Computer Networks Laboratory

NAME: PREM SAGAR J S	SRN: PES1UG20CS825	SEC: 'H'

Week #3

A) Understanding Persistent and Non-persistent HTTP Connections

EXECUTION STEPS

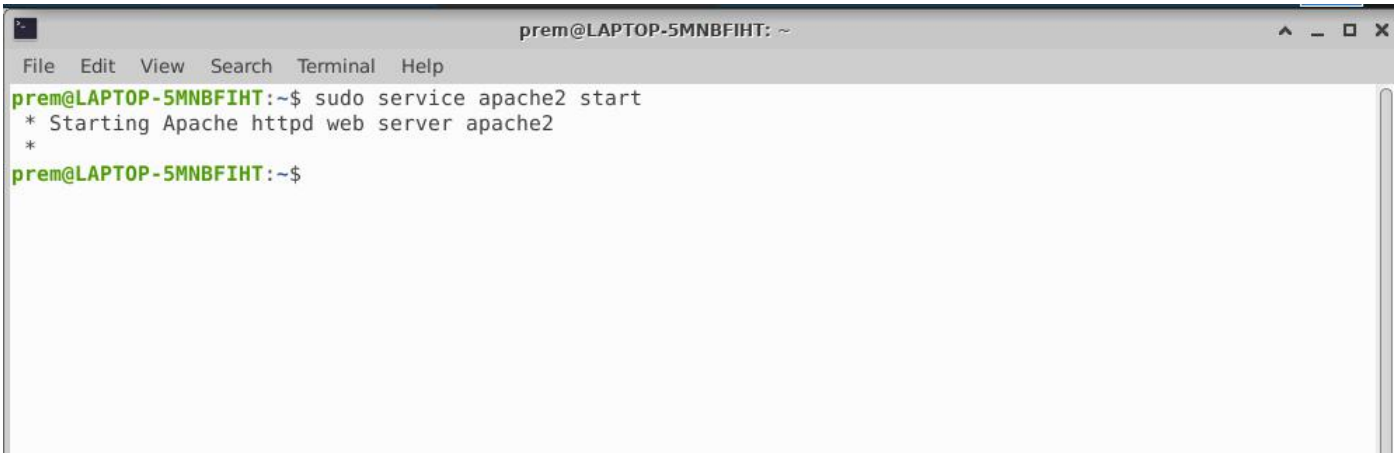
Step 1: Connect 2 desktops using switch and cables as shown below. (Use 2 VMs on Virtualbox or VMware instead of physical connections.)

Server	Client
Ubuntu (vm)	kali linux (vm)

Note: using Windows subsystem Linux for VM's

Server Side:

Step 2: Check your Web Server
At the end of the installation process, Ubuntu 16.04 starts Apache. The web server should already be up and running.



Step 3: Server IP address can be set by the following command

$sudo ip addr

```
prem@LAPTOP-5MNBFIHT: ~
File  Edit  View  Search  Terminal  Help
prem@LAPTOP-5MNBFIHT:~$ sudo ip addr
23: eth0: <> mtu 1500 group default qlen 1
    link/ether c8:d9:d2:ec:e2:cb
    inet 169.254.181.61/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::46a:2e38:789a:b53d/64 scope link dynamic
       valid_lft forever preferred_lft forever
12: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1
    link/ether 0a:00:27:00:00:0c
    inet 10.0.8.25/24 brd 10.0.8.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::1c0:6ae0:1420:5f76/64 scope link dynamic
       valid_lft forever preferred_lft forever
34: eth2: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1
    link/ether 00:15:5d:bb:6d:cd
    inet 172.17.208.225/28 brd 172.17.208.239 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::290e:34b0:c791:6bc/64 scope link dynamic
       valid_lft forever preferred_lft forever
15: eth3: <> mtu 1500 group default qlen 1
    link/ether 74:40:bb:4c:61:24
    inet 169.254.82.115/16 brd 169.254.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::6d9d:8257:546d:5273/64 scope link dynamic
       valid_lft forever preferred_lft forever
```

