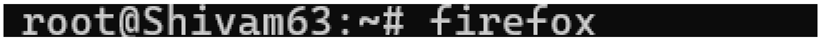
BHARATIYA VIDYA BHAVAN’S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan’s Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

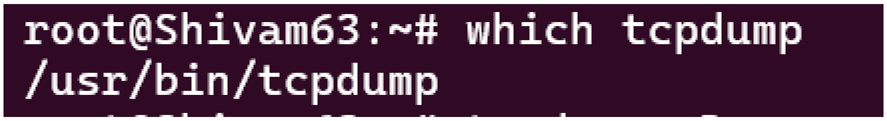
Name	**Shivam Santosh Kadam**
UID no.	**2023300099**
Experiment No.	**3(Home)**

AIM:	To analyze HTTP GET/response interactions, HTTP conditional GET, and HTTP authentication using tcpdump / Tshark.

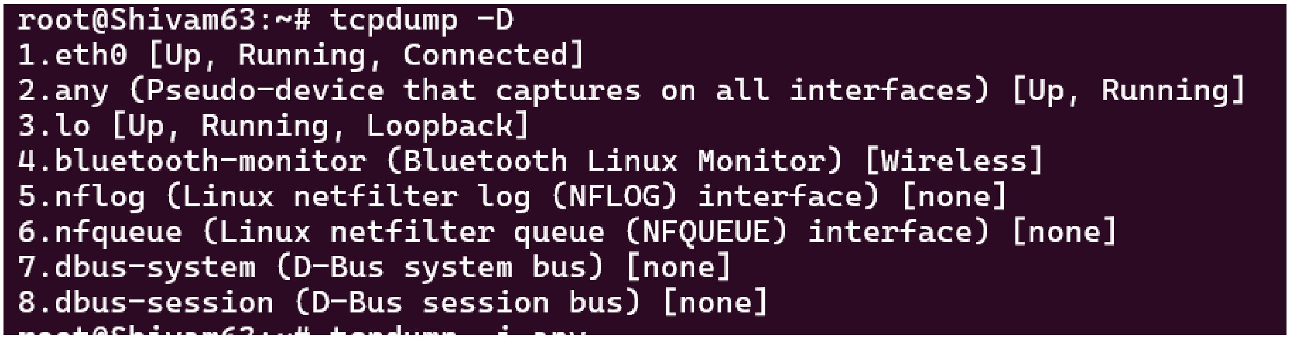
SOLUTION:



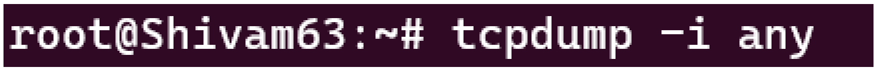
```
root@Shivam63:~# firefox
```



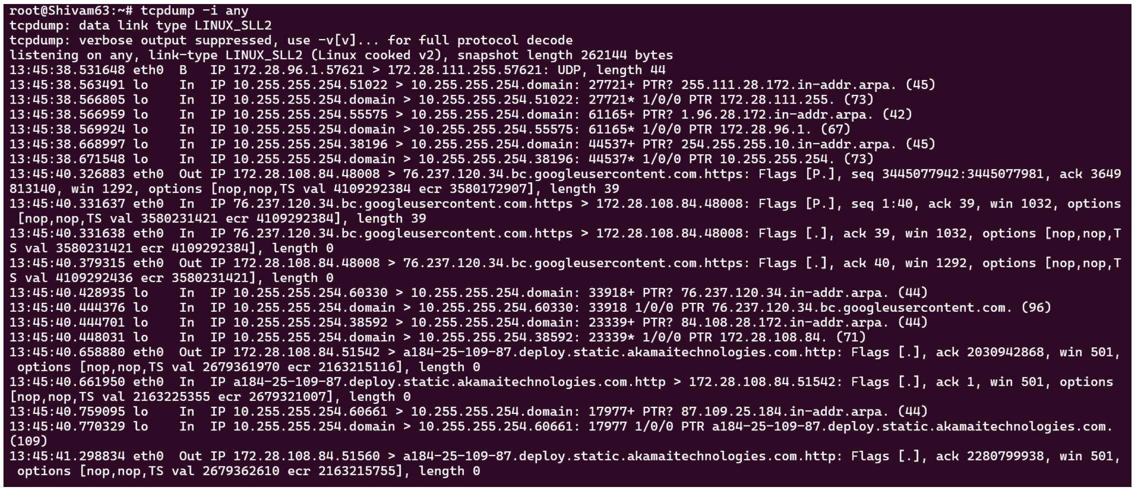
```
root@Shivam63:~# which tcpdump
/usr/bin/tcpdump
```



```
root@Shivam63:~# tcpdump -D
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
```



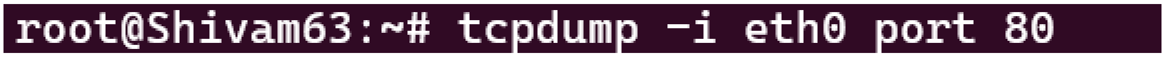
```
root@Shivam63:~# tcpdump -i any
```



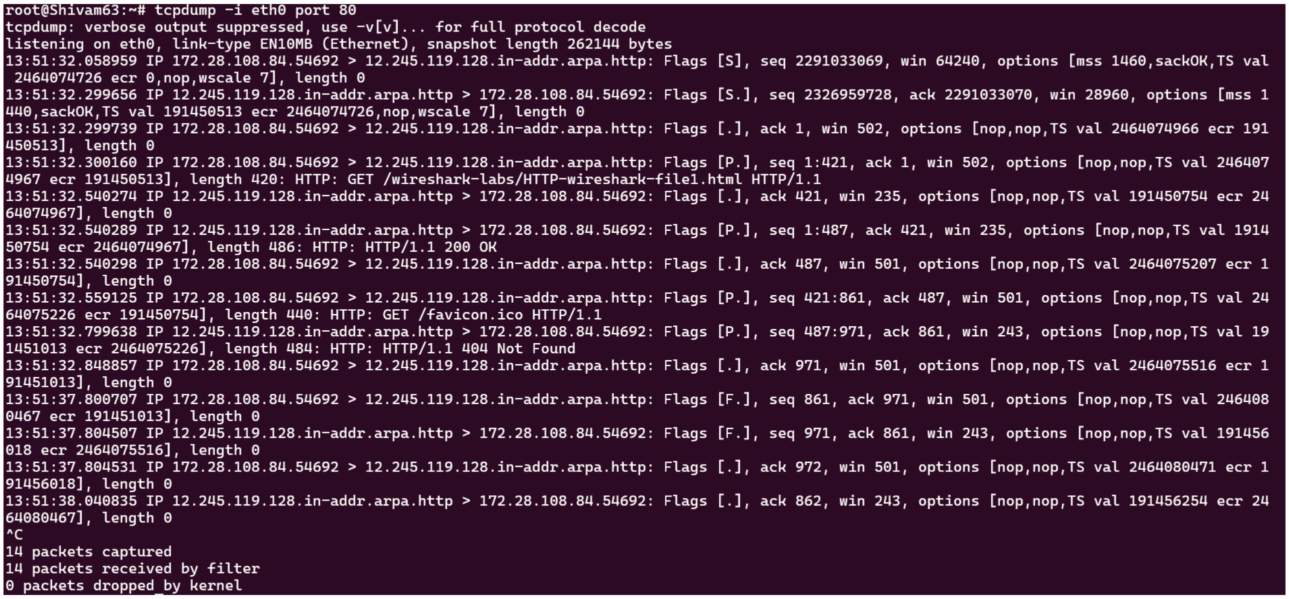
```
root@Shivam63:~# tcpdump -i any
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
13:45:38.531648 eth0  B   IP 172.28.96.1.57621 > 172.28.111.255.57621: UDP, length 44
13:45:38.563491 lo    In  IP 10.255.255.254.51022 > 10.255.255.254.domain: 27721+ PTR? 255.111.28.172.in-addr.arpa. (45)
13:45:38.566805 lo    In  IP 10.255.255.254.domain > 10.255.255.254.51022: 27721* 1/0/0 PTR 172.28.111.255. (73)
13:45:38.566959 lo    In  IP 10.255.255.254.55575 > 10.255.255.254.domain: 61165+ PTR? 1.96.28.172.in-addr.arpa. (42)
13:45:38.569924 lo    In  IP 10.255.255.254.domain > 10.255.255.254.55575: 61165* 1/0/0 PTR 172.28.96.1. (67)
13:45:38.668997 lo    In  IP 10.255.255.254.38196 > 10.255.255.254.domain: 44537+ PTR? 254.255.255.10.in-addr.arpa. (45)
13:45:38.671548 lo    In  IP 10.255.255.254.domain > 10.255.255.254.38196: 44537* 1/0/0 PTR 10.255.255.254. (73)
13:45:40.326883 eth0  Out IP 172.28.108.84.48008 > 76.237.120.34.bc.googleusercontent.com.https: Flags [P.], seq 3445077942:3445077981, ack 3649
813140, win 1292, options [nop,nop,TS val 4109292384 ecr 3580172907], length 39
13:45:40.331637 eth0  In  IP 76.237.120.34.bc.googleusercontent.com.https > 172.28.108.84.48008: Flags [P.], seq 1:40, ack 39, win 1032, options
 [nop,nop,TS val 3580231421 ecr 4109292384], length 39
13:45:40.331638 eth0  In  IP 76.237.120.34.bc.googleusercontent.com.https > 172.28.108.84.48008: Flags [.], ack 39, win 1032, options [nop,nop,T
S val 3580231421 ecr 4109292384], length 0
13:45:40.379315 eth0  Out IP 172.28.108.84.48008 > 76.237.120.34.bc.googleusercontent.com.https: Flags [.], ack 40, win 1292, options [nop,nop,T
S val 4109292436 ecr 3580231421], length 0
13:45:40.428935 lo    In  IP 10.255.255.254.60330 > 10.255.255.254.domain: 33918+ PTR? 76.237.120.34.in-addr.arpa. (44)
13:45:40.444376 lo    In  IP 10.255.255.254.domain > 10.255.255.254.60330: 33918 1/0/0 PTR 76.237.120.34.bc.googleusercontent.com. (96)
13:45:40.444701 lo    In  IP 10.255.255.254.38592 > 10.255.255.254.domain: 23339+ PTR? 84.108.28.172.in-addr.arpa. (44)
13:45:40.448031 lo    In  IP 10.255.255.254.domain > 10.255.255.254.38592: 23339* 1/0/0 PTR 172.28.108.84. (71)
13:45:40.658880 eth0  Out IP 172.28.108.84.51542 > a184-25-109-87.deploy.static.akamaitechnologies.com.http: Flags [.], ack 2030942868, win 501,
 options [nop,nop,TS val 2679361970 ecr 2163215116], length 0
13:45:40.661950 eth0  In  IP a184-25-109-87.deploy.static.akamaitechnologies.com.http > 172.28.108.84.51542: Flags [.], ack 1, win 501, options
[nop,nop,TS val 2163225355 ecr 2679321007], length 0
13:45:40.759095 lo    In  IP 10.255.255.254.60661 > 10.255.255.254.domain: 17977+ PTR? 87.109.25.184.in-addr.arpa. (44)
13:45:40.770329 lo    In  IP 10.255.255.254.domain > 10.255.255.254.60661: 17977 1/0/0 PTR a184-25-109-87.deploy.static.akamaitechnologies.com.
(109)
13:45:41.298834 eth0  Out IP 172.28.108.84.51560 > a184-25-109-87.deploy.static.akamaitechnologies.com.http: Flags [.], ack 2280799938, win 501,
 options [nop,nop,TS val 2679362610 ecr 2163215755], length 0
```

