CS301 Assignment-2

Prajapati Harsh Pareshkumar
12241300

Part-1 HTTP

[Solution-1]

We can apply filters into wireshark as **http.request.method == "GET"** to find the total number of GET requests.
Now, to count the number of total GET requests goto **statistics →HTTP →count number.**

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Total HTTP Packets	43				0.0102	100%	0.0800	0.421
Other HTTP Packets	0				0.0000	0.00%	-	-
HTTP Response Packets	13				0.0031	30.23%	0.0300	0.421
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
4xx: Client Error	0				0.0000	0.00%	-	-
3xx: Redirection	0				0.0000	0.00%	-	-
2xx: Success	13				0.0031	100.00%	0.0300	0.421
200 OK	13				0.0031	100.00%	0.0300	0.421
1xx: Informational	0				0.0000	0.00%	-	-
HTTP Request Packets	30				0.0071	69.77%	0.0500	0.423
GET	30				0.0071	100.00%	0.0500	0.423

In the above image, we can see that there are a total of 43 HTTP packets from which 30 packets were HTTP Request Packets.

Now, to find how many packets for embedded content and how many for text, we can set filters for the GET requests.

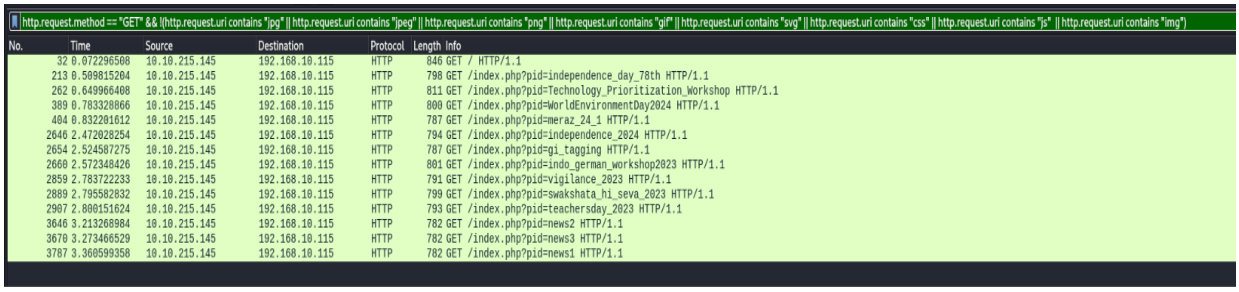
Command: **http.request.method == "GET" && (http.request.uri contains "css" || http.request.uri contains "js" || http.request.uri contains "png" || http.request.uri contains "jpg" || http.request.uri contains "jpeg" || http.request.uri contains "gif" || http.request.uri contains "ico" || http.request.uri contains "svg" || http.request.uri contains "img")**

The above filter will show only HTTP GET requests with embedded content which is shown in the below figure.There are 16 packets with embedded content.

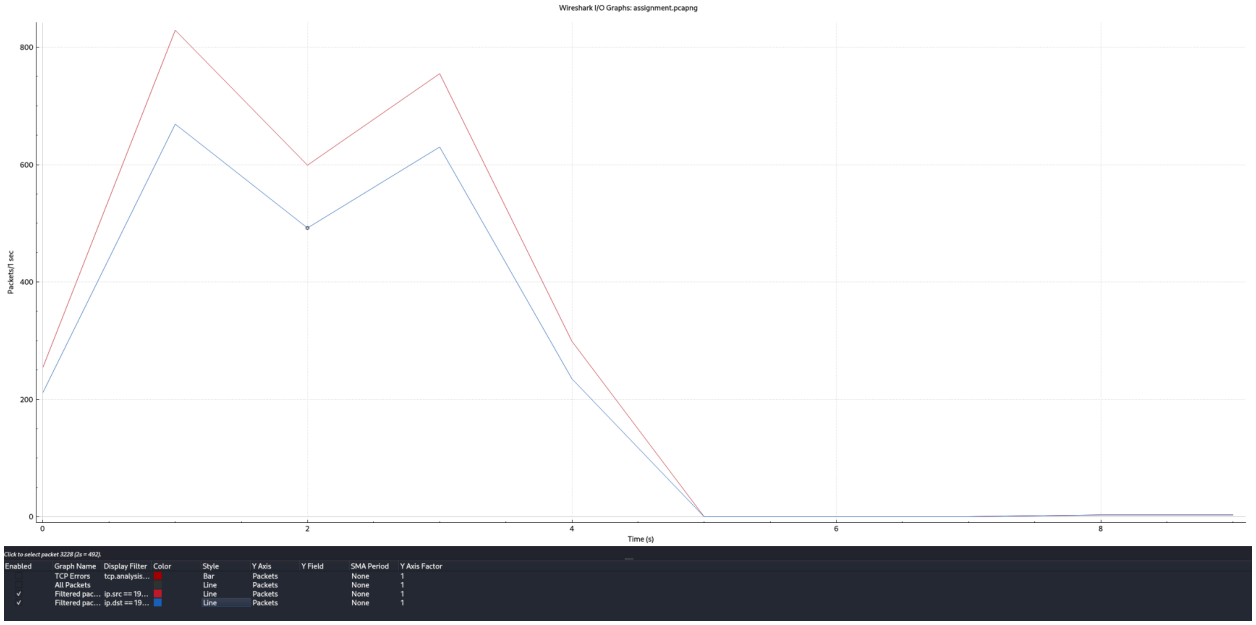
Now, to filter GET requests with text contents, make a filter to not show requests with embedded content.

Command: **http.request.method == "GET" && !(http.request.uri contains "css" || http.request.uri contains "js" || http.request.uri contains "png" || http.request.uri contains "jpg" || http.request.uri contains "jpeg" || http.request.uri contains "gif" || http.request.uri contains "ico" || http.request.uri contains "svg" || http.request.uri contains "img")**

Below figure shows the output of the command which are GET requests with text only.We can clearly see that total 14 requests are there.



Now, we have to plot an I/O graph for the packets sent to the host.
For that, goto **Statistics → I/O Graphs** and then apply filters in the panel below.



In the above Graph, Blue line-graph is for the packets sent to host iitbhilai.ac.in (filter: ip.dst==192.168.10.115) and red line-graph is for packets received from the same host (filter: ip.src==192.168.10.115).

