double click on the unencrypted file (yourpartner’s name.txt) and read the secret message that your partner wrote

Step 7 – Encrypting and Signing a file

By signing the file with your private key, your intended recipient is assured that it actually came from you and that it has not been modified in transit.

1. From the My Documents folder, open the *yourname.txt* file in Notepad
2. Highlight all of the text
3. Right click on it and select Copy
4. Right click on the PGP Lock in the tray (near the clock)
5. Select Clipboard
6. Click on Encrypt & Sign
7. Double click your partner’s public key and click OK
8. Enter your passphrase and click OK
9. Now the text on the clipboard is signed

10. Go back to Notepad and delete the old text
11. Paste the encrypted/signed text into Notepad
12. Save the file as *yourname-signed.txt* in the *pgp-bin* share

Step 8 – Verifying the signature

1. Open the *yourpartner's-signed.txt* file from the *pgp-bin* in Notepad
2. Highlight all of the text in the file
3. Right Click and select Copy
4. Right click on the PGP lock in the system tray
5. Select Clipboard
6. Click on Decrypt & Verify
7. Notice the status of the signed message (should be 'good')

Step 9 – Sending Secure Email with PGP

1. Open Microsoft Outlook by clicking the icon located on the desktop
2. Click on the Tools menu and select E-mail Accounts
3. Click Add a new e-mail account and then click Next
4. Under Server Type, click IMAP and then click Next
5. In the Your Name box, type your computer's hostname (i.e., isftsstudent1)
6. In the E-mail address box, type your computer's hostname and then @192.168.30.19 (i.e., isftsstudent1@192.168.30.19)
7. In the User Name box, type your computer's hostname (i.e., isftsstudent1)
8. In the Password box, type tartans (for the purposes of this Step, check the Remember password box)
9. In the Server Information boxes (Incoming & Outgoing) you will need to enter the following IP Address "192.168.30.19" in both fields. Once this has been completed click "Next" continue.
10. After your email account is created in Outlook
(may take a few moments to synchronize), Click the PGP Menu (in Outlook at top of screen) and select Options

11. Under Email Options, check all of the boxes and then click OK

12. Now click the Tools menu and select Options
13. Click the Mail Format tab, <u>Uncheck</u> the *Use Microsoft Word…*boxes and then click OK
14. Click File|New|Mail Message
15. In the To: box, type your partner's email address (i.e., isftsstudent2@192.168.30.19)
16. Type anything as a subject and then type a short message to your partner in the body of the message, then click Send
17. Type your passphrase and then click OK (Click the Send/Receive button several times until your message arrives)
18. When you read the message from your partner, notice that your message is verified, decrypted, and displayed in the PGP Secure Viewer.

Step 10 - Adding a Public Key and sending Secure Email

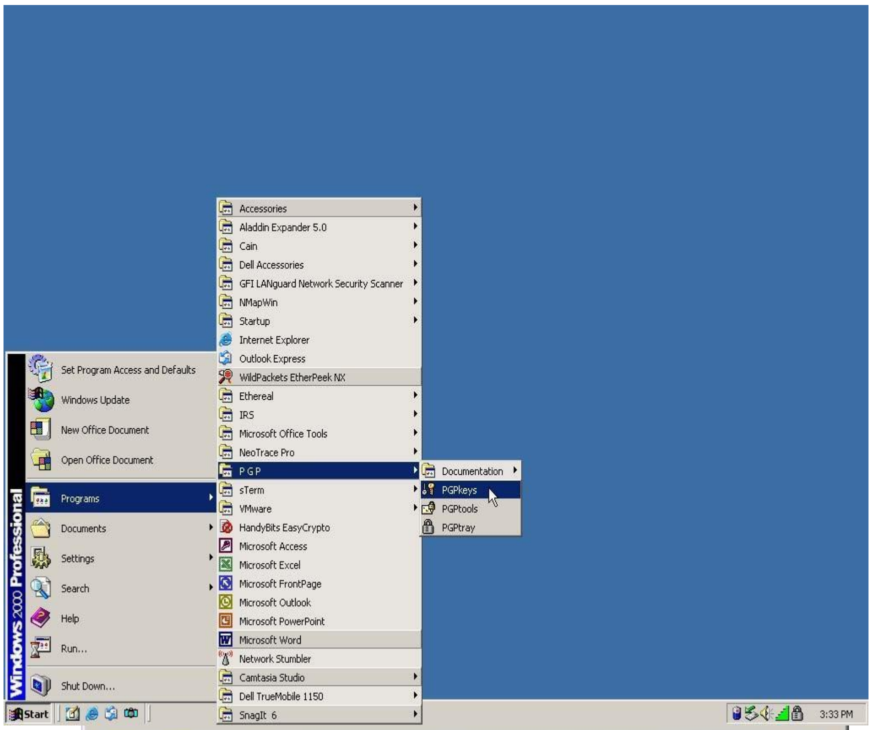
1. In this Step you will need to encrypt an email to the Instructor using PGP. Therefore, the first step is to obtain the instructor's Public PGP Key. The Instructor's PGP Key (isftsinstructor@192.168.30.19) is located in the shared pgp-bin directory at 192.168.30.250.

