IT Management & Audits

Practical Lab Manual

Network Traffic Analyzer

Practical P03

Learning Domain

Network Security & Traffic Analysis

Course Learning Outcomes

CLO03: Analyze network traffic patterns and identify security anomalies

Unit

Unit III: Network Security Fundamentals

Time Allocation: 3 hours

Learning Mode: Hands-on (80%) + Theory (20%)

Difficulty Level: Intermediate

Network Traffic Analyzer

Practical P03

Quick Reference

Practical Code	P03
Practical Name	Network Traffic Analyzer
Slot	T/P-3
Duration	3 hours
CLO Mapping	CLO03
Unit	Unit III: Network Security Fundamentals
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	HIGH
Screenshot Count	5 Required

Prerequisites

- Basic understanding of networking concepts (IP addressing, ports, protocols)
- Familiarity with Python programming fundamentals
- Understanding of the OSI model and TCP/IP stack
- Completion of P01 (Lab Setup & Infrastructure) recommended
- Command-line interface proficiency (bash or PowerShell)

Tools Required

Tool	**Version**	**Free**	**Notes**
Python	3.8+	✓	Required
Scapy	2.5+	✓	Packet manipulation library
Wireshark	4.0+	✓	Network protocol analyzer
matplotlib	3.5+	✓	Visualization library
Rich	Latest	✓	Terminal formatting
pip	Latest	✓	Included with Python
Terminal/CLI	-	✓	bash or PowerShell
Web Browser	Latest	✓	For viewing HTML reports

Learning Objectives

- ✓ Understand network traffic fundamentals and packet structure
- ✓ Set up a Python-based network traffic analysis tool from a Git repository
- ✓ Capture and parse PCAP files to extract packet metadata
- ✓ Analyze traffic distribution by protocol, IP address, and port
- ✓ Detect network anomalies including port scans, traffic spikes, and suspicious DNS queries
- ✓ Generate traffic visualizations (protocol pie charts, traffic timelines, top talkers)
- ✓ Produce professional HTML analysis reports
- ✓ Apply network monitoring concepts to Indian regulatory compliance (CERT-In, RBI)

What You Will Learn

By the end of this practical, you will:

1. Understand network packet structure and common protocols (TCP, UDP, ICMP, DNS)
2. Clone and configure a network traffic analyzer tool with Python dependencies
3. Generate and parse PCAP files to extract packet metadata
4. Analyze traffic patterns (protocol distribution, bandwidth, top communicators)
5. Run anomaly detection for port scans, DDoS indicators, and suspicious DNS activity
6. Create traffic visualizations using matplotlib
7. Generate comprehensive HTML analysis reports
8. Apply CERT-In guidelines and RBI network monitoring mandates

Real-World Application

Network traffic analysis is critical for cybersecurity in India. CERT-In mandates that organizations report cyber incidents within 6 hours, requiring continuous network monitoring. The RBI requires financial institutions to implement intrusion detection systems and maintain traffic logs. Companies such as **Paytm**, **PhonePe**, and **Razorpay** deploy real-time monitoring to detect fraudulent transactions, DDoS attacks, and data exfiltration.

Hands-On Procedure

Part A: Environment Setup

Step 1: Clone the Network Traffic Analyzer Repository

Objective: Download and explore the network traffic analyzer tool structure on your local machine.

Instructions:

1. Open your terminal (bash on Linux/Mac, PowerShell on Windows)
2. Navigate to your working directory
3. Clone the repository from GitHub
4. Navigate into the project directory
5. Explore the project structure and understand each module

Code/Command:

```
cd ~/it-audit-labs
git clone https://github.com/it-audit-tools/network-traffic-
    analyzer.git
cd network-traffic-analyzer
ls -la
ls -la src/
# Key files: src/parser.py, analyzer.py, anomaly_detector.py,
#   visualizer.py, reporter.py, cli.py, samples/
```

Clone and Explore Repository

Expected Output

```
src/   parser.py, analyzer.py, anomaly_detector.py,
visualizer.py, reporter.py, cli.py
samples/   generate_sample_pcap.py, sample.pcap
requirements.txt, README.md
```

Read through `src/cli.py` to understand available commands before proceeding.

Step 2: Create Virtual Environment and Install Dependencies

Objective: Set up an isolated Python environment and install all required libraries for network traffic analysis.