[Solution-2]

In this question we have to calculate the total amount of data received for each HTTP GET request.

Steps to do it manually:
(1) filter the HTTP Response packets
(2) open the packet
(3) note the Request frame corresponding to response and the File Data

Response Frame	Request Frame	Data (Bytes)
115	32	36976
163	94	121033
174	101	19232
184	105	6065
197	166	31004
217	176	379
243	188	116840
282	202	36312
402	284	37045
2355	404	555508
2653	2356	302
3644	2889	2365936
5116	3646	593471

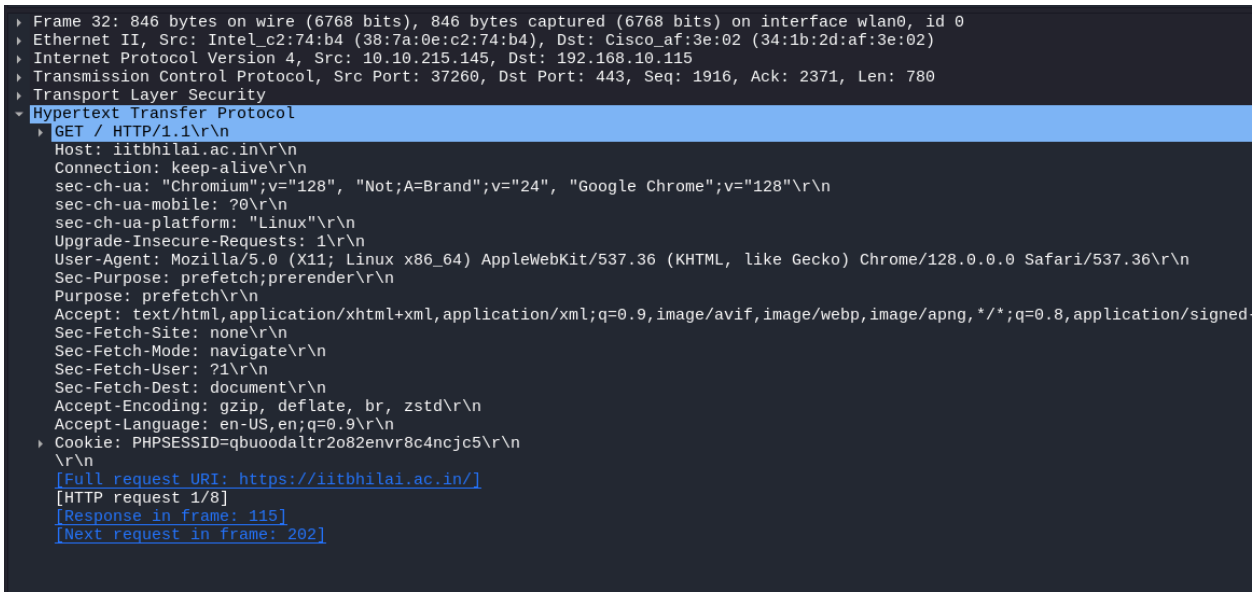
[Solution-3]

The above image is converted from the hex stream.

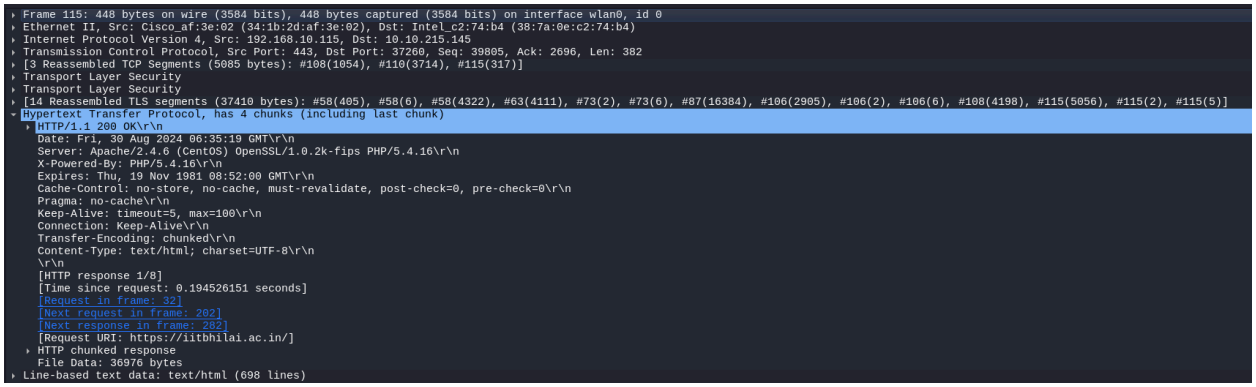
For this, first go to the response containing the image file and copy the file data as a hex stream from inside the packet.
Now, use any online hex to image converter to convert the hex into an image file.

[Solution-4]

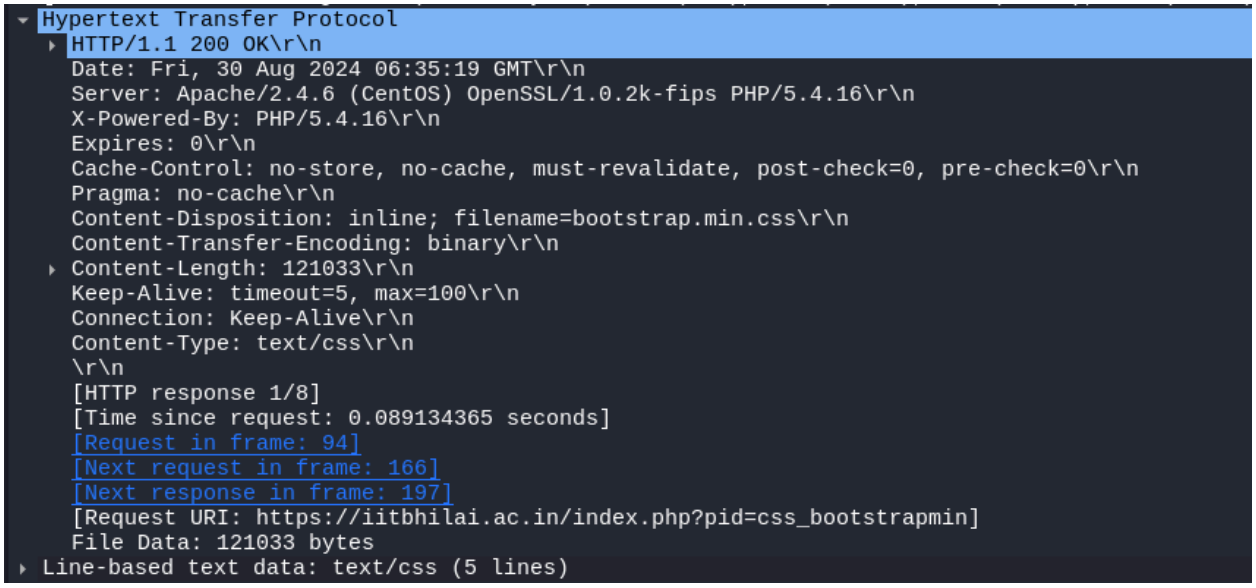
(a) No, in the first HTTP GET request from browser to server, there is no “IF MODIFIED SINCE” line in the packet.Hence, we can say that it was not a conditional GET request.