eth1 : IP 10.0..8.25

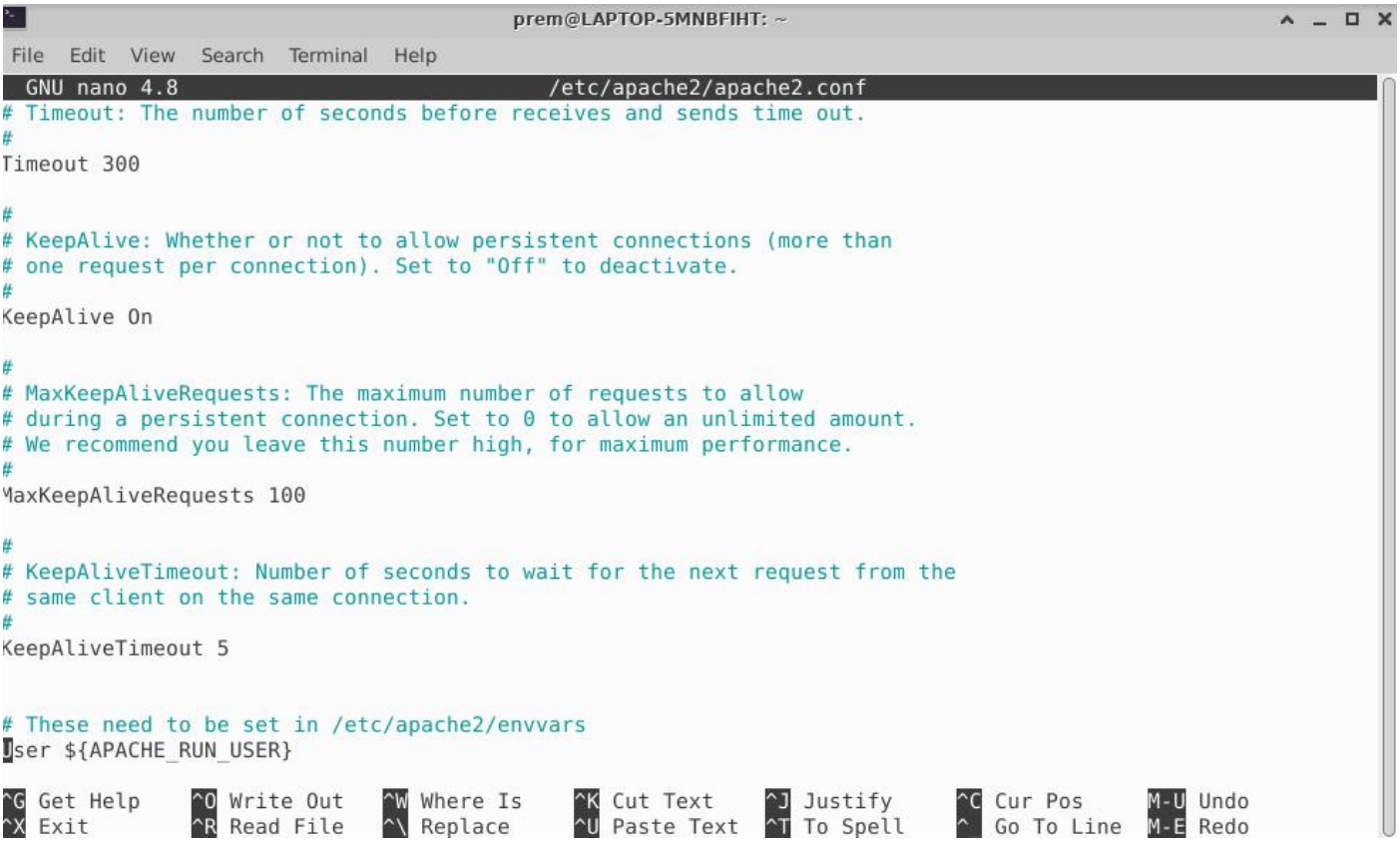
```
12: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1
    link/ether 0a:00:27:00:00:0c
    inet 10.0.8.25/24 brd 10.0.8.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::1c0:6ae0:1420:5f76/64 scope link dynamic
       valid_lft forever preferred_lft forever
```

Step 4: The **apache2.conf** file present in the **etc/apache2** directory is modified as:

