Experiment No. 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.
Date of Performance:
Date of Submission:

Experiment No. 7

Title: 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.

Aim: To student wireshark packet sniffer tool

Objectives:

- Identify security threats and malicious activity on a network
- Observe network traffic for debugging complex networks
- Filter traffic based on protocols, ports, and other parameters
- Capture packets and save them to a Pcap file for offline analysis
- Apply coloring rules to the packet list for better analysis

Theory:

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

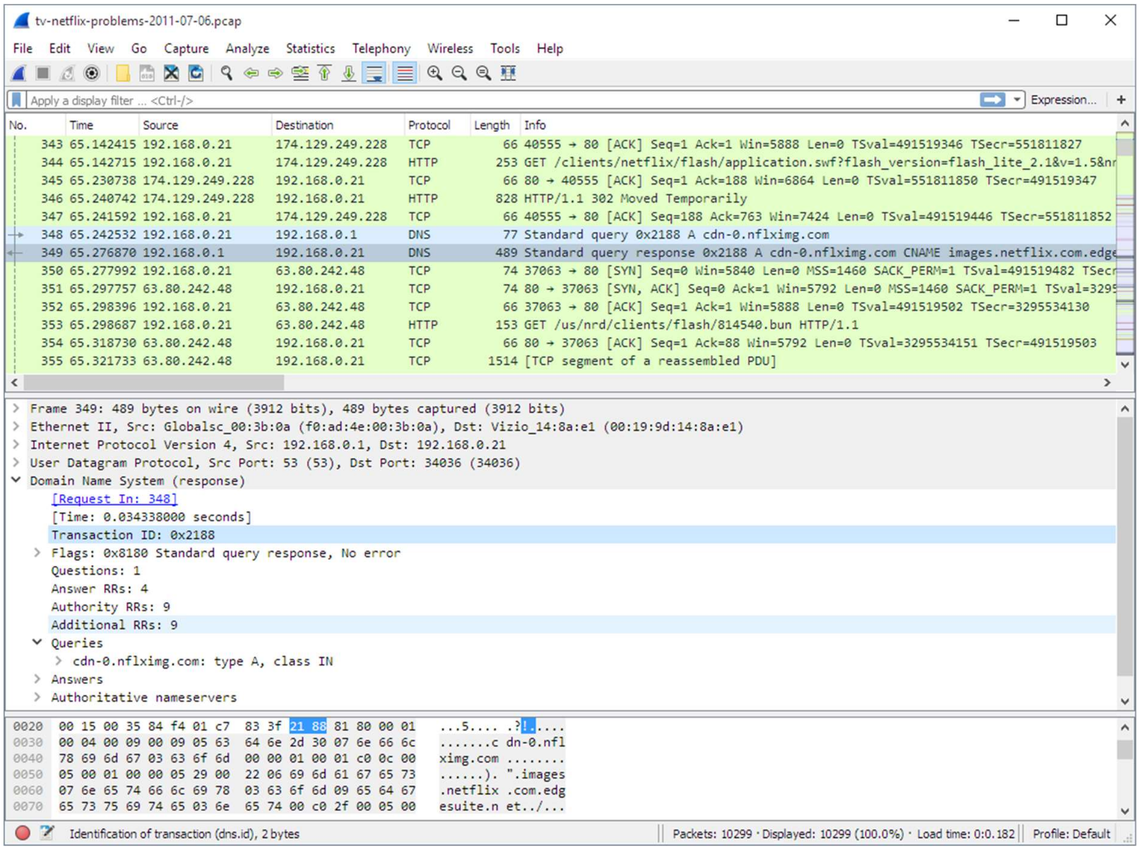
You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

The following are some of the many features Wireshark provides:

- Available for *UNIX* and *Windows*.
- *Capture* live packet data from a network interface.
- *Open* files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- *Import* packets from text files containing hex dumps of packet data.
- Display packets with *very detailed protocol information*.
- *Save* packet data captured.
- *Export* some or all packets in a number of capture file formats.
- *Filter packets* on many criteria.
- *Search* for packets on many criteria.

- *Colorize* packet display based on filters.
- Create various *statistics*.


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.  Time      Source          Destination     Protocol Length Info
 343 65.142415 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
 344 65.142715 192.168.0.21    174.129.249.228 HTTP      253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
 345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
 346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
 347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
 348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
 349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
 350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
 351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
 352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
 353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
 354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
 355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]
```

```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

Identification of transaction (dns.id), 2 bytes || Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182 || Profile: Default

1.1.3. Live capture from many different network media

Wireshark can capture traffic from many different network media types, including Ethernet, Wireless LAN, Bluetooth, USB, and more. The specific media types supported may be limited by several factors, including your hardware and operating system. An overview of the supported media types can be found at https://gitlab.com/wireshark/wireshark/-/wikis/CaptureSetup/NetworkMedia.

1.1.4. Import files from many other capture programs

Wireshark can open packet captures from a large number of capture programs. For a list of input formats see Section 5.2.2, "Input File Formats".