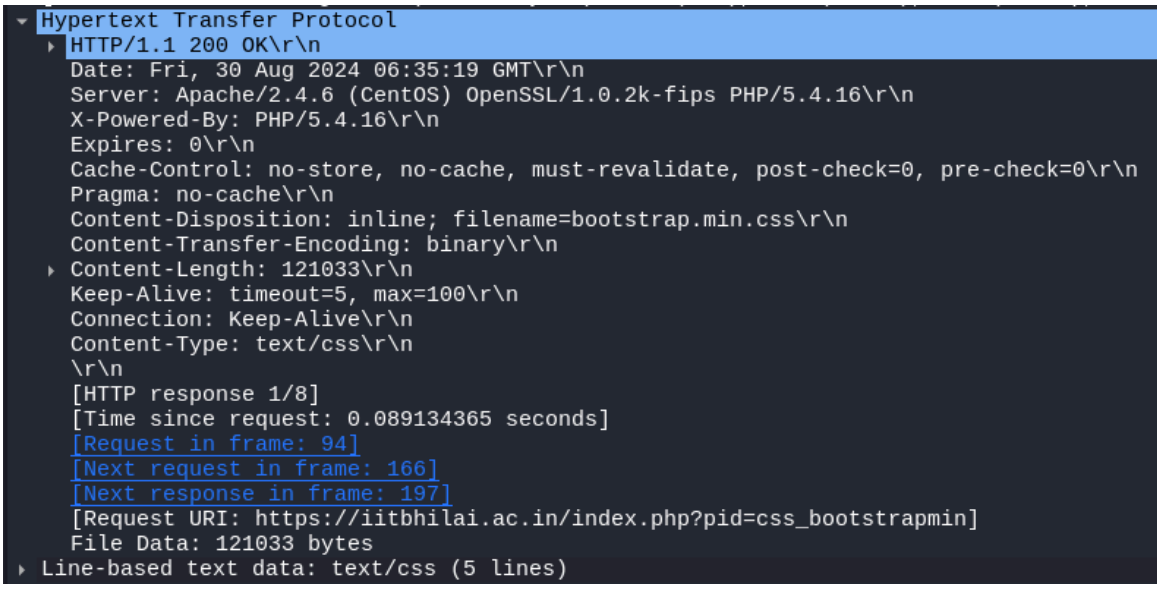
(b) Yes, the server sent the data in an explicit manner.
We can see that the first HTTP request was responded to in frame 115 and the content type is text/HTML.There it is also mentioned that response data was sent to the browser into **4 chunks of data**.Hence, we can say that data was sent in an explicit manner.



(c) There is no “IF MODIFIED SINCE” line in the 2nd packet of HTTP GET Request.



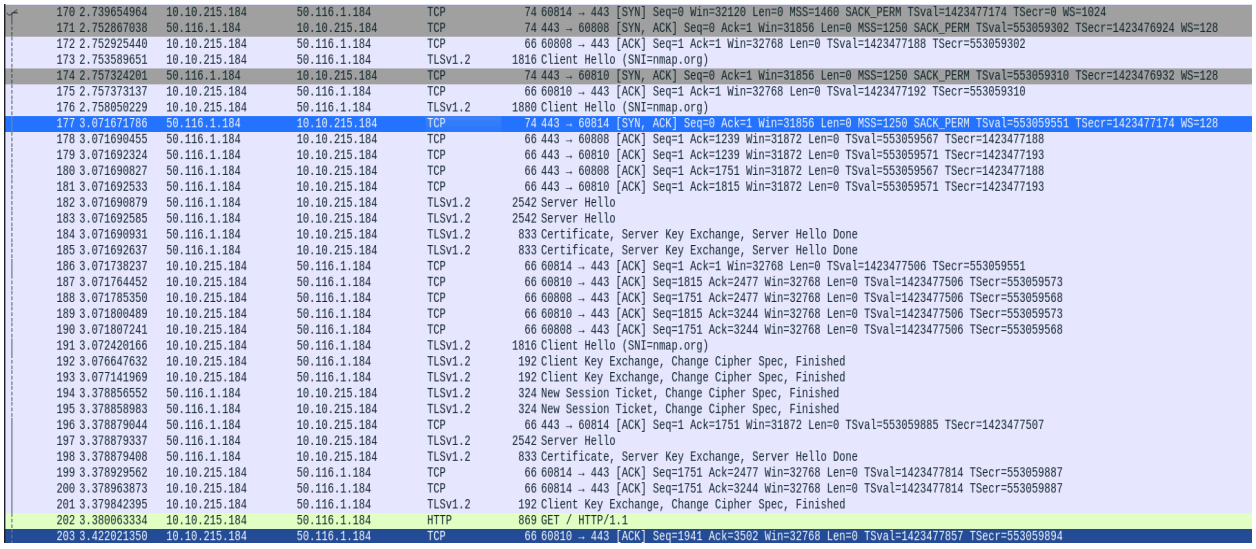
(d) HTTP status code 200 with OK message was returned to the 2nd HTTP GET Request.



With response, the server sent the content length of the data in the packet.
Hence, we can say that the data was sent explicitly in the response to the 2nd HTTP GET request.

[Solution-5]

For this question I visited **nmap.org** website.