Instructions:

1. Create a Python virtual environment
2. Activate the virtual environment
3. Install required dependencies from requirements.txt
4. Verify the installation by running the CLI with the help flag

Code/Command:

```
python -m venv venv
source venv/bin/activate          # Linux/Mac
# .\venv\Scripts\Activate.ps1    # Windows PowerShell
pip install -r requirements.txt
# Installs: scapy, matplotlib, rich, geoip2
python src/cli.py --help
```

Environment Setup

Expected Output

```
Usage: cli.py [OPTIONS] COMMAND [ARGS]...
Commands: parse, analyze, visualize, report
```

Scapy may require administrator/root privileges for live capture. This lab uses pre-captured PCAP files, so elevated privileges are not needed. If you encounter permission errors in Step 8, run the terminal as administrator.

Screenshot 1

What to paste: Terminal output showing the project directory structure (output of `ls -la` and `ls -la src/`) alongside the CLI help output from `python src/cli.py -help`.

Paste your screenshot here

Part B: Traffic Capture & Parsing

Step 3: Generate and Parse a Sample PCAP File

Objective: Generate a sample PCAP file containing various network traffic types, then parse it to extract packet metadata.
Instructions:

1. Run the sample PCAP generator script to create realistic network traffic
2. Verify the generated PCAP file exists and note its size
3. Use the CLI parse command to extract metadata from the PCAP file
4. Review the extracted fields: source/destination IPs, protocols, ports, timestamps, packet sizes

Code/Command:

```
python samples/generate_sample_pcap.py
ls -lh samples/sample.pcap
python src/cli.py parse --input samples/sample.pcap
python src/cli.py parse --input samples/sample.pcap --verbose
python src/cli.py parse --input samples/sample.pcap --limit 20
```

Generate and Parse PCAP

Expected Output

```
Generated: samples/sample.pcap (2.4 MB, 1500 packets)

| No.  | Source IP       | Dest IP         | Proto | SPort | DPort
| Size |
| 1   | 192.168.1.100  | 10.0.0.1        | TCP   | 52341 | 443   |
1280 |
| 2   | 10.0.0.1       | 192.168.1.100   | TCP   | 443   | 52341 |
540  |
| 3   | 192.168.1.105  | 8.8.8.8         | UDP   | 49152 | 53    |
72   |
... (showing 3 of 1500 packets)
Summary: 1500 packets | TCP: 890 | UDP: 450 | ICMP: 160
```

The `-verbose` flag shows TCP flags, TTL values, and payload snippets per packet.

Step 4: Analyze Traffic Distribution

Objective: Perform statistical analysis of the parsed traffic to understand protocol distribution, top communicators, port frequency, and bandwidth usage.
Instructions:

1. Run the analyze command to generate traffic statistics
2. Review the protocol breakdown (TCP, UDP, ICMP percentages)
3. Examine the top source IPs by packet count
4. Examine the top destination IPs and most-used ports
5. Review bandwidth consumption per IP address

Code/Command:

```
python src/cli.py analyze --input samples/sample.pcap
python src/cli.py analyze --input samples/sample.pcap \
    --show-protocols --show-top-ips \
    --show-ports --show-bandwidth
```

Traffic Distribution Analysis

Expected Output

Traffic analysis results showing four sections:

```
=== Protocol Distribution ===
TCP: 890 packets (59.3%) | UDP: 450 (30.0%) | ICMP: 160 (10.7%)

=== Top 5 Source IPs ===
1.  192.168.1.100 - 342 pkts | 2.  192.168.1.105 - 289 pkts | ...

=== Top 5 Destination Ports ===
1.  443 (HTTPS) - 412 | 2.  80 (HTTP) - 198 | 3.  53 (DNS) - 156 |
...

=== Bandwidth per IP (Top 5) ===
1.  192.168.1.100 - 0.82 MB | 2.  10.0.0.50 - 0.54 MB | ...
```

Screenshot 2

What to paste: Terminal output showing the parsed PCAP metadata table with columns for packet number, source IP, destination IP, protocol, source port, destination port, and packet size. Include the summary line showing total packets and protocol counts.

Paste your screenshot here

Part C: Anomaly Detection

Step 5: Run Anomaly Detection on Network Traffic

Objective: Use the anomaly detection module to identify suspicious network behavior including port scans, traffic spikes, and unusual DNS activity.