1.1.5. Export files for many other capture programs

Wireshark can save captured packets in many formats, including those used by other capture programs. For a list of output formats see Section 5.3.2, "Output File Formats".

1.1.6. Many protocol dissectors

There are protocol dissectors (or decoders, as they are known in other products) for a great many protocols: see Appendix C, Protocols and Protocol Fields.

1.1.7. Open Source Software

Wireshark is an open source software project, and is released under the GNU General Public License (GPL). You can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plugins, or built into the source, and they often do!

Wireshark should support any version of Windows that is still within its extended support lifetime. At the time of writing this includes Windows 11, 10, Server 2022, Server 2019, and Server 2016. It also requires the following:

The Universal C Runtime. This is included with Windows 10 and Windows Server 2019 and is installed automatically on earlier versions if Microsoft Windows Update is enabled. Otherwise you must install KB2999226 or KB3118401.

Any modern 64-bit Intel or Arm processor.

500 MB available RAM. Larger capture files require more RAM.

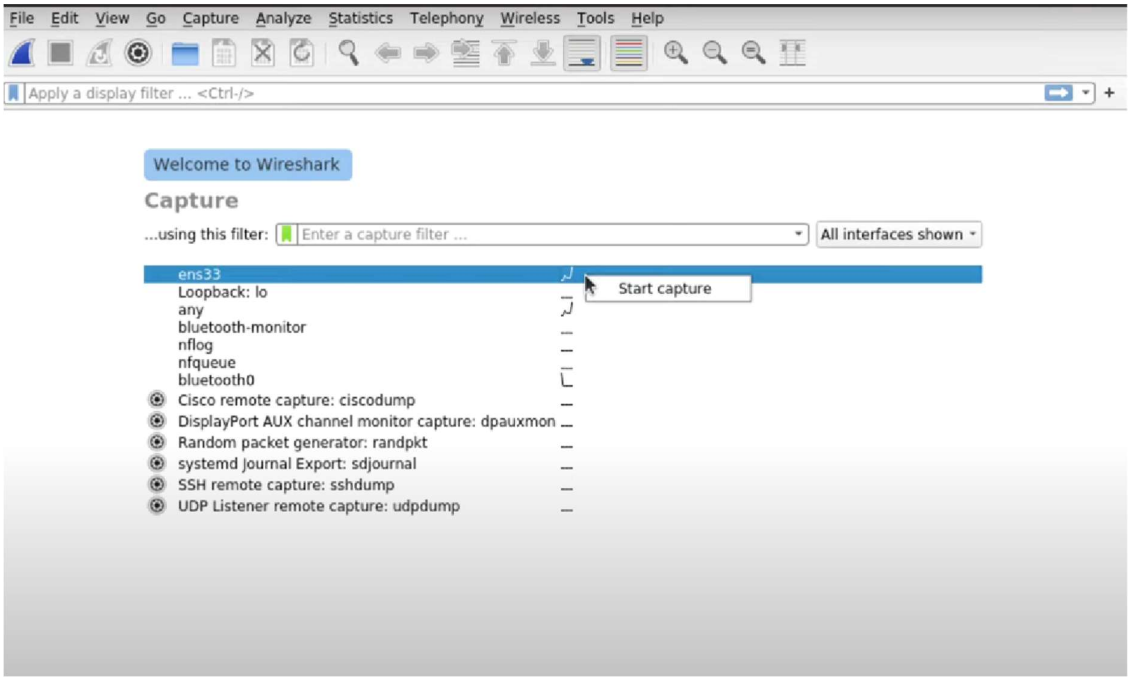
500 MB available disk space. Capture files require additional disk space.

Any modern display. 1280 × 1024 or higher resolution is recommended. Wireshark will make use of HiDPI or Retina resolutions if available. Power users will find multiple monitors useful.