Task A: Basic HTTP GET/response interaction

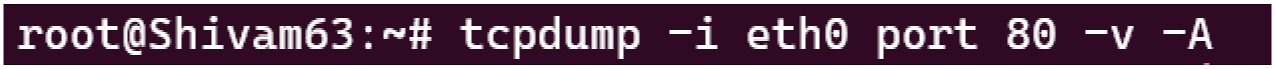
Filter the traffic by using port 80 (the default HTTP port)



```
root@Shivam63:~# tcpdump -i eth0 port 80
```



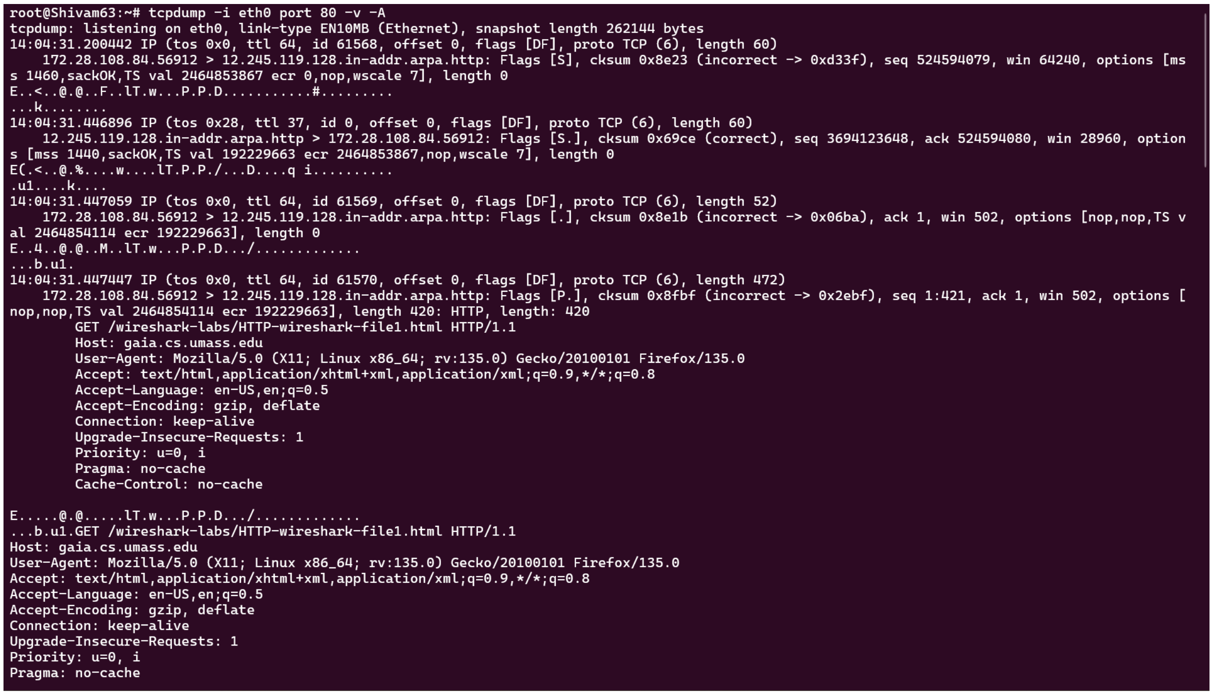
```
root@Shivam63:~# tcpdump -i eth0 port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:51:32.058959 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [S], seq 2291033069, win 64240, options [mss 1460,sackOK,TS val
 2464074726 ecr 0,nop,wscale 7], length 0
13:51:32.299656 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [S.], seq 2326959728, ack 2291033070, win 28960, options [mss 1
440,sackOK,TS val 191450513 ecr 2464074726,nop,wscale 7], length 0
13:51:32.299739 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 2464074966 ecr 191
450513], length 0
13:51:32.300160 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [P.], seq 1:421, ack 1, win 502, options [nop,nop,TS val 246407
4967 ecr 191450513], length 420: HTTP: GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
13:51:32.540274 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [.], ack 421, win 235, options [nop,nop,TS val 191450754 ecr 24
64074967], length 0
13:51:32.540289 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [P.], seq 1:487, ack 421, win 235, options [nop,nop,TS val 1914
50754 ecr 2464074967], length 486: HTTP: HTTP/1.1 200 OK
13:51:32.540298 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [.], ack 487, win 501, options [nop,nop,TS val 2464075207 ecr 1
91450754], length 0
13:51:32.559125 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [P.], seq 421:861, ack 487, win 501, options [nop,nop,TS val 24
64075226 ecr 191450754], length 440: HTTP: GET /favicon.ico HTTP/1.1
13:51:32.799638 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [P.], seq 487:971, ack 861, win 243, options [nop,nop,TS val 19
1451013 ecr 2464075226], length 484: HTTP: HTTP/1.1 404 Not Found
13:51:32.848857 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [.], ack 971, win 501, options [nop,nop,TS val 2464075516 ecr 1
91451013], length 0
13:51:37.800707 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [F.], seq 861, ack 971, win 501, options [nop,nop,TS val 246408
0467 ecr 191451013], length 0
13:51:37.804507 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [F.], seq 971, ack 861, win 243, options [nop,nop,TS val 191456
018 ecr 2464075516], length 0
13:51:37.804531 IP 172.28.108.84.54692 > 12.245.119.128.in-addr.arpa.http: Flags [.], ack 972, win 501, options [nop,nop,TS val 2464080471 ecr 1
91456018], length 0
13:51:38.040835 IP 12.245.119.128.in-addr.arpa.http > 172.28.108.84.54692: Flags [.], ack 862, win 243, options [nop,nop,TS val 191456254 ecr 24
64080467], length 0
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
```



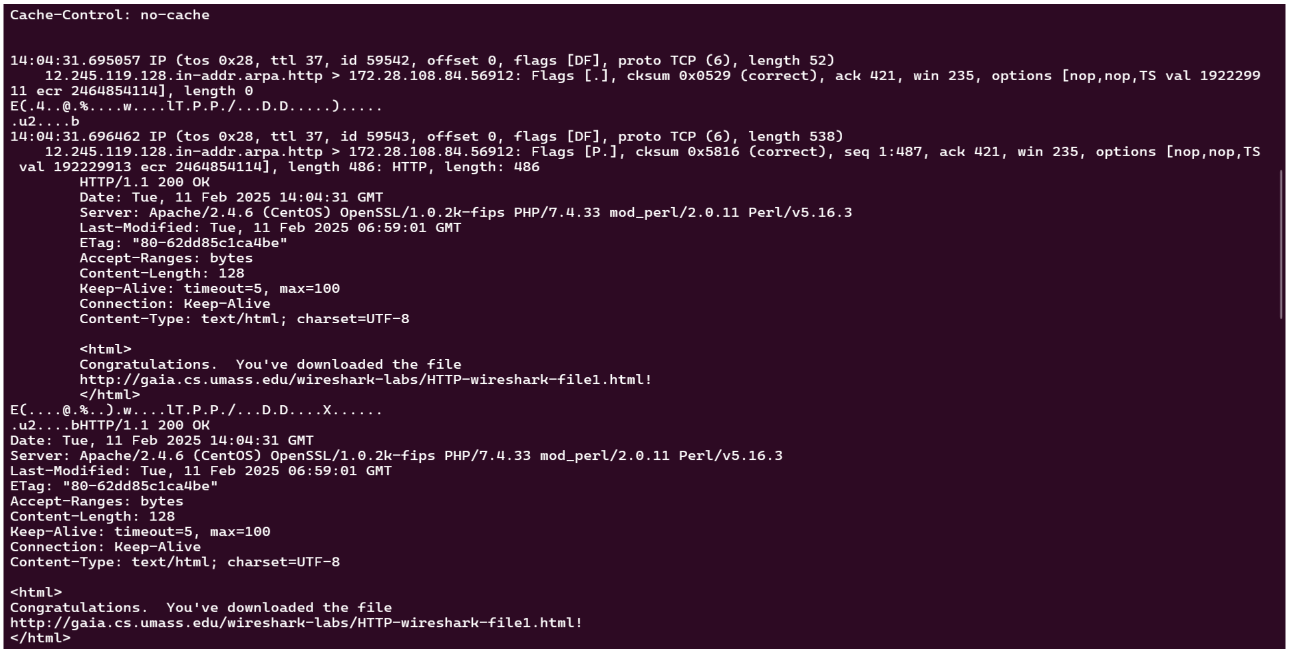
```
root@Shivam63:~# tcpdump -i eth0 port 80 -v -A
```

`-v`: This stands for "verbose" mode. It provides more detailed output than the standard `tcpdump` output. In verbose mode, `tcpdump` includes additional packet information such as timestamps, TTL (Time to Live), and more detailed IP address information.