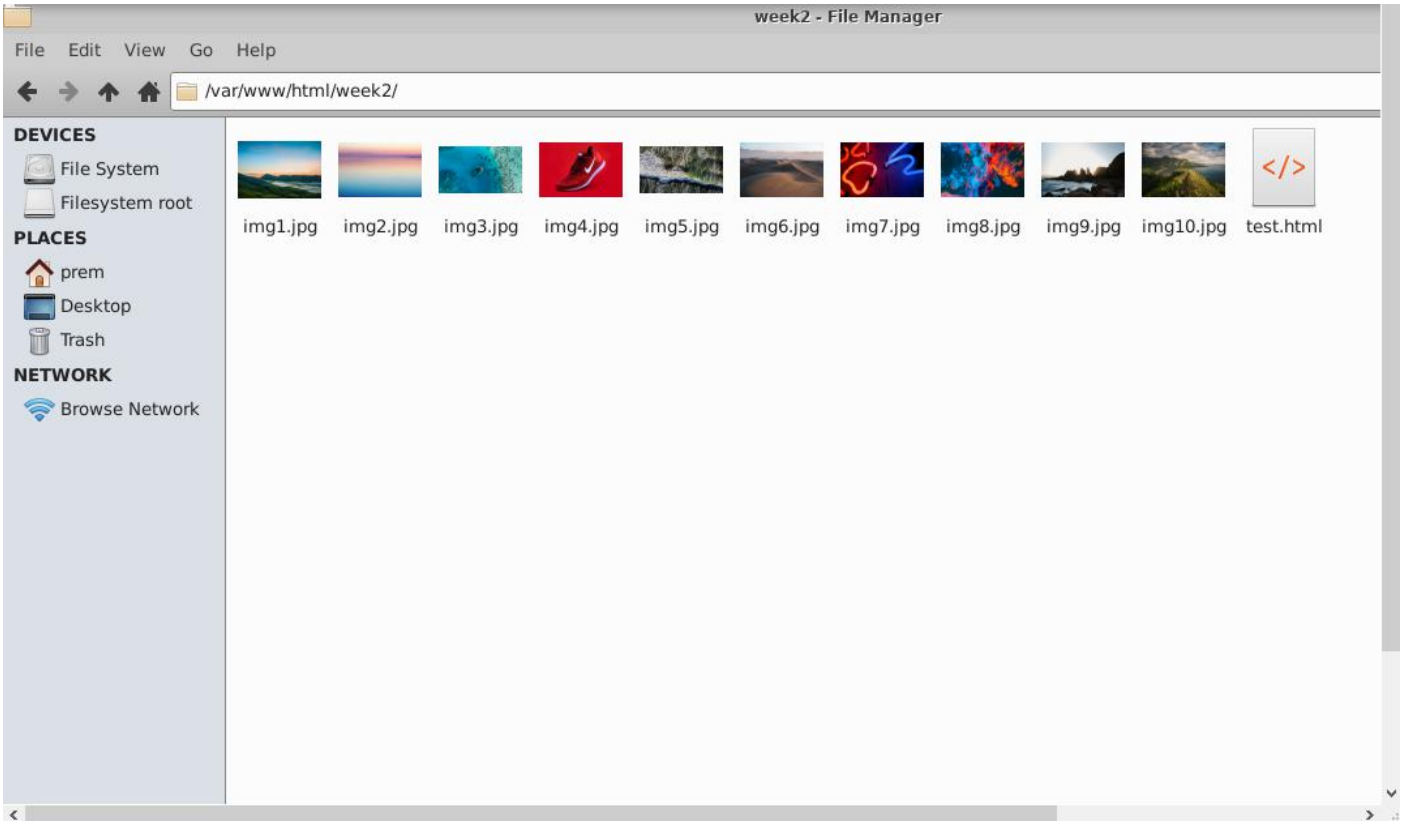
a) The **keep-alive** option was set (i.e. value was made **ON**)

b) The **MaximumKeepAliveRequests** were set to **2**

$sudo nano /etc/apache2/apache2.conf



Step 5: Store images in the server path. A html page consisting of 10 images having size > 2MB were placed and accessed by the client. This html page is stored in the location – **/var/www/html/file_name.html**.



