Government Polytechnic Kanpur
POST GRADUATE DIPLOMA IN CYBER SECURITY

LAB MANUAL

Networking Concepts & Security Lab Manual

Lab Title: [**Networking Concepts & Security Lab Manual**]
Subject: [Networking Concepts & Security]
Course: [POST GRADUATE DIPLOMA IN CYBER SECURITY]
Semester: [1st Semester)]
Session: [2024-2025]
Lab Instructor: [**Mr. Himanshu Singh**]

Submitted by:
Name:______________________________________
Roll Number:_______________________________
EnRollment Number:_________________________
Class: [INFORMATION TECHNOLOGY 5TH Semester]

INDEX

Networking Concepts & Security SYLLABUS:

DETAILED CONTENTS

Unit I:

Introduction to Network Security
Types of networks,
IP Address,
NAT,
IP Subnets,
DHCP Server,
Ports,
DNS,
Proxy Servers,
Virtual Private Networks,
DNS Server,
OSI and TCP/IP Model,
TCP Vs. UDP,
Routers,
Switches,
Endpoint solutions,
Access Directory,
TOR Network.
Networking Devices (Layer1,2,3) -
Different types of network layer attacks–
Firewall (ACL, Packet Filtering, DMZ, Alerts and
Audit Trails) – IDS, IPS and its types (Signature based, Anomaly based, Policy based, Honeypot based)
and setup.

Unit II:

Virtual Private Networks

VPN and its types
–Tunnelling Protocols
– Tunnel and Transport Mode
–Authentication Header
Encapsulation Security Payload (ESP)-
IPSEC Protocol Suite – IKE PHASE 1,
II – Generic Routing
Encapsulation (GRE).

Implementation of VPNs.

Unit III:

Network Attacks Part 1

Sniffing concepts,
Sniffing Techniques
MAC Attack,
DHCP attack,
ARP poisoning,
Spoofing,
DNS poisoning.
Wireshark,
packet analysis,
display and capture filters,
Ettercap,
sniffing counter
measures,
sniffing protection tools.
Denial of service (DOS)/Distributed Denial of service (DDOS):
Concepts, DOS/DDOS Technique,
Botnets,
DDOS, DOS/DDOS attacking tools,
DOS/DDOS counter Measures,
DOS/DDOS
protection tools.
Vulnerability scanning tools:
Concepts, Scanning Techniques,
Tools: Nessus,
OpenVAS,
Sparta,
Nexpose,
Nmap.
Network Scanning Report Generation,
Striping,
Router attacks,
VPN pentesting,
VOIP pentesting,
Enumeration techniques:
SMTP,
SNMP,
IPsec, VOIP,
RPC,

Telnet,
FTP,
TFTP,
SMP,
IPV6 and BGP.

Unit IV: Network Attacks Part 2

Network Exploitation OS Detection in network,
Scanning: nmap, open ports,
filtered ports,
service detection,
metasploit framework,
interface of metasploit framework,
network vulnerability
assessment, evade anti viruses and firewalls,
metasploit scripting,
exploits,
vulnerabilities,
payloads,
custom payloads
, nmap configuration,
Social Engineering toolkit,
Xero sploit Framework,
exploits
delivery, burp-suite,
End Point Security.

Unit V: Wireless Attacks

Wireless concept, wireless encryption, wireless threats, wireless hacking methodology, wireless hacking and security tools, Bluetooth hacking, countermeasures to wireless threats, Protocols, MAC
Filtering, Packet Encryption, Packet Sniffing, Types of authentications, ARP Replay attack, Fake Authentication Attack, De authentication, Attacks on WEP, WPA and WPA-2 Encryption, fake hotspots, evil twin attack, fluxion framework

List of Practical’s:

1. **Brute force attack using open-source tools.**
2. **Identifying network attacks using Nmap, Metasploit.**
3. **Selecting a Capture Interface and creating the first pcap file using Wireshark.**
4. **Using Capture filters in Wireshark.**
5. **Finding a Text String in a Trace File using Wireshark.**
6. **Understanding Packet Loss and Recovery process.**
7. **Identifying DOS & DDOS Attack.**
8. **VPN & VOIP pentesting using open-source tools.**
9. **Demonstration of IDS using or any other open-source tool.**
10. **Demonstration of IPS using snort or any other open-source tool.**