`-A`: This option tells `tcpdump` to display the packet data in ASCII format. This is useful when you're inspecting HTTP traffic, as it shows the contents of the packets (like HTTP headers and body) in a human-readable form.



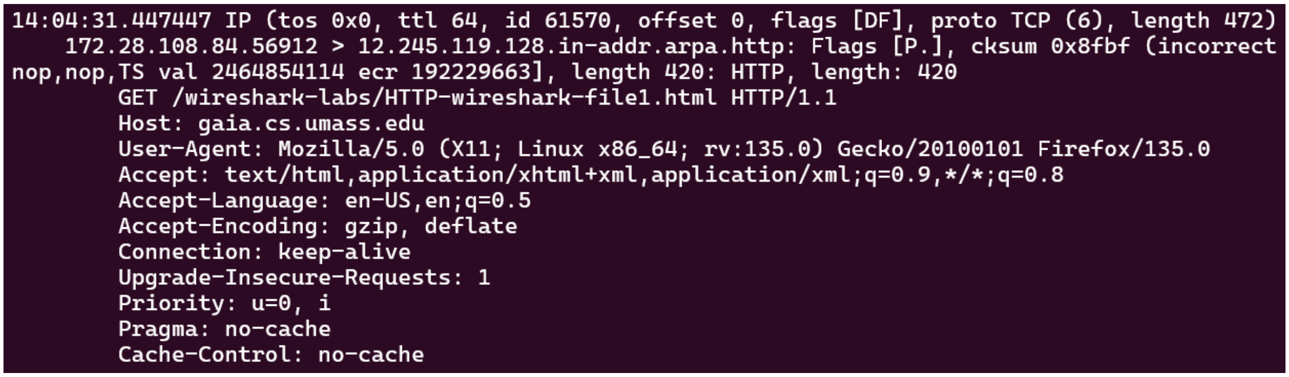
```
root@Shivam63:~# tcpdump -i eth0 port 80 -v -A
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:04:31.200442 IP (tos 0x0, ttl 64, id 61568, offset 0, flags [DF], proto TCP (6), length 60)
    172.28.108.84.56912 > 12.245.119.128.in-addr.arpa.http: Flags [S], cksum 0x8e23 (incorrect -> 0xd33f), seq 524594079, win 64240, options [ms
s 1460,sackOK,TS val 2464853867 ecr 0,nop,wscale 7], length 0
E..<..@.@..F..lT.w...P.P.D..........#.........
...k........
14:04:31.446896 IP (tos 0x28, ttl 37, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [S.], cksum 0x69ce (correct), seq 3694123648, ack 524594080, win 28960, option
s [mss 1440,sackOK,TS val 192229663 ecr 2464853867,nop,wscale 7], length 0
E(.<..@.%....w....lT.P.P./...D....q i.........
.u1....k....
14:04:31.447059 IP (tos 0x0, ttl 64, id 61569, offset 0, flags [DF], proto TCP (6), length 52)
    172.28.108.84.56912 > 12.245.119.128.in-addr.arpa.http: Flags [.], cksum 0x8e1b (incorrect -> 0x06ba), ack 1, win 502, options [nop,nop,TS v
al 2464854114 ecr 192229663], length 0
E..4..@.@..M..lT.w...P.P.D.../............
...b.u1.
14:04:31.447447 IP (tos 0x0, ttl 64, id 61570, offset 0, flags [DF], proto TCP (6), length 472)
    172.28.108.84.56912 > 12.245.119.128.in-addr.arpa.http: Flags [P.], cksum 0x8fbf (incorrect -> 0x2ebf), seq 1:421, ack 1, win 502, options [
nop,nop,TS val 2464854114 ecr 192229663], length 420: HTTP, length: 420
        GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
        Host: gaia.cs.umass.edu
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Connection: keep-alive
        Upgrade-Insecure-Requests: 1
        Priority: u=0, i
        Pragma: no-cache
        Cache-Control: no-cache

E.....@.@.....lT.w...P.P.D.../............
...b.u1.GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
```



```
Cache-Control: no-cache


14:04:31.695057 IP (tos 0x28, ttl 37, id 59542, offset 0, flags [DF], proto TCP (6), length 52)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [.], cksum 0x0529 (correct), ack 421, win 235, options [nop,nop,TS val 1922299
11 ecr 2464854114], length 0
E(.4..@.%....w....lT.P.P./...D.D.....).....
.u2....b
14:04:31.696462 IP (tos 0x28, ttl 37, id 59543, offset 0, flags [DF], proto TCP (6), length 538)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [P.], cksum 0x5816 (correct), seq 1:487, ack 421, win 235, options [nop,nop,TS
 val 192229913 ecr 2464854114], length 486: HTTP, length: 486
        HTTP/1.1 200 OK
        Date: Tue, 11 Feb 2025 14:04:31 GMT
        Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
        Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
        ETag: "80-62dd85c1ca4be"
        Accept-Ranges: bytes
        Content-Length: 128
        Keep-Alive: timeout=5, max=100
        Connection: Keep-Alive
        Content-Type: text/html; charset=UTF-8

        <html>
        Congratulations.  You've downloaded the file
        http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html!
        </html>
E(....@.%..).w....lT.P.P./...D.D....X......
.u2....bHTTP/1.1 200 OK
Date: Tue, 11 Feb 2025 14:04:31 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
ETag: "80-62dd85c1ca4be"
Accept-Ranges: bytes
Content-Length: 128
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html>
Congratulations.  You've downloaded the file
http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html!
</html>
```

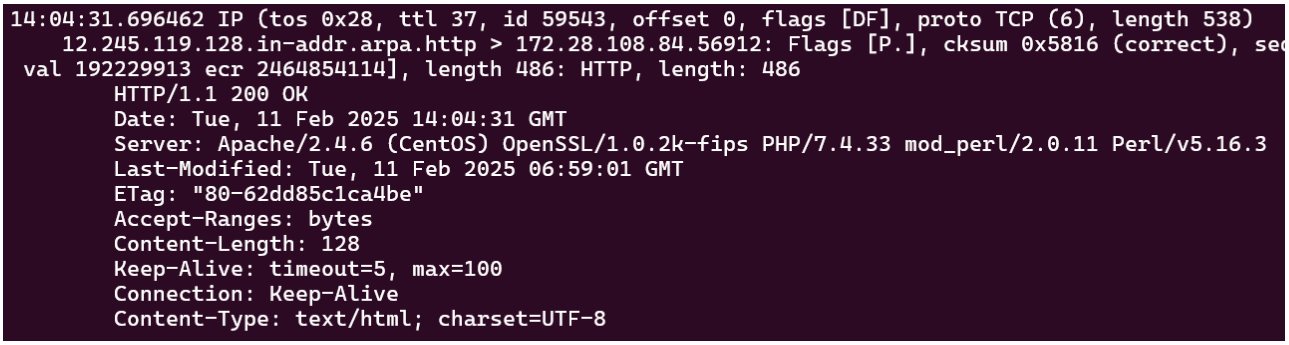
1. **Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?**
Answer: My browser is using **HTTP/1.1**, as indicated by the "GET" request line:GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1. This shows that the browser is requesting the resource using HTTP/1.1.

The server is also using **HTTP/1.1**, as seen in the response: HTTP/1.1 200K . This indicates that the server's response is also using HTTP/1.1.



```
14:04:31.447447 IP (tos 0x0, ttl 64, id 61570, offset 0, flags [DF], proto TCP (6), length 472)
    172.28.108.84.56912 > 12.245.119.128.in-addr.arpa.http: Flags [P.], cksum 0x8fbf (incorrect
nop,nop,TS val 2464854114 ecr 192229663], length 420: HTTP, length: 420
        GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
        Host: gaia.cs.umass.edu
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Connection: keep-alive
        Upgrade-Insecure-Requests: 1
        Priority: u=0, i
        Pragma: no-cache
        Cache-Control: no-cache
```

Clients version



```
14:04:31.696462 IP (tos 0x28, ttl 37, id 59543, offset 0, flags [DF], proto TCP (6), length 538)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [P.], cksum 0x5816 (correct), se
 val 192229913 ecr 2464854114], length 486: HTTP, length: 486
        HTTP/1.1 200 OK
        Date: Tue, 11 Feb 2025 14:04:31 GMT
        Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
        Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
        ETag: "80-62dd85c1ca4be"
        Accept-Ranges: bytes
        Content-Length: 128
        Keep-Alive: timeout=5, max=100
        Connection: Keep-Alive
        Content-Type: text/html; charset=UTF-8
```

Servers version

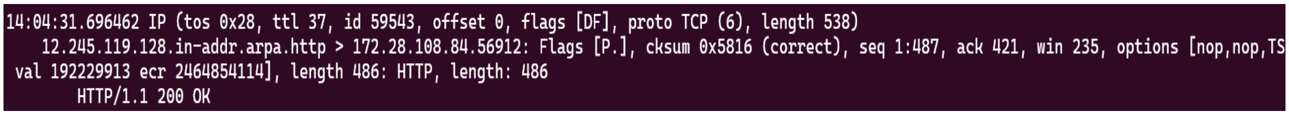
2. **What languages (if any) does your browser indicate that it can accept to the server? In the captured session, what other information (if any) does the browser provide the server with regarding the user/browser?**
Answer: The browser indicates that it can accept two languages from the server: en-US (English, United States) as the preferred language with full priority, and en (general English) as an acceptable alternative, but with a lower preference (q=0.5), as specified by the Accept-Language header: Accept-Language: en-US, en;q=0.5.

