WIESHARK

Wireshark is a free and open-source network protocol analyser. It is widely used by network administrators, cybersecurity professionals, and developers to capture and analyse network traffic in real time. It allows you to see what's happening on your network at a microscopic level.

Think of it as a microscope for your network—it shows every packet of data traveling through your network.

Uses of Wireshark

1. Network Troubleshooting
Detecting slow networks, dropped packets, or misconfigured devices.
Identifying the source of connectivity issues (like router or firewall problems).

2. Network Security Analysis
Detecting suspicious or malicious network activity.
Analysing malware or hacking attempts by inspecting packets.

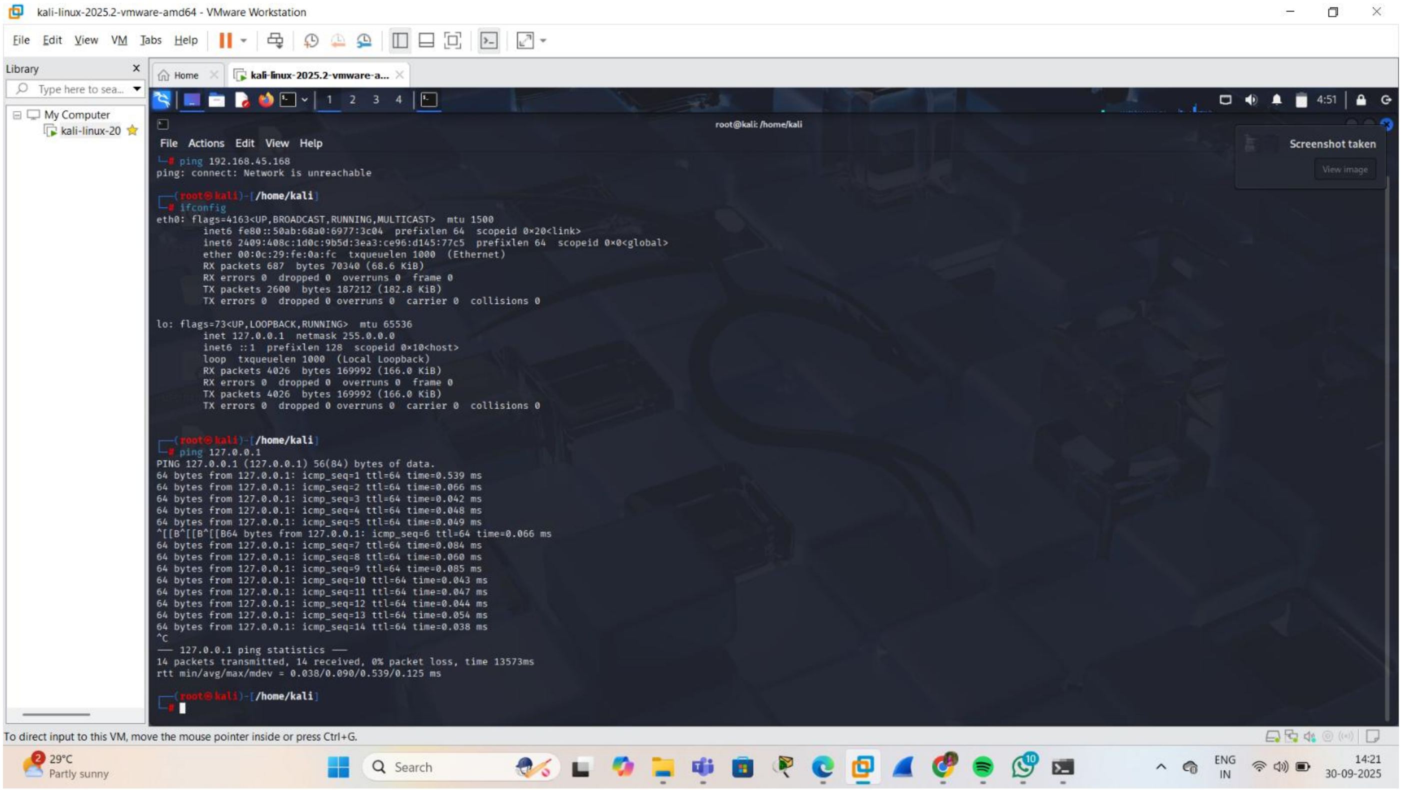
3. Protocol Analysis
Understanding how protocols like HTTP, TCP, DNS, FTP, or SMTP work.
Debugging protocol implementations in software development.