Step 6: Prepare a web page as shown below. The html file needs to add 10 images. (Kindly skip the style attribute in the below image)

```
Terminal
File  Edit  View  Search  Terminal  Help
<!DOCTYPE html>
<html>
<body>
<img src="img1.jpg">
<img src="img2.jpg">
<img src="img3.jpg">
<img src="img4.jpg">
<img src="img5.jpg">
<img src="img6.jpg">
<img src="img7.jpg">
<img src="img8.jpg">
<img src="img9.jpg">
<img src="img10.jpg">
</body>
</html>
~
~
~
~
~
~
~
~
</week2/test.html" [readonly][noeol][dos] 15L, 270C          1,1          All
```

Client side:

Client IP address can be set by the following command.
$sudo ip addr

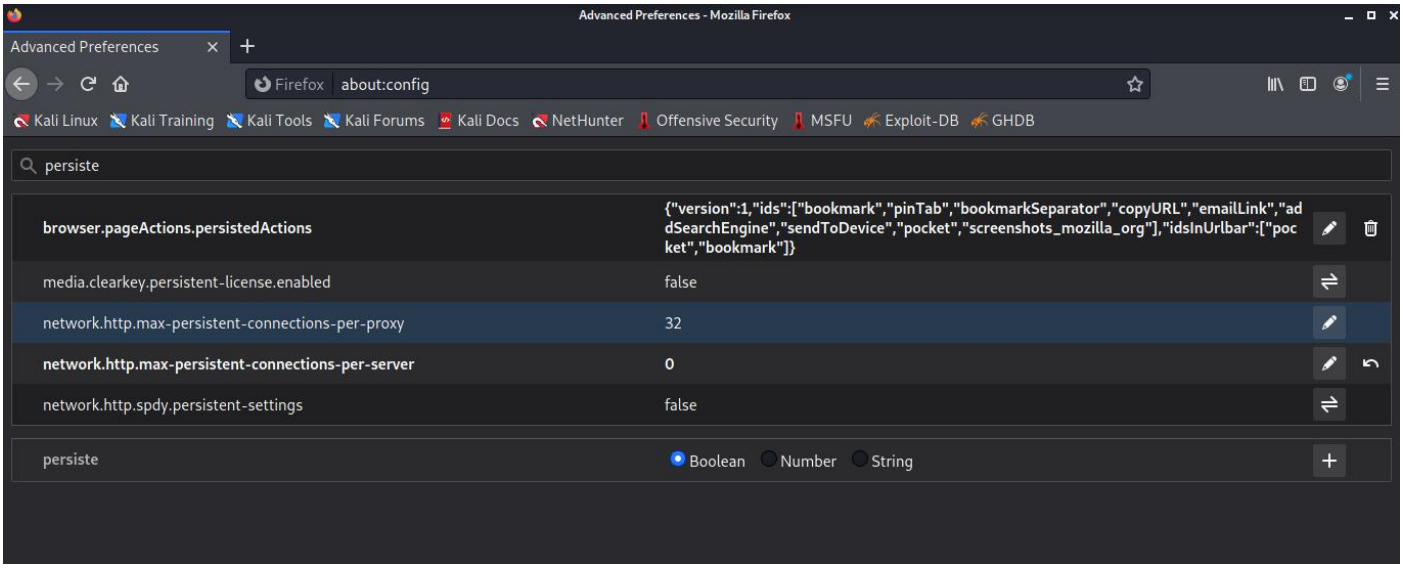
```
prem@LAPTOP-5MNBFIHT:~
File  Actions  Edit  View  Help
┌──(prem㉿LAPTOP-5MNBFIHT)-[~]
└─$ sudo service apache2 stop
[sudo] password for prem:
Stopping Apache httpd web server: apache2.
┌──(prem㉿LAPTOP-5MNBFIHT)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.208.231  netmask 255.255.255.240  broadcast 172.17.208.239
        inet6 fe80::215:5dff:fe52:8b91  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:52:8b:91  txqueuelen 1000  (Ethernet)
        RX packets 30558  bytes 40416760 (38.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10234  bytes 934673 (912.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 33391  bytes 76972933 (73.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33391  bytes 76972933 (73.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(prem㉿LAPTOP-5MNBFIHT)-[~]
└─$
```