The browser also provides the following additional information to the server:

- **User-Agent:** This tells the server that the client is using the Firefox browser (version 135.0) on a 64-bit Linux machine.
- **Host:** This indicates the domain name of the server to which the request is being sent.`H`
- **Accept:** The client specifies the types of content it can accept. The primary content type is `text/html`, followed by `application/xhtml+xml`, and `application/xml` (with a slight preference for the latter types). If none of those types are available, any other content type (`*/*`) is acceptable with lower priority (`q=0.8`).
- **Accept-Encoding:** This tells the server that the client supports `gzip` and `deflate` compression methods for the response.
- **Connection:** `keep-alive`This indicates that the client wants to keep the connection open for future requests, rather than closing it after the current transaction.
- **Upgrade-Insecure-Requests:** `1` This means that the client prefers secure (HTTPS) versions of resources if available, suggesting that it is a security-conscious request.
- **Pragma:** The client tells the server not to cache the response.
- **Cache-Control:** This is another directive telling the server not to cache the response.

3. **What is the IP address of your computer? Of the gaia.cs.umass.edu server?**
   Answer: In the captured tcpdump session, I can see that my computer's IP address (the client) is **172.28.108.84**, which is indicated as the source IP address in the packets sent from my machine. The IP address of the server, gaia.cs.umass.edu, is **12.245.119.128**, which appears as the destination IP address in the packets sent to the server.

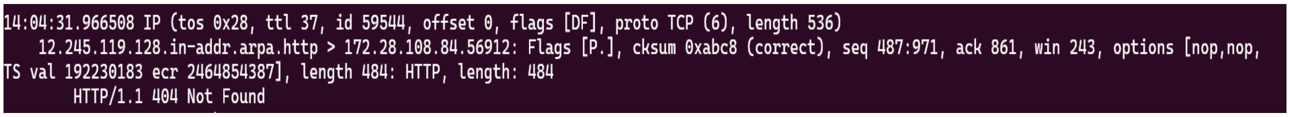
4. **What is the status code returned from the server to your browser?**
   Answer: The status code returned from the server to my browser is 200 OK. I can see this in the captured tcpdump output, which shows the line:



```
14:04:31.696462 IP (tos 0x28, ttl 37, id 59543, offset 0, flags [DF], proto TCP (6), length 538)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [P.], cksum 0x5816 (correct), seq 1:487, ack 421, win 235, options [nop,nop,TS
 val 192229913 ecr 2464854114], length 486: HTTP, length: 486
        HTTP/1.1 200 OK
```

This 200 OK status code means that the request I made was successful, and the server has sent back the requested content.

However, in a subsequent request, I received a 404 Not Found status code instead. This 404 Not Found response means that the server could not find the requested resource, which in this case appears to be the `/favicon.ico` file.

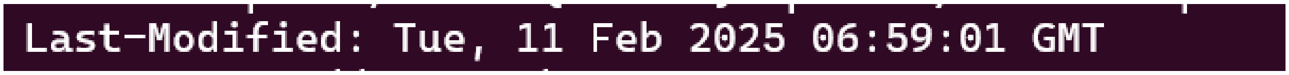


```
14:04:31.966508 IP (tos 0x28, ttl 37, id 59544, offset 0, flags [DF], proto TCP (6), length 536)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56912: Flags [P.], cksum 0xabc8 (correct), seq 487:971, ack 861, win 243, options [nop,nop,
TS val 192230183 ecr 2464854387], length 484: HTTP, length: 484
        HTTP/1.1 404 Not Found
```

5. **When was the HTML file that you are retrieving last modified at the server?**
   Answer:

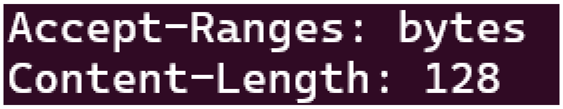
   The HTML file that you are retrieving was last modified on **Tue, 11 Feb 2025 06:59:01 GMT**. This timestamp helps the browser to decide if the file is still valid or if it needs to request a fresh copy.

   This information can be found in the `Last-Modified` header of the HTTP response:



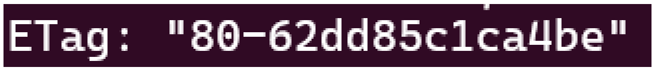
```
Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
```

6. **How many bytes of content are being returned to your browser?**
   Answer: The `Content-Length` header in the response will indicate the number of bytes of content. This field tells the browser how many bytes it should expect in the body of the response.



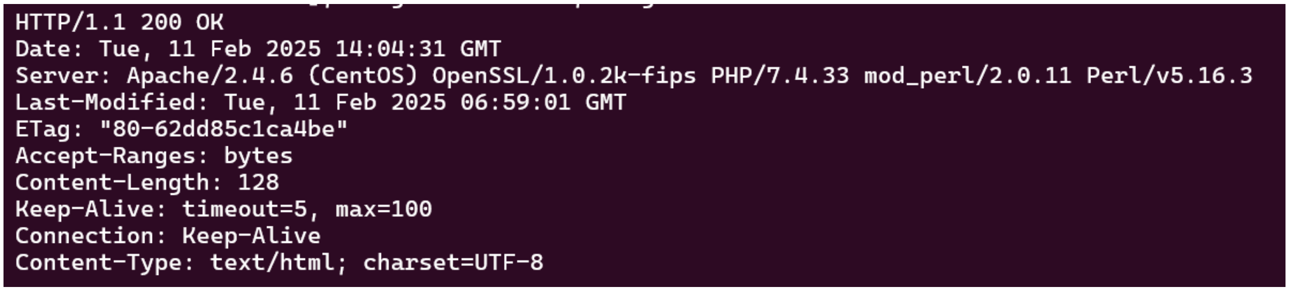
```
Accept-Ranges: bytes
Content-Length: 128
```

7. **By inspecting the raw data in the "packet bytes" pane, do you see any HTTP headers within the data that are not displayed in the "packet details" pane? If so, name one.**
   Answer: From the raw packet data, one HTTP header that appears in the captured packets but is not specifically shown in the "packet details" section of tcpdump is:



```
ETag: "80-62dd85c1ca4be"
```

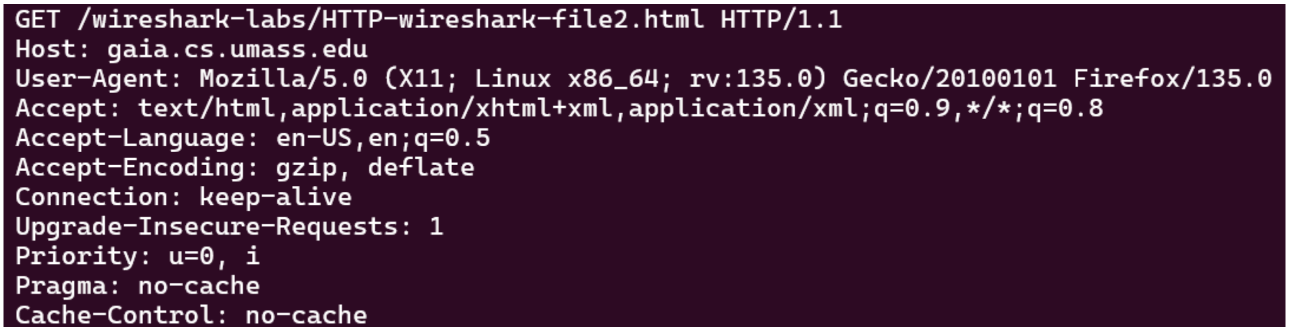
   This header is part of the HTTP response (see the response with HTTP/1.1 200 OK), but tcpdump doesn't display it in the structured format by default. It is shown in the raw packet data instead, specifically in this section



```
HTTP/1.1 200 OK
Date: Tue, 11 Feb 2025 14:04:31 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
ETag: "80-62dd85c1ca4be"
Accept-Ranges: bytes
Content-Length: 128
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Task B: HTTP CONDITIONAL GET/response interaction

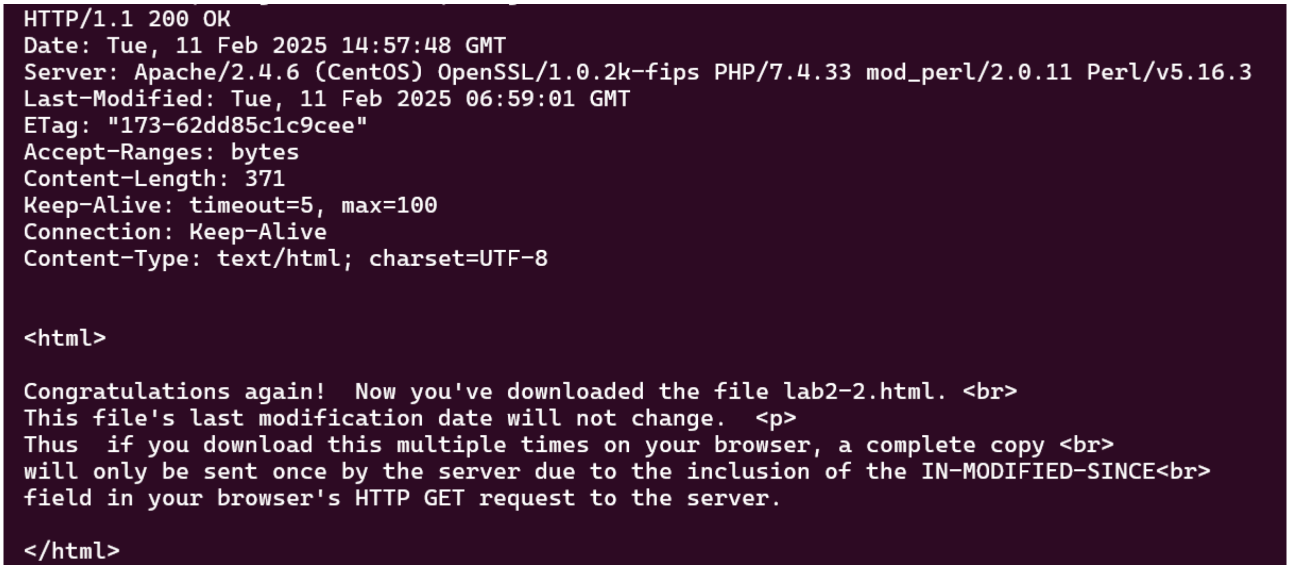
8. **Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?**
   Answer: No, in the first HTTP GET request, there is no "IF-MODIFIED-SINCE" line in the HTTP headers. This header would only appear in subsequent requests if the browser wants to tell the server that it has a cached copy of the requested resource and is checking if it has changed since the last request.