Instructions:

1. Run the analyze command with the anomaly detection flag enabled
2. Review detected port scan activity (single source hitting multiple ports)
3. Examine traffic spike alerts (unusual volume from a single source)
4. Check for suspicious DNS queries (potential DNS tunneling or exfiltration)
5. Note the severity levels assigned to each anomaly

Code/Command:

```
python src/cli.py analyze --input samples/sample.pcap \
    --detect-anomalies
# Custom thresholds for sensitivity tuning
python src/cli.py analyze --input samples/sample.pcap \
    --detect-anomalies --port-scan-threshold 15 \
    --spike-threshold 100 --dns-entropy-threshold 3.5
```

Anomaly Detection

Expected Output

```
=== ANOMALY DETECTION RESULTS ===
[HIGH] Port Scan Detected
Source:  10.0.0.50 | Unique ports:  16 | Timeframe:  4.2s
Ports:  22, 23, 25, 80, 443, 445, 3306, 3389, ...

[MEDIUM] Traffic Spike Detected
Source:  192.168.1.100 | 142 pkts (avg:  35) | 4.06x spike

[MEDIUM] Suspicious DNS Queries
Source:  192.168.1.105 | Entropy:  4.21 (threshold:  3.5)
Verdict:  Possible DNS tunneling/data exfiltration

Total:  3 anomalies (HIGH: 1 | MEDIUM: 2)
```

In production, these alerts would go to a SOC. Under CERT-In guidelines, confirmed port scans and data exfiltration must be reported within 6 hours.

The `-port-scan-threshold` controls how many unique ports trigger an alert. Lower values increase sensitivity but may produce false positives.

Screenshot 3

What to paste: Terminal output showing the anomaly detection results, including at least one port scan detection (with source IP and scanned ports), one traffic spike alert, and one suspicious DNS query alert with severity levels.

Paste your screenshot here

Part D: Visualization & Reporting

Step 6: Generate Traffic Visualizations

Objective: Create visual charts from the traffic analysis data including protocol distribution pie chart, traffic timeline, and top talkers bar chart.
Instructions:

1. Create an output directory for chart files
2. Run the visualize command to generate all charts
3. Verify the generated chart files in the output directory
4. Open the charts to review the visual representations

Code/Command:

```
mkdir -p charts
python src/cli.py visualize --input samples/sample.pcap \
    --output charts/
ls -la charts/
# Outputs: protocol_distribution.png, traffic_timeline.png,
#   top_talkers.png, port_frequency.png, bandwidth_usage.png
start charts/protocol_distribution.png   # Windows
```

Generate Traffic Visualizations

Expected Output

```
Generating visualizations...  [1/5] to [5/5] saved
All charts saved to:  charts/ (5 PNG images)
```

TCP should dominate (55–65%), followed by UDP (25–35%) and ICMP (5–15%). ICMP exceeding 30% could indicate an ICMP flood attack.

Step 7: Generate Analysis Report

Objective: Create a comprehensive HTML report that summarizes all traffic analysis findings, anomaly detections, and statistics in a shareable format.
Instructions:

1. Generate a terminal-formatted summary report for quick review
2. Generate a full HTML report with embedded charts and tables
3. Open the HTML report in your web browser
4. Review the report sections: executive summary, traffic statistics, anomaly findings, charts, and recommendations

Code/Command:

```
python src/cli.py report --input samples/sample.pcap \
    --format terminal
python src/cli.py report --input samples/sample.pcap \
    --format html --output traffic_report.html
start traffic_report.html    # Windows (use open on Mac)
```

Generate Analysis Reports

Expected Output

```
=== NETWORK TRAFFIC ANALYSIS REPORT ===
Packets:  1,500 | Bandwidth:  2.40 MB
Unique Src IPs:  12 | Unique Dst IPs:  18
Anomalies:  3 (HIGH: 1, MEDIUM: 2)
HTML report saved to:  traffic_report.html
```

The HTML report is designed for sharing with management and audit teams, aligning with CERT-In incident reporting templates.

Screenshot 4

What to paste: The generated traffic visualization charts. Include the protocol distribution pie chart (showing TCP/UDP/ICMP percentages) and the top talkers bar chart (showing source IPs by packet count). You may combine multiple chart images or paste the most representative chart.

Paste your screenshot here

Part E: Custom Analysis

Step 8: Capture and Analyze Your Own Network Traffic

Objective: Capture live network traffic using Wireshark (or use an alternate provided PCAP file) and analyze it using the network traffic analyzer tool.