PART 1: NON-PERSISTENT CONNECTION

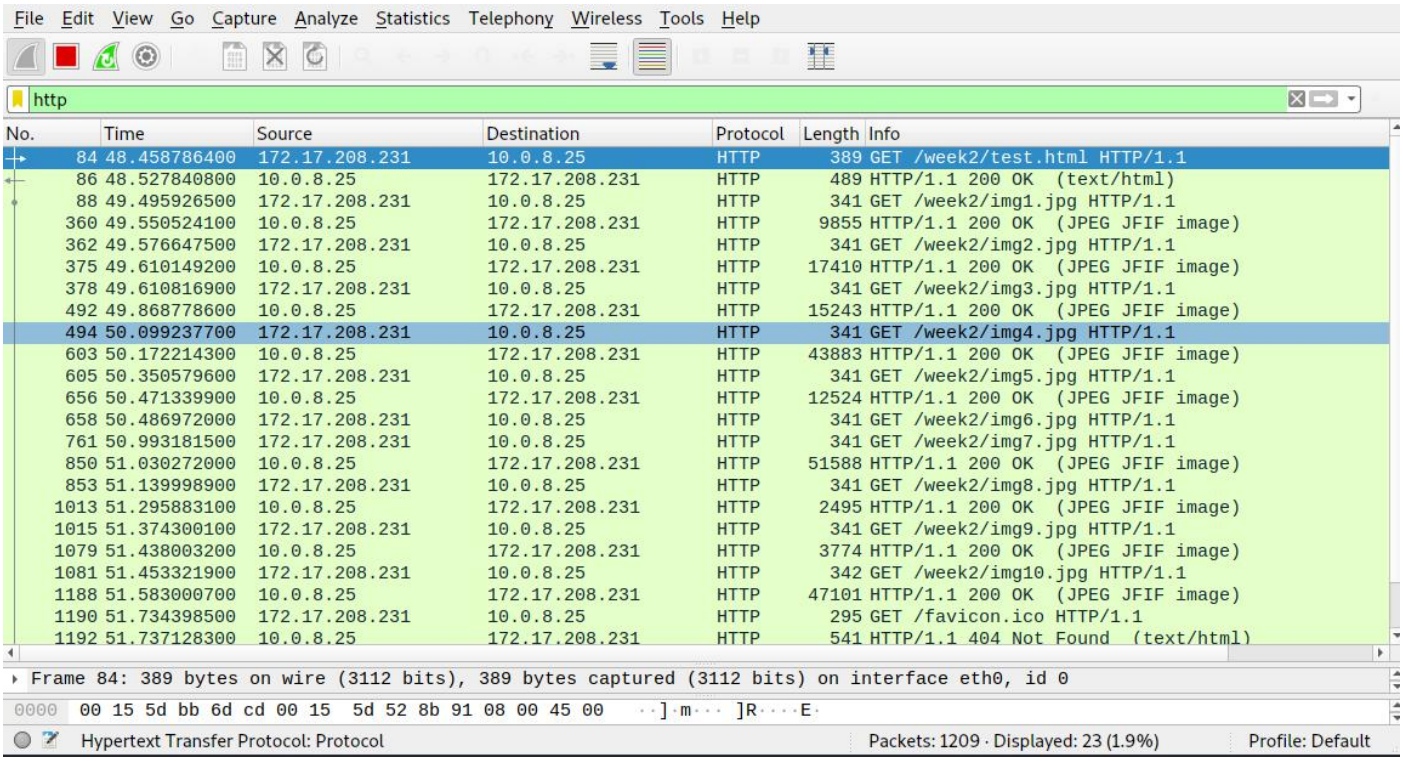
Step 1: This is done by setting the value of max-persistent-connection-per-server to 0 in the client computer



Step 2: Access web page on client-side browser (Firefox)
The client could access the file as:
172.16.10.1/file_name.html where--> **172.16.10.1** is Server's IP

→ File can be accessed as 10.0.8.25/week2/test.html

Step 3: Use wireshark**.** Open wireshark in the server computer while client is trying to access the server's local host webpage. Apply 'http' filter and note the time to capture all the 10 images.

