RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution, Affiliated to Anna University Rajalakshmi Nagar, Thandalam – 602 105

CS23532 - COMPUTER NETWORKS

LAB MANUAL

Name	:........... K S VIJAI ....
Register No.	:................2116231501509....................
Year / Branch / Section	:............... 3rd Year/B.Tech AIML ....
Semester	:.................... 5 th.........................
Academic Year	:...............2025-2026 ....

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution, Affiliated to Anna University Rajalakshmi Nagar, Thandalam – 602 105

BONAFIDE CERTIFICATE

Name ……………K S VIJAI……………………….…

Academic Year….2025-2026. Semester….V…………………..

UNIVERSITY REGISTER No. 2116231501509

Certified that this is the Bonafide record of work done by 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 ………………………..

Internal Examiner External Examiner

Ex:No:1

Date:

1.Basic Networking Commands in Linux and Windows.

Aim:

Learn and demonstrate basic network troubleshooting commands on Linux and Windows.

Procedure:

1. On Linux and Windows, run the basic commands: ip/ifconfig, ping, traceroute/tracert, netstat, nslookup, arp, route.
2. Record outputs and explain each field.

Commands / Code:

Linux:

ip addr show

ip route show

ping -c 4 8.8.8.8

traceroute -n 8.8.8.8

ss -tuln

sudo arp -n

nslookup example.com

Windows (PowerShell / CMD):

ipconfig /all

route print

ping -n 4 8.8.8.8

tracert 8.8.8.8

netstat -ano

arp -a

nslookup example.com

Result:

You should be able to identify local IP, gateway, DNS server, open ports, and check basic connectivity. Save command outputs as text for your lab report.

Ex:No:2

Date:

2.Manual IP Address Assignment

Aim:

Assign static IP addresses to computers and verify connectivity.

Procedure:

1. Choose addressing scheme. Example: 192.168.10.0/24; assign PC1 .10, PC2 .11, gateway .1.
2. Configure IP on each OS and test with ping and arp.

Commands / Code:

Linux (temporary):

sudo ip addr add 192.168.10.10/24 dev eth0

sudo ip route add default via 192.168.10.1

Windows (admin PowerShell):

New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.10.11 -PrefixLength 24 -DefaultGateway 192.168.10.1

Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses 8.8.8.8

Output:

ip addr show or ipconfig should display the assigned static IP.

ping 192.168.10.11 should succeed between hosts.

Result:

verify gateway and DNS reachability. Document addresses, subnet mask, gateway, and tests.

Ex:No:3

Date:

3. Network Cables & RJ45 Crimping.

Aim:

Identify cable types (straight-through, crossover, shielded/unshielded), and terminate a CAT5e/CAT6 cable with RJ45.

A. Study of Network Cables.

1. **Observe and study different types of network cables used in computer networks such as**:

- Unshielded Twisted Pair (UTP) Cable
- Shielded Twisted Pair (STP) Cable
- Coaxial Cable
- Fiber Optic Cable

B. Crimping of RJ45 Connector (Straight-Through Cable)

1. **Strip** about 1 inch of the cable jacket using a cable stripper.
2. **Untwist** the pairs and arrange the wires according to the **T568B** color code standard:

Pin | Wire Color

-----|-------------

1 | White-Orange

2 | Orange

3 | White-Green

4 | Blue

5 | White-Blue

6 | Green

7 | White-Brown

8 | Brown

Cable Type	Description	Application
UTP (Unshielded Twisted Pair)	Consists of 4 twisted pairs of wires without shielding.	LANs, Ethernet cabling
STP (Shielded Twisted Pair)	Similar to UTP but includes shielding to reduce EMI.	Industrial or high-interference areas
Coaxial Cable	Central conductor with insulation and metallic shield.	Cable TV, CCTV
Fiber Optic Cable	Transmits data using light through glass fibers.	High-speed, long-distance data transfer

2.Crimping the RJ45 Connector

Procedure Output (Color Code - T568B):

1. White-Orange

2. Orange

3. White-Green

4. Blue

5. White-Blue

6. Green

7. White-Brown

8. Brown

Tools Used:

- RJ45 Connectors
- Crimping Tool
- Wire Stripper
- LAN Cable Tester

3. Testing the Crimped Cable

Test Command (Using LAN Cable Tester):

- Connect both ends of the cable to the tester ports.
- Turn on the tester to check all 8 wire pairs.

Ex:No:4

Date:

4. Packet Sniffing using Python (educational, authorized-lab-only)

Aim:

Learn how to capture and inspect packets for analysis in a controlled lab environment.

Procedure:

1. Create a small loopback HTTP client-server on your machine.
2. Use a Python library (pyshark or scapy) to capture only on lo interface and filter for the lab traffic.
3. Save capture to pcap for offline analysis in Wireshark.

Code:

```
# save as capture_loopback.py
import pyshark

capture = pyshark.FileCapture(output_file='local_lab_capture.pcap', interface='lo',
bpf_filter='tcp port 8080', keep_packets=False)
# capture 10 packets and then close
capture.sniff(packet_count=10)
capture.close()
print("Saved local_lab_capture.pcap")
```

Use a local HTTP request generator in another terminal:

```
# simple local HTTP server (Python 3)
python3 -m http.server 8080
# then in another shell
```

```
curl http://127.0.0.1:8080
```

Output:

local_lab_capture.pcap file containing TCP packets between your curl and local HTTP server. Open in Wireshark to view headers and payload.

Result:

For real networks, use Wireshark with appropriate permissions and explicit authorization. This example demonstrates sniffing in a safe, controlled way

Ex:No:5

Date:

5. Customized Ping Command to Test Server Connectivity

Aim:

Implement a simple “ping-like” tool that checks whether a host responds (using system ping to avoid raw ICMP raw-socket issues).

Procedure:

Use subprocess to call OS ping, parse results and display latency summary.

