Installation of Wire shark, tcpdump, etc and observe data transferred in client server communication using UDP/TCP and identify the UDP/TCP datagram

able of Contents

1. Introduction

In this lab, you will learn how to install and use network analysis tools like Wireshark and tcpdump. You will also capture and analyze network traffic to observe data transferred in client-server communication using UDP and TCP protocols.

2. Objectives

- Install Wireshark and tcpdump.
- Capture network traffic using Wireshark and tcpdump.
- Observe and analyze data transferred in client-server communication using UDP/TCP.
- Identify and interpret UDP/TCP datagrams.

3. Prerequisites

- Basic knowledge of networking concepts.

- A computer with internet access.
- Administrative privileges to install software.

4. Lab Setup

- A computer running a Linux or Windows operating system.
- Wireshark and tcpdump installed.
- A server and client setup for testing UDP/TCP communication (can be done on the same machine using different ports or on different machines).

5. Installing Wireshark

On Windows

1. Download the Wireshark installer from the Wireshark website.
2. Run the installer and follow the on-screen instructions to complete the installation.

On Linux

For Debian-based distributions (e.g., Ubuntu):

```
sudo apt update

sudo apt install wireshark
```

For Red Hat-based distributions (e.g., CentOS, Fedora):

```
sudo yum install wireshark
```

6. Installing tcpdump

On Windows

1. Download the WinDump from the WinDump website.
2. Install WinDump by following the on-screen instructions.

On Linux

For Debian-based distributions (e.g., Ubuntu):

```
sudo apt update

sudo apt install tcpdump
```

For Red Hat-based distributions (e.g., CentOS, Fedora):

```
sudo yum install tcpdump
```

. Capturing and Analyzing Network Traffic

Using Wireshark

1. Open Wireshark.
2. Select the network interface to capture traffic from (usually `eth0` or `wlan0` for wired or wireless connections, respectively).
3. Click on the "Start" button to begin capturing packets.
4. To stop capturing, click on the "Stop" button.
5. Analyze the captured packets by applying filters and examining the packet details.

Using tcpdump

1. Open a terminal.
2. Start capturing packets on a network interface (e.g., `eth0`) with the following command:

```
sudo tcpdump -i eth0
```

3.To stop capturing, press `Ctrl+C`.

4.To save the captured packets to a file for later analysis, use the `-w` option:

```
sudo tcpdump -i eth0 -w capture.pcap
```

5.To read a saved capture file:

```
tcpdump -r capture.pcap
```

8. Client-Server Communication Using UDP

UDP Server

Create a simple UDP server in Python:

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024)
    print("Received message:", data.decode())
```

UDP Client

Create a simple UDP client in Python:

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, UDP!"

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(MESSAGE.encode(), (UDP_IP, UDP_PORT))
```

9. Client-Server Communication Using TCP

TCP Server

Create a simple TCP server in Python:

```
import socket

TCP_IP = "127.0.0.1"
TCP_PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((TCP_IP, TCP_PORT))
sock.listen(1)

conn, addr = sock.accept()
print("Connection address:", addr)
while True:
    data = conn.recv(1024)
    if not data:
        break
    print("Received message:", data.decode())
    conn.send(data)
conn.close()
```

TCP Client

Create a simple TCP client in Python:

```
import socket

TCP_IP = "127.0.0.1"
TCP_PORT = 5005
MESSAGE = "Hello, TCP!"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((TCP_IP, TCP_PORT))
sock.send(MESSAGE.encode())
data = sock.recv(1024)
sock.close()

print("Received message:", data.decode())
```

10. Identifying UDP/TCP Datagrams

Using Wireshark

1. Start capturing packets in Wireshark.
2. Run the UDP/TCP client and server scripts.
3. Stop capturing packets in Wireshark.
4. Use filters to isolate UDP (`udp`) and TCP (`tcp`) packets.
5. Analyze the details of the packets to identify the datagrams.

Using tcpdump

1. Start capturing packets with tcpdump:

```
sudo tcpdump -i eth0 -w capture.pcap
```

2. Run the UDP/TCP client and server scripts.
3. Stop capturing packets by pressing `Ctrl+C`.
4. Read the captured packets:

```
tcpdump -r capture.pcap
```

5. Use filters to isolate UDP and TCP packets:

```
tcpdump -r capture.pcap udp
tcpdump -r capture.pcap tcp
```

6. Analyze the details of the packets to identify the datagrams.

11. Conclusion

In this lab, you learned how to install and use Wireshark and tcpdump to capture and analyze network traffic. You also observed data transferred in client-server communication using UDP and TCP protocols and identified UDP/TCP datagrams.

12. References

- Wireshark Official Website
- tcpdump Official Website
- Python socket module documentation