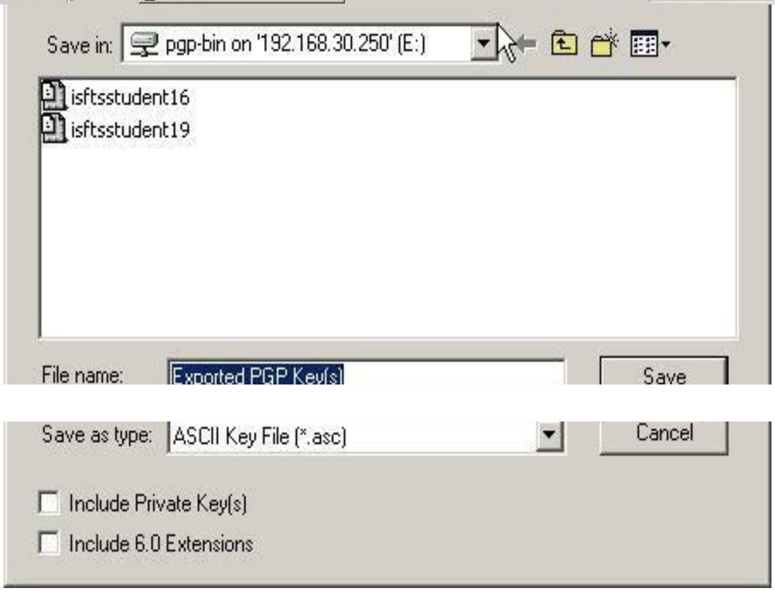
2. Before you can send a secure email you'll need to import the Instructor's Public Key to your Key Ring. To review the process make reference to Step #3.
3. Go into Outlook and send a PGP Secure email message to isftsinstructor@192.168.30.19; the instructor will verify that a message is received from each student

8. Test cases:

1. Try to use another private key instead of original one for decryption process.
2. Use digital signature verification with different public keys.

9. Sample output




Save in: pgp-bin on '192.168.30.250' (E:)

isftsstudent16
isftsstudent19

File name: Exported PGP Key(s)

Save as type: ASCII Key File (*.asc)

☐ Include Private Key(s)
☐ Include 6.0 Extensions

10. Practical Questions:

1. What is PGP?
2. How do PGP works?
3. How PGP encryption works
4. Discuss **about the PGP keys.**
5. Differentiate **PGP and S/MIME.**

11. Exercise Questions

1. Implement PGP software on windows platform.
2. Implement PGP software on Unix platform

PRACTICAL 10: Case Study: Intrusion Detection Systems and Honey pots.

Practical significance:

An Intrusion Detection System (IDS) is a monitoring system that detects suspicious activities and generates alerts when they are detected. Based upon these alerts, a security operations center (SOC) analyst or incident responder can investigate the issue and take the appropriate actions to remediate the threat.
Intrusion detection systems are designed to be deployed in different environments. And like many cybersecurity solutions, an IDS can either be host-based or network-based.

- **Host-Based IDS (HIDS):** A host-based IDS is deployed on a particular endpoint and designed to protect it against internal and external threats. Such an IDS may have the ability to monitor network traffic to and from the machine, observe running processes, and inspect the system's logs. A host-based IDS's visibility is limited to its host machine, decreasing the available context for decision-making, but has deep visibility into the host computer's internals.
- **Network-Based IDS (NIDS):** A network-based IDS solution is designed to monitor an entire protected network. It has visibility into all traffic flowing through the network and makes determinations based upon packet metadata and contents. This wider viewpoint provides more context and the ability to detect widespread threats; however, these systems lack visibility into the internals of the endpoints that they protect.

Due to the different levels of visibility, deploying a HIDS or NIDS in isolation provides incomplete protection to an organization's system. A unified threat management solution, which integrates multiple technologies in one system, can provide more comprehensive security.
Detection Method of IDS Deployment.

Beyond their deployment location, IDS solutions also differ in how they identify potential intrusions:

- **Signature Detection:** Signature-based IDS solutions use fingerprints of known threats to identify them. Once malware or other malicious content has been identified, a signature is generated and added to the list used by the IDS solution to test incoming content. This enables an IDS to achieve a high threat detection rate with no false positives because all alerts are generated based upon detection of known-malicious content. However, a signature-based IDS is limited to detecting known threats and is blind to zero-day vulnerabilities.

- **Anomaly Detection:** Anomaly-based IDS solutions build a model of the "normal" behavior of the protected system. All future behavior is compared to this model, and any anomalies are labeled as potential threats and generate alerts. While this approach can detect novel or zero-day threats, the difficulty of building an accurate model of "normal" behavior means that these systems must balance false positives (incorrect alerts) with false negatives (missed detections).
- **Hybrid Detection:** A hybrid IDS uses both signature-based and anomaly-based detection. This enables it to detect more potential attacks with a lower error rate than using either system in isolation.