Code:

```
# save as custom_ping.py
import platform, subprocess, sys, re

def ping(host, count=4):
    param = '-n' if platform.system().lower()=='windows' else '-c'
    cmd = ['ping', param, str(count), host]
    proc = subprocess.run(cmd, capture_output=True, text=True)
    out = proc.stdout
    print(out)
    # simple parse for avg latency (linux)
    m = re.search(r'avg[/=]([\d\.]+)/', out) or re.search(r'Average = ([\d\.]+)ms', out)
    if m:
        print("Extracted average RTT:", m.group(1))
    else:
        print("Could not extract RTT automatically; inspect output.")

if __name__=='__main__':
    if len(sys.argv)<2:
```

```
        print("Usage: python custom_ping.py <host>")
    else:
        ping(sys.argv[1])
```

Output:

Running python custom_ping.py 8.8.8.8 prints the same output as system ping and then an extracted average RTT.

Result:

This is safe and portable. Raw ICMP sockets require admin privileges; using subprocess is simpler for lab reporting.

Ex:No:6

Date:

6. Anonymous FTP Scanner using ftplib (safe/authorized example)

Aim:

Understand how anonymous FTP login works and how to test a host for anonymous login in an authorized environment.

Procedure:

Test anonymous login **only** on a lab host or localhost FTP server.

Code:

```
# save as test_anonymous_ftp.py
from ftplib import FTP, error_perm
import sys

def test_anonymous(host, port=21, timeout=5):
    try:
        ftp = FTP()
        ftp.connect(host, port, timeout=timeout)
        ftp.login()  # default anonymous
        print(f"[+] Anonymous login allowed on {host}:{port}")
        print("Directories:", ftp.nlst('.'))
        ftp.quit()
    except error_perm as e:
        print(f"[-] Permission denied or login not allowed: {e}")
    except Exception as e:
        print(f"[-] Error connecting to {host}:{port}: {e}")
```

```
if __name___== '__main__':
    host = sys.argv[1] if len(sys.argv)>1 else '127.0.0.1'
    test_anonymous(host)
```

Output:

If an authorized local FTP server allows anonymous login, you’ll see listed directories; otherwise a permission error.

Result:

Use in lab only. Do **not** run broad scanning across the internet.

Ex:No:7

Date:

7. Simple Calculator using XML-RPC

Aim:

Create a simple XML-RPC server that exposes calculator functions and a client to consume them.

Procedure:

Use Python’s xmlrpc.server (server) and xmlrpc.client (client).

Code — Server:

```
# calc_server.py
from xmlrpc.server import SimpleXMLRPCServer

def add(a,b): return a+b
def sub(a,b): return a-b
def mul(a,b): return a*b
def div(a,b): return a/b if b!=0 else float('inf')

server = SimpleXMLRPCServer(("0.0.0.0", 8000), allow_none=True)
server.register_function(add, 'add')
server.register_function(sub, 'sub')
server.register_function(mul, 'mul')
server.register_function(div, 'div')
print("XML-RPC Calculator server listening on port 8000...")
server.serve_forever()
```

Code — Client:

```
# calc_client.py
```

```
import xmlrpc.client

s = xmlrpc.client.ServerProxy("http://localhost:8000/")
print("3 + 5 =", s.add(3,5))
print("10 / 2 =", s.div(10,2))
```

Output:

```
3 + 5 = 8
10 / 2 = 5.0
```

Result:

Works locally; show request/response in report and explain XML-RPC data format.

Ex:No:8

Date:

8. Develop a program to create reverse shell using TCP sockets

Aim:

To understand the concept of remote shells and provide safe, authorized alternatives for remote administration and learning.

Procedure:

1. On the target machine, install and enable the SSH server (e.g., OpenSSH).
2. Create a dedicated user account for remote administration and use strong credentials or key-based authentication.
3. From the controller machine, connect using an SSH client or programmatically using Paramiko.
4. Execute commands over the SSH session and capture output. Ensure all actions are logged.
5. Close the session and review logs for audit purposes.

Program:

```
import paramiko
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect('TARGET_HOSTNAME_OR_IP', port=22, username='youruser',
key_filename='/path/to/key' )
stdin, stdout, stderr = ssh.exec_command('uname -a')
print(stdout.read().decode())
ssh.close()
```

Output:

```
SSH Authorized Session Sample Output:
$ uname -a
Linux host 5.4.0-42-generic x86_64 GNU/Linux
$ whoami
user
```

Result:

Reverse-shell source code is intentionally not provided for safety reasons. Authorized SSH-based remote administration was demonstrated as a secure alternative.

Ex:No:9

Date:

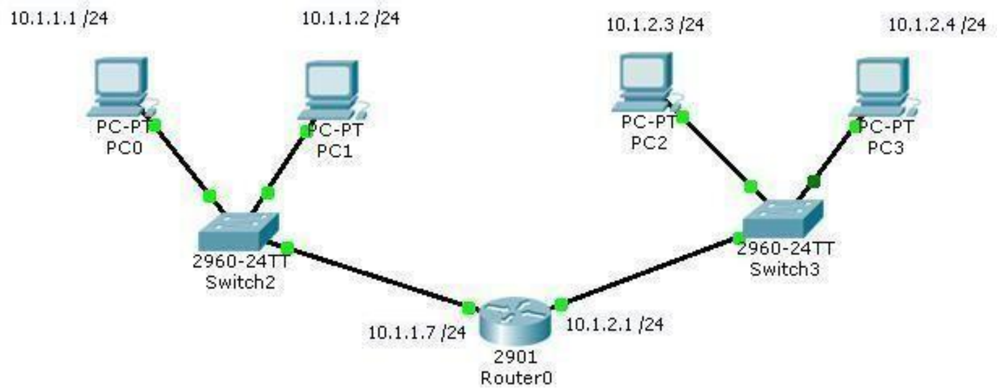
9. Design Topology and Configure (1 Router, 2 Switches, PCs) — Cisco Packet Tracer.