```
GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache
```

9. **Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?**
   Answer: Yes, the server explicitly returns the contents of the file.

The **HTTP status code** is `200 OK`, which indicates that the server has successfully processed the request and is returning the requested content. The **body of the response** contains the HTML content.



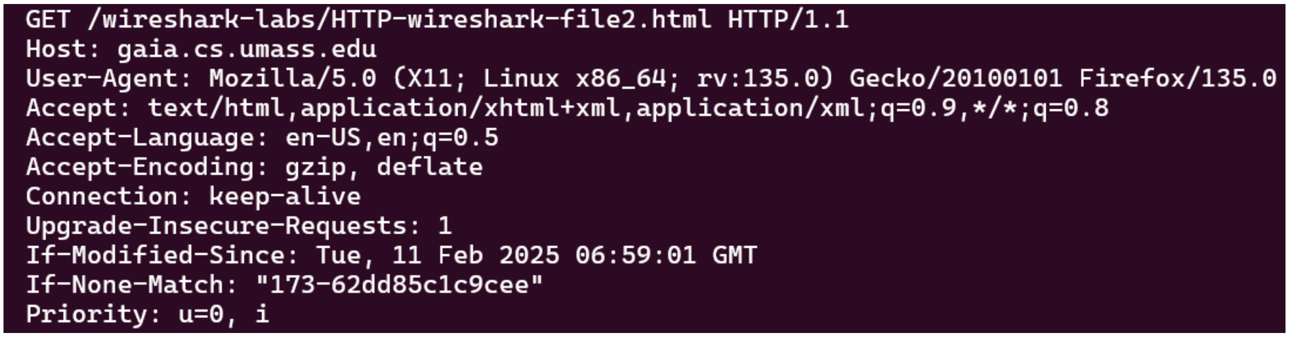
```
HTTP/1.1 200 OK
Date: Tue, 11 Feb 2025 14:57:48 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
ETag: "173-62dd85c1c9cee"
Accept-Ranges: bytes
Content-Length: 371
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8


<html>

Congratulations again!  Now you've downloaded the file lab2-2.html. <br>
This file's last modification date will not change.  <p>
Thus  if you download this multiple times on your browser, a complete copy <br>
will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>
field in your browser's HTTP GET request to the server.

</html>
```

10. **Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?**
    Answer:

Yes, the second HTTP GET request includes an **"If-Modified-Since"** header, which instructs the server to return the requested resource only if it has been modified after the specified date and time: **February 11, 2025, 06:59:01 GMT**.

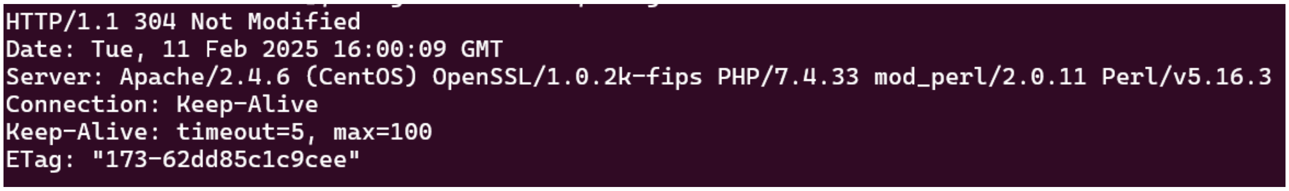
This header is used by browsers to optimize network efficiency. When the browser already has a cached version of the resource, it sends the "If-Modified-Since" header to check if the resource has changed on the server since the last retrieval. If the resource hasn't been modified, the server responds with a 304 Not Modified status, indicating that the content hasn't changed, and therefore, the server doesn't need to send the content again.

In this case, the browser is asking the server if the resource has been updated since the specified timestamp, and if not, it expects the server to refrain from resending the content. This mechanism helps reduce unnecessary data transfer when the content remains unchanged.



```
GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Tue, 11 Feb 2025 06:59:01 GMT
If-None-Match: "173-62dd85c1c9cee"
Priority: u=0, i
```

11. **What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.**
    Answer: The HTTP status code returned by the server in response to the second HTTP GET request is 304 Not Modified.



```
HTTP/1.1 304 Not Modified
Date: Tue, 11 Feb 2025 16:00:09 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
ETag: "173-62dd85c1c9cee"
```

The **304 Not Modified** status indicates that the resource has not changed since the date specified in the **If-Modified-Since** header (February 11, 2025, 06:59:01 GMT). As a result, the server does not send the file content again.

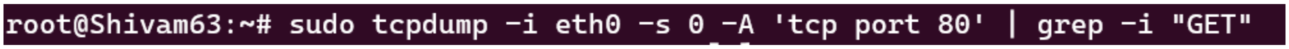
The **ETag** value ("173-62dd85c1c9cee") included in the response confirms that the resource hasn't been modified. Since the ETag matches the version already stored in the browser's cache, the server determines the content is unchanged and simply instructs the browser to use its cached version.

In summary, the server did not return the file's contents because the resource was unmodified. Instead, it sent a **304 Not Modified** response, allowing the browser to continue using its cached copy, thus optimizing bandwidth usage.

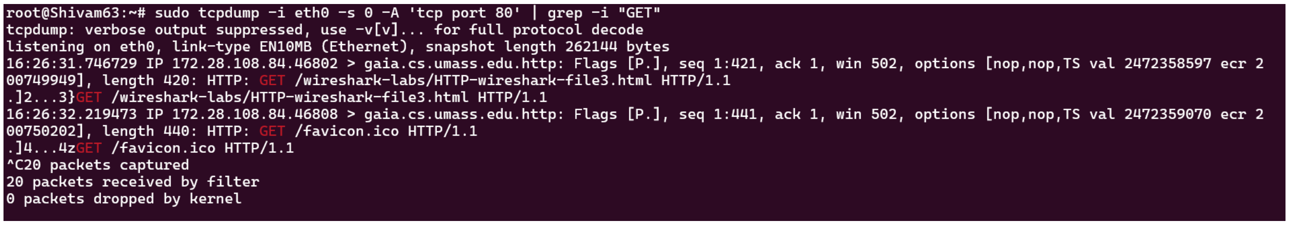
Task C: Retrieving Long Documents

12. **How many HTTP GET request messages were sent by your browser?**
    Answer:

There are two HTTP GET requests. The first GET request is for the resource `/wireshark-labs/HTTP-wireshark-file3.html`, and the second GET request is for the `/favicon.ico` resource. These GET requests are represented in the output as `GET` method lines in the HTTP headers.



