NAME - GAUTAM KUMAR MAHAR(2103114) AND AUMKAR LOREKAR (2003108)

LAB 5 (HTTP, Web and DNS)

Question 1.

Choose a name for your team. Create a website for your team focussed on any theme/topic of your choice, using only raw HTML and a text editor. (Some suggested themes: Internet memes, Networking-related jokes, Networking history...)

◦ The website should consist of at least 2 webpages, where one of the pages should be named "index.html"

◦ Each webpage should contain a hyperlink to the other.

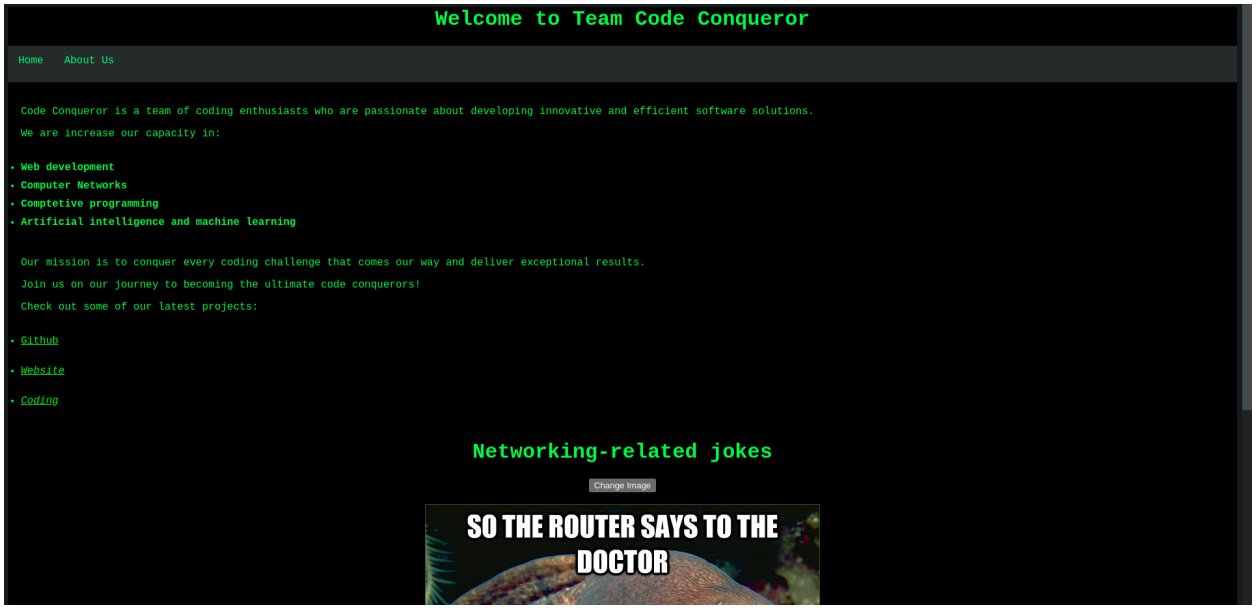
◦ The webpages should contain:

▪ at least one link to some external website

▪ at least one image

▪ some formatted text such as headings and bold text. Do not include inappropriate content on the web pages. Store the objects (.html files, images etc) in a single folder on the local machine and test your web pages by opening the .html files in a browser.

Answer-



Check out some of our latest projects:

- Github
- *Website*
- *Coding*

Networking-related jokes

Change Image

About Code Conqueror

Home About Us

Code Conqueror is a team of coding enthusiasts who are passionate about developing innovative and efficient software solutions.

We are a diverse team with expertise in various programming languages and technologies.

Our team members include:

- Gautam Kumar Mahar - Web Developer
- Gautam Kumar Mahar - Comptetive Programming
- Aumkar- AI/ML Engineer

Our goal is to provide high-quality software solutions that meet the needs of our clients and exceed their expectations.

Get in touch with us to learn more about how we can help you achieve your coding goals!

Here are some helpful coding resources (I also use these)-

- Codecademy
- Udemy
- Stack Overflow

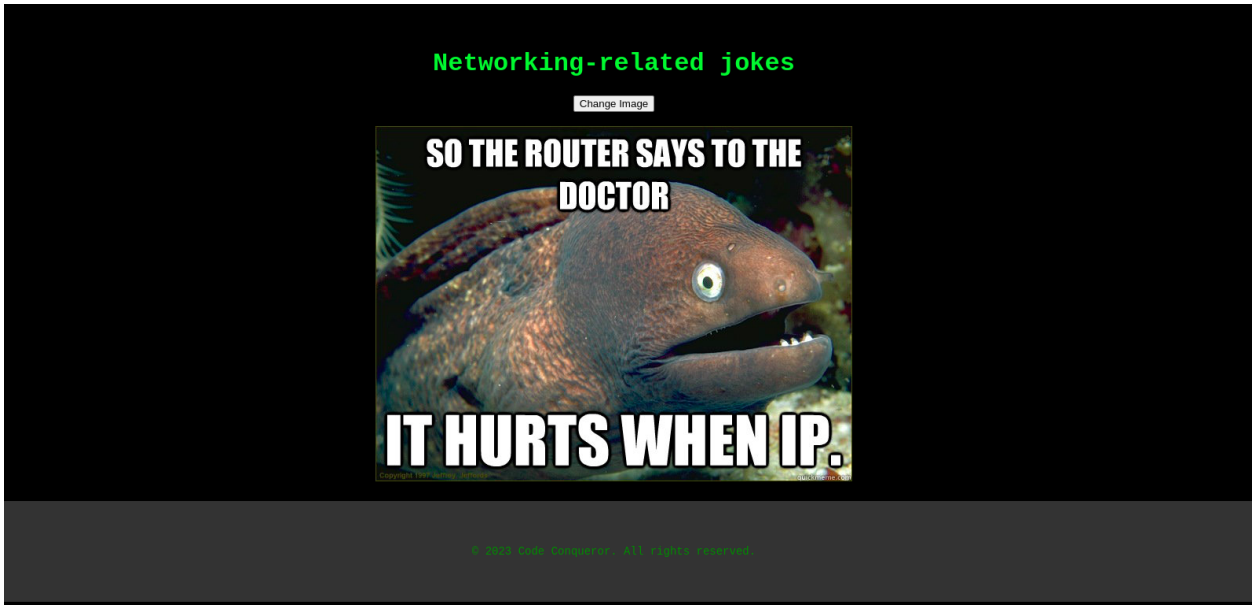
Question 2

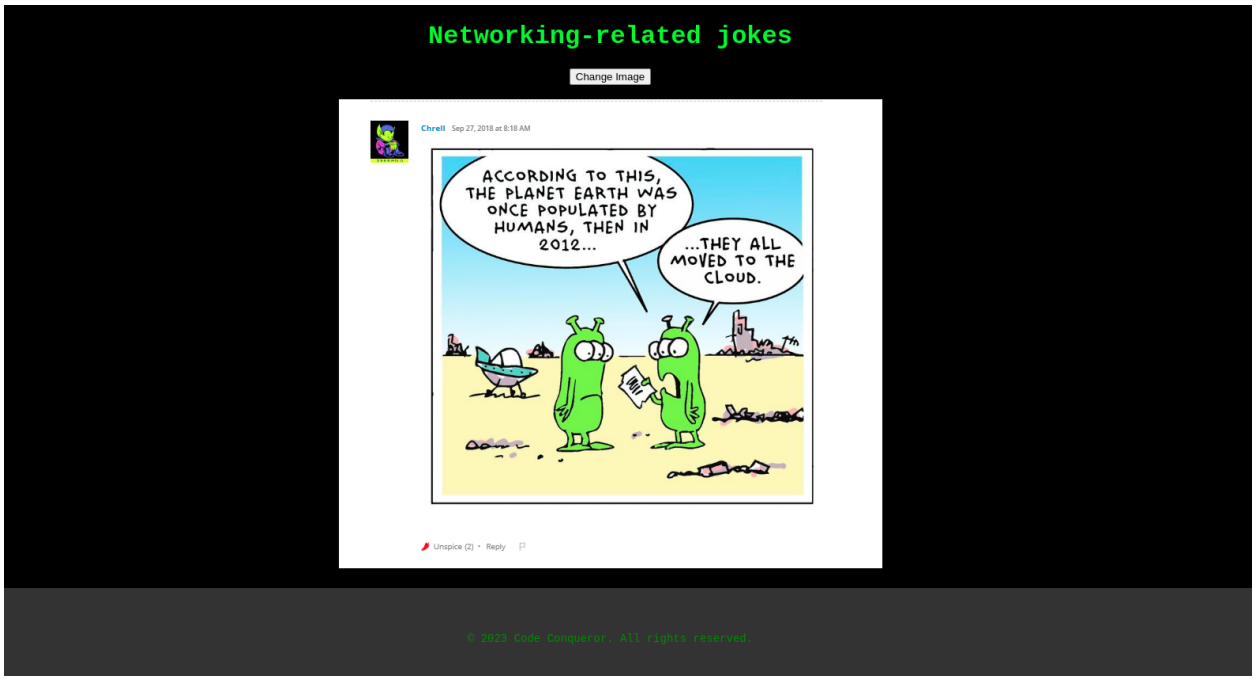
The essential components/technologies that go into a modern webpage are: ◦ HTML: to define the content of web pages ◦ CSS: to specify the style/layout of web pages and ◦ JavaScript: to program the behaviour of web pages Tutorials and references for all three can be found here: https://www.w3schools.com/default.asp.