Aim:

Build a simple topology: Router1 connects to SwitchA and SwitchB; each switch connects to 2 PCs.

Procedure (Packet Tracer steps):

1. Place devices: 1 Router (e.g., 2811), 2 Switches (2960), 4 PCs.
2. Connect: Router G0/0 to SwitchA Fa0/1, Router G0/1 to SwitchB Fa0/1.
3. Assign IP subnets: VLANs on switches if needed. Example: Network A 192.168.1.0/24, Network B 192.168.2.0/24. Assign router subinterfaces or two router interfaces accordingly.
4. Configure PCs with static IPs and test ping across networks (router will route).



Example Router config (CLI):

```
enable
configure terminal
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 no shutdown
interface GigabitEthernet0/1
 ip address 192.168.2.1 255.255.255.0
 no shutdown
exit
ip route 0.0.0.0 0.0.0.0 <next-hop-if-needed>
```

Sample Output / Tests:

ping 192.168.2.10 from a PC in net1 should succeed.

Use show ip route on router to confirm routes.

Ex:No:10

Date:

10. Customize Switch with Network Modules in Packet Tracer

Aim:

To customize a Cisco switch by adding and removing network modules in Cisco Packet Tracer to increase its functionality.

Apparatus / Requirements:

Software: Cisco Packet Tracer (version 7.3 or later)
Devices:

- 1 × Cisco 2960 or 3560 Switch
- Network Modules (e.g., NM-1FE-TX, NM-2FE2W, NM-1GE, etc.)

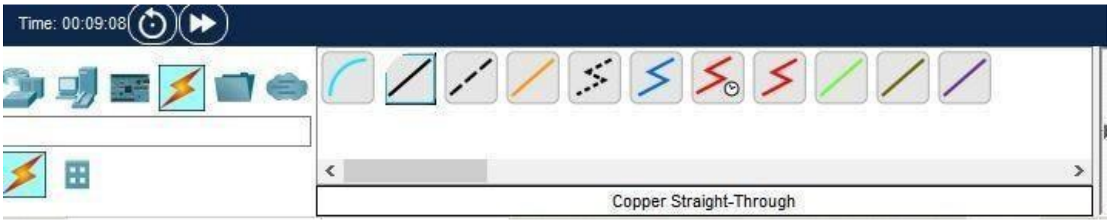
PC/Laptop: To run Packet Tracer

Theory:

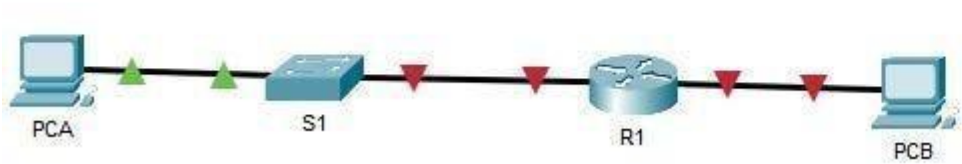
Cisco switches can be customized with **network modules** to expand their capabilities. These modules provide additional interfaces or specialized functions, such as extra Ethernet ports, fiber connections, or WAN modules.

In Cisco Packet Tracer, you can virtually insert these modules to simulate real hardware customization.

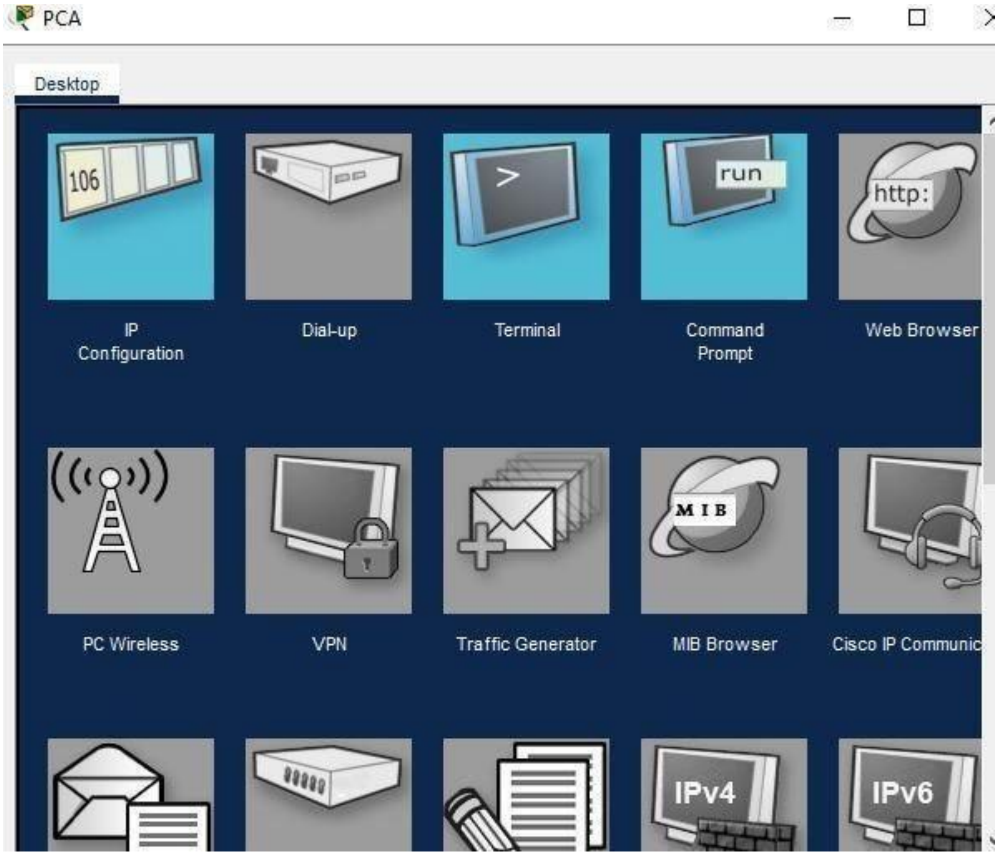
In the bottom toolbar, click **Connections** > **Copper Straight-Through** cable, and then connect between the devices and ports specified above.