Instructions:

1. **Option A (Wireshark Capture):**
   a. Open Wireshark and select your active network interface
   b. Start capture and browse a few websites (generate traffic for 30–60 seconds)
   c. Stop the capture and save as a PCAP file
2. **Option B (Alternate PCAP):** Use the provided alternate sample file
3. Run the full analysis pipeline on your captured traffic
4. Compare your real traffic patterns with the sample PCAP analysis
5. Document any anomalies or interesting patterns you observe

Code/Command:

```
# Option B (if no Wireshark): use alternate sample
cp samples/alternate_sample.pcap my_capture.pcap

# Parse, analyze, visualize, and report
python src/cli.py parse --input my_capture.pcap --limit 30
python src/cli.py analyze --input my_capture.pcap \
    --detect-anomalies
mkdir -p my_charts
python src/cli.py visualize --input my_capture.pcap \
    --output my_charts/
python src/cli.py report --input my_capture.pcap \
    --format html --output my_traffic_report.html
```

Custom Traffic Analysis

Expected Output

Custom traffic analysis results vary based on captured traffic. Expect predominantly HTTPS (port 443) traffic, DNS queries, unique IPs for visited websites, and potentially different anomaly patterns than the sample PCAP.

Use Wireshark capture filter `not broadcast and not multicast` to reduce noise. Close unnecessary applications before capture.

Only capture traffic on authorized networks. Unauthorized packet capture may violate the Information Technology Act, 2000 (India). Always obtain permission before monitoring production networks.

Screenshot 5

What to paste: The HTML analysis report opened in your web browser, showing the executive summary section with total packets, protocol breakdown, anomaly count, and at least one embedded chart. If using Wireshark capture, also show the Wireshark capture window briefly.

Paste your screenshot here

Conceptual Background

Network Traffic Fundamentals

Network traffic analysis involves capturing, inspecting, and interpreting data packets traversing a network. The OSI model defines 7 layers, while the practical TCP/IP model uses 4:

OSI Layer	Name	TCP/IP Layer	Key Protocols
7	Application	Application	HTTP, HTTPS, DNS, FTP, SSH
6	Presentation	Application	SSL/TLS, JPEG, ASCII
5	Session	Application	NetBIOS, RPC
4	Transport	Transport	TCP, UDP
3	Network	Internet	IP, ICMP, ARP
2	Data Link	Network Access	Ethernet, Wi-Fi (802.11)
1	Physical	Network Access	Cables, Radio signals

Packet Structure

A network packet consists of nested headers: **Ethernet Header** (MAC addresses, EtherType), **IP Header** (source/destination IPs, TTL, protocol), **Transport Header** (ports, TCP flags/sequence numbers), and **Payload** (application data, encrypted for HTTPS).

Protocol Analysis

Key protocols for traffic analysis: **TCP** (reliable, connection-oriented transfer), **UDP** (fast, connectionless transfer), **ICMP** (diagnostics like ping/traceroute), **DNS** (port 53, name resolution), **HTTP/HTTPS** (ports 80/443, web traffic), **SSH** (port 22, secure remote access), **FTP** (port 21, file transfer), and **SMTP** (port 25, email delivery). See Appendix A for a complete port reference.

Common Anomaly Patterns

Network anomalies indicate potential security threats. Key patterns to detect:

Anomaly	Description	Detection Method
Port Scan	Single source probes many ports on a target (SYN scan, connect scan, UDP scan)	Single source → many ports in short time window
DDoS	Flood attack: volume-based (UDP/ICMP flood), protocol-based (SYN flood), or application-based (HTTP flood)	Traffic spike → single destination, many sources
DNS Tunneling	Data encoded as Base64 subdomains in DNS queries for exfiltration or C2	High Shannon entropy (> 3.5) in domain names
Data Exfiltration	Unauthorized outbound data transfer via large uploads, off-hours activity, or covert channels	Unusual upload volumes, uncommon destinations

Wireshark Basics

Wireshark is the most widely used open-source network protocol analyzer, supporting live capture, deep protocol inspection, display filters, built-in statistics, and PCAP export.

India-Specific Context: CERT-In Guidelines

The Indian Computer Emergency Response Team (CERT-In), under MeitY, is the nodal agency for cybersecurity incident response. Under the April 2022 directives, organizations must: (1) report incidents within 6 hours of detection, (2) maintain logs for 180 days within Indian jurisdiction, (3) synchronize system clocks with NTP servers from NIC/NPL, (4) designate a Point of Contact for CERT-In communication, and (5) provide information to CERT-In upon request.

