What is a Packet Sniffer?

On Internet data is sent through files like a web page or a zip file, but what goes on underneath, files are sent in the form of packets.

packets contains data along with headers, these packets are combined and files are generated on the client.

Packet Sniffer is a program that captures sent over internet.

Example of a packet sniffer is Wireshark.

Wireshark captures packets on adapters like ethernet adapters, wireless or (wifi) adapters or loopback address or adapter.

You can download Wireshark from www.wireshark.org

Wireshark can capture data packets using filters

Example of Wireshark Filters are

ip.addr == 10.0.0.1

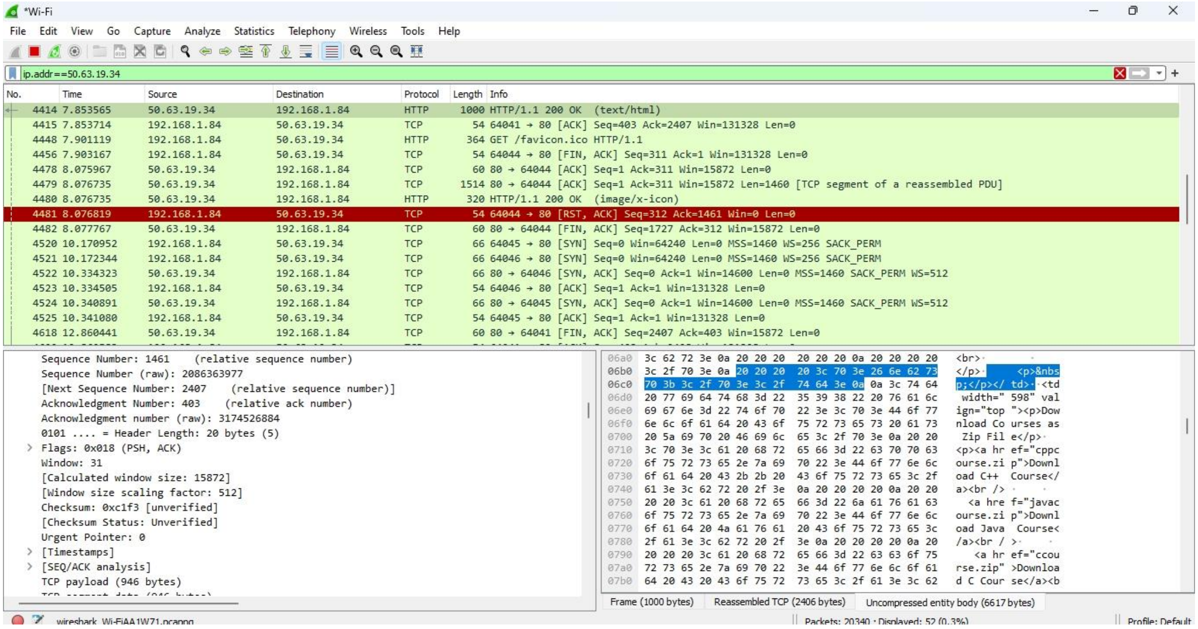
means listen for packets on ip address 10.0.0.1