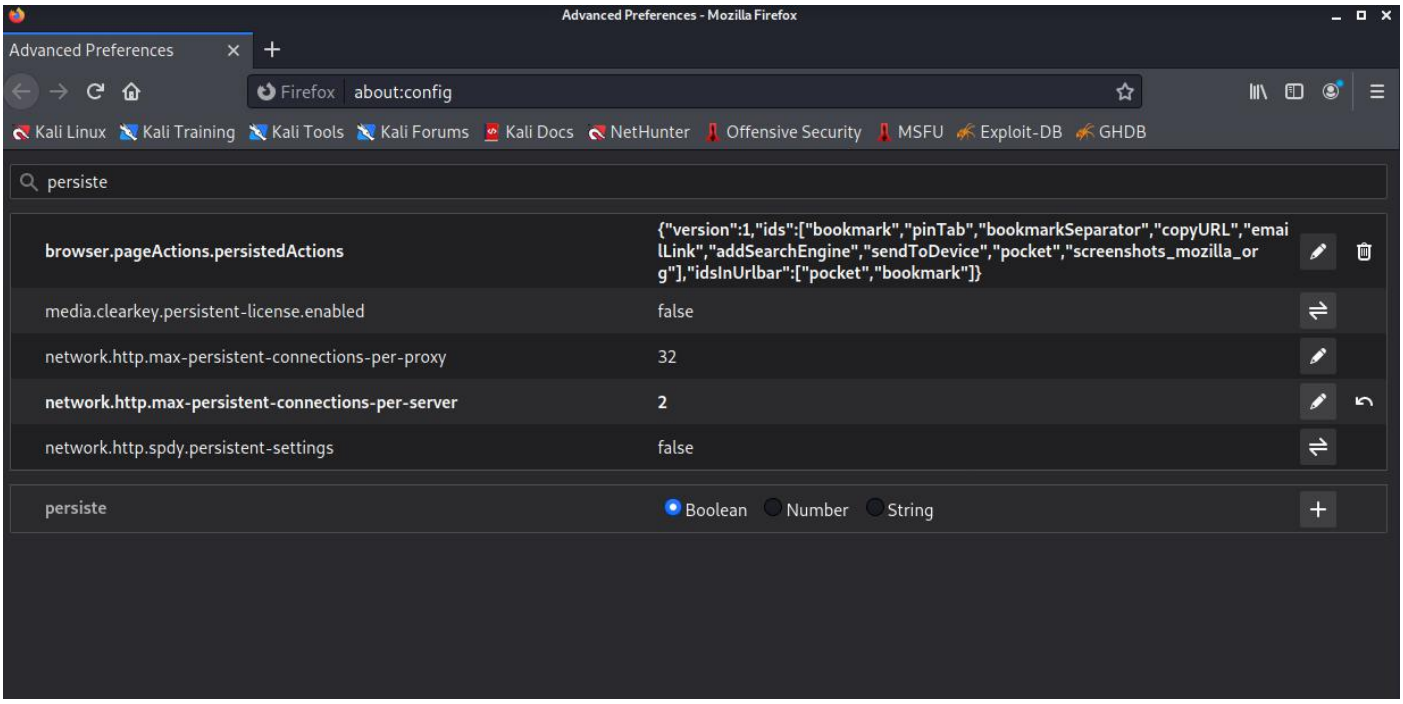


No.	Time	Source	Destination	Protocol	Length	Info
84	48.458786400	172.17.208.231	10.0.8.25	HTTP	389	GET /week2/test.html HTTP/1.1
86	48.527840800	10.0.8.25	172.17.208.231	HTTP	489	HTTP/1.1 200 OK (text/html)
88	49.495926500	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img1.jpg HTTP/1.1
360	49.550524100	10.0.8.25	172.17.208.231	HTTP	9855	HTTP/1.1 200 OK (JPEG JFIF image)
362	49.576647500	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img2.jpg HTTP/1.1
375	49.610149200	10.0.8.25	172.17.208.231	HTTP	17410	HTTP/1.1 200 OK (JPEG JFIF image)
378	49.610816900	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img3.jpg HTTP/1.1
492	49.868778600	10.0.8.25	172.17.208.231	HTTP	15243	HTTP/1.1 200 OK (JPEG JFIF image)
494	50.099237700	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img4.jpg HTTP/1.1
603	50.172214300	10.0.8.25	172.17.208.231	HTTP	43883	HTTP/1.1 200 OK (JPEG JFIF image)
605	50.350579600	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img5.jpg HTTP/1.1
656	50.471339900	10.0.8.25	172.17.208.231	HTTP	12524	HTTP/1.1 200 OK (JPEG JFIF image)
658	50.486972000	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img6.jpg HTTP/1.1
761	50.993181500	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img7.jpg HTTP/1.1
850	51.030272000	10.0.8.25	172.17.208.231	HTTP	51588	HTTP/1.1 200 OK (JPEG JFIF image)
853	51.139998900	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img8.jpg HTTP/1.1
1013	51.295883100	10.0.8.25	172.17.208.231	HTTP	2495	HTTP/1.1 200 OK (JPEG JFIF image)
1015	51.374300100	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img9.jpg HTTP/1.1
1079	51.438003200	10.0.8.25	172.17.208.231	HTTP	3774	HTTP/1.1 200 OK (JPEG JFIF image)
1081	51.453321900	172.17.208.231	10.0.8.25	HTTP	342	GET /week2/img10.jpg HTTP/1.1
1188	51.583000700	10.0.8.25	172.17.208.231	HTTP	47101	HTTP/1.1 200 OK (JPEG JFIF image)
1190	51.734398500	172.17.208.231	10.0.8.25	HTTP	295	GET /favicon.ico HTTP/1.1
1192	51.737128300	10.0.8.25	172.17.208.231	HTTP	541	HTTP/1.1 404 Not Found (text/html)