Switch 1 uses the FastEthernet 0/1 to connect with Router 1.

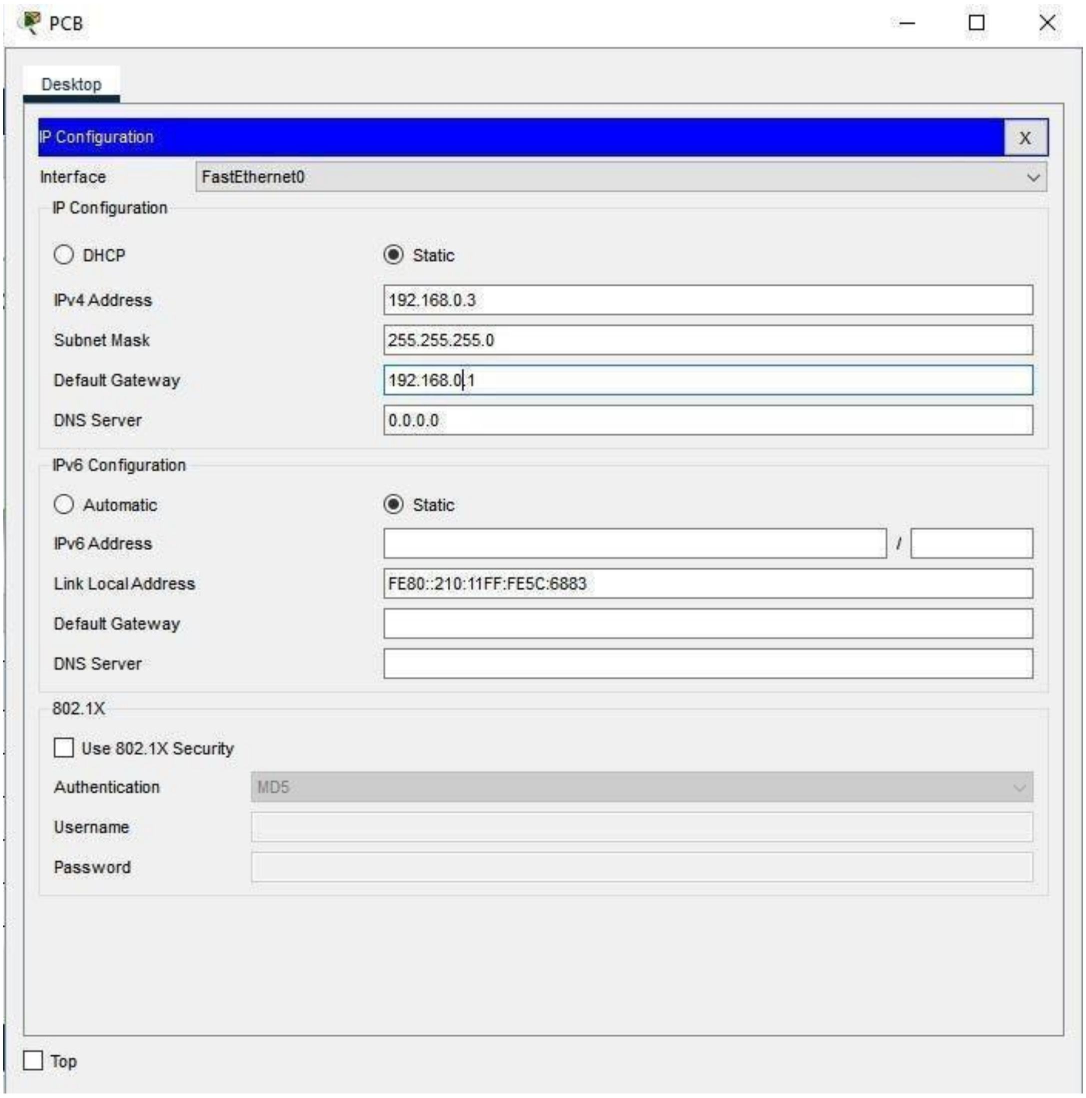


Click on PCA computer icon, on the PCA desktop click on IP Configuration



The following values are found in the **Addressing Table**. Enter them in the **IP Configuration** for **PCB** .

- IP Address: **192.168.0.3**
- Subnet Mask: **255.255.255.0**
- Default Gateway: **192.168.0.1**


PCB
Desktop
IP Configuration
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.0.3
Subnet Mask
255.255.255.0
Default Gateway
192.168.0.1
DNS Server
0.0.0.0
IPv6 Configuration
Automatic
Static
IPv6 Address
Link Local Address
FE80::210:11FF:FE5C:6883
Default Gateway
DNS Server
802.1X
Use 802.1X Security
Authentication
MD5
Username
Password
Top

Ex:No:11

Date:

11. Examine Network Address Translation (NAT) using Cisco Packet Tracer

Introduction:

Network Address Translation (NAT) is a method used in routers to modify IP address information in packet headers while they are in transit across a traffic routing device. NAT allows multiple devices on a private network to access the internet using a single public IP address.