• Go through the initial tutorials for Javascript at https://www.w3schools.com/js/default.asp.

• Using Javascript, add a button to your webpage (created in q1) such that each time the user clicks this button, one of the images in your webpage changes to some other image.

Ans:

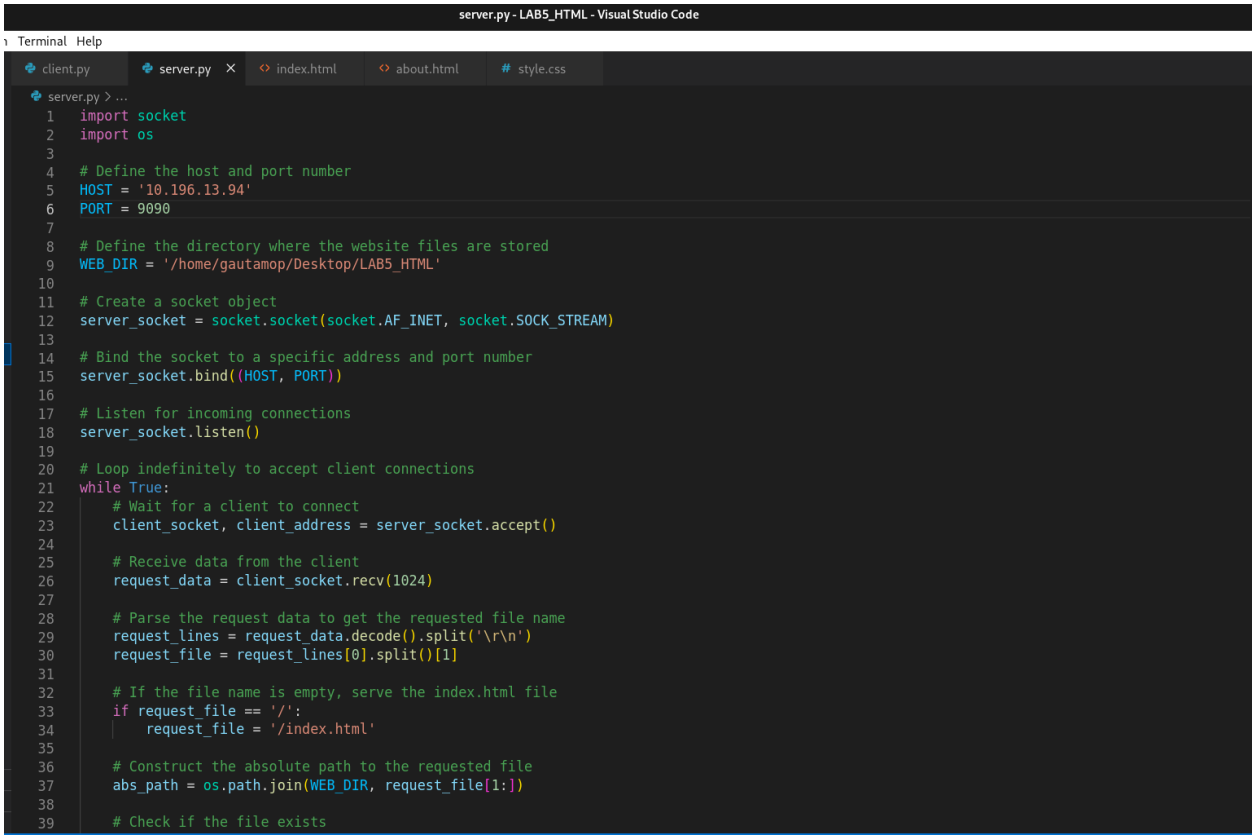




Question 3

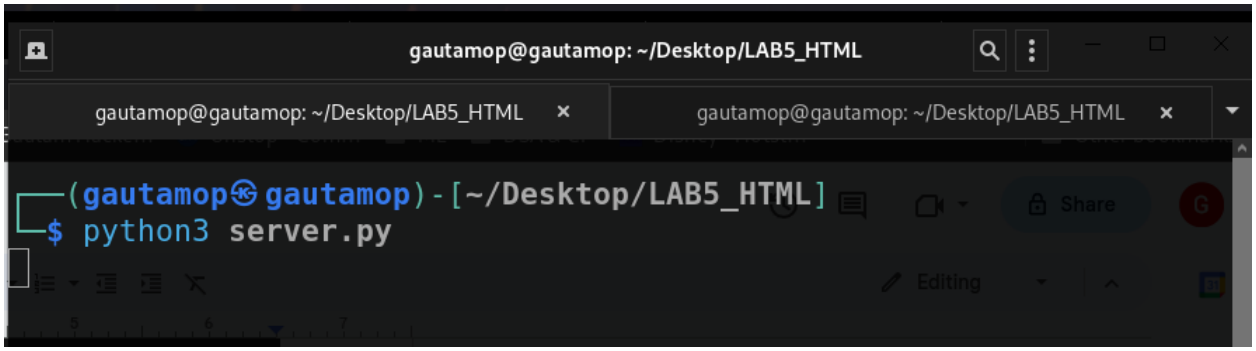
Create a web server for your website using Python. The web server process should accept a TCP connection from a Client process on port 9090 and generate responses to GET requests for objects contained in your website's folder. (You can modify and use the template for a simple Python Webserver provided on Google Classroom.)

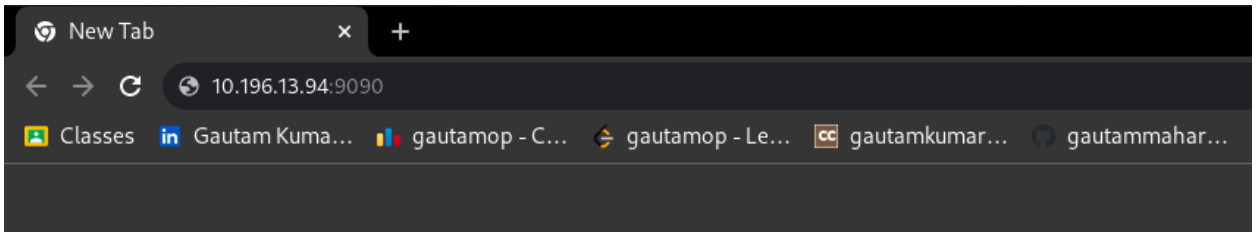
Answer

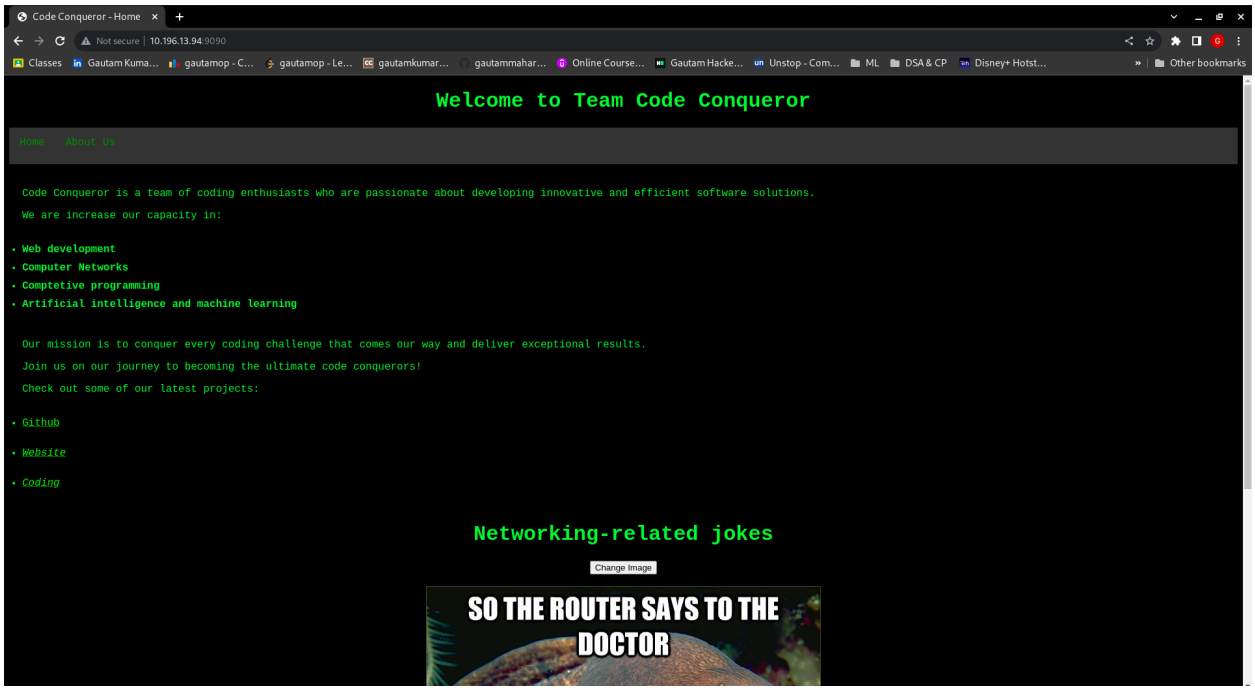


```
    # Parse the request data to get the requested file name
    request_lines = request_data.decode().split('\r\n')
    request_file = request_lines[0].split()[1]

    # If the file name is empty, serve the index.html file
    if request_file == '/':
        request_file = '/index.html'

    # Construct the absolute path to the requested file
    abs_path = os.path.join(WEB_DIR, request_file[1:])

    # Check if the file exists
    if os.path.exists(abs_path):
        # Open the file and read its contents
        with open(abs_path, 'rb') as file:
            file_contents = file.read()

        # Send a response back to the client
        response_data = b"HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n" + file_contents
        client_socket.sendall(response_data)
    else:
        # If the file doesn't exist, send a 404 error response
        response_data = b"HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n<html><body><h1>404 Not Found</h1></body></html>"
        client_socket.sendall(response_data)

    # Close the connection
    client_socket.close()

# Close the server socket
server_socket.close()
```