```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -A 'tcp port 80' | grep -i "GET"
```



```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -A 'tcp port 80' | grep -i "GET"
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:26:31.746729 IP 172.28.108.84.46802 > gaia.cs.umass.edu.http: Flags [P.], seq 1:421, ack 1, win 502, options [nop,nop,TS val 2472358597 ecr 2
00749949], length 420: HTTP: GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
.]2...3}GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
16:26:32.219473 IP 172.28.108.84.46808 > gaia.cs.umass.edu.http: Flags [P.], seq 1:441, ack 1, win 502, options [nop,nop,TS val 2472359070 ecr 2
00750202], length 440: HTTP: GET /favicon.ico HTTP/1.1
.]4...4zGET /favicon.ico HTTP/1.1
^C20 packets captured
20 packets received by filter
0 packets dropped by kernel
```

`-i eth0`: Specifies the network interface to capture traffic from (replace `eth0` with the appropriate interface, such as `wlan0` for wireless).

`-s 0`: Ensures that the full packet is captured (the default capture size is 68 bytes, which might truncate HTTP data).

`-A`: Prints the contents of the packet in ASCII, which is useful for reading the HTTP data.

`'tcp port 80'`: Filters the traffic to capture only packets that are using port 80, which is the default port for HTTP.

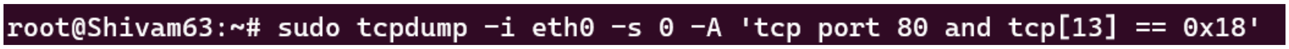
`grep -i "GET"`: Filters the output to show only HTTP GET requests (the `-i` option makes the search case-insensitive).

13. **How many data-containing TCP segments were needed to carry the single HTTP response?**

    Answer:

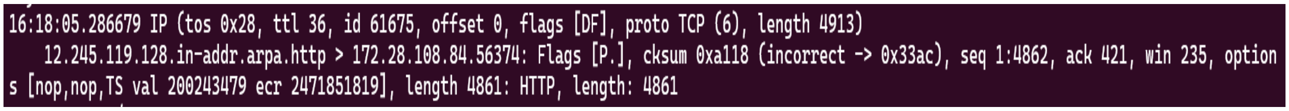
The HTTP response for the file `/wireshark-labs/HTTP-wireshark-file3.html` is quite large, and it's broken up 2 segments. Here's a breakdown:

1. **First segment**: The first segment of the HTTP response is 4861 bytes long (`seq 1:4862`), which contains the actual HTTP response headers and the first portion of the body.
2. **Second segment**: The second segment (`seq 1:486`) likely contains additional data.

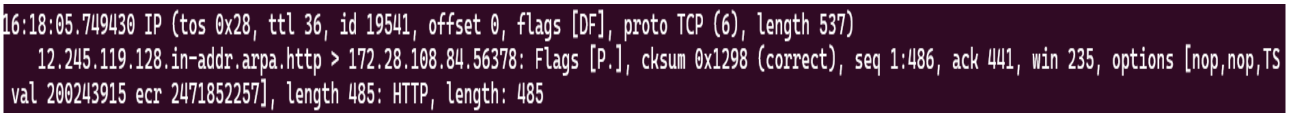


```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -A 'tcp port 80 and tcp[13] == 0x18'
```

`'tcp[13] == 0x18'`: Filter packets with the `PSH` (push) and `ACK` (acknowledgment) flags set, which are commonly seen in data-carrying segments.



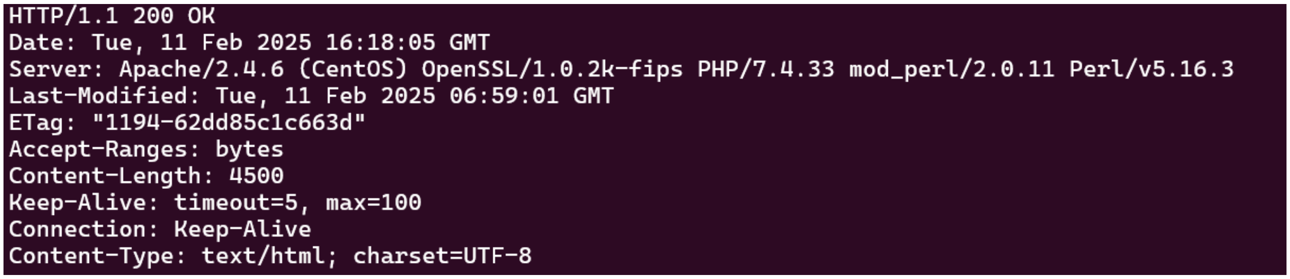
```
16:18:05.286679 IP (tos 0x28, ttl 36, id 61675, offset 0, flags [DF], proto TCP (6), length 4913)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56374: Flags [P.], cksum 0xa118 (incorrect -> 0x33ac), seq 1:4862, ack 421, win 235, option
s [nop,nop,TS val 200243479 ecr 2471851819], length 4861: HTTP, length: 4861
```



```
16:18:05.749430 IP (tos 0x28, ttl 36, id 19541, offset 0, flags [DF], proto TCP (6), length 537)
    12.245.119.128.in-addr.arpa.http > 172.28.108.84.56378: Flags [P.], cksum 0x1298 (correct), seq 1:486, ack 441, win 235, options [nop,nop,TS
 val 200243915 ecr 2471852257], length 485: HTTP, length: 485
```

14. **What is the status code and phrase associated with the response to the HTTP GET request?**

    Answer: The response to the HTTP GET request returns the status code `200 OK`, indicating a successful request. This status code is found in the line `HTTP/1.1 200 OK`, which is part of the response headers.



```
HTTP/1.1 200 OK
Date: Tue, 11 Feb 2025 16:18:05 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 11 Feb 2025 06:59:01 GMT
ETag: "1194-62dd85c1c663d"
Accept-Ranges: bytes
Content-Length: 4500
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

15. **Is there any HTTP header information in the transmitted data associated with TCP segmentation?**

    Answer: Yes, there is HTTP header information in the transmitted data, especially as part of the HTTP response. Specifically, the headers from the response include:

These headers are part of the HTTP response and are transmitted as part of the **TCP segments**. When the server sends a large HTTP response, it is often split across multiple segments due to size limitations in TCP. The HTTP headers and body are carried in these segments, ensuring that they reach the client correctly. Each segment also contains the necessary TCP metadata, such as sequence numbers and acknowledgment information, for the reliable delivery of the data.

Task D: HTML Documents with Embedded Objects

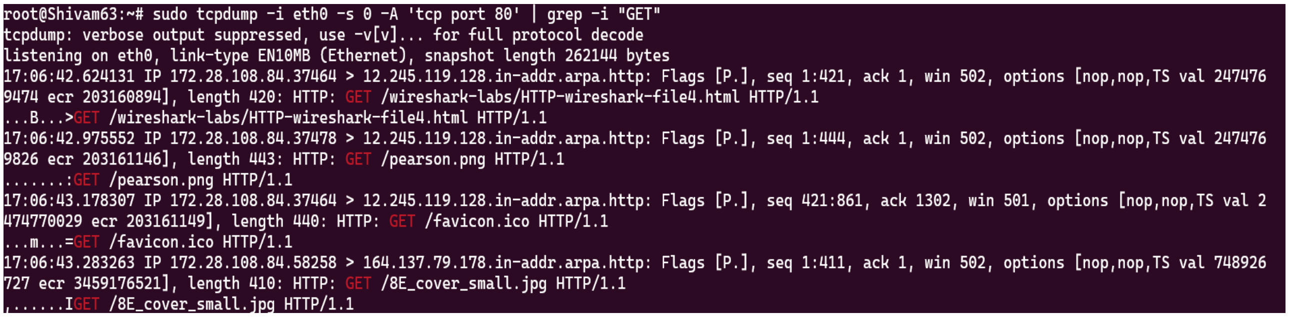
16. **How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?**

    Answer:

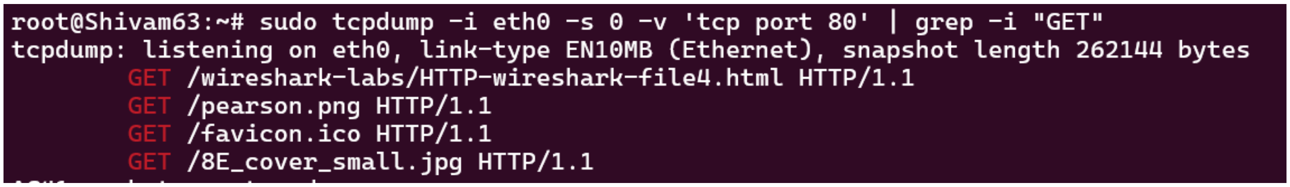
In the **first output (without cache)**, we observe 4 `GET` requests being captured. The requests include:

1. `GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1`
2. `GET /pearson.png HTTP/1.1`
3. `GET /favicon.ico HTTP/1.1`
4. `GET /8E_cover_small.jpg HTTP/1.1`.

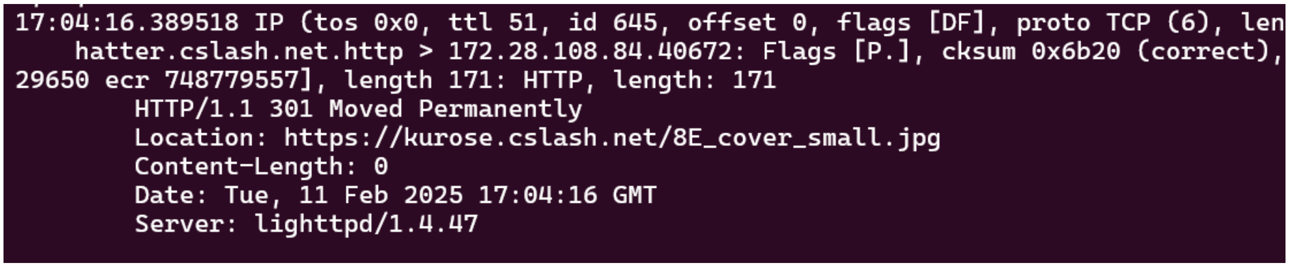
These multiple requests are sent because the client is requesting different resources, such as an HTML file, images, and a favicon, over the network. Additionally, a `301 Moved Permanently` HTTP response is seen for one of these requests, indicating that the requested resource has been permanently redirected to a new location. This response likely leads to the client making another request for the redirected resource, further contributing to the 4 `GET` requests observed. The total of 4 `GET` requests suggests that the resources were not yet cached, and the browser was requesting them anew from the server.



```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -A 'tcp port 80' | grep -i "GET"
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:06:42.624131 IP 172.28.108.84.37464 > 12.245.119.128.in-addr.arpa.http: Flags [P.], seq 1:421, ack 1, win 502, options [nop,nop,TS val 247476
9474 ecr 203160894], length 420: HTTP: GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
...B...>GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
17:06:42.975552 IP 172.28.108.84.37478 > 12.245.119.128.in-addr.arpa.http: Flags [P.], seq 1:444, ack 1, win 502, options [nop,nop,TS val 247476
9826 ecr 203161146], length 443: HTTP: GET /pearson.png HTTP/1.1
.......:GET /pearson.png HTTP/1.1
17:06:43.178307 IP 172.28.108.84.37464 > 12.245.119.128.in-addr.arpa.http: Flags [P.], seq 421:861, ack 1302, win 501, options [nop,nop,TS val 2
474770029 ecr 203161149], length 440: HTTP: GET /favicon.ico HTTP/1.1
...m...=GET /favicon.ico HTTP/1.1
17:06:43.283263 IP 172.28.108.84.58258 > 164.137.79.178.in-addr.arpa.http: Flags [P.], seq 1:411, ack 1, win 502, options [nop,nop,TS val 748926
727 ecr 3459176521], length 410: HTTP: GET /8E_cover_small.jpg HTTP/1.1
,......IGET /8E_cover_small.jpg HTTP/1.1
```