Reportable incidents include: targeted scanning/probing, system compromise, unauthorized access, website defacement, large-scale attacks, data breaches, and attacks on critical infrastructure. The RBI additionally requires financial institutions to deploy NIDS/NIPS, implement SIEM solutions, conduct periodic vulnerability assessments, maintain audit trails, and establish SOCs for continuous monitoring.

Real-World Example: Network Monitoring in Indian Digital Payments

Company: Paytm / PhonePe (major Indian digital payment platforms)

Network Security Approach:

- Real-time transaction traffic monitoring across payment gateways
- Deep packet inspection for UPI transaction integrity
- Anomaly detection on API traffic to identify fraudulent transaction bursts
- DDoS mitigation using traffic scrubbing centers
- DNS monitoring to prevent phishing attacks on payment URLs
- Compliance with CERT-In 6-hour incident reporting

Assessment & Deliverables

Assessment Questions

Answer the following questions in your submission:

Q1. Explain the difference between TCP and UDP protocols. In what scenarios would each be preferred? Give examples of applications using each.

Q2. What is a port scan? Describe two different types of port scans and explain how each can be detected through traffic analysis.

Q3. Your anomaly detector flagged a "suspicious DNS query" with high entropy in the subdomain. What could this indicate? What steps would you take to investigate further?

Q4. How does the OSI model help in understanding network traffic analysis? Which layers are most relevant for security monitoring?

Q5. Under CERT-In directives, what is the mandatory incident reporting timeline? List at least four types of incidents that must be reported.

Q6. Explain the concept of DNS tunneling. How can an attacker use DNS queries to exfiltrate data from a network?

Q7. Your traffic analysis shows that 85% of outbound traffic from a single host is going to an unknown external IP address during non-business hours. What are the potential implications and what actions would you recommend?

Q8. How do RBI network monitoring mandates differ from general CERT-In requirements? Why do financial institutions need additional network security measures?

Grading Rubric

Criteria	**Description**	**Points**	**Score**
Setup	Repo cloned, venv created, deps installed	10	___/10
PCAP Parsing	Sample PCAP generated and parsed correctly	10	___/10
Traffic Analysis	Protocol, IP, port, bandwidth analysis	15	___/15
Anomaly Detection	Port scans, spikes, DNS anomalies found	20	___/20
Visualization	Charts generated (pie, bar, timeline)	10	___/10
Report Generation	HTML report created and reviewed	10	___/10
Custom Analysis	Own traffic captured and analyzed	15	___/15
Documentation	Assessment answers complete	10	___/10
	TOTAL	**100**	___/100

Deliverables Checklist

Item	**Description**	**Type**	**Status**
Screenshot 1	Repo structure & CLI help	Paste	☐
Screenshot 2	Parsed PCAP metadata table	Paste	☐
Screenshot 3	Anomaly detection results	Paste	☐
Screenshot 4	Traffic visualization charts	Paste	☐
Screenshot 5	HTML report in browser	Paste	☐
PCAP File	Custom capture or alternate sample	File	☐
HTML Report	Generated traffic_report.html	File	☐
Answers	Assessment Q1–Q8 responses	Text	☐

Verification Checklist

Complete all items below before submitting:

☐ Repository cloned and virtual environment set up

□ CLI tool runs successfully with `-help`

□ Sample PCAP file generated using the provided script

□ PCAP file parsed with metadata table displayed

□ Traffic distribution analysis completed (protocols, IPs, ports, bandwidth)

□ Anomaly detection executed with results showing port scan, spike, and DNS alerts

□ Visualization charts generated and saved to output directory

□ HTML analysis report generated and opened in browser

□ Custom traffic captured (Wireshark) or alternate PCAP analyzed

□ Custom traffic report generated

□ All 5 required screenshots captured and pasted

□ All 8 assessment questions answered

Appendix A: Common Protocols Reference

Protocol	Port	Transport	Description
FTP	20/21	TCP	File transfer (data/control)
SSH	22	TCP	Secure shell remote access
Telnet	23	TCP	Unencrypted remote access (deprecated)
SMTP	25	TCP	Email sending
DNS	53	TCP/UDP	Domain name resolution
DHCP	67/68	UDP	Dynamic IP address assignment
HTTP	80	TCP	Unencrypted web traffic
NTP	123	UDP	Network time synchronization
SNMP	161	UDP	Network device management
HTTPS	443	TCP	Encrypted web traffic (TLS)
SMB	445	TCP	File sharing (Windows)
MS SQL	1433	TCP	Microsoft SQL Server
MySQL	3306	TCP	MySQL database
RDP	3389	TCP	Remote Desktop Protocol
PostgreSQL	5432	TCP	PostgreSQL database
HTTP Alt	8080	TCP	Alternative HTTP port