170	2.739654964	10.10.215.184	50.116.1.184	TCP	74	60814 → 443 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1423477174 TSecr=0 WS=1024
171	2.752867038	50.116.1.184	10.10.215.184	TCP	74	443 → 60808 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1250 SACK_PERM TSval=553059302 TSecr=1423476924 WS=128
172	2.752925440	10.10.215.184	50.116.1.184	TCP	66	60808 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=1423477188 TSecr=553059302
173	2.753589651	10.10.215.184	50.116.1.184	TLSv1.2	1816	Client Hello (SNI=nmap.org)
174	2.757324201	50.116.1.184	10.10.215.184	TCP	74	443 → 60810 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1250 SACK_PERM TSval=553059310 TSecr=1423476932 WS=128
175	2.757373137	10.10.215.184	50.116.1.184	TCP	66	60810 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=1423477192 TSecr=553059310
176	2.758050229	10.10.215.184	50.116.1.184	TLSv1.2	1880	Client Hello (SNI=nmap.org)
177	3.071671786	50.116.1.184	10.10.215.184	TCP	74	443 → 60814 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1250 SACK_PERM TSval=553059551 TSecr=1423477174 WS=128
178	3.071690455	50.116.1.184	10.10.215.184	TCP	66	443 → 60808 [ACK] Seq=1 Ack=1239 Win=31872 Len=0 TSval=553059567 TSecr=1423477188
179	3.071692324	50.116.1.184	10.10.215.184	TCP	66	443 → 60810 [ACK] Seq=1 Ack=1239 Win=31872 Len=0 TSval=553059571 TSecr=1423477193
180	3.071690827	50.116.1.184	10.10.215.184	TCP	66	443 → 60808 [ACK] Seq=1 Ack=1751 Win=31872 Len=0 TSval=553059567 TSecr=1423477188
181	3.071692533	50.116.1.184	10.10.215.184	TCP	66	443 → 60810 [ACK] Seq=1 Ack=1815 Win=31872 Len=0 TSval=553059571 TSecr=1423477193
182	3.071690879	50.116.1.184	10.10.215.184	TLSv1.2	2542	Server Hello
183	3.071692585	50.116.1.184	10.10.215.184	TLSv1.2	2542	Server Hello
184	3.071690931	50.116.1.184	10.10.215.184	TLSv1.2	833	Certificate, Server Key Exchange, Server Hello Done
185	3.071692637	50.116.1.184	10.10.215.184	TLSv1.2	833	Certificate, Server Key Exchange, Server Hello Done
186	3.071738237	10.10.215.184	50.116.1.184	TCP	66	60814 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=1423477506 TSecr=553059551
187	3.071764452	10.10.215.184	50.116.1.184	TCP	66	60810 → 443 [ACK] Seq=1815 Ack=2477 Win=32768 Len=0 TSval=1423477506 TSecr=553059573
188	3.071785350	10.10.215.184	50.116.1.184	TCP	66	60808 → 443 [ACK] Seq=1751 Ack=2477 Win=32768 Len=0 TSval=1423477506 TSecr=553059568
189	3.071800489	10.10.215.184	50.116.1.184	TCP	66	60810 → 443 [ACK] Seq=1815 Ack=3244 Win=32768 Len=0 TSval=1423477506 TSecr=553059573
190	3.071807241	10.10.215.184	50.116.1.184	TCP	66	60808 → 443 [ACK] Seq=1751 Ack=3244 Win=32768 Len=0 TSval=1423477506 TSecr=553059568
191	3.072420166	10.10.215.184	50.116.1.184	TLSv1.2	1816	Client Hello (SNI=nmap.org)
192	3.076647632	10.10.215.184	50.116.1.184	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Finished
193	3.077141969	10.10.215.184	50.116.1.184	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Finished
194	3.378856552	50.116.1.184	10.10.215.184	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Finished
195	3.378858983	50.116.1.184	10.10.215.184	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Finished
196	3.378879044	50.116.1.184	10.10.215.184	TCP	66	443 → 60814 [ACK] Seq=1 Ack=1751 Win=31872 Len=0 TSval=553059885 TSecr=1423477507
197	3.378879337	50.116.1.184	10.10.215.184	TLSv1.2	2542	Server Hello
198	3.378879408	50.116.1.184	10.10.215.184	TLSv1.2	833	Certificate, Server Key Exchange, Server Hello Done
199	3.378929562	10.10.215.184	50.116.1.184	TCP	66	60814 → 443 [ACK] Seq=1751 Ack=2477 Win=32768 Len=0 TSval=1423477814 TSecr=553059887
200	3.378963873	10.10.215.184	50.116.1.184	TCP	66	60814 → 443 [ACK] Seq=1751 Ack=3244 Win=32768 Len=0 TSval=1423477814 TSecr=553059887
201	3.379842395	10.10.215.184	50.116.1.184	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Finished
202	3.380063334	10.10.215.184	50.116.1.184	HTTP	869	GET / HTTP/1.1
203	3.422021350	10.10.215.184	50.116.1.184	TCP	66	60810 → 443 [ACK] Seq=1941 Ack=3502 Win=32768 Len=0 TSval=1423477857 TSecr=553059894

We can see the connection establishment process of the browser with the website in the above figure.
First we can see the TCP 3-way handshake taking place with [SYN], [SYN,ACK], and [ACK] packets.
Then there are some packets for the connection establishment like client hello, Server hello, Certificate exchange and Session ticket generation.

In frame 202, we can see that first GET request is being sent to the server from our browser.

(1) Total time to load the webpage can be calculated by subtracting first GET request's timestamp from last http response's timestamp.

Since I got only 1 http response and there is only one GET request.
We can simply subtract the timestamp of Request from Response.
GET request Timestamp = 3.95089
Response Timestamp = 3.38006

Total time to load the webpage = 3.95089 - 3.38006 = **0.57083 sec**.

(2) Now, we have to find total connections to load the webpage.
For that, goto Statistics → Conversations and filter only TCP connections.