◦ **Test your web server by entering the following URL in a browser's address bar: "http://localhost:9090" on the same machine**

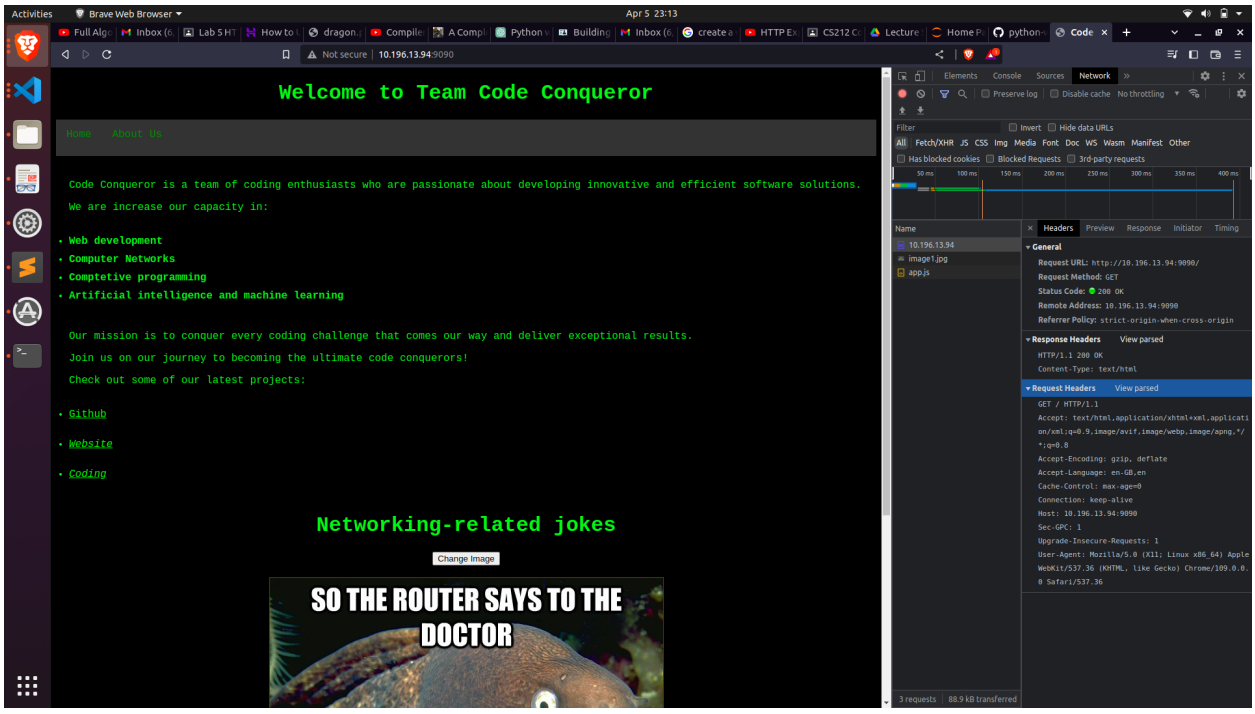






◦ **Check if your website can be accessed from another machine in the lab. Use the private IP address of the webserver (10.xx.xx.xx) instead of 'localhost'.**

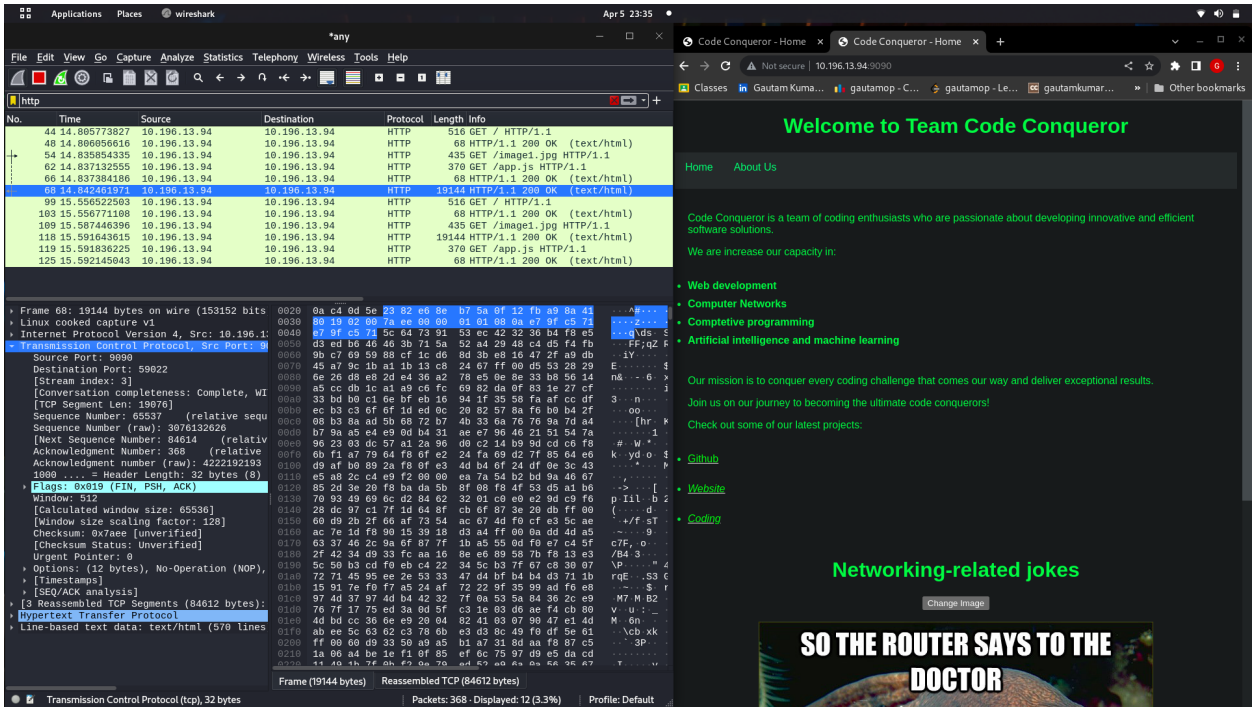
Other Machine POV:



