**TASK NO:01**

**What is the name of the website?**

Steps:

(Wireshark GUI)

Open your capture (.pcap) in Wireshark.

Apply the display filter:

http.request

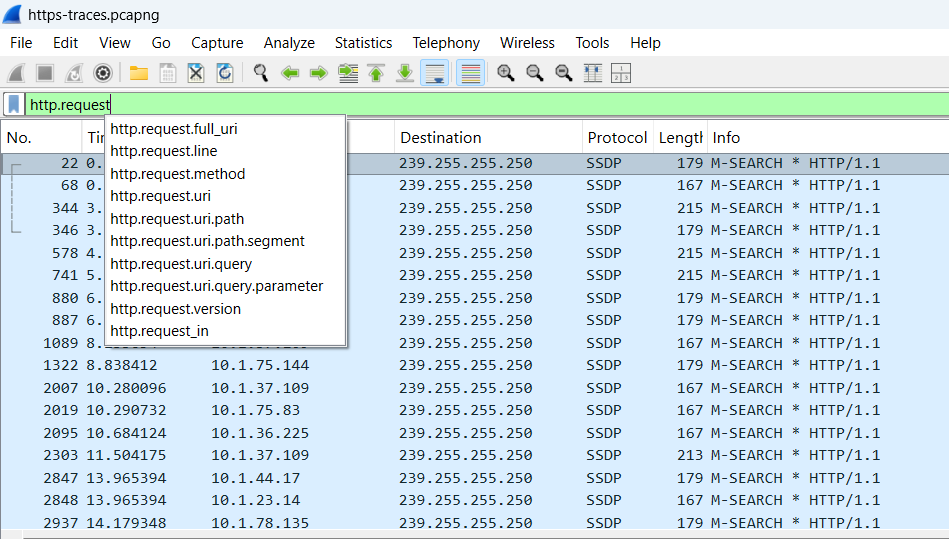
This shows HTTP requests only.

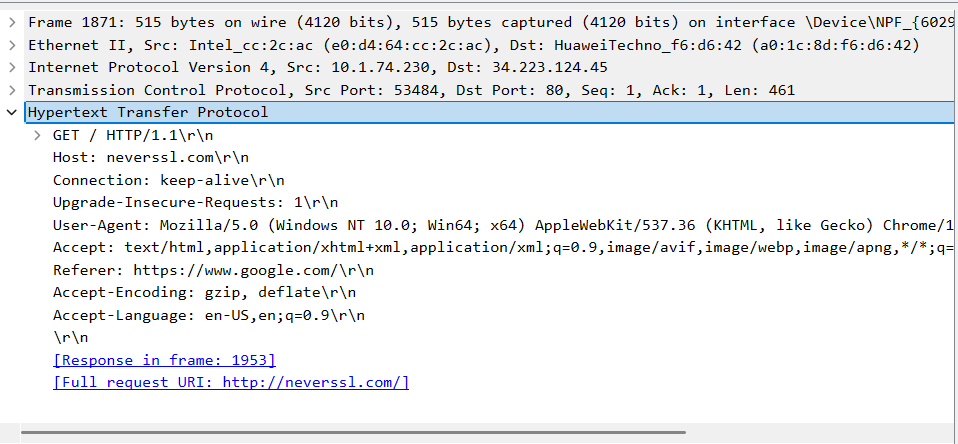
Click the first GET request packet in the list.

In the Packet Details pane expand Hypertext Transfer Protocol → look for the Host: header (or look at the Request URI).

Host: contains the website

If the request line contains a full URL (rare for proxies), the domain will also be there.





**TASK:02**

**Find the packet that contains the first GET request**

Steps (Wireshark GUI)

Filter: http.request && http.request.method == "GET"

Sort by No. (already chronological). The topmost result is the first GET.

Select that packet and note the Frame Number (shown in the packet list).