TCP · 1

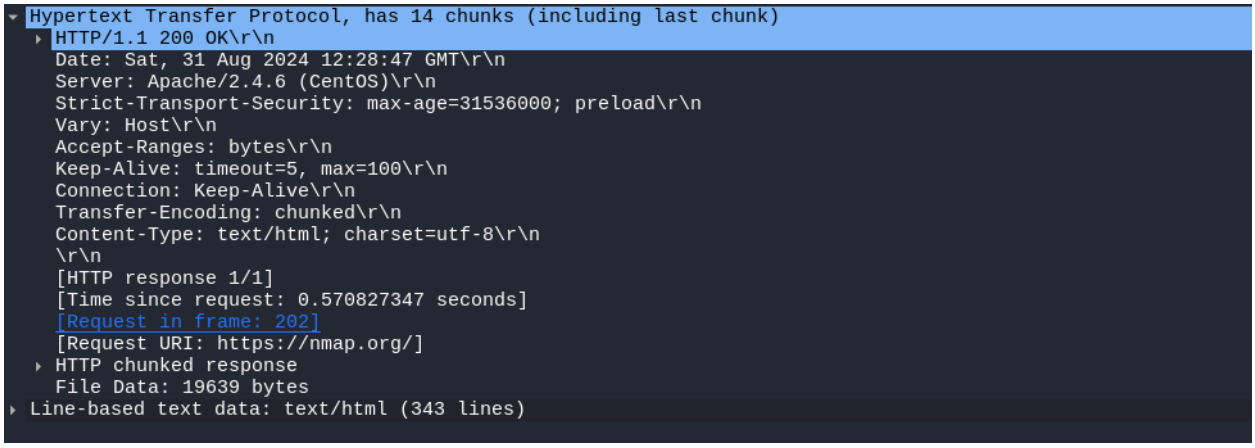
Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Total Packets	Percent Filtered	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.10.215.184	60808	50.116.1.184	443	2	3 kB	1	31	6.45%	1	869 bytes	1	2 kB	2.488900	6.2571	1111 bits/s	2536 bits/s

From above image, we can see that the webpage was loaded into only one TCP Connection.

(3) Now, we have to check whether the connection was persistent or inpersistent. For this Follow the TCP Stream of the above connection,
There we will see that this One connection was used for the multiple requests to the server from the browser.
Hence we can say that connection was **persistent**.
Also, the website is using **HTTP 1.1** which uses persistent connection by default until you have not modified it.

(4) HTTP protocol requests each single object with an individual GET request.
In my case, since there is only one HTTP GET request and one response for that we can say that only one object was transferred within the connection.The object type is text/HTML, which is nmap.org's home page.

(5) Since only one object (Home page) of the Website was received from the connection, we can say that it has taken the longest time to download.



```
▾ Hypertext Transfer Protocol, has 14 chunks (including last chunk)
  ▸ HTTP/1.1 200 OK\r\n
    Date: Sat, 31 Aug 2024 12:28:47 GMT\r\n
    Server: Apache/2.4.6 (CentOS)\r\n
    Strict-Transport-Security: max-age=31536000; preload\r\n
    Vary: Host\r\n
    Accept-Ranges: bytes\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Transfer-Encoding: chunked\r\n
    Content-Type: text/html; charset=utf-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.570827347 seconds]
    [Request in frame: 202]
    [Request URI: https://nmap.org/]
  ▸ HTTP chunked response
    File Data: 19639 bytes
▸ Line-based text data: text/html (343 lines)
```

In the above image we can see the response packet.
It is showing the TIme since First request as 0.570827347 which is the same as time to load the total web page since there was only one request and response needed to load this website.

[Solution-6]

Host : www.iitbhilai.ac.in

HTTP GET Request Header Fields

1. **Host**: iitbhilai.ac.in

   This header shows the name of the server.This is to indicate that exact website is being visited.

2. **Sec-Ch-Ua**: "Not/A)Brand";v="8", "Chromium";v="126"

   This header provides information about the user agent's branding, indicating compatibility with certain browsers or engines.

3. **Sec-Ch-Ua-Mobile**: ?0

   This header indicates the user's device.If the device is mobile, then shows ?1 else ?0.

4. **Sec-Ch-Ua-Platform**: Linux

   This header shows the operating system being used on the user's platform.

5. **Accept-Language**: en-US

   This header specifies the preferred language for the response content. It allows the server to deliver content in the preferred language if available.

6. **Upgrade-Insecure-Requests**: 1

   This Informs to server that the client prefers secure connections (HTTPS) and requests that insecure requests to be upgraded.

7. **User-Agent**: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36

   This provides information about the client software, including the browser name, version, and operating system. This can help the server tailor responses for compatibility.

8. **Accept**: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

   This header specifies the media types the client can handle. The server uses this to provide the most appropriate content type.

9. **Sec-Fetch-Site**: none

   This provides information about the context of the request, indicating whether it is a same-site, same-origin, or cross-site request. This is used for security purposes.

10. **Sec-Fetch-Mode**: navigate

    This field indicates the mode of the request(navigate cors or no-cors). It helps control how the request is handled by the browser, particularly in cross-origin requests.

11. **Sec-Fetch-User**: ?1

    This is a boolean that indicates whether the request is initiated by user interaction, such as a click or form submission.If it is not generated by User then the value will be **?0**.

12. **Sec-Fetch-Dest**: document

    This specifies the destination of the fetched content, indicating that the requested resource is meant to be a document, such as an HTML page.

13. **Accept-Encoding**: gzip, deflate, br

This indicates the content-encoding schemes that the client can decode. The server uses this to compress the response appropriately, improving transmission efficiency.

14. **Priority**: u=0, i

    This is a hint for HTTP/2 server prioritization, indicating the priority of the request, where u=0 indicates a default priority and i indicates a prioritized incremental load.

15. **Connection**: keep-alive

    This instructs the server to maintain the connection open for multiple requests rather than closing it after the current request/response cycle. This reduces latency for subsequent requests.

HTTP Response Header Fields

1. **Date**: Sat, 31 Aug 2024 09:17:04 GMT

   This field specifies the date and time at which the response was generated. This is used for synchronization purposes and also can be used for caching.

2. **Server**: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16

   This provides information about the server software handling the request. This can include web server software, OS, and any other relevant details.

3. **X-Powered-By**: PHP/5.4.16

This field indicates the server-side technology used to generate the response. It shows that PHP is being used in this case.