Question 4 -

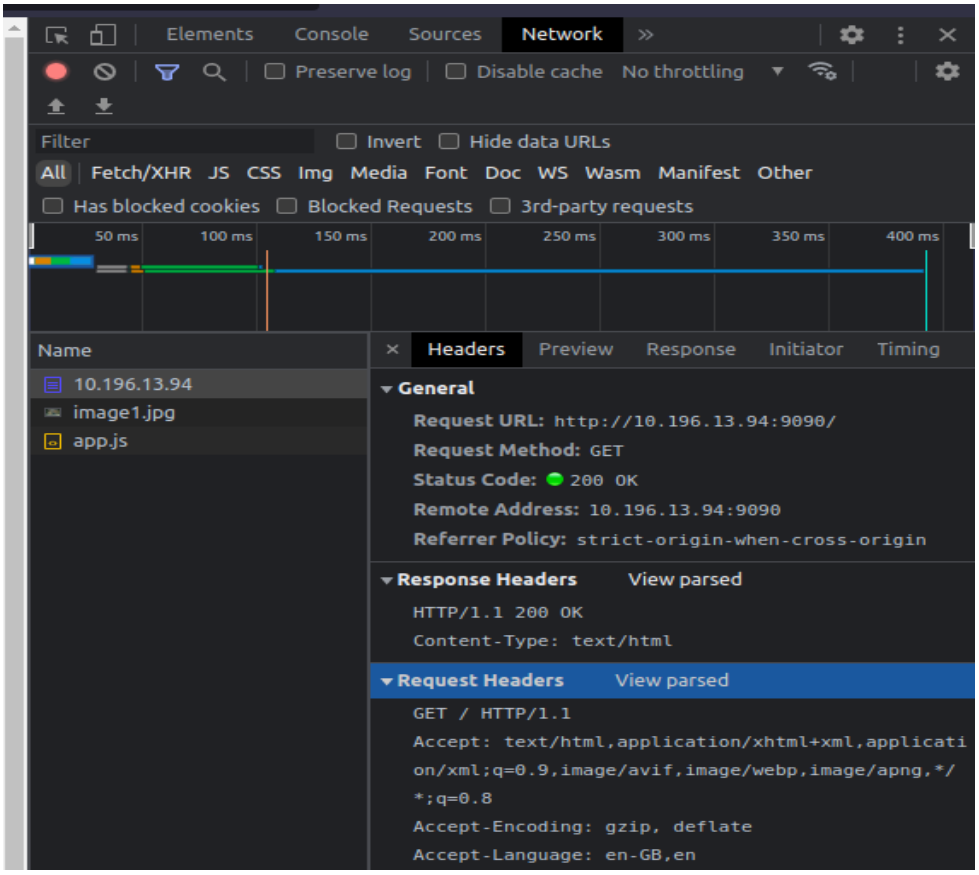
View the HTTP Request and Response traffic generated by the web browser using the browser's developer/Network tool. (For Google Chrome, press F12 to view the Network Tools)

◦ Check if you can see the "raw" HTTP requests and responses

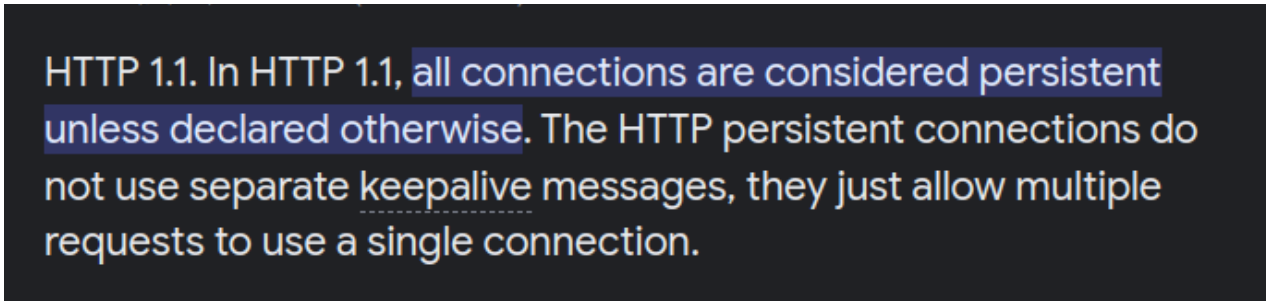


◦ Find out whether the persistent or non-persistent connection was used by the browser for viewing your webpages