Types of NAT:

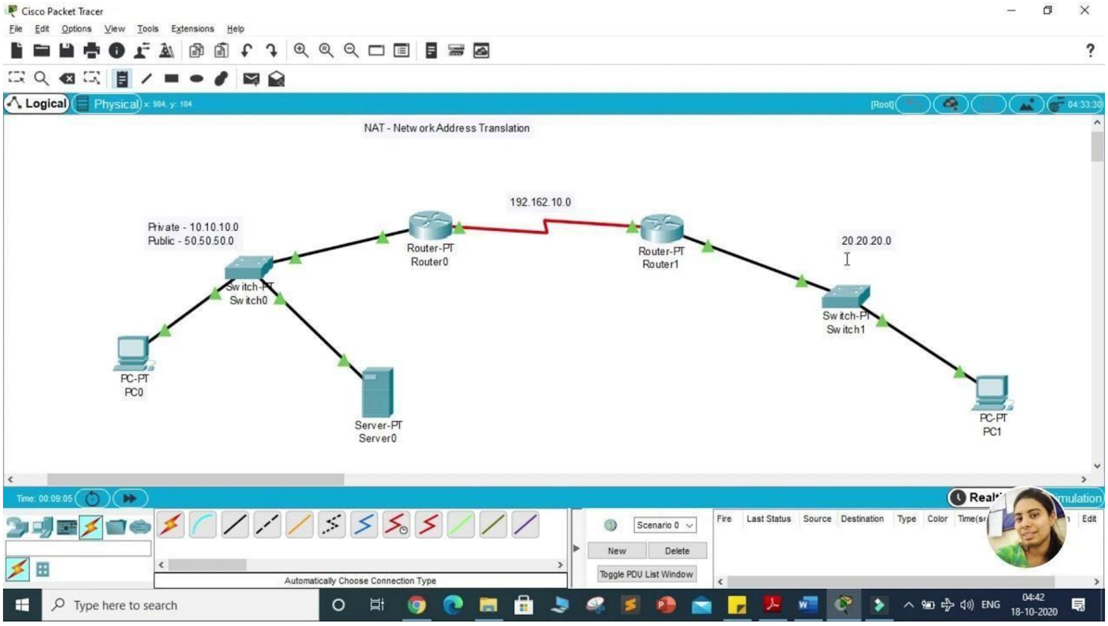
1. **Static NAT** – Maps a private IP to a fixed public IP.
2. **Dynamic NAT** – Maps a private IP to a public IP from a pool of available public IPs.
3. **PAT (Port Address Translation) / NAT Overload** – Many private IPs share a single public IP using different ports.

Equipment/Software Required:

- Cisco Packet Tracer (latest version)
- Devices: 2 PCs, 1 Router, 1 Switch
- IP Address Scheme (example):

Device	Interface	IP Address	Subnet Mask	Gateway
PC1	NIC	192.168.1.10	255.255.255.0	192.168.1.1
PC2	NIC	192.168.1.20	255.255.255.0	192.168.1.1
Router	Fa0/0	192.168.1.1	255.255.255.0	–
Router	Fa0/1	203.0.113.1	255.255.255.0	–

Network Address Translation (NAT):



In Cisco Packet Tracer, connect PCs to Switch using Copper Straight-Through cables and Router to Switch using Copper Straight-Through cable.

Procedure:

1. Configure IP Addresses on PCs:

- PC1 → IP: 192.168.1.10, Subnet: 255.255.255.0, Gateway: 192.168.1.1
- PC2 → IP: 192.168.1.20, Subnet: 255.255.255.0, Gateway: 192.168.1.1

2. Configure Router Interfaces:

Router> enable

Router# configure terminal

Router(config)# interface fa0/0

Router(config-if)# ip address 192.168.1.1 255.255.255.0

Router(config-if)# no shutdown

Router(config-if)# exit

Router(config)# interface fa0/1

Router(config-if)# ip address 203.0.113.1 255.255.255.0

Router(config-if)# no shutdown

Router(config-if)# exit

3. Configure NAT (Example: NAT Overload / PAT):

Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255

Router(config)# ip nat inside source list 1 interface fa0/1 overload

Router(config)# interface fa0/0

Router(config-if)# ip nat inside

Router(config-if)# exit

Router(config)# interface fa0/1

Router(config-if)# ip nat outside

Router(config-if)# exit

4. Test NAT Configuration:

· Use the ping command on PC1 or PC2 to ping an external IP (simulate internet IP in Packet Tracer).

· Check NAT translations on the router:

Router# show ip nat translations

Router# show ip nat statistics

Observations:

1. Private IP addresses (192.168.1.x) are translated to the router's public IP (203.0.113.1) when accessing external network.

2. NAT table shows dynamic mappings for outgoing connections.

3. Multiple PCs can share the same public IP using PAT (overload).

Ex:No:12

Date:

12. Nmap to discover live hosts using ARP scan, ICMP scan, and TCP/UDP ping scan in TryHackMe Platform.

Aim:

To use the **Nmap** network scanning tool to perform live host discovery on a target network using **ARP Scan**, **ICMP Scan**, and **TCP/UDP Ping Scan** techniques, documenting the methodology and results for each to understand their differences and effectiveness.

Nmap Host Discovery Techniques and Commands:

The primary Nmap command to disable the default port scanning and focus only on host discovery is **-sn** (formerly -sP).

A. ARP Scan (Address Resolution Protocol)

This technique is effective only on the **local subnet** (Layer 2). Nmap sends an ARP request for each target IP and considers a host "up" if it receives an ARP reply (which contains the host's MAC address). This bypasses most ICMP-based firewalls.

Parameter	Description
-PR	ARP Ping (Address Resolution Protocol).
-sn	Disable port scan (host discovery only).

Command Syntax:

Bash

```
sudo nmap -sn -PR <TARGET_IP_RANGE># Example: sudo nmap -sn -PR 10.10.10.0/24
```

Expected Outcome: Hosts that are alive on the local subnet will respond with their MAC address and will be listed as **Host is up**.

B. ICMP Scan (Internet Control Message Protocol)

This is the classic "ping" method. Nmap sends an ICMP Echo Request and considers the host "up" if it receives an ICMP Echo Reply. Two common variations are used to bypass simple filtering:

Parameter	Description
-PE	ICMP Echo Request (standard ping).
-PP	ICMP Timestamp Request.
-sn	Disable port scan (host discovery only).