4. **Set-Cookie**: PHPSESSID=ebh3m9ua2bevcpqc6u77emhff1; path=/

   This shows that the Server sends a cookie to the client, which can be used to maintain a session or store user-specific data. **PHPSESSID** is commonly used to manage user sessions in PHP applications.

5. **Expires**: Thu, 19 Nov 1981 08:52:00 GMT

   This field specifies the date/time after which the response is considered stale. Often used in conjunction with caching mechanisms.

6. **Cache-Control**: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

   This indicates how the response is cached by browsers and intermediate caches. In this case, it instructs not to store or cache the response and requires validation for reuse.

7. **Pragma**: no-cache

   A legacy HTTP/1.0 header for backward compatibility, used to control caching behavior. It instructs the client to not cache the response.

8. **Keep-Alive**: timeout=5, max=100

   This specifies the parameters of a persistent connection. In this case, the connection will remain open for 5 seconds or 100 requests, whichever comes first.

9. **Connection**: Keep-Alive

   This field indicates that the connection should be kept open for multiple requests, reducing latency for further requests.

10. **Content-Type**: text/html; charset=UTF-8

    This field specifies the media type of the response body and character encoding. Here, it indicates that the response is HTML content encoded in UTF-8.

11. **Content-Length**: 36976

    This field indicates the size of the response body in bytes. It helps the client know how much data to expect and can assist in efficiently handling the response.

Host : www.nmap.org

Request

Pretty Raw Hex

```
1  GET / HTTP/1.1
2  Host: nmap.org
3  Sec-Ch-Ua: "Not/A)Brand";v="8", "Chromium";v="126"
4  Sec-Ch-Ua-Mobile: ?0
5  Sec-Ch-Ua-Platform: "Linux"
6  Accept-Language: en-US
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.
   7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=0, i
16 Connection: keep-alive
17
18
```

We can see that all the fields in the request are same as for previous host.

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Date: Sat, 31 Aug 2024 13:30:46 GMT
3  Server: Apache/2.4.6 (CentOS)
4  Strict-Transport-Security: max-age=31536000; preload
5  Vary: Host
6  Accept-Ranges: bytes
7  Keep-Alive: timeout=5, max=100
8  Connection: Keep-Alive
9  Content-Type: text/html; charset=utf-8
10 Content-Length: 19639
```

Here, there are 3 new header fields.

Strict-Transport-Security: max-age=31536000; preload

This field enforces the use of HTTPS only.max-age is 31536000 seconds which is around 1 year.

Vary: Host

This field indicate that response may vary according to host.

Accept-Ranges: bytes

This field specifies that the server supports partial requests for the resource.This allows clients to request specific byte ranges which is useful for resuming interrupted downloads.

Part-2 DNS

[Solution-1]

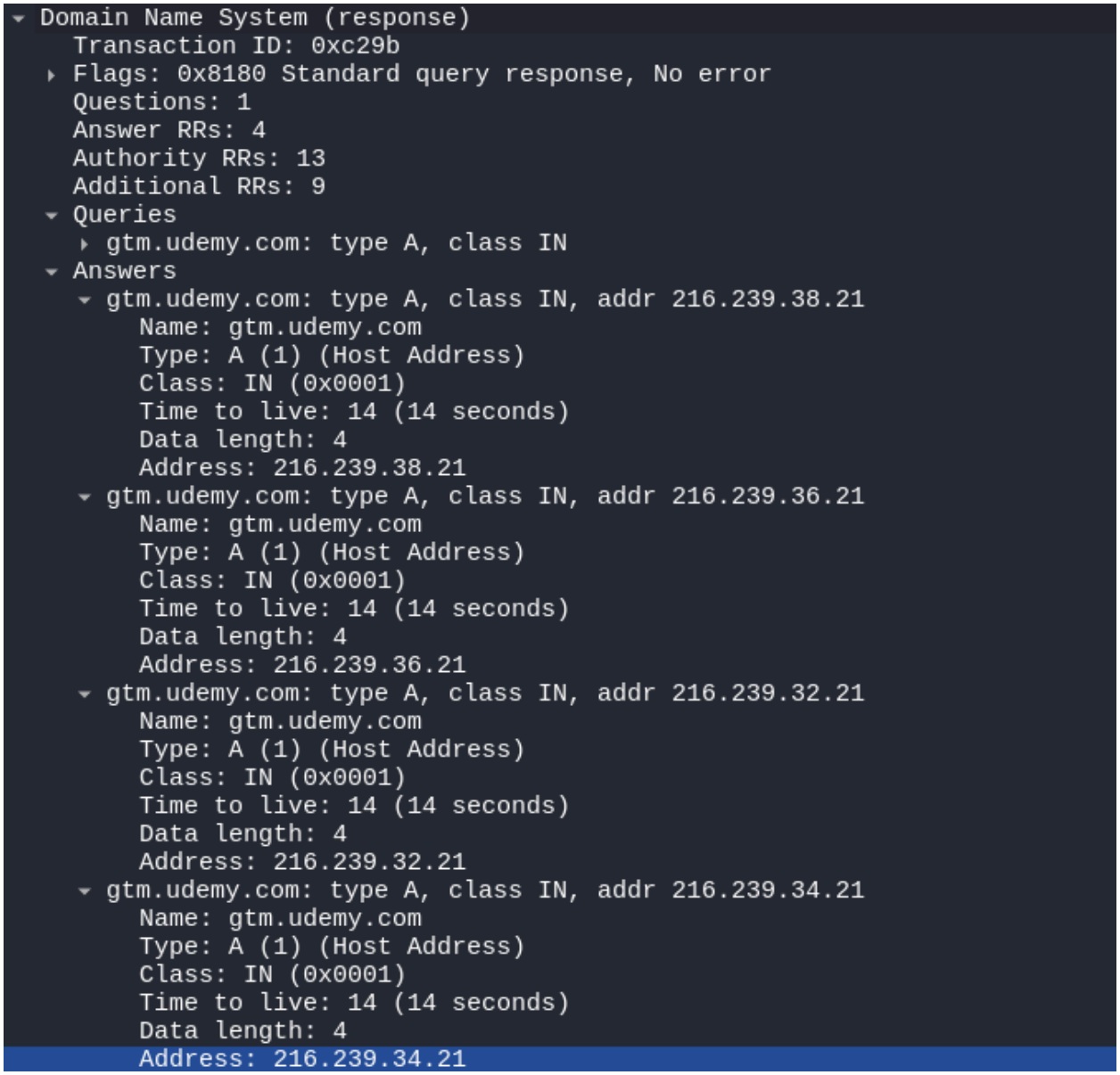
(a) I visited **udemy.com** along with **iitbhilai.ac.in** to note dns queries**.**
There are total of 78 packets including DNS queries and Responses.
We can see that for most of the hosts we can find two types of IP addresses,

(1) Type A : IPv4 Address
(2) Type AAAA : IPv6 Address

I mentioned only IPv4 addresses in the table.

Domain Name	IP Address
iitbhilai.ac.in	192.168.10.115
cse.google.com	172.217.174.238
safebrowsing.googleapis.com	142.250.192.42
d23hs77t6unaoa.cloudfront.net	108.159.80.111
sdk.iad-03.braze.com.cdn.cloudflare.net	104.18.36.46
gtm.udemy.com	216.239.36.21

(b)Yes, We can find out IP address of the server by exploring DNS packets.
In the DNS packet, if the packet is for the response of the DNS query then you can find out the IP address of the DNS in the answers subsection of the response packet.



gtm.udemy.com: type A, class IN, addr 216.239.38.21

Above line is the DNS answer for A type DNS query to udemy.com

This is how we can find out IP addresses for domains from captured packets.

We are getting more than one IP for the same domain which indicates that udemy.com has divided services in subdomains for load balancing.

[Solution-2]

dig @a.root-servers.net www.iitbhilai.ac.in
Above command returns name servers for the **.in** domain.