Ans - The connection is persistent Because

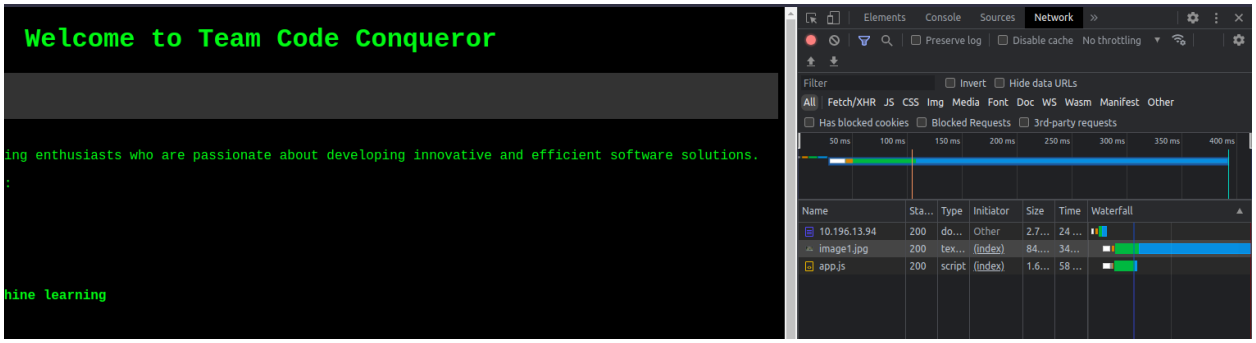


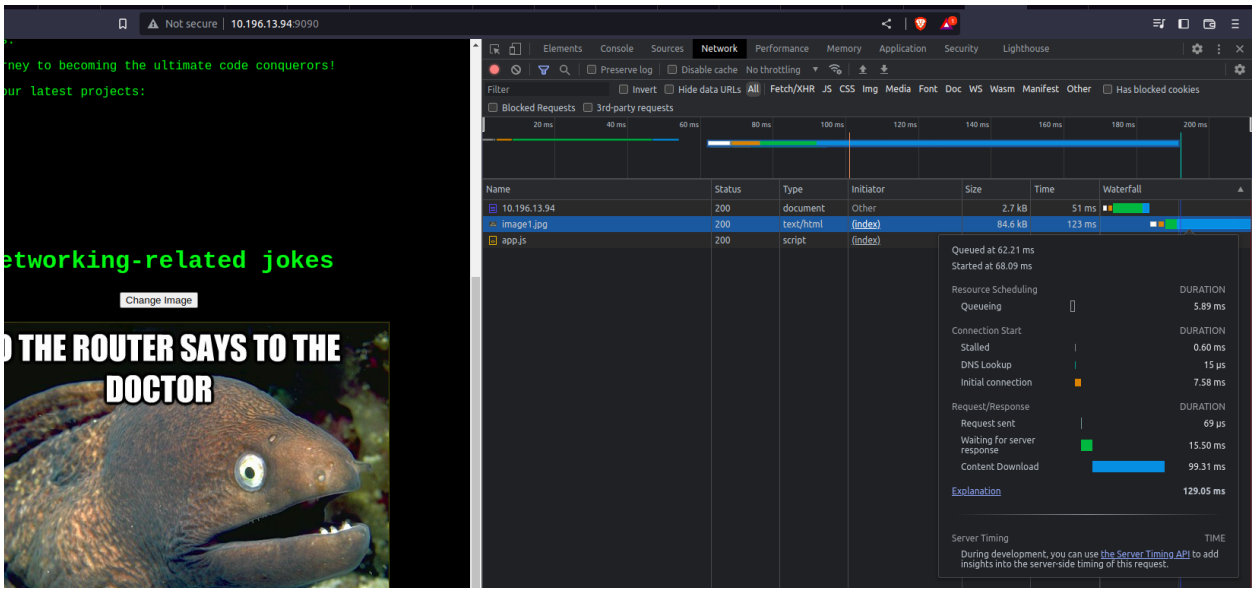
Here we can see HTTP/1.1 and

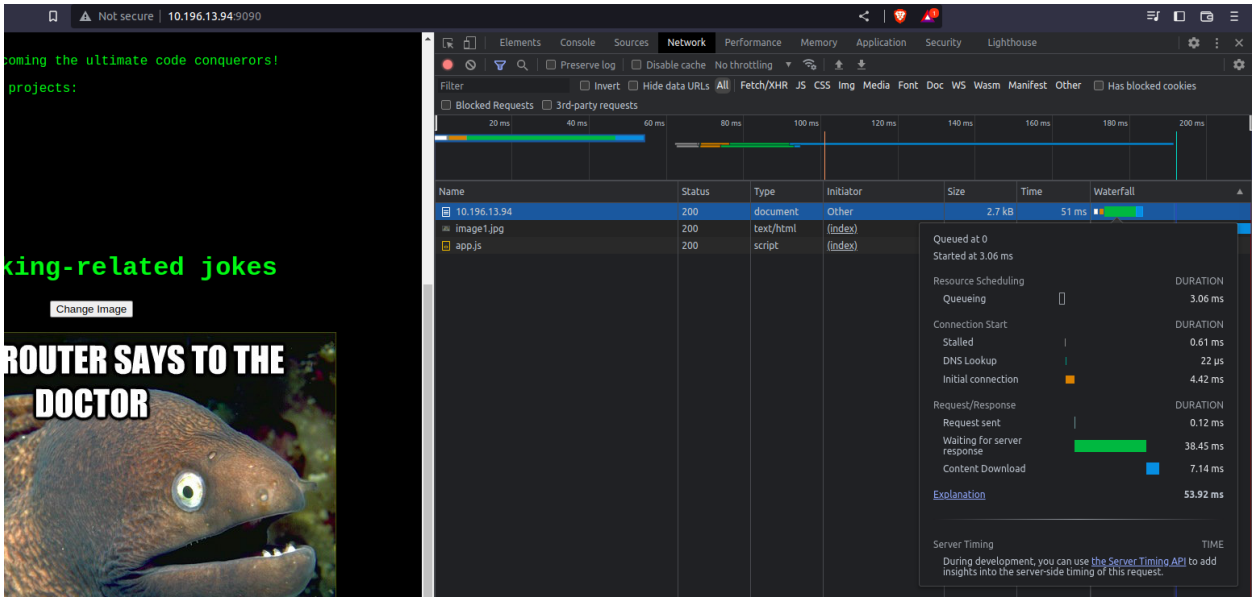


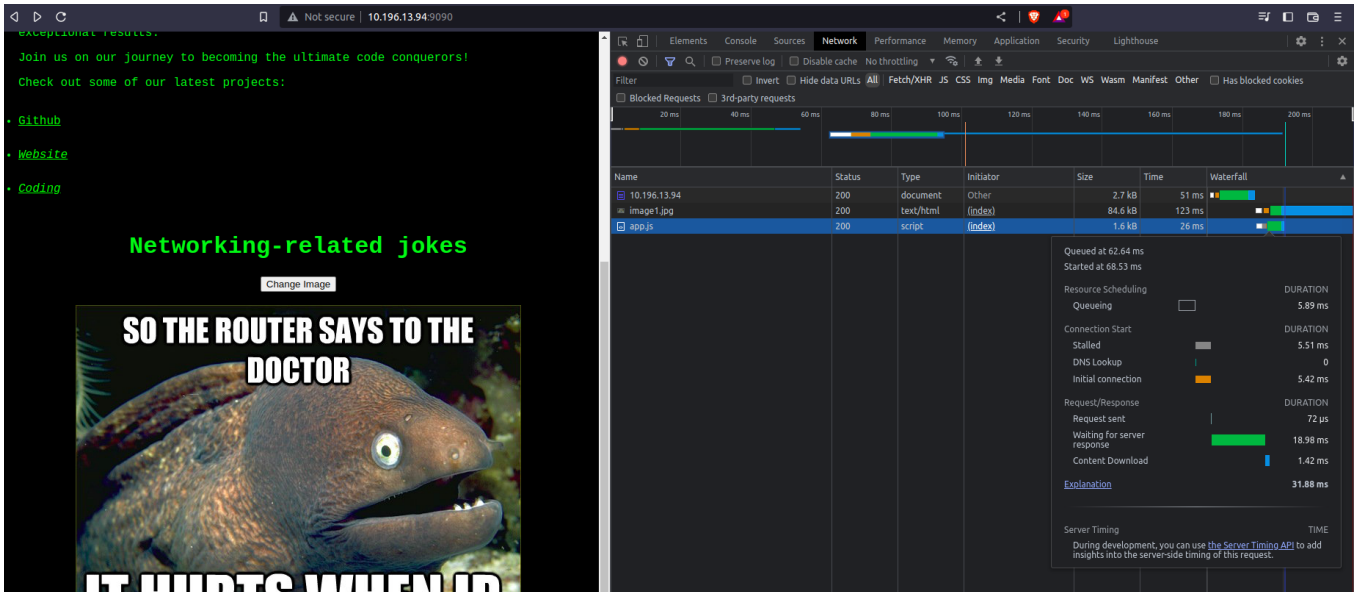
So, Now we can say that connect the connection is persistent.

◦ **Open another webpage hosted remotely, such as https://www.iitgoa.ac.in/~nehak/example.html View the HTTP traffic for this webpage in the Network tool. Does the browser download the components of the webpage (images etc) in a sequential or pipelined or parallel manner? [find out and show a demo to the TAs]**