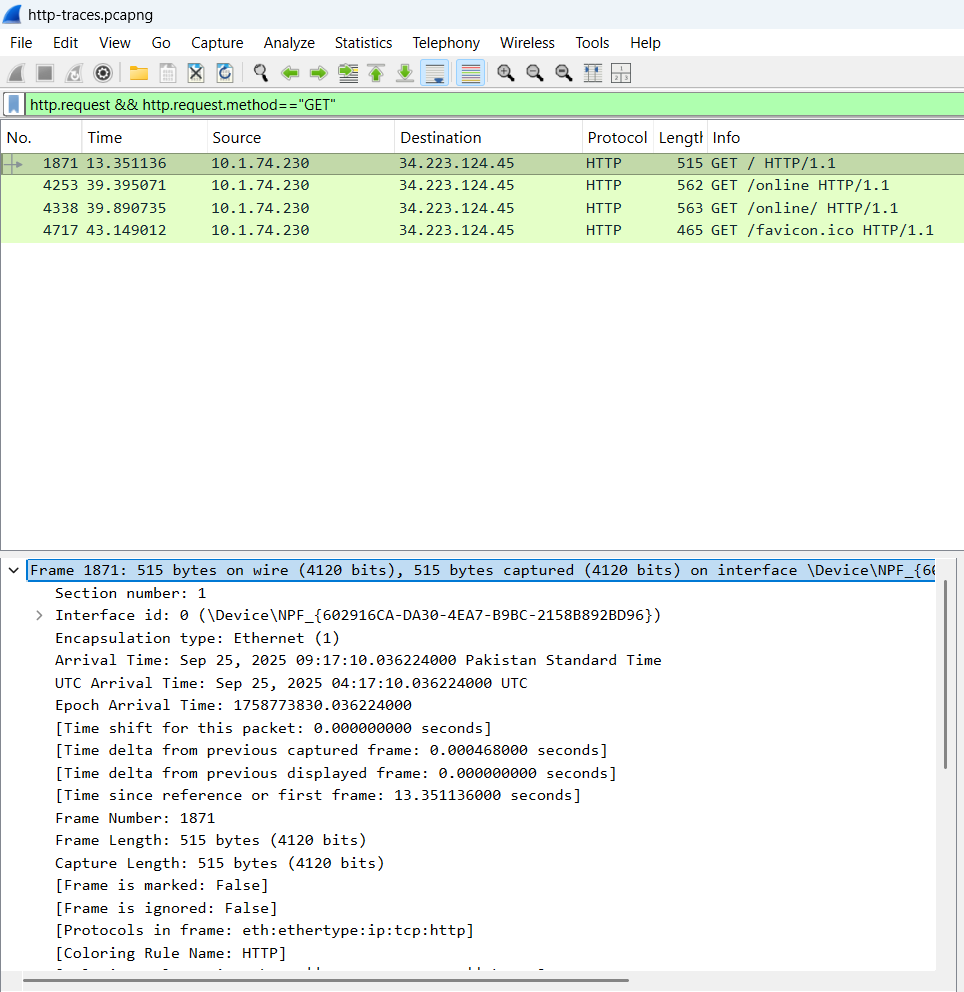
Alternative (Find dialog)

Edit → Find Packet → choose Display Filter → enter http.request.method == "GET" → click Find.

Steps (tshark / CLI)

tshark -r trace.pcap -Y 'http.request && http.request.method == "GET"' -T fields -e frame.number -e frame.time -e http.host -e http.request.uri | head -n 1

The first line printed gives the frame number of the first GET, timestamp, host and URI.