IDS vs Firewalls

Intrusion Detection Systems and firewalls are both cybersecurity solutions that can be deployed to protect an endpoint or network. However, they differ significantly in their purposes.

An IDS is a passive monitoring device that detects potential threats and generates alerts, enabling security operations center (SOC) analysts or incident responders to investigate and respond to the potential incident. An IDS provides no actual protection to the endpoint or network. A firewall, on the other hand, is designed to act as a protective system. It performs analysis of the metadata of network packets and allows or blocks traffic based upon predefined rules. This creates a boundary over which certain types of traffic or protocols cannot pass.

Since a firewall is an active protective device, it is more like an Intrusion Prevention System (IPS) than an IDS. An IPS is like an IDS but actively blocks identified threats instead of simply raising an alert. This complements the functionality of a firewall, and many next-generation firewalls (NGFWs) have integrated IDS/IPS functionality. This enables them to both enforce the predefined filtering rules (firewalls) and detect and respond to more sophisticated cyber threats (IDS/IPS). Learn more about the IPS vs IDS debate here.

HONEYPOTS:

What is a honeypot?---Honeypots lie under the category of NIDS. A honeypot is a security resource whose value lies in being probed, attacked, or compromised. A honeypot is a network-attached system set up as a decoy to lure cyber attackers and detect, deflect and study hacking attempts to gain unauthorized access to information systems. The function of a honeypot is to represent itself on the internet as a potential target for attackers -- usually, a server or other high-value asset -- and to gather information and notify defenders of any attempts to access the honeypot by unauthorized users.

Honeypot systems often use hardened operating systems (OSes) where extra security measures have been taken to minimize their exposure to threats. They are usually configured so they appear to offer attackers exploitable vulnerabilities.

How do honeypots work?

Generally, a honeypot operation consists of a computer, applications and data that simulate the behavior of a real system that would be attractive to attackers, such as a financial system, internet of things (IoT) devices, or a public utility or transportation network. It appears as part of a network but is actually isolated and closely monitored. Because there is no reason for legitimate users to access a honeypot, any attempts to communicate with it are considered hostile.

Honeypots are often placed in a demilitarized zone (DMZ) on the network. That approach keeps it isolated from the main production network, while still being a part of it. In the DMZ, a honeypot can be monitored from a distance while attackers access it, minimizing the risk of the main network being breached.

Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as:

- production honeypots
- research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats. Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

Based on design criteria, honeypots can be classified as:

- pure honeypots
- high-interaction honeypots
- low-interaction honeypots

Pure honeypots are full-fledged production systems. The activities of the attacker are monitored by using a bug tap that has been installed on the honeypot's link to the network. No other software needs to be installed. Even though a pure honeypot is useful, stealthiness of the defense mechanisms can be ensured by a more controlled mechanism.

High-interaction honeypots imitate the activities of the production systems that host a variety of services and, therefore, an attacker may be allowed a lot of services to waste their time. By employing virtual machines, multiple honeypots can be hosted on a single physical machine. Therefore, even if the honeypot is compromised, it can be restored more quickly. In general, high-interaction honeypots provide more security by being difficult to detect, but they are expensive to maintain. If virtual machines are not available, one physical computer must be maintained for each honeypot, which can be exorbitantly expensive. Example: Honeynet.

Low-interaction honeypots simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the virtual system's security. Example: Honeyd.

2. **Relevant Program Outcomes**:

PO 2, PO 4, PO 9

3. Competency and practical skills :

The practical is expected to develop the following skills

1. Ability to study and understand IDS and Honeypots.

4. Pre-requisites:

1. Student should have knowledge on Computer Networks

2. Student should have knowledge on C Programming.

5. Resources required:

S .No	Name of the Resource Broad Specification(Approx.)
1	Computer System 1. Processor – 2GHz 2. RAM – 4GB 3. VGA with 1024×768 screen resolution (exact hardware requirement will depend upon the distribution that we choose to work with)
2	Internet

6. Precautions:

1. Check whether the computer is getting proper power or not.
2. Ensure the keyboard, mouse and monitor are properly working.
3. Ensure that there are no power fluctuations while executing the program.
4. Safe working conditions help prevent injury to people and damage to computer equipment.
5. A safe work space is clean, organized, and properly lighted. Everyone must understand and follow safety procedures.
6. Follow electrical safety guidelines to prevent electrical fires, injuries, and fatalities in the home and the workplace. Power supplies and CRT monitors contain high voltage.

7. Practical/Case Study Related questions:

1. What are the types of IDS?
2. What are the differences between IDS and Firewalls?
3. What type of IDS is Honeypots?
4. What are various methods of detecting Intrusion in HIDS?
5. What are the types of Honeypots?
6. How do Honeypots work?