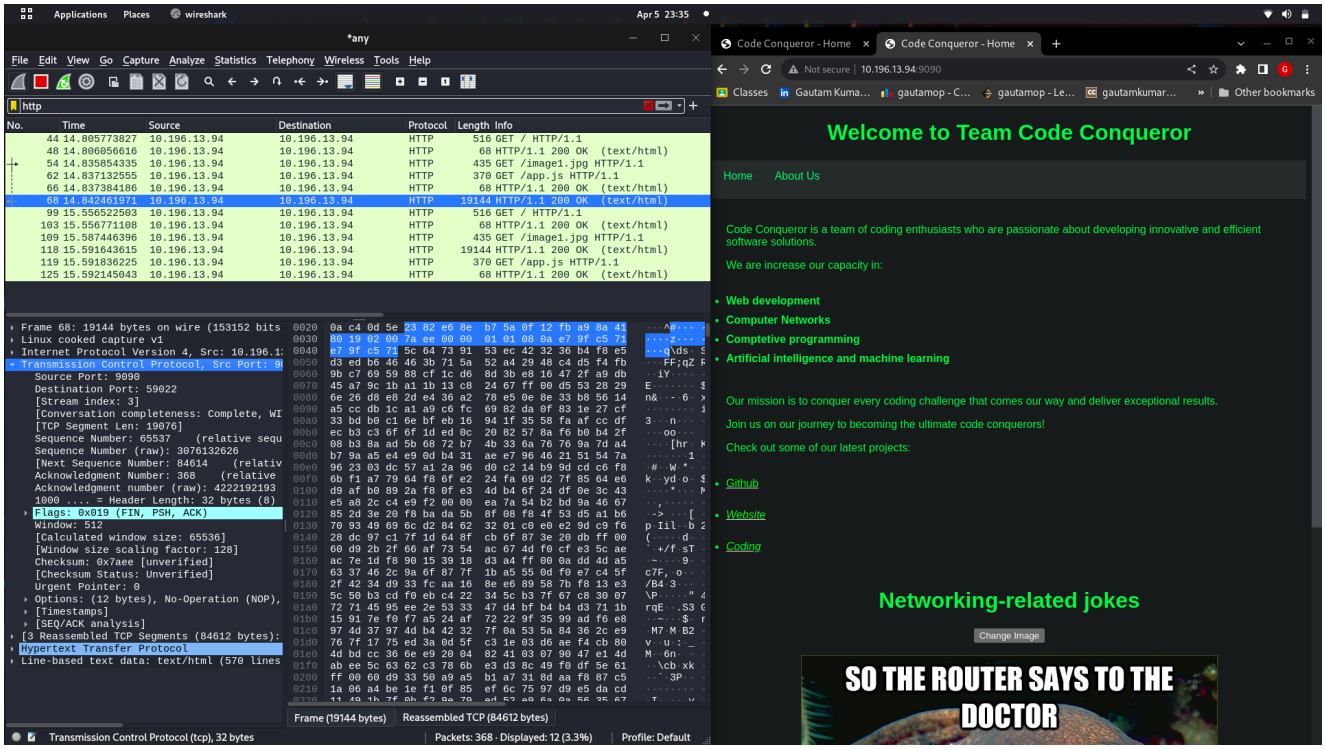


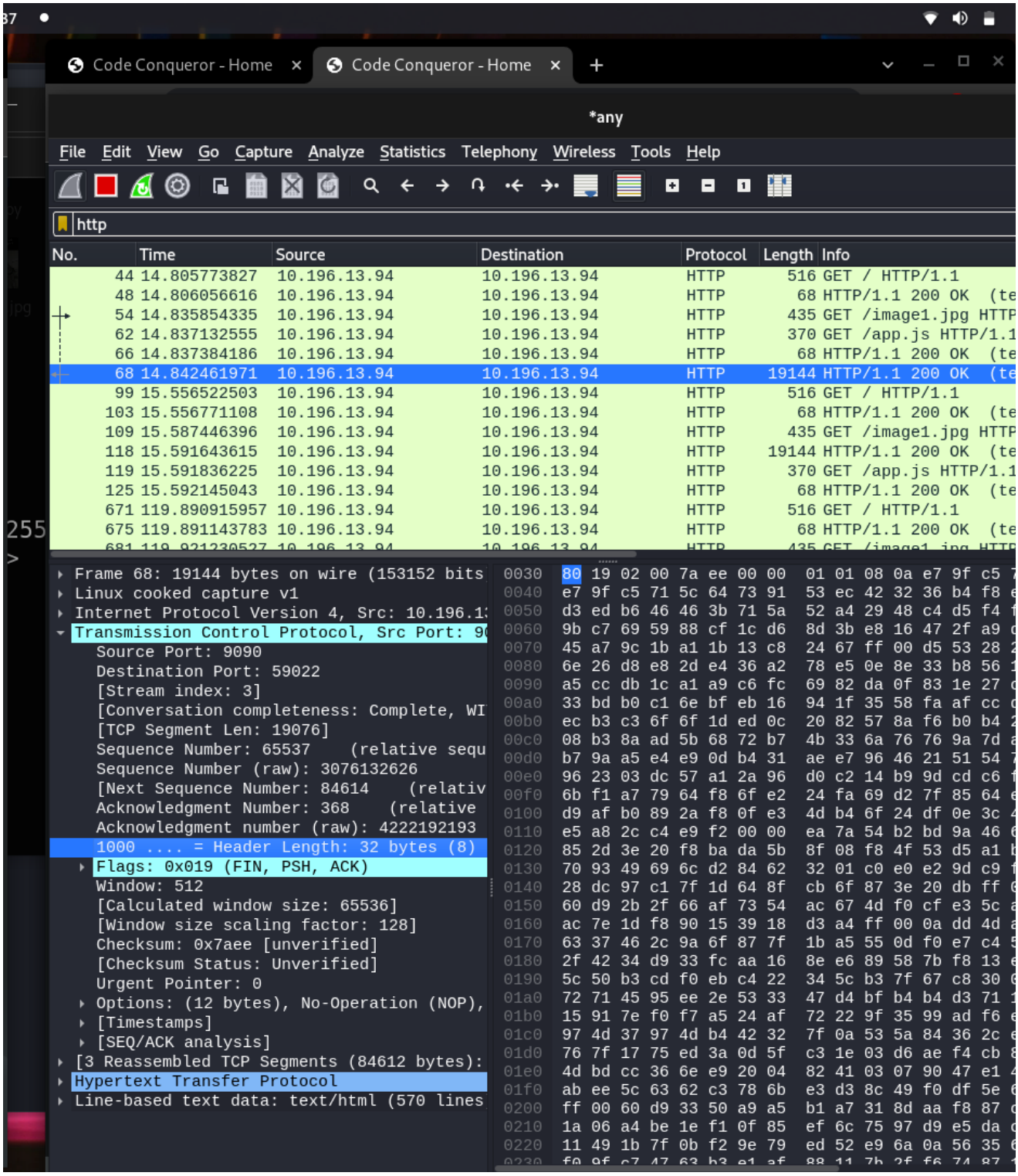




HERE, WE CAN SE THAT parallel manner

◦ Observe the HTTP requests in Wireshark. Look out for the TCP handshaking (SYN/ SYN-ACK/ACK sequence). Is there one persistent TCP connection per webpage?



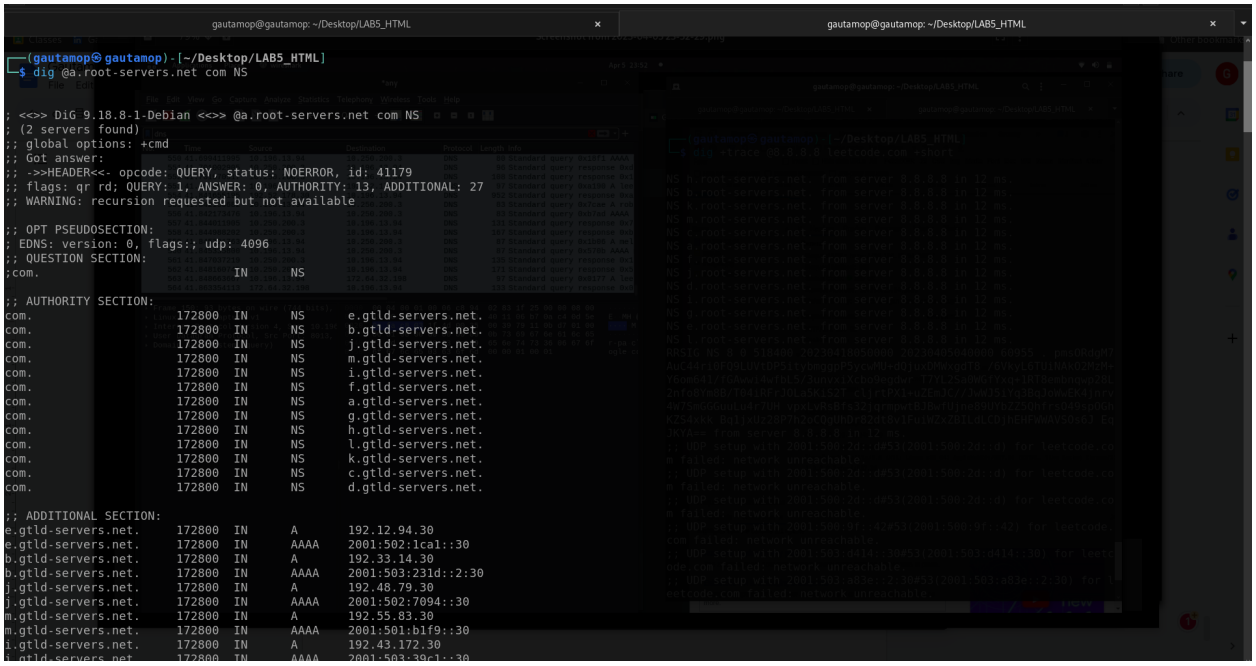

Code Conqueror - Home
Code Conqueror - Home
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
44 14.805773827 10.196.13.94 10.196.13.94 HTTP 516 GET / HTTP/1.1
48 14.806056616 10.196.13.94 10.196.13.94 HTTP 68 HTTP/1.1 200 OK (te
54 14.835854335 10.196.13.94 10.196.13.94 HTTP 435 GET /image1.jpg HTTP
62 14.837132555 10.196.13.94 10.196.13.94 HTTP 370 GET /app.js HTTP/1.1
66 14.837384186 10.196.13.94 10.196.13.94 HTTP 68 HTTP/1.1 200 OK (te
68 14.842461971 10.196.13.94 10.196.13.94 HTTP 19144 HTTP/1.1 200 OK (te
99 15.556522503 10.196.13.94 10.196.13.94 HTTP 516 GET / HTTP/1.1
103 15.556771108 10.196.13.94 10.196.13.94 HTTP 68 HTTP/1.1 200 OK (te
109 15.587446396 10.196.13.94 10.196.13.94 HTTP 435 GET /image1.jpg HTTP
118 15.591643615 10.196.13.94 10.196.13.94 HTTP 19144 HTTP/1.1 200 OK (te
119 15.591836225 10.196.13.94 10.196.13.94 HTTP 370 GET /app.js HTTP/1.1
125 15.592145043 10.196.13.94 10.196.13.94 HTTP 68 HTTP/1.1 200 OK (te
671 119.890915957 10.196.13.94 10.196.13.94 HTTP 516 GET / HTTP/1.1
675 119.891143783 10.196.13.94 10.196.13.94 HTTP 68 HTTP/1.1 200 OK (te
Frame 68: 19144 bytes on wire (153152 bits
Linux cooked capture v1
Internet Protocol Version 4, Src: 10.196.1
Transmission Control Protocol, Src Port: 9
Source Port: 9090
Destination Port: 59022
[Stream index: 3]
[Conversation completeness: Complete, WI
[TCP Segment Len: 19076]
Sequence Number: 65537 (relative sequ
Sequence Number (raw): 3076132626
[Next Sequence Number: 84614 (relativ
Acknowledgment Number: 368 (relative
Acknowledgment number (raw): 4222192193
1000 .... = Header Length: 32 bytes (8)
Flags: 0x019 (FIN, PSH, ACK)
Window: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0x7aee [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP),
[Timestamps]
[SEQ/ACK analysis]
[3 Reassembled TCP Segments (84612 bytes):
Hypertext Transfer Protocol
Line-based text data: text/html (570 lines
0030 80 19 02 00 7a ee 00 00 01 01 08 0a e7 9f c5
0040 e7 9f c5 71 5c 64 73 91 53 ec 42 32 36 b4 f8
0050 d3 ed b6 46 46 3b 71 5a 52 a4 29 48 c4 d5 f4
0060 9b c7 69 59 88 cf 1c d6 8d 3b e8 16 47 2f a9
0070 45 a7 9c 1b a1 1b 13 c8 24 67 ff 00 d5 53 28
0080 6e 26 d8 e8 2d e4 36 a2 78 e5 0e 8e 33 b8 56
0090 a5 cc db 1c a1 a9 c6 fc 69 82 da 0f 83 1e 27
00a0 33 bd b0 c1 6e bf eb 16 94 1f 35 58 fa af cc
00b0 ec b3 c3 6f 6f 1d ed 0c 20 82 57 8a f6 b0 b4
00c0 08 b3 8a ad 5b 68 72 b7 4b 33 6a 76 76 9a 7d
00d0 b7 9a a5 e4 e9 0d b4 31 ae e7 96 46 21 51 54
00e0 96 23 03 dc 57 a1 2a 96 d0 c2 14 b9 9d cd c6
00f0 6b f1 a7 79 64 f8 6f e2 24 fa 69 d2 7f 85 64
0100 d9 af b0 89 2a f8 0f e3 4d b4 6f 24 df 0e 3c
0110 e5 a8 2c c4 e9 f2 00 00 ea 7a 54 b2 bd 9a 46
0120 85 2d 3e 20 f8 ba da 5b 8f 08 f8 4f 53 d5 a1
0130 70 93 49 69 6c d2 84 62 32 01 c0 e0 e2 9d c9
0140 28 dc 97 c1 7f 1d 64 8f cb 6f 87 3e 20 db ff
0150 60 d9 2b 2f 66 af 73 54 ac 67 4d f0 cf e3 5c
0160 ac 7e 1d f8 90 15 39 18 d3 a4 ff 00 0a dd 4d
0170 63 37 46 2c 9a 6f 87 7f 1b a5 55 0d f0 e7 c4
0180 2f 42 34 d9 33 fc aa 16 8e e6 89 58 7b f8 13
0190 5c 50 b3 cd f0 eb c4 22 34 5c b3 7f 67 c8 30
01a0 72 71 45 95 ee 2e 53 33 47 d4 bf b4 b4 d3 71
01b0 15 91 7e f0 f7 a5 24 af 72 22 9f 35 99 ad f6
01c0 97 4d 37 97 4d b4 42 32 7f 0a 53 5a 84 36 2c
01d0 76 7f 17 75 ed 3a 0d 5f c3 1e 03 d6 ae f4 cb
01e0 4d bd cc 36 6e e9 20 04 82 41 03 07 90 47 e1
01f0 ab ee 5c 63 62 c3 78 6b e3 d3 8c 49 f0 df 5e
0200 ff 00 60 d9 33 50 a9 a5 b1 a7 31 8d aa f8 87
0210 1a 06 a4 be 1e f1 0f 85 ef 6c 75 97 d9 e5 da
0220 11 49 1b 7f 0b f2 9e 79 ed 52 e9 6a 0a 56 35

