BARI ANKIT (56)

Exp 7 : Study of packet sniffer tools wireshark, :- 1. Observer performance in promiscuous as well as non-promiscuous mode. 2. Show the packets can be traced based on different filters.

Study of Packet Sniffer Tools: Wireshark

Wireshark is one of the most popular and powerful network protocol analyzers. It allows users to capture, analyze, and display packets of data transmitted over a network. Wireshark is widely used for network troubleshooting, performance analysis, and security auditing.

In this study, we will explore Wireshark's performance in promiscuous vs. non-promiscuous mode, and how different filters can be used to trace packets effectively.

---

1. Performance in Promiscuous vs. Non-Promiscuous Mode

Promiscuous Mode:

- **Definition: Promiscuous mode allows a network interface card (NIC) to receive all packets on the network, even those not intended for the system. This mode is particularly useful for sniffing traffic on a network to monitor or analyze data packets flowing through.**
- **Use Case: It is commonly used for network monitoring and packet capture to observe all traffic on a network, even from devices that are not directly communicating with the machine running Wireshark.**

Non-Promiscuous Mode:

- **Definition: In non-promiscuous mode, a network interface card (NIC) only receives packets that are addressed to it. This is the default mode for most network interfaces.**
- **Use Case: Non-promiscuous mode is typically used when you want to monitor traffic only intended for your machine, and it’s less intrusive than promiscuous mode.**

How to Set the Modes in Wireshark:

- **By default, Wireshark will capture packets in promiscuous mode if your network adapter supports it. You can change this setting in Wireshark’s preferences.**

Steps to enable/disable promiscuous mode in Wireshark:

1. **Open Wireshark.**
2. **Go to Capture > Options (or click the gear icon next to the interface).**
3. **Under the interface options, look for the checkbox that says "Enable promiscuous mode".**

4. **Enable or disable the checkbox depending on whether you want to capture packets in promiscuous or non-promiscuous mode.**

Observing Performance Differences:

- **Promiscuous Mode Performance:**
  - **Higher packet capture: In this mode, your NIC captures all packets on the network, including those not addressed to your device. This means Wireshark will capture more traffic and can be useful for network troubleshooting or monitoring all devices in the network.**
  - **Potential overhead: Since your NIC is capturing all traffic, this can lead to increased network load and higher CPU usage on the system running Wireshark, especially in large or heavily trafficked networks.**
- **Non-Promiscuous Mode Performance:**
  - **Less traffic captured: In this mode, Wireshark only captures packets intended for your machine (broadcast, unicast, and multicast traffic). The amount of traffic observed is smaller, which means less data to process.**
  - **Lower network load: Since only relevant packets are captured, the system's CPU and memory usage are lower, which can be advantageous in low-impact scenarios.**

Performance Impact:

- **Promiscuous Mode:**
  - **The amount of captured data is larger, so analyzing large packets or frequent traffic might cause higher memory usage.**
  - **In some cases, performance degradation can occur, especially on older machines or slow network environments.**
- **Non-Promiscuous Mode:**
  - **Less data to capture and analyze, resulting in lower overhead on system resources.**
  - **Less visibility: You can't capture all the network traffic in non-promiscuous mode, limiting visibility to only traffic that is meant for your machine.**

---

2. Packet Tracing Using Filters in Wireshark

Wireshark allows users to filter captured traffic using a variety of powerful display filters and capture filters. Filters help narrow down the large amount of network traffic to focus on specific protocols, IP addresses, ports, or even specific packet fields.

Display Filters:

Display filters in Wireshark are used to filter packets after they have been captured. They allow you to focus on specific traffic types, protocols, or fields in the packet header.

Common Wireshark Display Filters:

1. **Filter by IP Address:**
    - **Show packets from or to a specific IP address.**

plaintext

Copy

```
ip.addr == 192.168.1.1
```

    - **This filter will show all packets where 192.168.1.1 is either the source or destination IP.**
2. **Filter by Protocol:**
    - **Show packets of a specific protocol, such as HTTP, TCP, or UDP.**

plaintext

Copy

```
http
```

    - **This filter will show HTTP traffic only.**
3. **Filter by Port:**
    - **Show traffic that is using a specific port (e.g., HTTP typically runs on port 80).**

plaintext

Copy

```
tcp.port == 80
```

    - **This filter will capture TCP packets that use port 80 (usually HTTP traffic).**
4. **Filter by MAC Address:**
    - **Filter packets by the source or destination MAC address.**

plaintext

Copy

```
eth.addr == 00:1a:2b:3c:4d:5e
```

    - **This filter shows packets with a specific MAC address.**
5. **Filter by TCP Flags:**
    - **Filter packets based on specific TCP flags (e.g., SYN, ACK, FIN).**

plaintext

Copy

```
tcp.flags.syn == 1
```

    - **This will capture SYN packets, typically seen in the initial steps of a TCP handshake.**

6. **Filter by HTTP Requests:**
    - **Capture only HTTP GET requests.**

plaintext

Copy

```
http.request.method == "GET"
```

    - **This filter will show all HTTP GET requests.**
7. **Filter by Time:**
    - **Show packets captured during a specific time range.**

plaintext

Copy

```
frame.time >= "2022-03-01 12:00:00" && frame.time <= "2022-03-01 12:10:00"
```

    - **This will display packets captured during the specified time range.**

Capture Filters:

Capture filters are used to filter packets before they are captured, which reduces the amount of data being collected. Capture filters are more limited than display filters but can help reduce network load and storage requirements during the capture phase.

Common Capture Filters:

1. **Filter by IP Address:**

plaintext

Copy

```
host 192.168.1.1
```

    - **Capture traffic to and from the specified IP address.**
2. **Filter by Protocol:**

plaintext

Copy

```
tcp
```

    - **Capture only TCP traffic.**
3. **Filter by Port:**

plaintext

Copy

```
port 80
```

    - **Capture only traffic on port 80 (HTTP).**
4. **Filter by Source or Destination Port:**

plaintext

Copy

src port 443

- Capture traffic from port 443 (HTTPS).

5. Filter by MAC Address:

plaintext

Copy

ether host 00:1a:2b:3c:4d:5e

- Capture packets to/from a specific MAC address.

6. Filter by Interface:

plaintext

Copy

interface eth0

- Capture packets only from the eth0 network interface.

Example Use Cases for Filters:

- Track HTTP Traffic: If you're interested in HTTP traffic (e.g., analyzing web requests or responses), you can filter HTTP traffic using the display filter:

plaintext

Copy

http

- Capture Traffic from a Specific Device: To capture packets to or from a specific device with an IP address of 192.168.1.1, use the display filter:

plaintext

Copy

ip.addr == 192.168.1.1

- Troubleshoot DNS Resolution Issues: If you're troubleshooting DNS resolution issues, filter packets related to DNS using the filter:

plaintext

Copy

dns

- Monitor Port 22 (SSH): To monitor SSH traffic (port 22), use:

plaintext

Copy

tcp.port == 22

- Capture TCP Handshake: To observe a TCP handshake, you can filter by

the SYN flag:

plaintext

Copy

tcp.flags.syn == 1