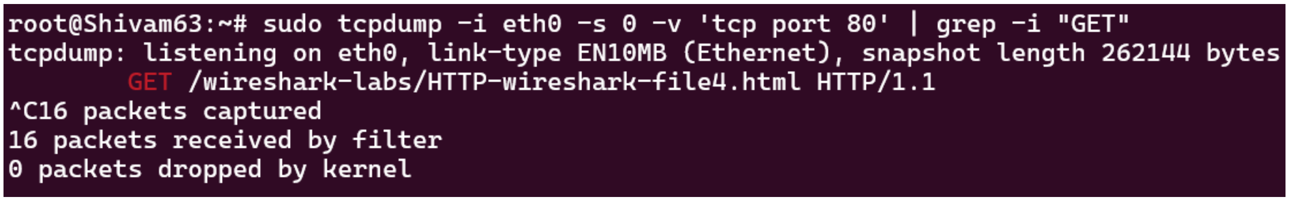


```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -v 'tcp port 80' | grep -i "GET"
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
        GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
        GET /pearson.png HTTP/1.1
        GET /favicon.ico HTTP/1.1
        GET /8E_cover_small.jpg HTTP/1.1
```



```
17:04:16.389518 IP (tos 0x0, ttl 51, id 645, offset 0, flags [DF], proto TCP (6), len
    hatter.cslash.net.http > 172.28.108.84.40672: Flags [P.], cksum 0x6b20 (correct),
29650 ecr 748779557], length 171: HTTP, length: 171
        HTTP/1.1 301 Moved Permanently
        Location: https://kurose.cslash.net/8E_cover_small.jpg
        Content-Length: 0
        Date: Tue, 11 Feb 2025 17:04:16 GMT
        Server: lighttpd/1.4.47
```

Cache Version

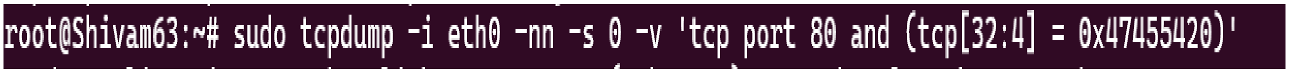


```
root@Shivam63:~# sudo tcpdump -i eth0 -s 0 -v 'tcp port 80' | grep -i "GET"
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
        GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
^C16 packets captured
16 packets received by filter
0 packets dropped by kernel
```

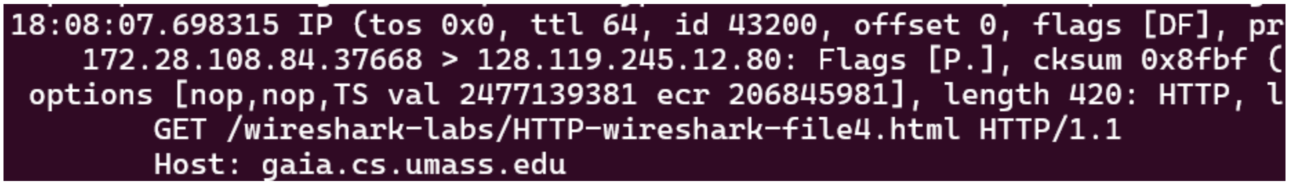
In **this output (with cache)**, only 1 GET request is captured, which is for the resource /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1. This behavior is due to caching. When the resource is cached, the browser doesn't need to send additional GET requests for the same resources. Any subsequent requests for those resources are served from the cache rather than the server, so no new GET requests are triggered. The presence of only 1 GET request suggests that the initial request for the page loaded the resources, and the cache prevented the need for additional requests, including the ones for images and other content.

17. **Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.**

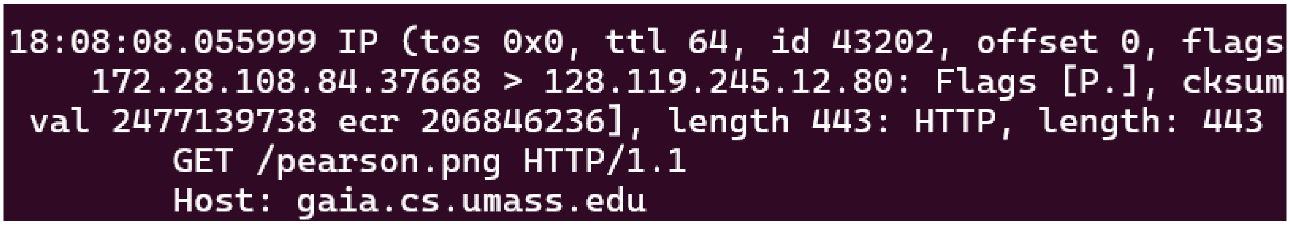
Answer:



```
root@Shivam63:~# sudo tcpdump -i eth0 -nn -s 0 -v 'tcp port 80 and (tcp[32:4] = 0x47455420)'
```



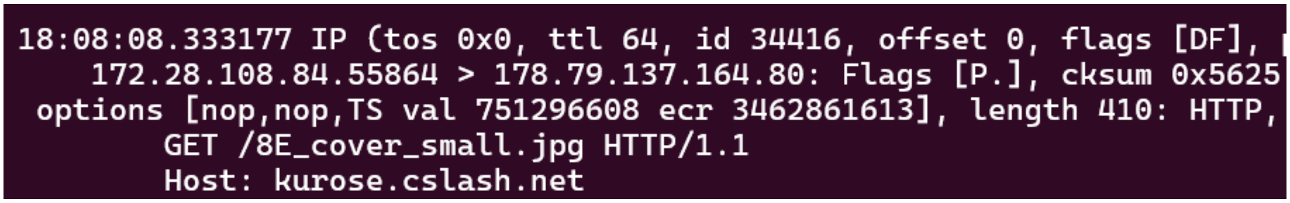
```
18:08:07.698315 IP (tos 0x0, ttl 64, id 43200, offset 0, flags [DF], pr
    172.28.108.84.37668 > 128.119.245.12.80: Flags [P.], cksum 0x8fbf (
 options [nop,nop,TS val 2477139381 ecr 206845981], length 420: HTTP, l
        GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
        Host: gaia.cs.umass.edu
```



```
18:08:08.055999 IP (tos 0x0, ttl 64, id 43202, offset 0, flags
    172.28.108.84.37668 > 128.119.245.12.80: Flags [P.], cksum
 val 2477139738 ecr 206846236], length 443: HTTP, length: 443
        GET /pearson.png HTTP/1.1
        Host: gaia.cs.umass.edu
```

The **first and second requests** (GET /wireshark-labs/HTTP-wireshark-file4.html and GET /pearson.png) are going to the same IP address 128.119.245.12 (i.e., both images are being requested from the same server).

The **third request** (GET /8E_cover_small.jpg) is going to a **different IP address** 178.79.137.164, indicating that this image is hosted on a different server.



```
18:08:08.333177 IP (tos 0x0, ttl 64, id 34416, offset 0, flags [DF],
    172.28.108.84.55864 > 178.79.137.164.80: Flags [P.], cksum 0x5625
 options [nop,nop,TS val 751296608 ecr 3462861613], length 410: HTTP,
        GET /8E_cover_small.jpg HTTP/1.1
        Host: kurose.cslash.net
```