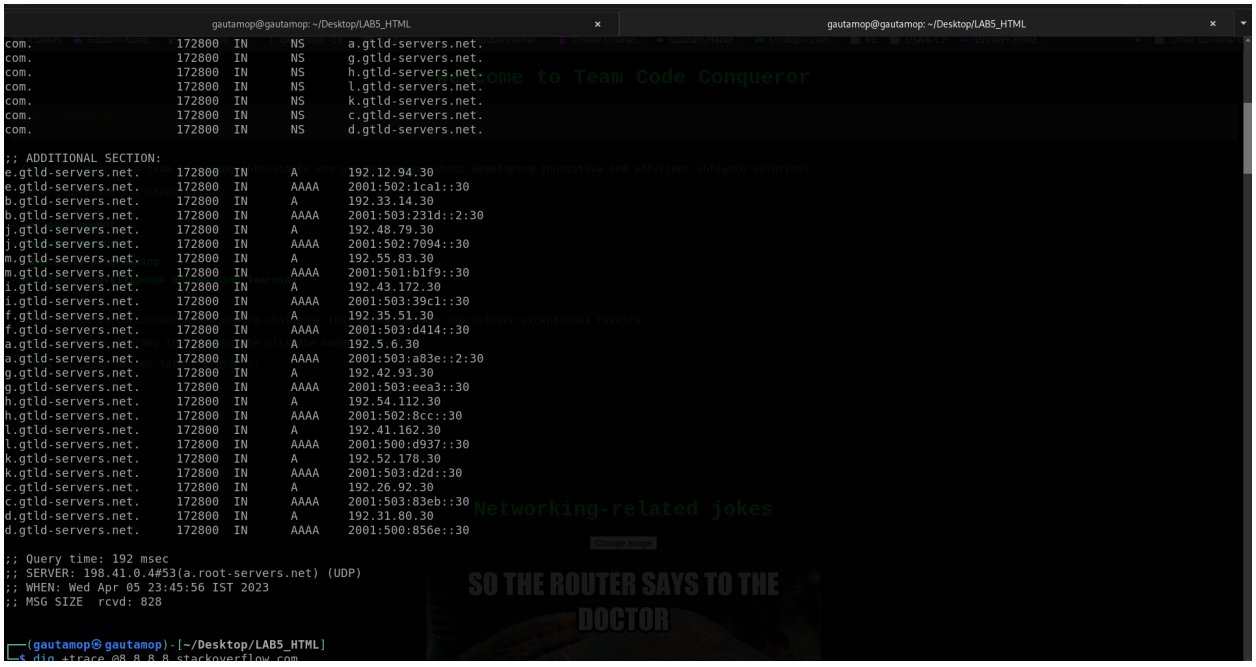
Question 5-

You can use the dig command to send DNS queries to any specified DNS server.
The general format of the command: $ dig @<DNS server> <name to be looked up> <type> Example: $ dig @a.root-servers.net com NS

◦ Go through this short tutorial for the dig command
https://www.hostinger.in/tutorials/how-to-use-the-dig-command-in-linux/

◦ Find out the appropriate command options (for dig) to send a DNS query to the root server "a.root-servers.net", requesting the address of all name-servers (NS) for the "com" top-level domains.

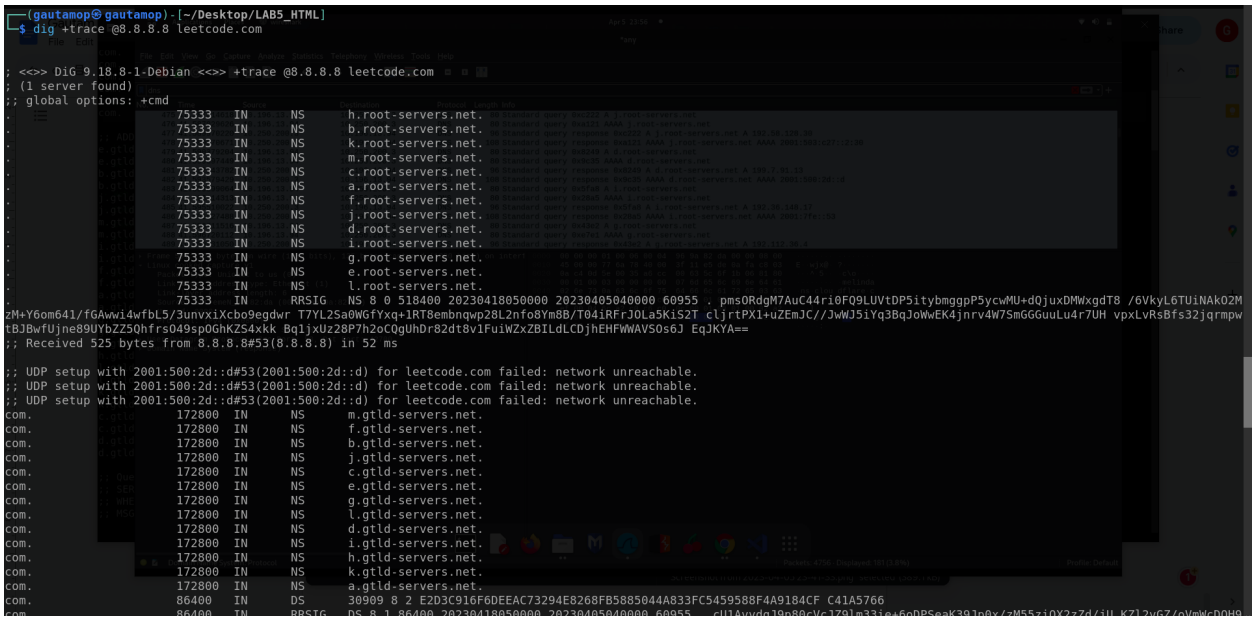


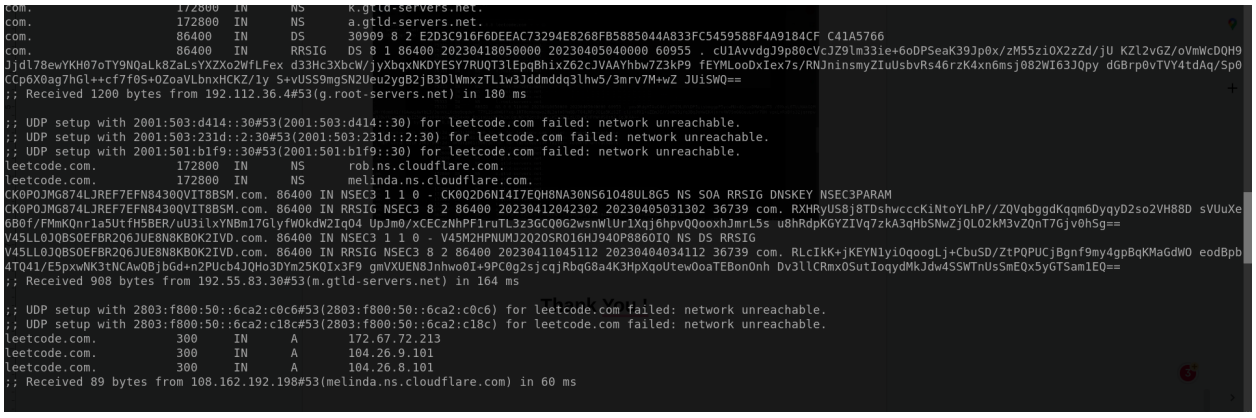


◦ **Try running dig with the “+trace” option to see all the steps performed in iteratively querying the hierarchy of DNS servers. Try to make sense of the output. Example: dig +trace @8.8.8.8 stackoverflow.com**