List of Practical’s:

1. Brute force attack using open-source tools.
2. Identifying network attacks using Nmap, Metasploit.
3. Selecting a Capture Interface and creating the first pcap file using Wireshark.
4. Using Capture filters in Wireshark.
5. Finding a Text String in a Trace File using Wireshark.
6. Understanding Packet Loss and Recovery process.
7. Identifying DOS & DDOS Attack.
8. VPN & VOIP pentesting using open-source tools.
9. Demonstration of IDS using or any other open-source tool.
10. Demonstration of IPS using snort or any other open-source tool.

Lab Manual: Understanding Packet Loss and Recovery Process

Objective:

To understand how packet loss occurs in a network, how it's detected using network monitoring tools like Wireshark, and the recovery mechanisms implemented to mitigate the effects of packet loss.

Requirements:

- Wireshark (installed on your system)
- A network environment where packet loss can be simulated (optional: network emulator or packet generator like `tc` on Linux)
- A PCAP file with instances of packet loss (can be captured during testing)

Background Information:

Packet Loss: Packet loss occurs when one or more packets of data traveling across a computer network fail to reach their destination. It can be caused by:

- Network congestion
- Faulty hardware

- Software issues
- High latency connections
- Interference in wireless networks

Effects of Packet Loss:

- Reduced data throughput
- Increased latency
- Degraded quality of real-time applications (e.g., voice, video)

Packet Recovery Mechanisms:

- **TCP (Transmission Control Protocol):** TCP includes mechanisms like retransmission, timeouts, and acknowledgment (ACK) to recover from packet loss.
- **UDP (User Datagram Protocol):** UDP does not inherently recover from packet loss; additional application-level protocols (e.g., RTP for video streaming) need to handle recovery.

Step-by-Step Instructions:

1. Capture or Open a Trace File

a. **Open a Pre-existing Capture File:**

- Launch Wireshark.
- Go to `File -> Open` and select a PCAP file that has instances of packet loss.

b. **Capture Network Traffic:**

- If you need to capture live traffic:
  - Select your network interface in Wireshark.
  - Start capturing by clicking on the blue shark fin icon (`Start Capture`).
  - Allow some traffic to flow through the network.
  - Stop the capture by clicking the red square (`Stop Capture`).

If you want to simulate packet loss, you can use tools like `tc` (Linux command line utility) to manipulate traffic parameters (e.g., introducing packet loss).

Command Example (Linux):

```
sudo tc qdisc add dev eth0 root netem loss 10%
```

2. Analyze Network Traffic for Packet Loss

Once you have a PCAP file loaded, look for the signs of packet loss. Packet loss in Wireshark is typically identified by:

- Duplicate ACKs
- Retransmissions
- TCP Fast Retransmissions
- Out-of-order packets
- Gaps in the sequence numbers

3. Use Display Filters to Identify Packet Loss

Wireshark allows you to filter specific packet behaviors that indicate loss and recovery.

Common filters to detect packet loss in TCP traffic:

- **TCP Retransmissions:**

```
tcp.analysis.retransmission
```

- **TCP Fast Retransmissions:**

```
tcp.analysis.fast_retransmission
```

- **Duplicate ACKs:**

```
tcp.analysis.duplicate_ack
```

- **Lost Segments:**

```
tcp.analysis.lost_segment
```

Steps:

- Type any of the above filters into the Wireshark display filter bar.
- Apply the filter to view only the packets that match the criteria.
- In the packet list, you should see retransmitted packets or acknowledgment messages that signal a lost or out-of-order segment.