tcp.port==4000

means listen on tcp port equal to 4000

http means listen for packets sent through http protocol

Screenshot of wireshark



ip.addr==50.63.19.34

```
No.    Time         Source          Destination     Protocol  Length Info
 4414 7.853565      50.63.19.34     192.168.1.84    HTTP      1000 HTTP/1.1 200 OK  (text/html)
 4415 7.853714      192.168.1.84    50.63.19.34     TCP         54 64041 → 80 [ACK] Seq=403 Ack=2407 Win=131328 Len=0
 4448 7.901119      192.168.1.84    50.63.19.34     HTTP       364 GET /favicon.ico HTTP/1.1
 4456 7.903167      192.168.1.84    50.63.19.34     TCP         54 64044 → 80 [FIN, ACK] Seq=311 Ack=1 Win=131328 Len=0
 4478 8.075967      50.63.19.34     192.168.1.84    TCP         60 80 → 64044 [ACK] Seq=1 Ack=311 Win=15872 Len=0
 4479 8.076735      50.63.19.34     192.168.1.84    TCP       1514 80 → 64044 [ACK] Seq=1 Ack=311 Win=15872 Len=1460 [TCP segment of a reassembled PDU]
 4480 8.076735      50.63.19.34     192.168.1.84    HTTP       320 HTTP/1.1 200 OK  (image/x-icon)
 4481 8.076819      192.168.1.84    50.63.19.34     TCP         54 64044 → 80 [RST, ACK] Seq=312 Ack=1461 Win=0 Len=0
 4482 8.077767      50.63.19.34     192.168.1.84    TCP         60 80 → 64044 [FIN, ACK] Seq=1727 Ack=312 Win=15872 Len=0
 4520 10.170952     192.168.1.84    50.63.19.34     TCP         66 64045 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 4521 10.172344     192.168.1.84    50.63.19.34     TCP         66 64046 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 4522 10.334323     50.63.19.34     192.168.1.84    TCP         66 80 → 64046 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=512
 4523 10.334505     192.168.1.84    50.63.19.34     TCP         54 64046 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 4524 10.340891     50.63.19.34     192.168.1.84    TCP         66 80 → 64045 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=512
 4525 10.341080     192.168.1.84    50.63.19.34     TCP         54 64045 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 4618 12.860441     50.63.19.34     192.168.1.84    TCP         60 80 → 64041 [FIN, ACK] Seq=2407 Ack=403 Win=15872 Len=0
```

```
    Sequence Number: 1461    (relative sequence number)
    Sequence Number (raw): 2086363977
    [Next Sequence Number: 2407    (relative sequence number)]
    Acknowledgment Number: 403    (relative ack number)
    Acknowledgment number (raw): 3174526884
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window: 31
    [Calculated window size: 15872]
    [Window size scaling factor: 512]
    Checksum: 0xc1f3 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (946 bytes)
```

```
06a0  3c 62 72 3e 0a 20 20 20  20 20 20 0a 20 20 20 20   <br>·      ·
06b0  3c 2f 70 3e 0a 20 20 20  20 3c 70 3e 26 6e 62 73   </p>·    <p>&nbs
06c0  70 3b 3c 2f 70 3e 3c 2f  74 64 3e 0a 0a 3c 74 64   p;</p></ td>··<td
06d0  20 77 69 64 74 68 3d 22  35 39 38 22 20 76 61 6c    width=" 598" val
06e0  69 67 6e 3d 22 74 6f 70  22 3e 3c 70 3e 44 6f 77   ign="top "><p>Dow
06f0  6e 6c 6f 61 64 20 43 6f  75 72 73 65 73 20 61 73   nload Co urses as
0700  20 5a 69 70 20 46 69 6c  65 3c 2f 70 3e 0a 20 20    Zip Fil e</p>·
0710  3c 70 3e 3c 61 20 68 72  65 66 3d 22 63 70 70 63   <p><a hr ef="cppc
0720  6f 75 72 73 65 2e 7a 69  70 22 3e 44 6f 77 6e 6c   ourse.zi p">Downl
0730  6f 61 64 20 43 2b 2b 20  43 6f 75 72 73 65 3c 2f   oad C++  Course</
0740  61 3e 3c 62 72 20 2f 3e  0a 20 20 20 20 0a 20 20   a><br /> ·    ·
0750  20 20 3c 61 20 68 72 65  66 3d 22 6a 61 76 61 63     <a hre f="javac
0760  6f 75 72 73 65 2e 7a 69  70 22 3e 44 6f 77 6e 6c   ourse.zi p">Downl
0770  6f 61 64 20 4a 61 76 61  20 43 6f 75 72 73 65 3c   oad Java  Course<
0780  2f 61 3e 3c 62 72 20 2f  3e 0a 20 20 20 20 0a 20   /a><br / >·    ·
0790  20 20 20 3c 61 20 68 72  65 66 3d 22 63 63 6f 75      <a hr ef="ccou
07a0  72 73 65 2e 7a 69 70 22  3e 44 6f 77 6e 6c 6f 61   rse.zip" >Downloa
07b0  64 20 43 20 43 6f 75 72  73 65 3c 2f 61 3e 3c 62   d C Cour se</a><b
```

Frame (1000 bytes)
Reassembled TCP (2406 bytes)
Uncompressed entity body (6617 bytes)
wireshark_Wi-FiAA1W71.pcapng
Packets: 20340 · Displayed: 52 (0.3%)
Profile: Default

What is a Port Scanner ?