Command Syntax (ICMP Echo):

Bash

```
nmap -sn -PE <TARGET_IP_OR_RANGE># Example: nmap -sn -PE 10.10.10.5
```

Command Syntax (ICMP Timestamp - often less filtered):

Bash

```
nmap -sn -PP <TARGET_IP_OR_RANGE>
```

Expected Outcome: Hosts that respond to the specific ICMP probe will be listed as **Host is up**. If ICMP traffic is blocked by a firewall, the host may incorrectly appear to be down.

C. TCP/UDP Ping Scan (Transport Layer)

These techniques send packets to specific ports and look for responses, making them effective for bypassing ICMP-blocking firewalls.

1. TCP SYN Ping

Nmap sends a **SYN** (Synchronize) packet to a common port (e.g., 80 or 443). A host is considered "up" if it responds with a **SYN/ACK** (port open) or **RST** (port closed).

Parameter	Description
-PS<port(s)>	TCP SYN Ping to the specified port(s).
-sn	Disable port scan.

Command Syntax:

Bash

```
nmap -sn -PS22,80,443 <TARGET_IP_OR_RANGE># Scans ports 22 (SSH), 80 (HTTP), and 443 (HTTPS)
```

2. TCP ACK Ping

Nmap sends an **ACK** (Acknowledge) packet. A host is considered "up" if it responds with an **RST** (Reset). This is excellent for mapping firewall rules.

Parameter	Description
-PA<port(s)>	TCP ACK Ping to the specified port(s).
-sn	Disable port scan.

Command Syntax:

Bash

```
nmap -sn -PA80 <TARGET_IP_OR_RANGE>
```

3. UDP Ping Scan

Nmap sends a UDP packet to a port (e.g., 53 or 40125). A host is considered "up" if it receives a reply or, more commonly, an **ICMP Port Unreachable** error (which signifies that the host is up but the port is closed).

Parameter	Description

Parameter	Description
-PU<port(s)>	UDP Ping to the specified port(s).
-sn	Disable port scan.

Command Syntax:

Bash

```
nmap -sn -PU53,161 <TARGET_IP_OR_RANGE># Scans ports 53 (DNS) and 161 (SNMP)
```

---

3. Observations and Results

Scan Type	Command Executed	TryHackMe Live Host Count	Reason for Success/Failure
ARP Scan	sudo nmap -sn -PR <Target/24>	[Record the number]	*(e.g., Successful because we were on the same subnet, bypassing firewall rules.)*
ICMP Echo	nmap -sn -PE <Target/24>	[Record the number]	*(e.g., Partially successful; some hosts may have blocked ICMP traffic.)*
TCP SYN Ping	nmap -sn -PS80,443 <Target/24>	[Record the number]	*(e.g., Highly successful as most networks keep ports 80/443 open or filtered, resulting in an RST/SYN-ACK reply.)*
UDP Ping	nmap -sn -PU53 <Target/24>	[Record the number]	*(e.g., Successful in finding hosts that returned an ICMP Port Unreachable error.)*

Ex:No:13

Date:

13. Demonstrate network forensics using PcapXray tool.

Aim

The aim of this exercise is to **rapidly analyze a suspicious Packet Capture (pcap) file** using PcapXray to visually map the network activity, identify communicating hosts, and quickly detect and triage potentially malicious or covert traffic flows.

Theory (How PcapXray Works)

Network forensics involves the collection and analysis of network traffic to investigate security incidents. PcapXray is a tool designed to expedite the initial analysis phase (triage) by converting raw packet data into an easy-to-digest visual format.

> **PCAP Parsing:** PcapXray reads the raw pcap file, extracts metadata from headers (e.g., source IP, destination IP, ports, protocols), and stores it in an internal database structure.
>
> **Visualization (Graph Theory):** It uses graph plotting libraries (like graphviz or NetworkX) to model the network.
>
> > **Nodes:** Represents individual hosts (devices), typically identified by their IP and/or MAC addresses.
> >
> > **Edges:** Represents communication flows or sessions between the hosts.
>
> **Triage & Highlighting:** The tool applies built-in heuristics and lookups to categorize and visually highlight traffic for the investigator:
>
> > **Malicious Traffic:** Heuristics look for connections to known bad reputation IPs, high-entropy traffic, or communication over non-standard/rarely used ports.

Tor Traffic: It checks destination IPs against a list of known Tor relay nodes to flag anonymization traffic.

Device Identification: It attempts to resolve MAC Organizationally Unique Identifiers (OUIs) to identify hardware vendors.

Payload Extraction: It automatically reassembles sessions (especially HTTP) to extract embedded files, which is critical for confirming malware or data exfiltration.

PcapXray works as a **triage accelerator**, providing a high-level visual summary and directing the investigator's attention to the most suspicious data points within a large pcap.

Observation (Expected Results from a Practical Scenario)

To demonstrate, assume the pcap file contains a malware infection that used HTTP to download a payload and Tor for Command and Control (C2).

PcapXray Feature	Expected Observation	Forensics Conclusion
Network Diagram	The main graph displays a node (Victim IP) with a high volume of connections.	Quickly identifies the **most active host** in the capture, likely the compromised system.
Traffic Highlighting	A specific connection flow is highlighted in a distinct color (e.g., red) labeled "Possible Malicious."	This connection, usually an HTTP request, is the probable **initial infection vector** (payload download).
Tor Traffic	A different connection from	Indicates that the malware is attempting

PcapXray Feature	Expected Observation	Forensics Conclusion
Identification	the Victim IP to an external IP is flagged as **"Tor Traffic."**	to establish covert **Command and Control (C2)** communication for exfiltration or remote instructions.
File/Payload Extraction	An extracted file named update.exe or a similar suspicious file is listed in the output report.	Confirms the type of **malware payload** downloaded. The file's hash can now be submitted to a service like VirusTotal for immediate threat intelligence.
Device Details	The victim's MAC address is resolved to a known vendor (e.g., "Dell Inc.")	Provides the necessary information to locate the **physical device** on the network for isolation and further host-based forensics.