4. Performance Monitoring
Measuring latency, packet loss, and bandwidth usage.
Monitoring network performance over time.

5. Education and Training
Learning about network protocols and packet structures.
Practicing cybersecurity analysis in lab environments.

Advantages of Wireshark

1. Free and Open-Source
No cost and actively maintained by the community.
2. Detailed Packet Analysis
Provides deep insights into every packet, including headers and payload.
3. Supports Multiple Protocols
Can decode thousands of protocols (HTTP, TCP, UDP, DNS, SSL, etc.).
4. Real-Time and Offline Analysis
Capture live traffic or analyse previously saved network captures.
5. Powerful Filtering
Use display filters and capture filters to isolate specific traffic.
For example: ip.addr == 192.168.1.10 or tcp.port == 80.
6. Cross-Platform
Works on Windows, Linux, macOS, and more.
7. Community and Documentation
Extensive guides, tutorials, and user forums available for learning.



kali-linux-2025.2-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
kali-linux-20
Home
kali-linux-2025.2-vmware-a...
1 2 3 4
4:51
root@kali: /home/kali
File Actions Edit View Help
Screenshot taken
View image
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
29°C
Partly sunny
Search
ENG
IN
14:21
30-09-2025

```
└─# ping 192.168.45.168
ping: connect: Network is unreachable

┌──(root㉿kali)-[/home/kali]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::50ab:68a0:6977:3c04  prefixlen 64  scopeid 0x20<link>
        inet6 2409:408c:1d0c:9b5d:3ea3:ce96:d145:77c5  prefixlen 64  scopeid 0x0<global>
        ether 00:0c:29:fe:0a:fc  txqueuelen 1000  (Ethernet)
        RX packets 687  bytes 70340 (68.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2600  bytes 187212 (182.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4026  bytes 169992 (166.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4026  bytes 169992 (166.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(root㉿kali)-[/home/kali]
└─# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.539 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.049 ms
^[[B^[[B^[[B64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.066 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.060 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.085 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.047 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.044 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.038 ms
^C
── 127.0.0.1 ping statistics ──
14 packets transmitted, 14 received, 0% packet loss, time 13573ms
rtt min/avg/max/mdev = 0.038/0.090/0.539/0.125 ms

┌──(root㉿kali)-[/home/kali]
└─# 
```

Ping is a command-line utility used to test network connectivity and measure the response time between devices on a network.

How Ping Works

1. ICMP Echo Request: The ping command sends an ICMP (Internet Control Message Protocol) echo request packet to a specified IP address or hostname.
2. Response: The device receiving the packet responds with an ICMP echo reply packet.
3. Round-trip time: The ping command measures the time it takes for the packet to travel from the source device to the destination device and back.

Ping Command Usage

1. Basic syntax: ping <hostname or IP address>
2. Options:
   - -c <count>: Specify the number of echo requests to send.
   - -i <interval>: Specify the interval between echo requests.
   - -s <size>: Specify the size of the echo request packet.
   - -t: Ping continuously until stopped.

In wireshark, notified that TCP, UDP, TLSv1.2 protocols,
TCP (Transmission Control Protocol) is a fundamental protocol in the internet protocol suite that ensures reliable, error-checked, and sequential delivery of data between devices over IP networks.