Port Scanner is a software that checks for open and close ports on a server.

open ports means ports on which server is listening.

listening means server will listen to requests from clients.

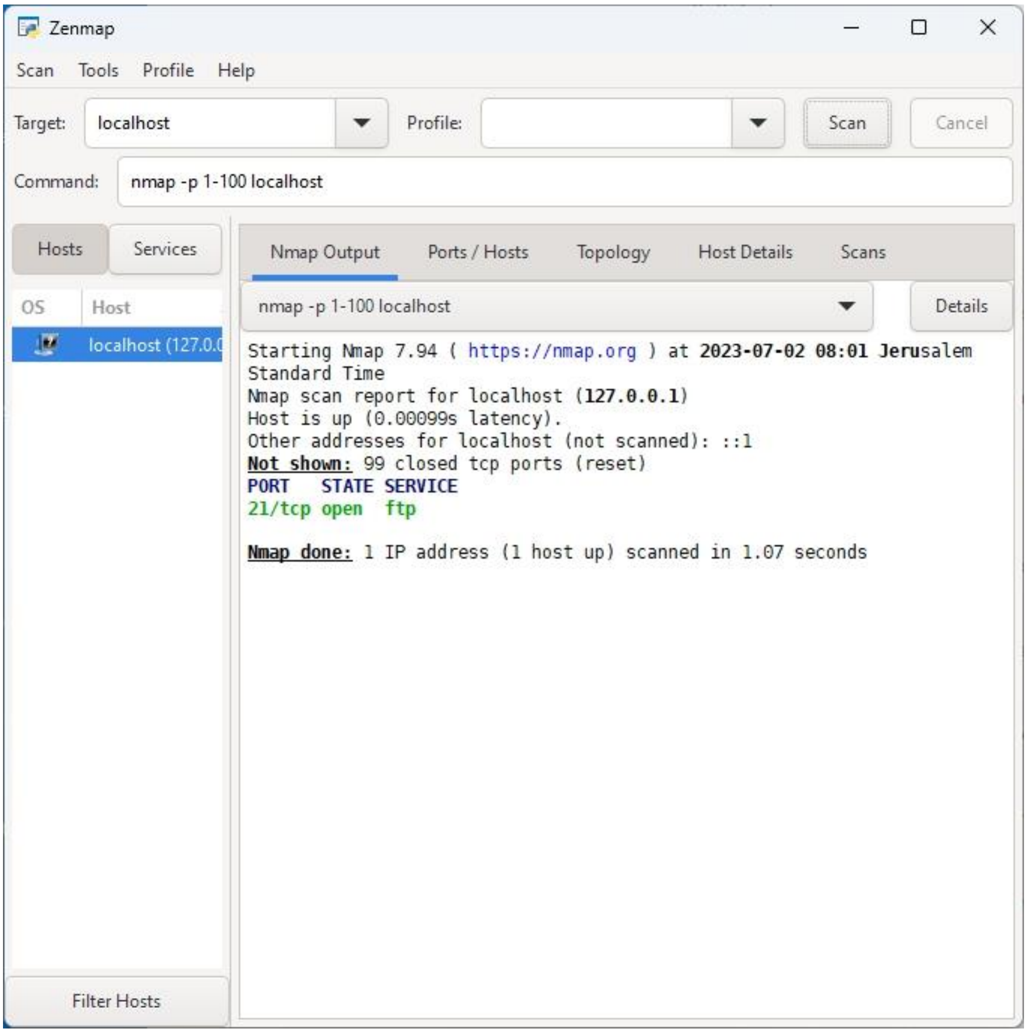
close ports means server cannot listen for client requests.

Example of Port scanner is nmap.

nmap command to scan ports from 1 to 100 on localhost is

nmap -p 1-100 localhost

Following is the screenshot of nmap



```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-07-02 08:01 Jerusalem
Standard Time
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00099s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 99 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 1.07 seconds
```