4. Examining TCP Recovery Mechanisms

Wireshark provides tools to view how TCP attempts to recover from packet loss. These mechanisms include:

a. **Duplicate Acknowledgments (Dup ACK):**

- If a packet is lost, the receiver detects a missing sequence number and sends duplicate ACKs for the last successful packet received.
- In the packet details, you'll see duplicate ACKs with the same acknowledgment number.

b. **Retransmissions:**

- After receiving a series of duplicate ACKs, the sender initiates retransmission of the lost packet.
- Wireshark will highlight this in the packet analysis as a "TCP Retransmission" event.
- Examine the sequence number in the retransmitted packet and compare it to the earlier sequence to confirm a retransmission.

c. **TCP Fast Retransmission:**

- If the sender receives three duplicate ACKs, it assumes packet loss and performs a fast retransmission, skipping the standard timeout period.
- These can be identified in Wireshark using the `tcp.analysis.fast_retransmission` filter.

Example:

- After applying the `tcp.analysis.retransmission` filter, you will see retransmitted packets. Inspect the packet details to observe the sequence numbers and TCP flags, which indicate the recovery in progress.

5. *Identifying UDP Packet Loss*

Since UDP doesn't handle packet loss at the transport layer, recovery mechanisms (if any) must be implemented at the application layer. This makes it harder to identify packet loss directly in UDP traffic. However, certain indicators can help:

a. **Missing Sequence Numbers (Application Level):**

- For application protocols built on UDP (like RTP for video or voice), sequence numbers are typically added at the application layer.
- Use Wireshark to inspect the protocol headers and see if any packets are missing based on sequence numbers.
- For RTP, you can use the `rtp.seq` field to track sequence numbers.

b. **Custom Protocol Analysis:**

- Some application-level protocols include their own error detection and recovery mechanisms. Look for retransmissions or re-requests in protocols like TFTP, RTP, or other media streams.

6. Simulating Packet Loss (Optional)

If you have a controlled environment (testbed), you can simulate packet loss and recovery to study its effects:

a. **Using a Network Emulator:**

- Tools like `tc` (on Linux) or `WANem` allow you to artificially introduce packet loss and study its impact on network traffic.

Example (Linux `tc` command to introduce 10% packet loss):

```
sudo tc qdisc add dev eth0 root netem loss
10%
```

b. **Monitoring the Effects:**

- Capture traffic with Wireshark while running an application (e.g., a file transfer using FTP or HTTP) and analyze the resulting packet loss and recovery efforts (such as retransmissions).

7. Calculating Packet Loss Percentage

You can also calculate the percentage of packet loss in Wireshark using the `Statistics` menu:

a. **TCP Stream Analysis:**

- Go to `Statistics -> TCP Stream Graphs -> Throughput`.

- This graph allows you to see where packet loss and retransmissions occurred in the stream.

b. **Packet Loss by Packet Count:**

- Use `Statistics -> Summary` to view the overall number of packets sent and received.
- To calculate packet loss, you can compare the number of retransmitted packets to the total number of packets in the capture.

Example Scenario:

Objective:

Understand packet loss during a file download over TCP and observe recovery through retransmissions.

1. **Capture network traffic** while downloading a file over HTTP.
2. **Simulate packet loss** using a network emulator (`tc`) to introduce 5% packet loss during the download.
3. Apply filters like `tcp.analysis.retransmission` and `tcp.analysis.duplicate_ack` to identify lost and retransmitted packets.
4. **Follow the TCP stream** to see how packet loss affects the throughput and recovery process.
5. Calculate the **percentage of packet loss** by analyzing the statistics in Wireshark.

Lab Questions:

1. What are the main causes of packet loss in a network?
2. How does TCP recover from packet loss, and what role do duplicate ACKs play?
3. How would you identify packet loss in a UDP-based protocol like RTP?
4. Can you observe packet loss directly, or do you rely on secondary indicators (e.g., retransmissions)?
5. How does packet loss impact real-time applications like voice or video calls compared to file transfers?

Conclusion:

Packet loss is a critical factor in network performance, affecting everything from web browsing to real-time applications like voice and video. By using Wireshark to identify and analyze packet loss events, you can better understand how different protocols like TCP and UDP handle recovery and maintain reliable communication. This lab provided an introduction to detecting packet loss, understanding recovery mechanisms, and simulating loss for testing purposes.