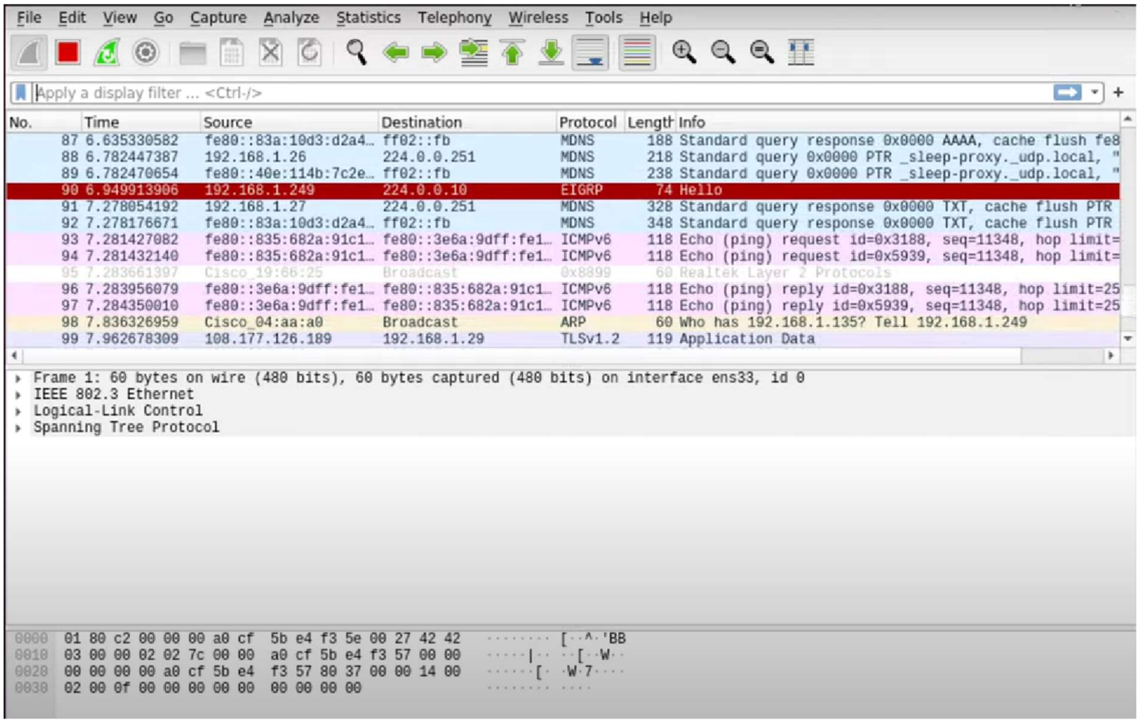
A supported network card for capturing

- Ethernet. Any card supported by Windows should work. See the wiki pages on Ethernet capture and offloading for issues that may affect your environment.
- 802.11. See the Wireshark wiki page. Capturing raw 802.11 information may be difficult without special equipment.
- Other media. See https://gitlab.com/wireshark/wireshark/-/wikis/CaptureSetup/NetworkMedia.

Implementation:



ens33
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
bluetooth0
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump



```
No.   Time          Source                 Destination            Protocol Length Info
   87 6.635330582   fe80::83a:10d3:d2a4…   ff02::fb               MDNS      188 Standard query response 0x0000 AAAA, cache flush fe8
   88 6.782447387   192.168.1.26           224.0.0.251            MDNS      218 Standard query 0x0000 PTR _sleep-proxy._udp.local, "
   89 6.782470654   fe80::40e:114b:7c2e…   ff02::fb               MDNS      238 Standard query 0x0000 PTR _sleep-proxy._udp.local, "
   90 6.949913906   192.168.1.249          224.0.0.10             EIGRP      74 Hello
   91 7.278054192   192.168.1.27           224.0.0.251            MDNS      328 Standard query response 0x0000 TXT, cache flush PTR
   92 7.278176671   fe80::83a:10d3:d2a4…   ff02::fb               MDNS      348 Standard query response 0x0000 TXT, cache flush PTR
   93 7.281427082   fe80::835:682a:91c1…   fe80::3e6a:9dff:fe1…   ICMPv6    118 Echo (ping) request id=0x3188, seq=11348, hop limit=
   94 7.281432140   fe80::835:682a:91c1…   fe80::3e6a:9dff:fe1…   ICMPv6    118 Echo (ping) request id=0x5939, seq=11348, hop limit=
   95 7.283661397   Cisco_19:66:25         Broadcast              0x8899     60 Realtek Layer 2 Protocols
   96 7.283956079   fe80::3e6a:9dff:fe1…   fe80::835:682a:91c1…   ICMPv6    118 Echo (ping) reply id=0x3188, seq=11348, hop limit=25
   97 7.284350010   fe80::3e6a:9dff:fe1…   fe80::835:682a:91c1…   ICMPv6    118 Echo (ping) reply id=0x5939, seq=11348, hop limit=25
   98 7.836326959   Cisco_04:aa:a0         Broadcast              ARP        60 Who has 192.168.1.135? Tell 192.168.1.249
   99 7.962678309   108.177.126.189        192.168.1.29           TLSv1.2   119 Application Data

▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
▸ IEEE 802.3 Ethernet
▸ Logical-Link Control
▸ Spanning Tree Protocol

0000  01 80 c2 00 00 00 a0 cf  5b e4 f3 5e 00 27 42 42   ········ [··^·'BB
0010  03 00 00 02 02 7c 00 00  a0 cf 5b e4 f3 57 00 00   ·····|·· ··[··W··
0020  00 00 00 00 a0 cf 5b e4  f3 57 80 37 00 00 14 00   ······[· ·W·7····
0030  02 00 0f 00 00 00 00 00  00 00 00 00               ········ ····
```

Conclusion: Comment on study of Wireshark tool

The experiment involved downloading and installing Nmap, a powerful network scanning tool, and utilizing it with various options to conduct a range of scans, including open port scanning, OS fingerprinting, ping scanning, TCP port scanning, and UDP port scanning. By achieving the objectives of learning about Nmap and its features, learners gained valuable insights into network reconnaissance techniques and security assessments. Nmap's versatility and extensive range of functionalities enable users to comprehensively analyze network configurations, identify potential vulnerabilities, and assess overall network security posture.