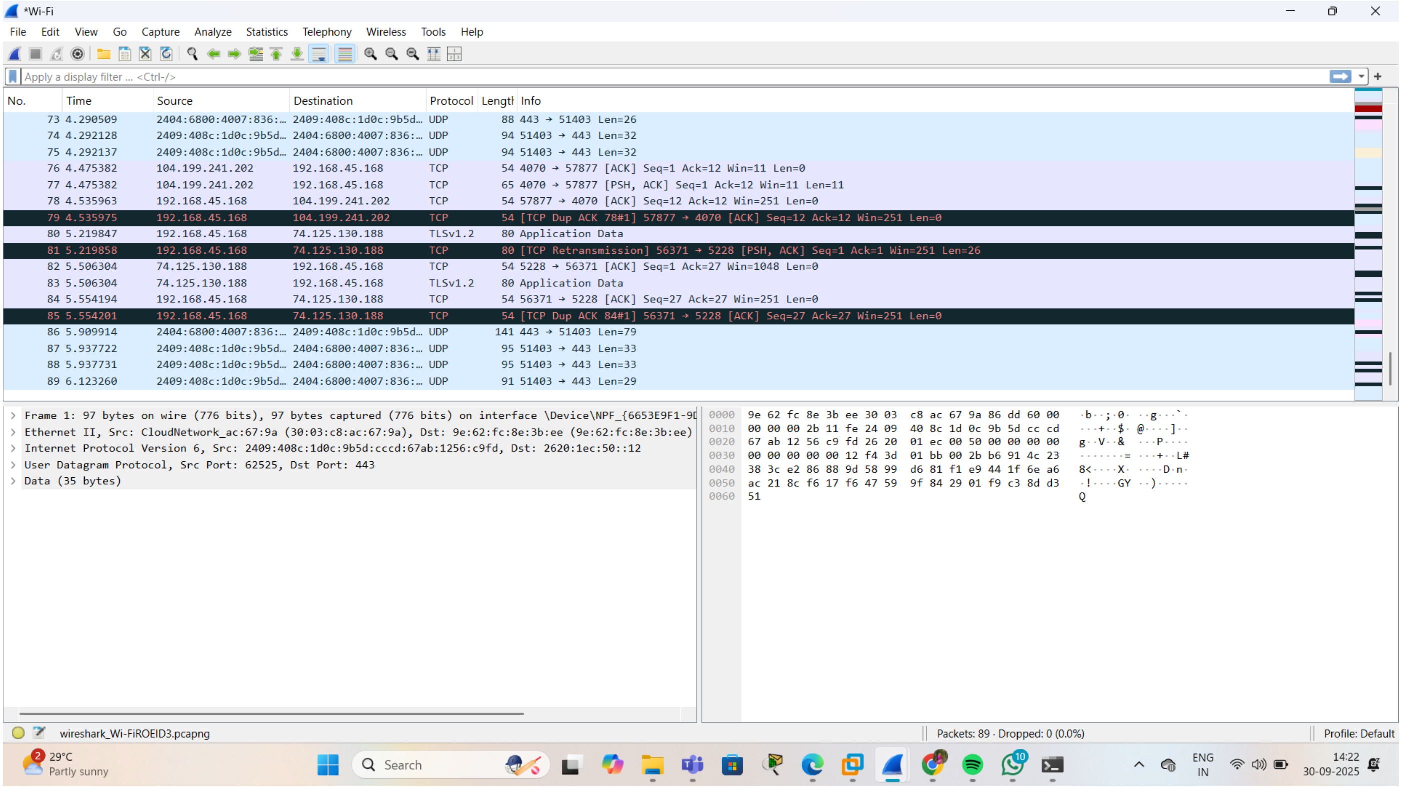
How TCP Works
1. Three-way handshake: TCP establishes a connection through a three-way handshake process (SYN, SYN-ACK, ACK).
2. Data transfer: Data is broken into packets and transmitted over the network.
3. Acknowledgment: The receiver acknowledges receipt of packets, and the sender retransmits any lost or corrupted packets.

UDP (User Datagram Protocol) is a connectionless protocol that enables fast and efficient transmission of data packets over IP networks.

Key Features
1. Connectionless: UDP doesn't establish a connection before sending data.
2. Best-effort delivery: UDP doesn't guarantee delivery of packets; packets may be lost, duplicated, or arrive out of order.
3. No sequencing: UDP doesn't sequence packets, so they may arrive out of order.
4. No error correction: UDP doesn't perform error correction; packets with errors are discarded.

TLS (Transport Layer Security) is a cryptographic protocol that provides secure communication over the internet.