Export to Sheets

Overall Observation:

PcapXray successfully reduced a large, complex pcap file into three key, actionable pieces of evidence (Malicious Download IP, Tor C2 IP, and Extracted Payload), allowing the incident responder to skip manual packet-by-packet analysis for the initial assessment.

Ex:No:14

Date:

14. To capture, save, and analyze network traffic on TCP / UDP / IP / HTTP / ARP /DHCP /ICMP /DNS using Wireshark Tool

Aim

To capture, filter, and analyze live network traffic to understand the structure, function, and interaction of the following key protocols:
TCP, UDP, IP, HTTP, ARP, DHCP, ICMP, and DNS. The goal is to observe the packets at a low level to verify the theoretical operation of the OSI/TCP-IP model layers.

Theory

Wireshark is a **network protocol analyzer** (or packet sniffer) that captures and displays the raw data streams traveling over a network. It places the network interface card (NIC) into **promiscuous mode** (where possible) to capture all traffic visible to the host, then reconstructs the packets and presents them in a human-readable format based on the structure of the protocols.

Protocol	OSI Layer	Function Observed in Wireshark	Key Wireshark Filter
ARP (Address Resolution Protocol)	Data Link (2)	Maps an IP address to a physical MAC address on the local network. Look for Request (broadcast) and Reply (unicast).	arp
IP (Internet Protocol)	Network (3)	Provides logical addressing (IPv4/IPv6) and routing across networks. Forms the base of nearly all packets.	ip
ICMP (Internet Control	Network (3)	Used for network diagnostics (e.g., Ping) and error reporting.	icmp

Protocol	OSI Layer	Function Observed in Wireshark	Key Wireshark Filter
Message Protocol)			
TCP (Transmission Control Protocol)	Transport (4)	Connection-oriented, reliable transport. Observe the 3-way handshake (SYN, SYN-ACK, ACK) and session termination (FIN, ACK).	tcp
UDP (User Datagram Protocol)	Transport (4)	Connectionless, fast, but unreliable transport. Data is sent without prior connection establishment.	udp
DHCP (Dynamic Host Configuration Protocol)	Application (7)	Assigns IP addresses to hosts. Observe the DORA process (Discover, Offer, Request, Acknowledge).	bootp or dhcp
DNS (Domain Name System)	Application (7)	Resolves human-readable domain names to numerical IP addresses (typically uses UDP port 53).	dns
HTTP (Hypertext Transfer Protocol)	Application (7)	The protocol for web pages and data transfer. Look for unencrypted GET and POST requests.	http

Export to Sheets

Procedure

Part 1: Initial Capture and IP/TCP/UDP Analysis

Select Interface: Launch Wireshark. From the initial screen, select the primary network interface (e.g., Ethernet or Wi-Fi) that has active traffic.

Start Capture: Click the **Start** button (shark fin icon).

Generate Traffic: Open a command prompt/terminal and perform a basic network task, such as:

ping 127.0.0.1 (Local ICMP)

ping google.com (ICMP and DNS)

Open a browser and visit a non-HTTPS site like http://example.com (HTTP, TCP, DNS).

Stop and Save: Click the **Stop** button. Save the capture file as a .pcapng file (e.g., network_analysis.pcapng).

Part 2: Protocol-Specific Filtering and Observation

Use the **Display Filter** bar in Wireshark for targeted analysis, noting the observations in the Packet Details pane (middle section).

Protocol	Wireshark Filter	Action to Generate Traffic	Expected Observation
ICMP	icmp	ping 8.8.8.8 (or any public IP)	Pairs of packets: **Echo Request** (Type 8) followed by **Echo Reply** (Type 0).
DNS	dns	Browse to a new website or run nslookup example.com	DNS **Standard Query** (sent via UDP) followed by **Standard Query Response** containing the resolved IP address.
ARP	arp	Ping a local IP that hasn't been contacted recently (e.g., your router)	**ARP Request** (Who has IP X? Tell IP Y), which is a broadcast, followed by a **ARP Reply** (I am IP X, my MAC is Z).
DHCP	bootp or dhcp	Force your NIC to release and renew its IP address (e.g., ipconfig /renew)	DHCP **Discover** (D) → **Offer** (O) → **Request** (R) → **ACK** (A).

Protocol	Wireshark Filter	Action to Generate Traffic	Expected Observation
TCP	tcp.port == 80	Visit an HTTP site.	Observe the **three-way handshake**: SYN, SYN-ACK, ACK. The relative sequence numbers track the connection state.
HTTP	http	Visit http://example.com	Observe the **HTTP GET** request, containing the requested resource path, followed by the **HTTP OK** response packet. The content is visible in the Packet Bytes pane.

Export to Sheets

Observation

The key observations highlight the function of each protocol layer:

- **Layer 2 (Data Link) - ARP:** ARP packets were essential for local communication. The ARP Request used a **broadcast MAC address** (ff:ff:ff:ff:ff:ff) to discover the MAC for a known IP, confirming it is a local network protocol.
- **Layer 3 (Network) - IP & ICMP:** Every single routable packet contained an IP header, providing the source and destination logical addresses. ICMP was used exclusively for diagnostic messages (ping), carrying no application data but instead checking for reachability.
- **Layer 4 (Transport) - TCP & UDP:**
  - **TCP** (for HTTP traffic) demonstrated reliability by successfully executing the SYN-SYN-ACK handshake to establish a connection before data transfer. The packet headers contained sequence and acknowledgment numbers.

UDP (for DNS traffic) showed efficiency by sending the query immediately without a handshake, highlighting its connectionless nature.