Appendix B: Wireshark Filter Quick Reference

Type	Filter	Description
Capture	`host 192.168.1.100`	Traffic to/from specific IP
Capture	`net 192.168.1.0/24`	Traffic from a subnet
Capture	`port 80`	Traffic on port 80
Capture	`tcp`	Only TCP traffic
Capture	`not broadcast`	Exclude broadcast traffic
Display	`ip.addr == 192.168.1.100`	Show specific IP
Display	`tcp.port == 443`	Show HTTPS traffic
Display	`dns`	Show DNS queries/responses
Display	`tcp.flags.syn == 1`	Show SYN packets
Display	`icmp`	Show ICMP traffic
Display	`frame.len > 1000`	Large packets (> 1000 bytes)

Appendix C: Scapy Command Reference

```python
from scapy.all import *
packets = rdpcap("sample.pcap")    # Read PCAP
pkt = packets[0]
if pkt.haslayer(IP):
    print(f"Src:␣{pkt[IP].src},␣Dst:␣{pkt[IP].dst}")
```

```
if pkt.haslayer(TCP):
    print(f"Dport:␣{pkt[TCP].dport},␣Flags:␣{pkt[TCP].flags}")
tcp_pkts = [p for p in packets if p.haslayer(TCP)]
wrpcap("output.pcap", packets)    # Write PCAP
```

Scapy Quick Reference

Appendix D: Troubleshooting Guide

Common Issues and Solutions

Problem: `ImportError: No module named scapy` or `PermissionError` when running analysis.
Solutions:

1. Ensure virtual environment is activated: `source venv/bin/activate`
2. Reinstall Scapy: `pip install scapy`
3. On Linux, Scapy may need root for live capture: `sudo python src/cli.py ...`
4. On Windows, install Npcap (required for Scapy): `https://npcap.com`
5. Verify installation: `python -c "from scapy.all import *; print('OK')"`

Problem: `FileNotFoundError` or `Scapy error: Not a pcap file` when parsing.
Solutions:

1. Verify the PCAP file exists: `ls -la samples/sample.pcap`
2. Regenerate the sample: `python samples/generate_sample_pcap.py`
3. If using Wireshark export, ensure format is “Wireshark/pcap” (not pcapng)
4. In Wireshark: File → Save As → select “Wireshark/tcpdump/... - pcap” format
5. Check file is not corrupted: `file samples/sample.pcap` (should show “pcap capture file”)

Problem: Charts not saving, blank images, or `_tkinter.TclError: no display`
Solutions:

1. On headless servers, set backend before import: `export MPLBACKEND=Agg`
2. Ensure output directory exists: `mkdir -p charts`
3. Check write permissions: `ls -la charts/`
4. Install missing system dependency (Linux): `sudo apt install python3-tk`
5. Verify matplotlib version: `pip show matplotlib`
6. Try reinstalling: `pip install -force-reinstall matplotlib`

Appendix E: Additional Resources

Official Documentation

→ CERT-In Official Site: `https://www.cert-in.org.in`

→ Wireshark Documentation: `https://www.wireshark.org/docs/`

→ Scapy Documentation: `https://scapy.readthedocs.io`

→ RBI IT Framework: `https://www.rbi.org.in`

→ NIST Cybersecurity Framework: `https://www.nist.gov/cyberframework`

RFC References and Learning Resources

- RFC 793 (TCP), RFC 768 (UDP), RFC 792 (ICMP), RFC 1035 (DNS), RFC 8446 (TLS 1.3)
- "Practical Packet Analysis" – Chris Sanders (Wireshark guide)
- "Network Security Monitoring" – Chris Sanders (No Starch Press)
- "Black Hat Python" – Justin Seitz (Scapy and network tools)

Tools Used in This Practical

Tool	Purpose	Cost
Python 3.8+	Programming language runtime	Free
Scapy	Packet manipulation and PCAP parsing	Free
Wireshark	Network protocol analyzer (GUI)	Free
matplotlib	Data visualization and charting	Free
Rich	Terminal output formatting and tables	Free
geoip2	IP geolocation lookups	Free
pip	Python package manager	Free
Git	Version control and repository clone	Free

—END OF LAB MANUAL—

Document Version: 1.0

IT Management & Audits – Practical Lab Series