Key Features
1. Encryption: TLS encrypts data in transit, ensuring confidentiality and integrity.
2. Authentication: TLS verifies the identity of the server (and optionally the client) to prevent impersonation.
3. Data integrity: TLS ensures that data is not tampered with or altered during transmission.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time       Source                Destination           Protocol Lengtl Info
   73 4.290509   2404:6800:4007:836:… 2409:408c:1d0c:9b5d… UDP        88 443 → 51403 Len=26
   74 4.292128   2409:408c:1d0c:9b5d… 2404:6800:4007:836:… UDP        94 51403 → 443 Len=32
   75 4.292137   2409:408c:1d0c:9b5d… 2404:6800:4007:836:… UDP        94 51403 → 443 Len=32
   76 4.475382   104.199.241.202      192.168.45.168       TCP        54 4070 → 57877 [ACK] Seq=1 Ack=12 Win=11 Len=0
   77 4.475382   104.199.241.202      192.168.45.168       TCP        65 4070 → 57877 [PSH, ACK] Seq=1 Ack=12 Win=11 Len=11
   78 4.535963   192.168.45.168       104.199.241.202      TCP        54 57877 → 4070 [ACK] Seq=12 Ack=12 Win=251 Len=0
   79 4.535975   192.168.45.168       104.199.241.202      TCP        54 [TCP Dup ACK 78#1] 57877 → 4070 [ACK] Seq=12 Ack=12 Win=251 Len=0
   80 5.219847   192.168.45.168       74.125.130.188       TLSv1.2    80 Application Data
   81 5.219858   192.168.45.168       74.125.130.188       TCP        80 [TCP Retransmission] 56371 → 5228 [PSH, ACK] Seq=1 Ack=1 Win=251 Len=26
   82 5.506304   74.125.130.188       192.168.45.168       TCP        54 5228 → 56371 [ACK] Seq=1 Ack=27 Win=1048 Len=0
   83 5.506304   74.125.130.188       192.168.45.168       TLSv1.2    80 Application Data
   84 5.554194   192.168.45.168       74.125.130.188       TCP        54 56371 → 5228 [ACK] Seq=27 Ack=27 Win=251 Len=0
   85 5.554201   192.168.45.168       74.125.130.188       TCP        54 [TCP Dup ACK 84#1] 56371 → 5228 [ACK] Seq=27 Ack=27 Win=251 Len=0
   86 5.909914   2404:6800:4007:836:… 2409:408c:1d0c:9b5d… UDP       141 443 → 51403 Len=79
   87 5.937722   2409:408c:1d0c:9b5d… 2404:6800:4007:836:… UDP        95 51403 → 443 Len=33
   88 5.937731   2409:408c:1d0c:9b5d… 2404:6800:4007:836:… UDP        95 51403 → 443 Len=33
   89 6.123260   2409:408c:1d0c:9b5d… 2404:6800:4007:836:… UDP        91 51403 → 443 Len=29
```

```
> Frame 1: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface \Device\NPF_{6653E9F1-9D
> Ethernet II, Src: CloudNetwork_ac:67:9a (30:03:c8:ac:67:9a), Dst: 9e:62:fc:8e:3b:ee (9e:62:fc:8e:3b:ee)
> Internet Protocol Version 6, Src: 2409:408c:1d0c:9b5d:cccd:67ab:1256:c9fd, Dst: 2620:1ec:50::12
> User Datagram Protocol, Src Port: 62525, Dst Port: 443
> Data (35 bytes)
```

```
0000  9e 62 fc 8e 3b ee 30 03  c8 ac 67 9a 86 dd 60 00   ·b··;·0· ··g···`·
0010  00 00 00 2b 11 fe 24 09  40 8c 1d 0c 9b 5d cc cd   ···+··$· @····]··
0020  67 ab 12 56 c9 fd 26 20  01 ec 00 50 00 00 00 00   g··V··&  ···P····
0030  00 00 00 00 00 12 f4 3d  01 bb 00 2b b6 91 4c 23   ·······= ···+··L#
0040  38 3c e2 86 88 9d 58 99  d6 81 f1 e9 44 1f 6e a6   8<····X· ····D·n·
0050  ac 21 8c f6 17 f6 47 59  9f 84 29 01 f9 c3 8d d3   ·!····GY ··)·····
0060  51                                                 Q
```

wireshark_Wi-FiROEID3.pcapng

Packets: 89 · Dropped: 0 (0.0%)

Profile: Default