```
┌──(prajapati㉿kali)-[~]
└─$ dig @a.root-servers.net www.iitbhilai.ac.in +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @a.root-servers.net www.iitbhilai.ac.in +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55443
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; AUTHORITY SECTION:
in.                     172800  IN      NS      ns2.registry.in.
in.                     172800  IN      NS      ns5.registry.in.
in.                     172800  IN      NS      ns4.registry.in.
in.                     172800  IN      NS      ns1.registry.in.
in.                     172800  IN      NS      ns6.registry.in.
in.                     172800  IN      NS      ns3.registry.in.

;; ADDITIONAL SECTION:
ns2.registry.in.        172800  IN      A       37.209.194.12
ns2.registry.in.        172800  IN      AAAA    2001:dcd:2::12
ns5.registry.in.        172800  IN      A       156.154.100.20
ns5.registry.in.        172800  IN      AAAA    2001:502:2eda::20
ns4.registry.in.        172800  IN      A       37.209.198.12
ns4.registry.in.        172800  IN      AAAA    2001:dcd:4::12
ns1.registry.in.        172800  IN      A       37.209.192.12
ns1.registry.in.        172800  IN      AAAA    2001:dcd:1::12
ns6.registry.in.        172800  IN      A       156.154.101.20
ns6.registry.in.        172800  IN      AAAA    2001:502:ad09::20
ns3.registry.in.        172800  IN      A       37.209.196.12
ns3.registry.in.        172800  IN      AAAA    2001:dcd:3::12

;; Query time: 156 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net) (UDP)
;; WHEN: Sat Aug 31 15:36:35 IST 2024
;; MSG SIZE  rcvd: 429
```

dig @<Authority_Section> www.iitbhilai.ac.in

Replace one of the names from Authority Section besides @ and run the command again. This will give related dns servers with the website.

```
┌──(prajapati㉿kali)-[~]
└─$ dig @ns1.registry.in www.iitbhilai.ac.in +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @ns1.registry.in www.iitbhilai.ac.in +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37930
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; AUTHORITY SECTION:
iitbhilai.ac.in.        3600    IN      NS      dns1.iitbhilai.ac.in.
iitbhilai.ac.in.        3600    IN      NS      dns2.iitbhilai.ac.in.

;; ADDITIONAL SECTION:
dns2.iitbhilai.ac.in.   3600    IN      A       103.147.138.111
dns1.iitbhilai.ac.in.   3600    IN      A       103.147.138.110

;; Query time: 32 msec
;; SERVER: 37.209.192.12#53(ns1.registry.in) (UDP)
;; WHEN: Sat Aug 31 15:37:18 IST 2024
;; MSG SIZE  rcvd: 118
```

Again doing the same as the previous step has given the IP of iitbhilai.ac.in.

```
┌──(prajapati㉿kali)-[~]
└─$ dig @dns2.iitbhilai.ac.in. www.iitbhilai.ac.in +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @dns2.iitbhilai.ac.in. www.iitbhilai.ac.in +norecurse
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39984
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; ANSWER SECTION:
www.iitbhilai.ac.in.    8641    IN      A       192.168.10.115

;; AUTHORITY SECTION:
iitbhilai.ac.in.        8641    IN      NS      dns2.iitbhilai.ac.in.

;; ADDITIONAL SECTION:
dns2.iitbhilai.ac.in.   8641    IN      A       192.168.10.72

;; Query time: 4 msec
;; SERVER: 192.168.10.72#53(dns2.iitbhilai.ac.in.) (UDP)
;; WHEN: Sat Aug 31 15:38:01 IST 2024
;; MSG SIZE  rcvd: 99
```

Answer: www.iitbhilai.ac.in → 192.168.10.115

Below, I repeated the same method for 2 other websites.

(i) kali.org

(ii) website.com

```
┌──(prajapati㉿kali)-[~]
└─$ dig @a.root-servers.net www.kali.org +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @a.root-servers.net www.kali.org +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17337
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.kali.org.                  IN      A

;; AUTHORITY SECTION:
org.                    172800  IN      NS      a2.org.afilias-nst.info.
org.                    172800  IN      NS      b2.org.afilias-nst.org.
org.                    172800  IN      NS      d0.org.afilias-nst.org.
org.                    172800  IN      NS      a0.org.afilias-nst.info.
org.                    172800  IN      NS      b0.org.afilias-nst.org.
org.                    172800  IN      NS      c0.org.afilias-nst.info.