**FRAME NUMBER:1871**

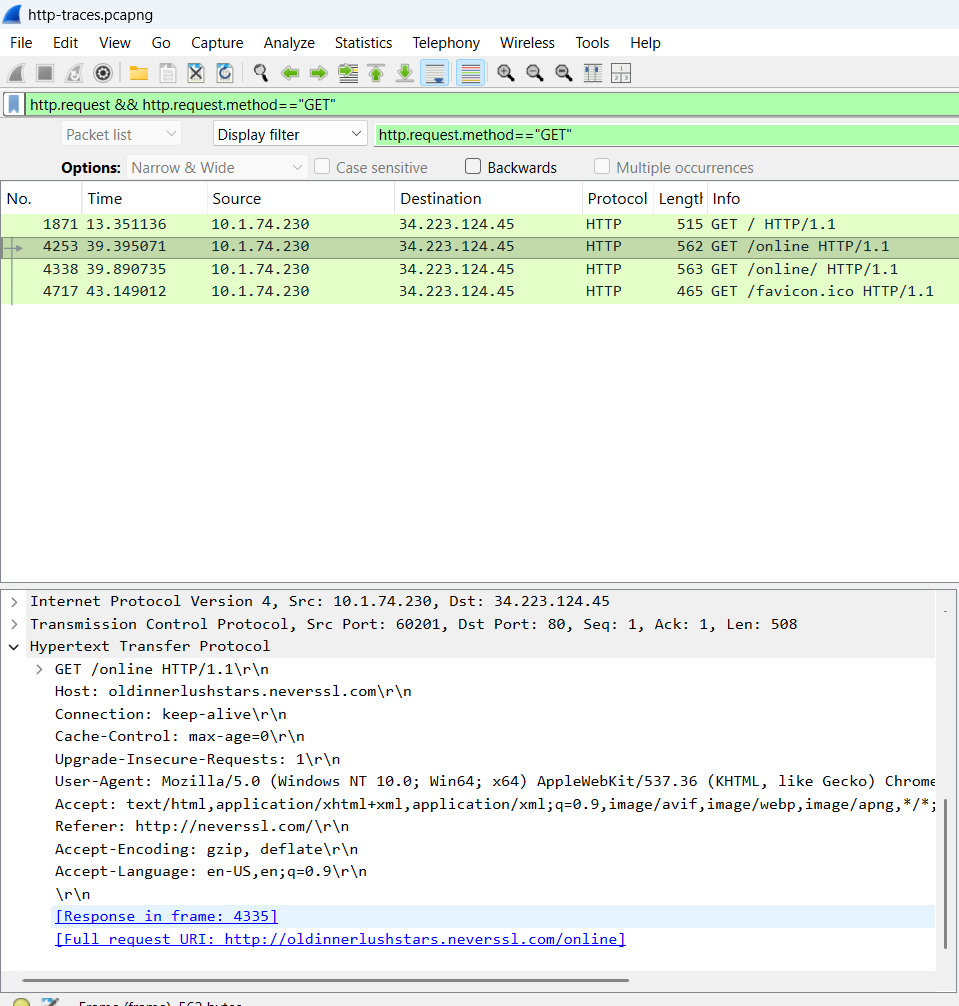
**TASK NO:03**

Describe all headers and their values in this GET request message

How to extract headers (Wireshark GUI)

Open the packet found in step 2.

Expand Hypertext Transfer Protocol — you’ll see the Request-Line and then each header as Header-Name: value.



**TASK 4:**

**Identify the status code in the first server response**

Steps (Wireshark GUI):

After the first GET, filter http.response or simply look at the packet immediately following the GET in the same TCP stream.