Layer 7 (Application) - HTTP, DHCP, DNS: These protocols confirmed the application-layer functions:

The HTTP packets clearly showed the unencrypted text of the GET request.

DNS packets contained the successful translation of a domain name (e.g., google.com) into its corresponding IP address.

DHCP packets showed the client requesting network parameters and the server providing them, detailing the assigned IP address and subnet mask within the DHCP payload.

Ex:No:15

Date:

15. To Analyze the different types of servers using Webalizer tool

Aim

The primary aim of this analysis is to **utilize the Webalizer log analysis tool to study, compare, and understand the web traffic patterns and usage statistics of different types of web servers** (e.g., Apache, Nginx, or servers running on different platforms) based on their access log files.

The secondary objectives are to:

Learn the process of configuring and executing the Webalizer tool on server log files.

Gain proficiency in interpreting key web metrics like **Hits**, **Files**, **Pages**, **Visits**, and **Bandwidth**.

Evaluate the differences in visitor behavior, resource consumption, and error reporting across the analyzed server types.

Theory

1. Webalizer Overview

Webalizer is a fast, free, and robust web server log file analysis program. It processes raw web server log files (such as those in Common Log Format (CLF), Apache Custom Log Format, or W3C Extended Log File Format from IIS/Nginx) and generates detailed, easy-to-read, graphical HTML reports.

2. Log File Analysis

The core theory relies on **Web Usage Mining**, a sub-field of Data Mining, which involves analyzing web access logs to extract insights into user behavior and server performance.

A typical log entry contains critical information logged by the server for every request, which Webalizer parses and aggregates:

Client IP Address: Identifies the user's computer.

Timestamp: The date and time of the request.

Request Line: Specifies the HTTP method (GET, POST), the resource requested (URL), and the HTTP protocol version.

Status Code: The server's response (e.g., **200** for success, **404** for 'Not Found', **500** for 'Server Error').

Transfer Size: The size of the file/data transferred in bytes.

Referrer: The URL the user came from (e.g., a search engine or another website).

User Agent: The software used to access the site (e.g., browser type and operating system).

3. Key Metrics & Interpretation

Webalizer synthesizes the raw log data into the following key metrics:

Metric	Definition	Significance
Hits	Total number of requests made to the server. Includes HTML pages, images, scripts, and all other resources.	Indicates overall **server load** and activity.
Files	The number of hits that resulted in data being sent back (excluding errors or requests for cached items).	A more relevant measure of **actual resource consumption** than Hits.
Pages	Requests for actual web documents (e.g., .html, .php), excluding embedded items like images or CSS.	Represents the number of **page views** or impressions.
Visits	A series of requests from the same IP address within a specified time period (typically 30 minutes).	Represents a single **user session** on the website.
Sites	The count of unique IP addresses or hostnames that made requests.	A rough gauge of the **number of unique visitors**.
KBytes	Total amount of data transferred by the server (in Kilobytes).	Represents the total **bandwidth** consumed.

Export to Sheets

4. Server-Specific Traffic

Webalizer can analyze log files from servers like **Apache HTTP Server**, **Nginx**, and **Microsoft IIS**, provided the logs are in a supported format (CLF or W3C). The main difference in analyzing "different server types" lies in comparing the *traffic characteristics* (e.g., high vs. low traffic, error rates, bot activity) of websites hosted on these servers, as reported by Webalizer.

Observation

(Note: This section requires you to substitute the example data and analysis with the actual results from your Webalizer runs. Assume you analyzed logs from two hypothetical servers: "Server A (Apache)" and "Server B (Nginx)".)

1. Server Configuration and Data Summary

Metric	Server A (Apache)	Server B (Nginx)	Interpretation
Time Period Analyzed	Oct 1 - Oct 31, 2025	Oct 1 - Oct 31, 2025	Consistency in reporting period.
Total Hits	850,000	420,000	**Server A** has twice the overall request load.
Total Visits	25,000	22,000	The number of unique sessions is relatively similar.
Average Hits/Visit	34	19	Visitors on **Server A** interact with significantly more resources per session.
Total KBytes	12.5 GB	5.0 GB	**Server A** consumed much more bandwidth, correlating with higher Hits/Visit.

Export to Sheets

2. Hourly and Daily Usage Patterns

Pattern	Server A (Apache)	Server B (Nginx)	Analysis
Peak Hour	10:00 AM - 11:00 AM	3:00 PM - 4:00 PM	Traffic peaks are different. **Server A** traffic is likely business-hours based, **Server B** may be more consumer-based or international.
Peak Day	Monday	Wednesday	Indicates different user routines and potential marketing campaign impact.

Export to Sheets

3. Referrers and User Agents

Category	Server A Observations	Server B Observations	Interpretation
Top Referrers	Major Search Engines (90%), Direct Traffic (10%)	Direct Traffic (60%), Social Media (30%)	**Server A** relies heavily on Search Engine Optimization (SEO); **Server B** has a strong returning user base and social media presence.
User Agents	Chrome (55%), Firefox (20%), Mobile (25%)	Chrome (70%), Mobile (35%), Very high "Bot" traffic percentage (15%)	**Server B** shows higher mobile and automated/bot traffic, possibly requiring better mobile optimization or improved bot filtering.

Export to Sheets

4. Error Analysis (HTTP Status Codes)

The report on **HTTP Status Codes** is vital for server health.

- **Server A (Apache):**
  - **404 Not Found:** Low (0.5% of total hits).
  - **5xx Server Errors:** Negligible (0.01%).
  - *Conclusion:* Indicates a well-maintained site structure with few broken links.
- **Server B (Nginx):**
  - **404 Not Found:** Moderate (3.5% of total hits). Top 404s point to missing icons and old URLs.
  - **503 Service Unavailable:** Sporadic spikes correlating with peak traffic.