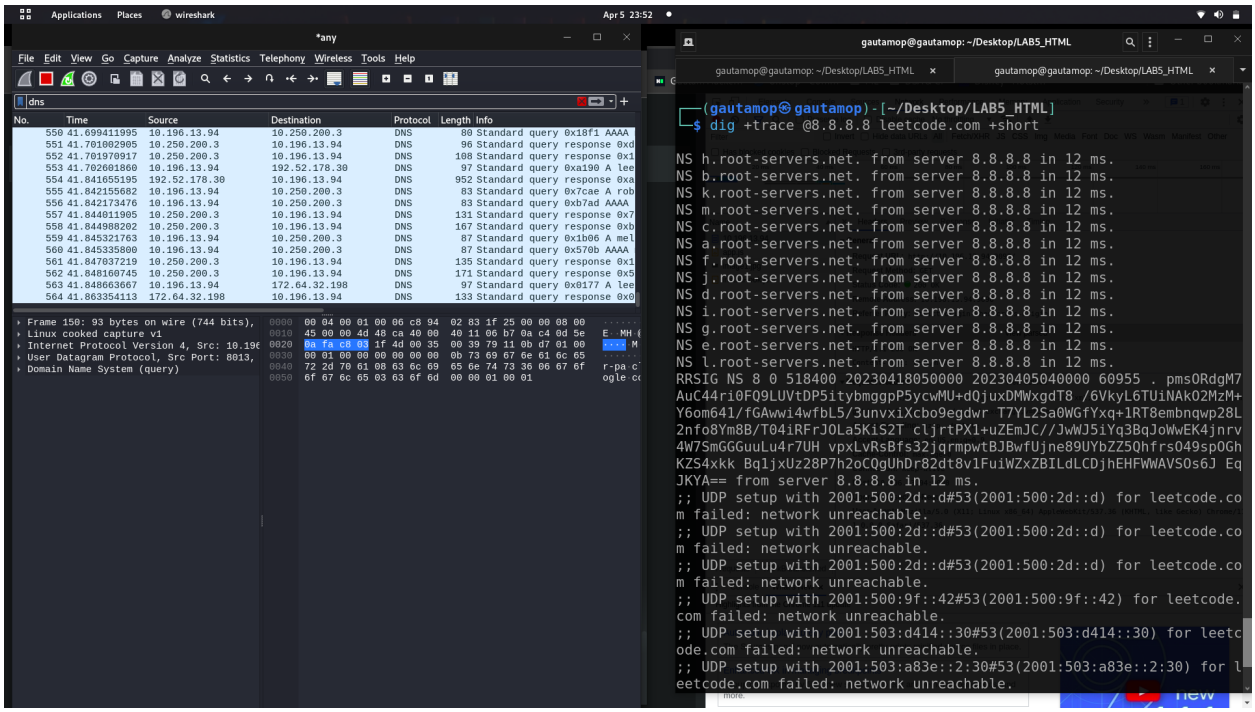
◦ **Open Wireshark and observe the DNS traffic on your computer using the filter “DNS”**

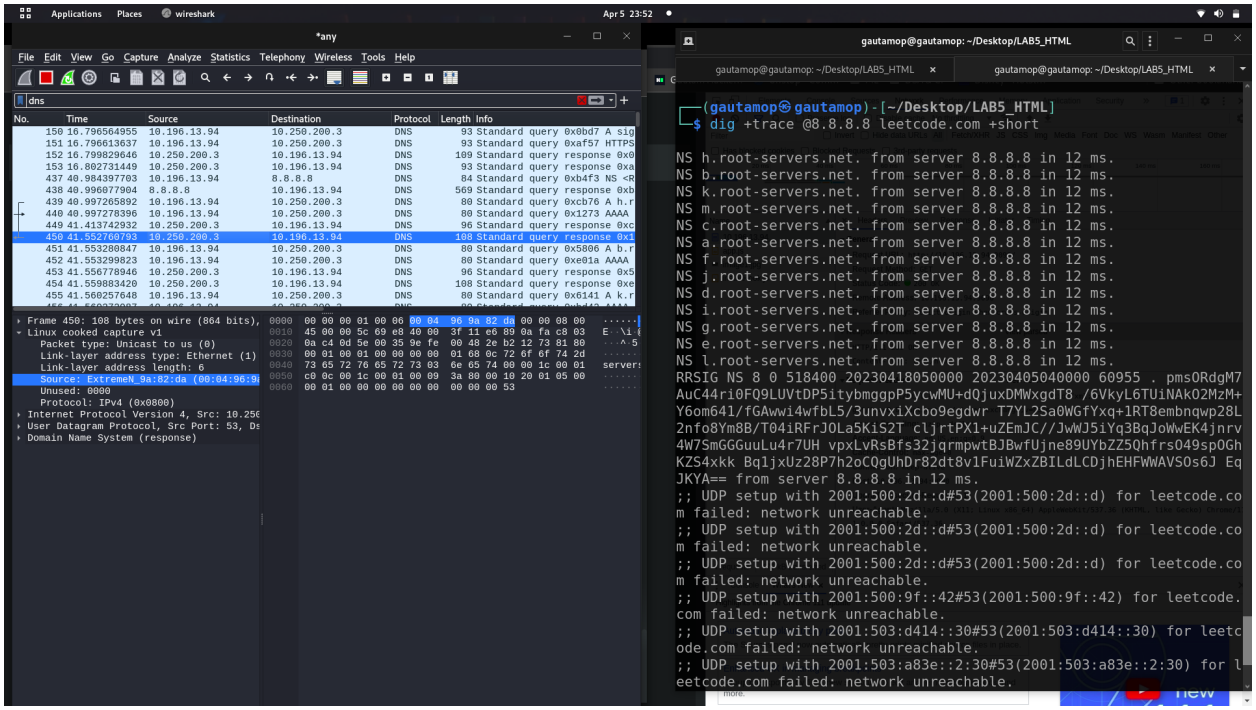
We try with leetcode



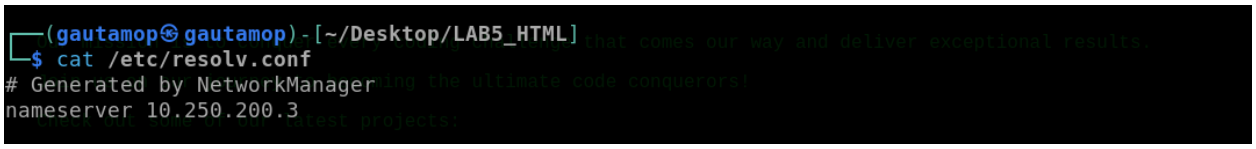


```
┌──(gautamop㉿gautamop)-[~/Desktop/LAB5_HTML]
└─$ dig +trace @8.8.8.8 leetcode.com +short

NS h.root-servers.net. from server 8.8.8.8 in 68 ms.
NS b.root-servers.net. from server 8.8.8.8 in 68 ms.
NS k.root-servers.net. from server 8.8.8.8 in 68 ms.
NS m.root-servers.net. from server 8.8.8.8 in 68 ms.
NS c.root-servers.net. from server 8.8.8.8 in 68 ms.
NS a.root-servers.net. from server 8.8.8.8 in 68 ms.
NS f.root-servers.net. from server 8.8.8.8 in 68 ms.
NS j.root-servers.net. from server 8.8.8.8 in 68 ms.
NS d.root-servers.net. from server 8.8.8.8 in 68 ms.
NS i.root-servers.net. from server 8.8.8.8 in 68 ms.
NS g.root-servers.net. from server 8.8.8.8 in 68 ms.
NS e.root-servers.net. from server 8.8.8.8 in 68 ms.
NS l.root-servers.net. from server 8.8.8.8 in 68 ms.
RRSIG NS 8 0 518400 20230418050000 20230405040000 60955 . pmsORdgM7AuC44ri0FQ9LUVtDP5itybmggpP5ycwMU+dQjuxDMWxgdT8 /6VkyL6TUiNAkO2MzM+Y6om641/fGAwwi4wfbL5/3unvxiXcbo9egdwr T
7YL2Sa0WGfYxq+1RT8embnqwp28L2nfo8Ym8B/T04iRFrJOLa5KiS2T cljrtPX1+uZEmJC//JwWJ5iYq3BqJoWwEK4jnrv4W7SmGGGuuLu4r7UH vpxLvRsBfs32jqrmpwtBJBwfUjne89UYbZZ5Qhfrs049spOGhKZS4xkk Bq1
jxUz28P7h2oCQgUhDr82dt8v1FuiWZxZBILdLCDjhEHFWWAVSOs6J EqJKYA== from server 8.8.8.8 in 68 ms.
;; UDP setup with 2001:500:200::b#53(2001:500:200::b) for leetcode.com failed: network unreachable.
;; UDP setup with 2001:500:200::b#53(2001:500:200::b) for leetcode.com failed: network unreachable.
;; UDP setup with 2001:500:200::b#53(2001:500:200::b) for leetcode.com failed: network unreachable.
A 172.67.72.213 from server 173.245.58.198 in 64 ms.
A 104.26.8.101 from server 173.245.58.198 in 64 ms.
A 104.26.9.101 from server 173.245.58.198 in 64 ms.
```





◦ **Find the address of your local DNS server**



◦ **Find out the addresses of any 3 root-level DNS servers**

Thank You