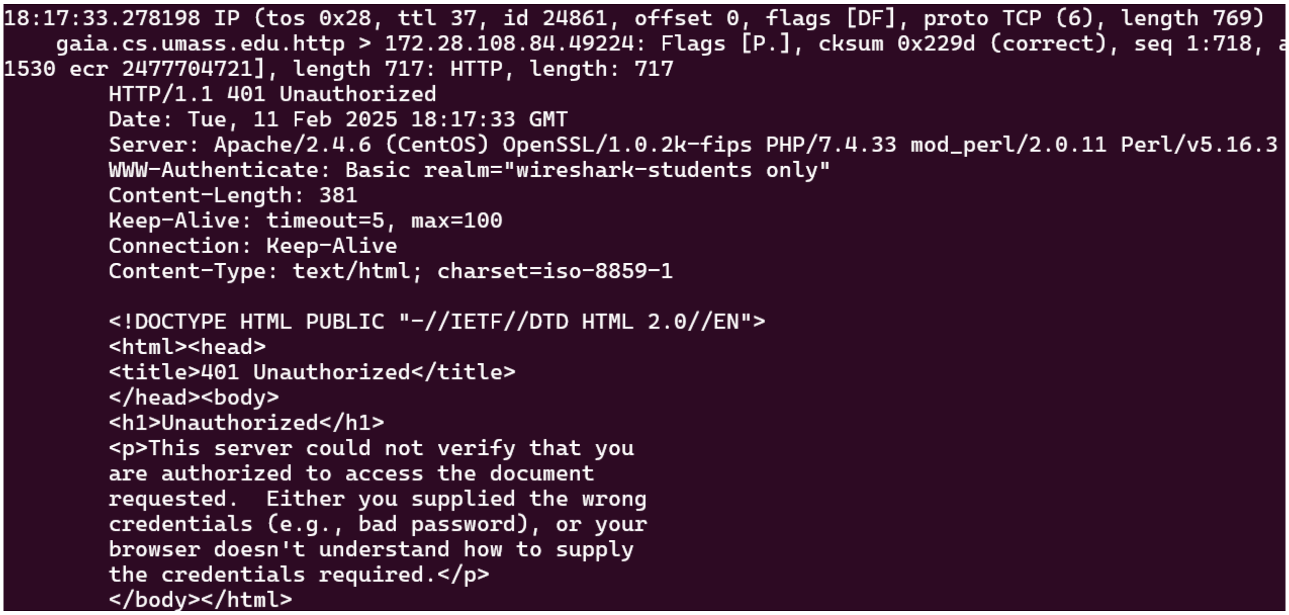
he **first image** (`GET /pearson.png`) is requested **just 0.357684 seconds** after the **HTML page** request (`GET /wireshark-labs/HTTP-wireshark-file4.html`). The **second image** (`GET /8E_cover_small.jpg`) is requested **within 0.277178 seconds** after the **second request** (`GET /pearson.png`), but from a **different server**.

The fact that two images are being requested almost simultaneously from different IP addresses suggests that the browser is downloading them in parallel from two separate servers, rather than one after the other. Therefore, we can conclude that the images were downloaded in parallel from two different web servers.

Task E: HTTP Authentication

18. **What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?**

Answer: When I first enter the URL in your browser, it sends an initial HTTP GET request to the server. The server responds with a **401 Unauthorized** status code, indicating that authentication is required. The response includes the **WWW-Authenticate** header, specifying the authentication method (**Basic Authentication**) and the **realm** ("wireshark-students only"), which defines the protected area that requires authentication. This tells the browser that it needs to supply credentials to access the requested resource.

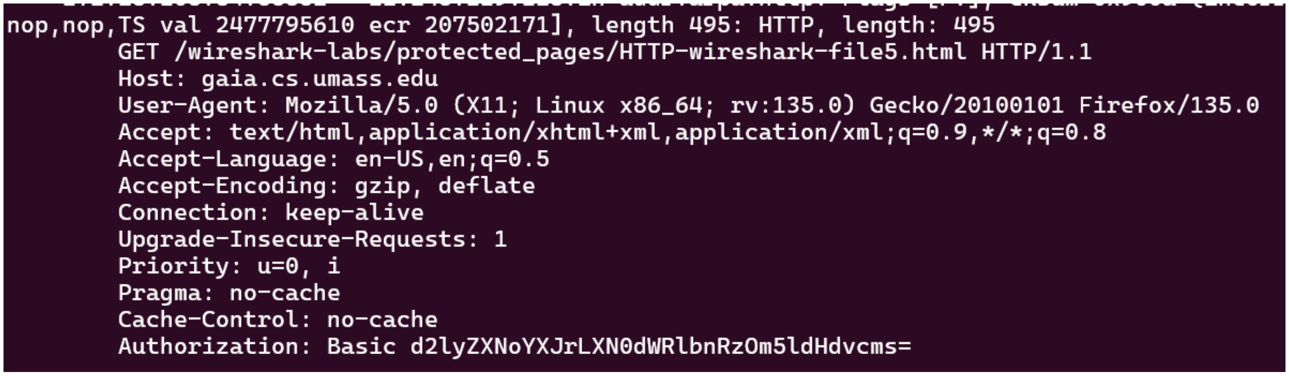


```
18:17:33.278198 IP (tos 0x28, ttl 37, id 24861, offset 0, flags [DF], proto TCP (6), length 769)
    gaia.cs.umass.edu.http > 172.28.108.84.49224: Flags [P.], cksum 0x229d (correct), seq 1:718, a
1530 ecr 2477704721], length 717: HTTP, length: 717
        HTTP/1.1 401 Unauthorized
        Date: Tue, 11 Feb 2025 18:17:33 GMT
        Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
        WWW-Authenticate: Basic realm="wireshark-students only"
        Content-Length: 381
        Keep-Alive: timeout=5, max=100
        Connection: Keep-Alive
        Content-Type: text/html; charset=iso-8859-1

        <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
        <html><head>
        <title>401 Unauthorized</title>
        </head><body>
        <h1>Unauthorized</h1>
        <p>This server could not verify that you
        are authorized to access the document
        requested.  Either you supplied the wrong
        credentials (e.g., bad password), or your
        browser doesn't understand how to supply
        the credentials required.</p>
        </body></html>
```

19. **When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?**

Answer:

In the second HTTP GET message, the key addition to the request header is the **Authorization** field. This field is used for authentication and includes the **Basic Authentication** credentials. These credentials are encoded in Base64 format to securely transmit the username and password. In this case, the header appears as:



```
nop,nop,TS val 2477795610 ecr 207502171], length 495: HTTP, length: 495
        GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
        Host: gaia.cs.umass.edu
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Connection: keep-alive
        Upgrade-Insecure-Requests: 1
        Priority: u=0, i
        Pragma: no-cache
        Cache-Control: no-cache
        Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=
```

This encoded string represents the username and password pair, which are typically sent by the browser after receiving a 401 Unauthorized status code from the server. The server originally responded with a 401 status, signaling that the requested resource requires authentication. The Authorization header in the second GET request is the browser's attempt to authenticate the user with the provided credentials. The server can then decode and verify these credentials to grant access to the requested resource, which in this case is a protected page.

20. **What does the "Connection: close" and "Connection: Keep-alive" header field imply in HTTP protocol? When should one be used over the other?**

Answer:

The Connection header field in HTTP is used to control the behavior of network connections between a client (browser) and a server. This header helps define whether the connection should be terminated after a single HTTP request/response or kept open for additional requests.

1. Connection: close

The Connection: close header indicates that the connection between the client and server will be **closed** immediately after the current HTTP request/response is completed.

- **Key Characteristics:**
    - After the server sends the requested data to the client, it closes the TCP connection.
    - The client must establish a new connection to the server for every subsequent HTTP request.
    - This was the default behavior in **HTTP/1.0**, where each HTTP transaction (request/response pair) occurred over a separate TCP connection.
- **Advantages:**
    - Simple to implement: Each request is isolated, so there's no need to manage persistent connections.
    - Suitable for environments with low traffic or for single-request operations.
- **Disadvantages:**
    - High overhead: Establishing and tearing down a new TCP connection for each request is resource-intensive and adds latency.
    - Inefficient for modern web pages, which often require multiple resources (e.g., CSS, JavaScript, images).
- **Use Cases:**
    - Applications or services where only a single request is expected, and maintaining a persistent connection is unnecessary.
    - Situations where the server or client has limited resources and cannot handle the overhead of managing many open connections.

2. Connection: Keep-Alive

The Connection: Keep-Alive header is used to keep the connection between the client and server **open** after the initial request/response transaction. This allows multiple HTTP requests to be sent over the same connection without repeatedly establishing and tearing it down.

- **Key Characteristics:**
    - The client and server reuse the same TCP connection for multiple requests/responses.
    - This behavior is the default in **HTTP/1.1**, which assumes persistent connections

unless explicitly told otherwise via the Connection: close header.
  - Keep-Alive can include additional parameters, such as:
    - timeout: Specifies how long the connection should remain open.
    - max: Specifies the maximum number of requests allowed over the connection.
- **Advantages:**
  - **Reduced latency:** Reusing the same TCP connection eliminates the need to re-establish new connections for each request.
  - **Lower resource usage:** Less overhead in setting up and tearing down TCP connections.
  - **Improved performance:** Essential for modern web pages, which often require many resources (e.g., CSS, JavaScript, images) to be fetched in quick succession.
- **Disadvantages:**
  - Requires the server and client to manage open connections, which could consume resources if too many are kept alive unnecessarily.
  - Can lead to resource exhaustion if connections are left open for too long without proper management.
- **Use Cases:**
  - High-traffic websites or applications where multiple resources (e.g., web assets) are requested in a short time.
  - Real-time applications or APIs that require frequent communication between the client and server.

When to Use Connection: close:

1. **Single-request transactions:** A client or application only sends one HTTP request to the server (e.g., API ping).
2. **Resource-limited servers:** Servers with limited memory and processing power might prefer to close connections immediately to free resources.
3. **Security concerns:** If persistent connections pose risks (e.g., long-lived connections from unauthenticated users), closing them immediately may be preferred.

When to Use Connection: Keep-Alive:

1. **Multi-resource web pages:** Modern websites often require multiple resources like images, CSS files, and JavaScript files, which can all be fetched over the same connection.
2. **High-performance applications:** Persistent connections reduce latency and are better suited for high-traffic applications or APIs.
3. **Real-time communication:** Scenarios like live chat or streaming applications benefit from maintaining open connections for quick data exchanges.

CONCLUSION:	In this experiment, I analyzed various HTTP interactions such as GET requests, server responses, and conditional GETs using tcpdump. I explored HTTP status codes, headers, and the role of caching in optimizing web traffic. I also examined HTTP authentication and the significance of persistent connections like "Keep-Alive" for improving performance. The experiment provided valuable insights into the technical workings of HTTP communication and its impact on web performance and security.