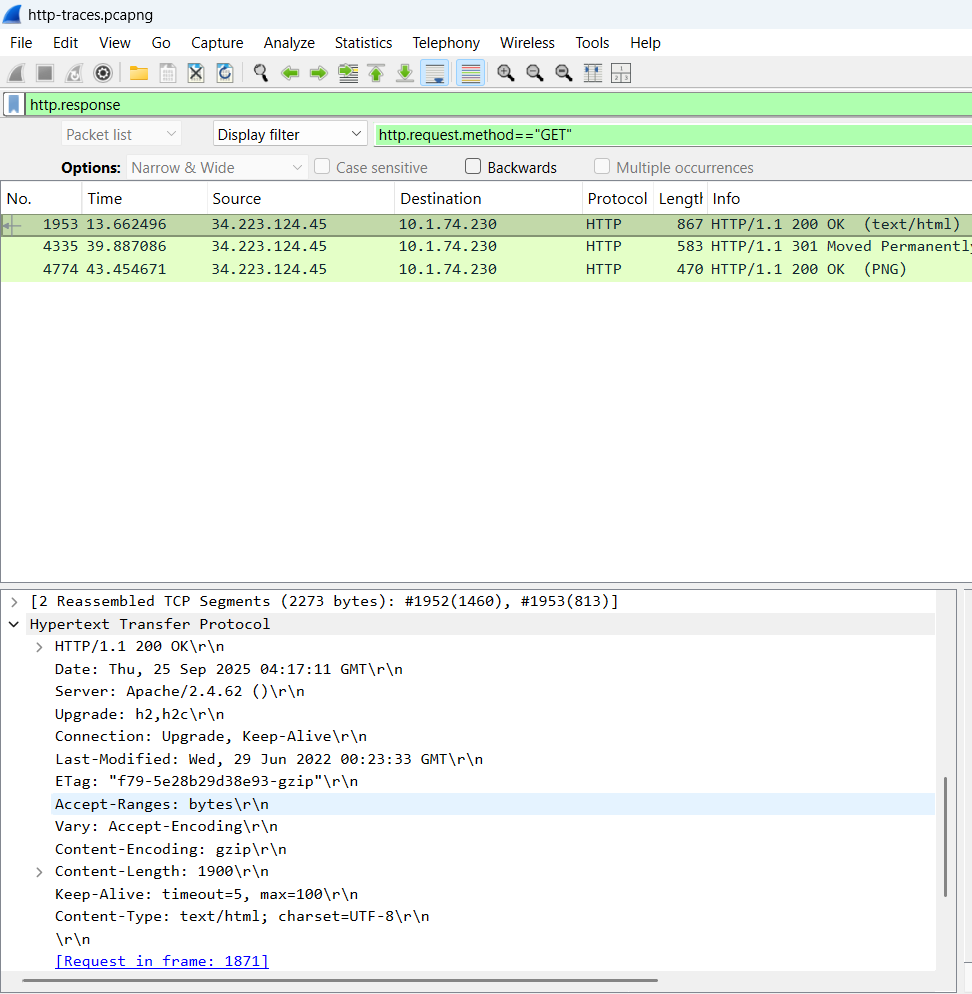
Select the response packet. Expand Hypertext Transfer Protocol — the status line is at top like HTTP/1.1 200 OK.

The numeric code (200) is the status code; the phrase (OK) is the reason phrase.

Note the frame number of the response and the status code.

tshark -r trace.pcap -Y 'http.response' -T fields -e frame.number -e frame.time -e http.response.code -e http.response.phrase | head -n 5

The first line printed is the first server response captured.



**TASK :05**

**How many HTTP response messages are exchanged in total?**

Method A — Wireshark GUI (fast)

Apply display filter: http.response

Look at the lower status bar — it shows count like Displayed: N of M packets; or check the packet list (it will only show responses).

Or go Statistics → HTTP (or Statistics → Protocol Hierarchy) to get counts and a response-code breakdown.

tshark -r trace.pcap -Y 'http.response' -T fields -e frame.number | wc -l

Output is the total number of HTTP responses in the capture.