;; ADDITIONAL SECTION:
a2.org.afilias-nst.info. 172800 IN      A       199.249.112.1
a2.org.afilias-nst.info. 172800 IN      AAAA    2001:500:40::1
b2.org.afilias-nst.org. 172800  IN      A       199.249.120.1
b2.org.afilias-nst.org. 172800  IN      AAAA    2001:500:48::1
d0.org.afilias-nst.org. 172800  IN      A       199.19.57.1
d0.org.afilias-nst.org. 172800  IN      AAAA    2001:500:f::1
a0.org.afilias-nst.info. 172800 IN      A       199.19.56.1
a0.org.afilias-nst.info. 172800 IN      AAAA    2001:500:e::1
b0.org.afilias-nst.org. 172800  IN      A       199.19.54.1
b0.org.afilias-nst.org. 172800  IN      AAAA    2001:500:c::1
c0.org.afilias-nst.info. 172800 IN      A       199.19.53.1
c0.org.afilias-nst.info. 172800 IN      AAAA    2001:500:b::1

;; Query time: 172 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net) (UDP)
;; WHEN: Sat Aug 31 15:39:14 IST 2024
;; MSG SIZE  rcvd: 443
```

```
┌──(prajapati㉿kali)-[~]
└─$ dig @a2.org.afilias-nst.info. www.kali.org +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @a2.org.afilias-nst.info. www.kali.org +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55179
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.kali.org.                  IN      A

;; AUTHORITY SECTION:
kali.org.               3600    IN      NS      nina.ns.cloudflare.com.
kali.org.               3600    IN      NS      nash.ns.cloudflare.com.

;; Query time: 240 msec
;; SERVER: 199.249.112.1#53(a2.org.afilias-nst.info.) (UDP)
;; WHEN: Sat Aug 31 15:40:38 IST 2024
;; MSG SIZE  rcvd: 96
```

```
┌──(prajapati㉿kali)-[~]
└─$ dig @nina.ns.cloudflare.com. www.kali.org +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @nina.ns.cloudflare.com. www.kali.org +norecurse
; (6 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5991
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.kali.org.                  IN      A

;; ANSWER SECTION:
www.kali.org.           300     IN      A       104.18.5.159
www.kali.org.           300     IN      A       104.18.4.159

;; Query time: 104 msec
;; SERVER: 173.245.58.136#53(nina.ns.cloudflare.com.) (UDP)
;; WHEN: Sat Aug 31 15:41:13 IST 2024
;; MSG SIZE  rcvd: 73
```

Answer: www.kali.org → 104.18.5.159

```
┌──(prajapati㉿kali)-[~]
└─$ dig @a.root-servers.net www.website.com +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @a.root-servers.net www.website.com +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42965
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.website.com.               IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.

;; ADDITIONAL SECTION:
l.gtld-servers.net.     172800  IN      A       192.41.162.30
l.gtld-servers.net.     172800  IN      AAAA    2001:500:d937::30
j.gtld-servers.net.     172800  IN      A       192.48.79.30
j.gtld-servers.net.     172800  IN      AAAA    2001:502:7094::30
h.gtld-servers.net.     172800  IN      A       192.54.112.30
h.gtld-servers.net.     172800  IN      AAAA    2001:502:8cc::30
d.gtld-servers.net.     172800  IN      A       192.31.80.30
d.gtld-servers.net.     172800  IN      AAAA    2001:500:856e::30
b.gtld-servers.net.     172800  IN      A       192.33.14.30
b.gtld-servers.net.     172800  IN      AAAA    2001:503:231d::2:30
f.gtld-servers.net.     172800  IN      A       192.35.51.30
f.gtld-servers.net.     172800  IN      AAAA    2001:503:d414::30
k.gtld-servers.net.     172800  IN      A       192.52.178.30
k.gtld-servers.net.     172800  IN      AAAA    2001:503:d2d::30
m.gtld-servers.net.     172800  IN      A       192.55.83.30
m.gtld-servers.net.     172800  IN      AAAA    2001:501:b1f9::30
i.gtld-servers.net.     172800  IN      A       192.43.172.30
i.gtld-servers.net.     172800  IN      AAAA    2001:503:39c1::30
g.gtld-servers.net.     172800  IN      A       192.42.93.30
g.gtld-servers.net.     172800  IN      AAAA    2001:503:eea3::30
a.gtld-servers.net.     172800  IN      A       192.5.6.30
a.gtld-servers.net.     172800  IN      AAAA    2001:503:a83e::2:30
c.gtld-servers.net.     172800  IN      A       192.26.92.30
c.gtld-servers.net.     172800  IN      AAAA    2001:503:83eb::30
e.gtld-servers.net.     172800  IN      A       192.12.94.30
e.gtld-servers.net.     172800  IN      AAAA    2001:502:1ca1::30

;; Query time: 200 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net) (UDP)
;; WHEN: Sat Aug 31 15:53:42 IST 2024
;; MSG SIZE  rcvd: 840
```

```
┌──(prajapati㉿kali)-[~]
└─$ dig @l.gtld-servers.net. www.website.com +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @l.gtld-servers.net. www.website.com +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49793
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.website.com.               IN      A

;; AUTHORITY SECTION:
website.com.            172800  IN      NS      ns1.website.com.
website.com.            172800  IN      NS      ns2.website.com.

;; ADDITIONAL SECTION:
ns1.website.com.        172800  IN      A       162.159.8.245
ns2.website.com.        172800  IN      A       162.159.9.164

;; Query time: 268 msec
;; SERVER: 192.41.162.30#53(l.gtld-servers.net.) (UDP)
;; WHEN: Sat Aug 31 15:54:12 IST 2024
;; MSG SIZE  rcvd: 112
```

```
┌──(prajapati㉿kali)-[~]
└─$ dig @ns1.website.com. www.website.com +norecurse

; <<>> DiG 9.20.1-1-Debian <<>> @ns1.website.com. www.website.com +norecurse
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32598
;; flags: qr aa; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.website.com.               IN      A

;; ANSWER SECTION:
www.website.com.        300     IN      A       104.22.67.195
www.website.com.        300     IN      A       172.67.27.106
www.website.com.        300     IN      A       104.22.66.195

;; Query time: 72 msec
;; SERVER: 162.159.8.245#53(ns1.website.com.) (UDP)
;; WHEN: Sat Aug 31 15:54:34 IST 2024
;; MSG SIZE  rcvd: 92
```

Answer : www.website.com → 104.22.67.195
www.website.com → 104.22.66.195
www.website.com → 172.67.27.106