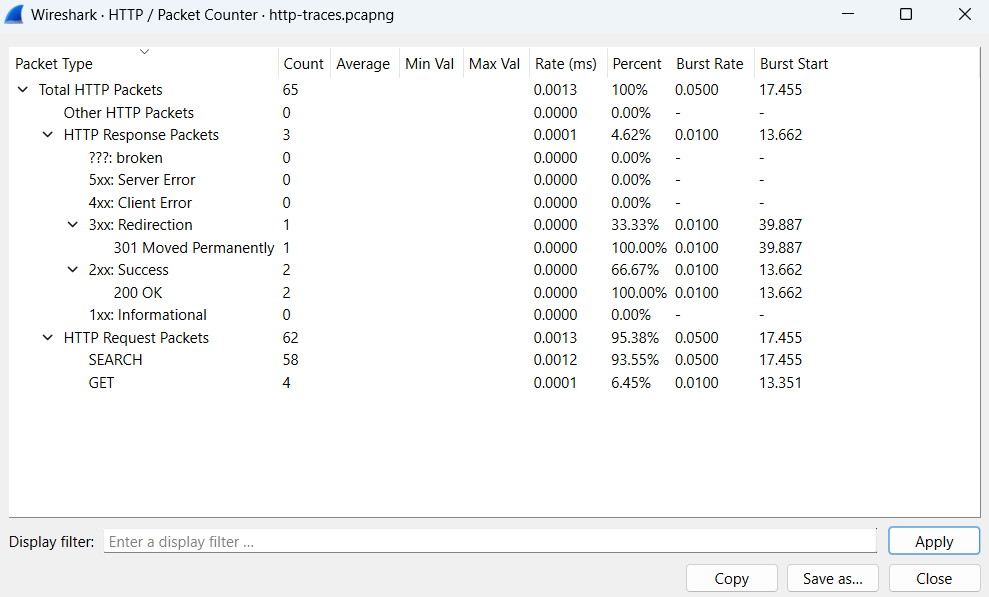
Extra: Breakdown by code

tshark -r trace.pcap -Y 'http.response' -T fields -e http.response.code | sort | uniq -c

That gives counts per status code (useful to show e.g., how many 200 vs 301).

What to report

The total number of HTTP responses and (optionally) a small table with counts per status code.



**TASK:06**

**Determine whether the connection is persistent or not. Justify with evidence.**

**Short summary of criteria:**

Persistent (keep-alive) evidence: multiple HTTP requests/responses occur over the same TCP connection (same tcp.stream) with no Connection: close header and without an immediate TCP FIN/RST. HTTP/1.1 is persistent by default unless Connection: close.

Non-persistent evidence: Connection: close header present, or each request/response pair is followed by TCP FIN/RST closing the connection, or HTTP/1.0 without Connection: keep-alive.

Detailed steps to prove it (Wireshark GUI)

Identify the TCP stream of the first GET: Right-click the GET packet → Follow → TCP Stream. The filter tcp.stream == N will be applied.

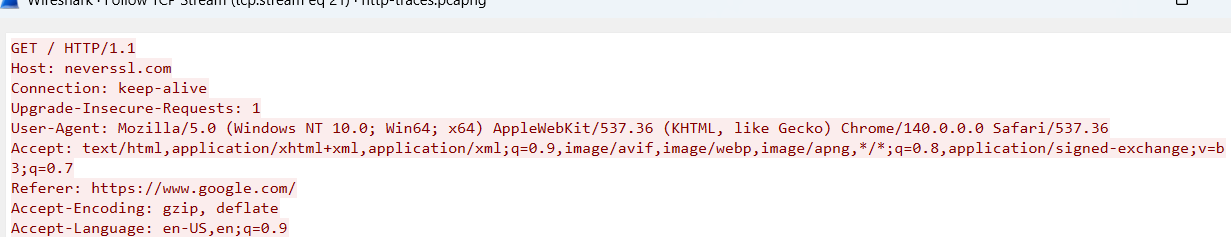
With that stream filter active, look at the packet list — count how many HTTP requests and responses appear in that TCP stream.

If you see more than one HTTP request in that same stream (without a TCP FIN/RST between requests), that is strong evidence of persistence.

Inspect headers for Connection: (in both request and response).

Connection: keep-alive -> explicit persistence.

Connection: close -> explicit non-persistence.Check for TCP FIN/RST packets immediately after the response; presence means connection closed, so likely non-persistent for that request.



**Answer: PERSISTENT**

**CONNECTION=KEEP ALIVE**