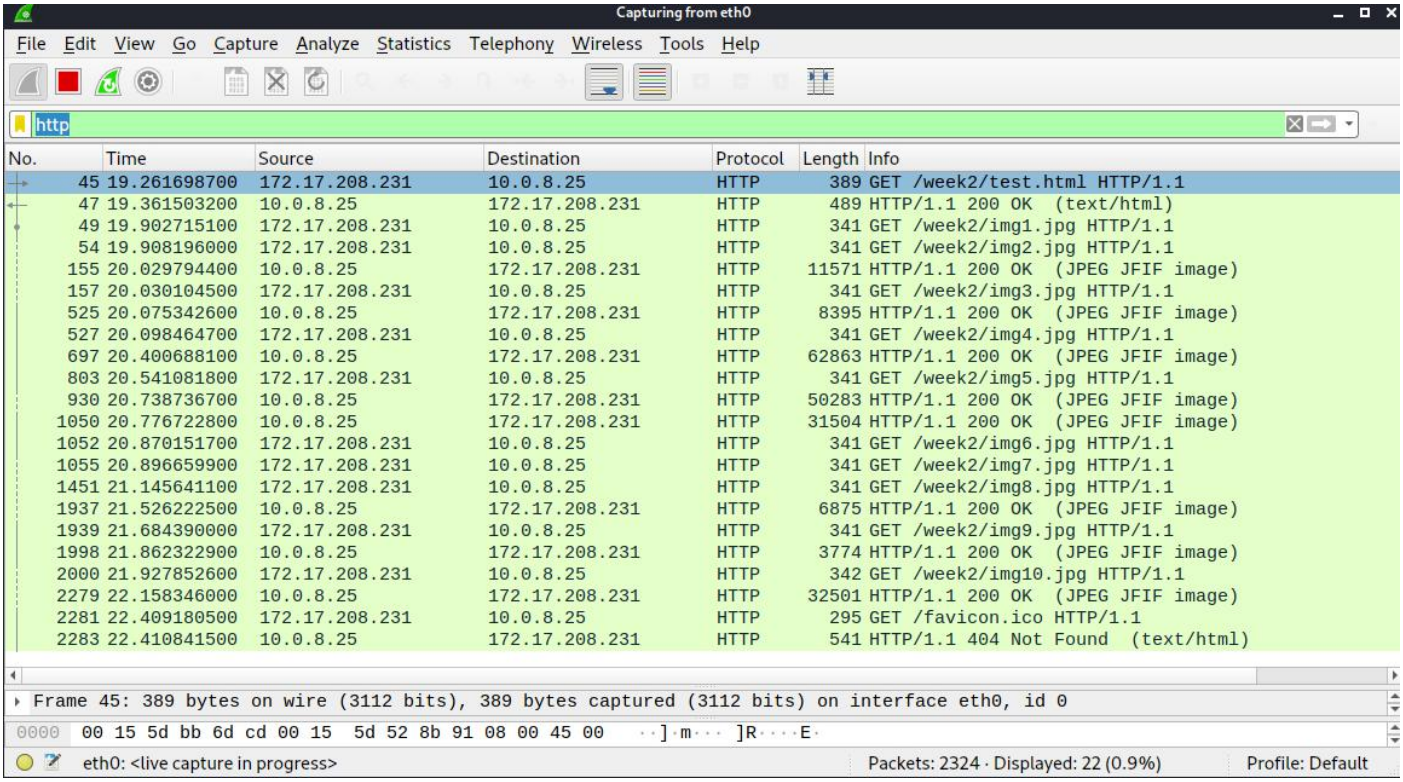
Here it is 51.583000700 - 48.495926500 = 3.0870742

PART 2: PERSISTENT CONNECTIONS

Step 1: For 2 persistent connections, set the value of **max-persistent-connection-per-server to 2** in the client computer.

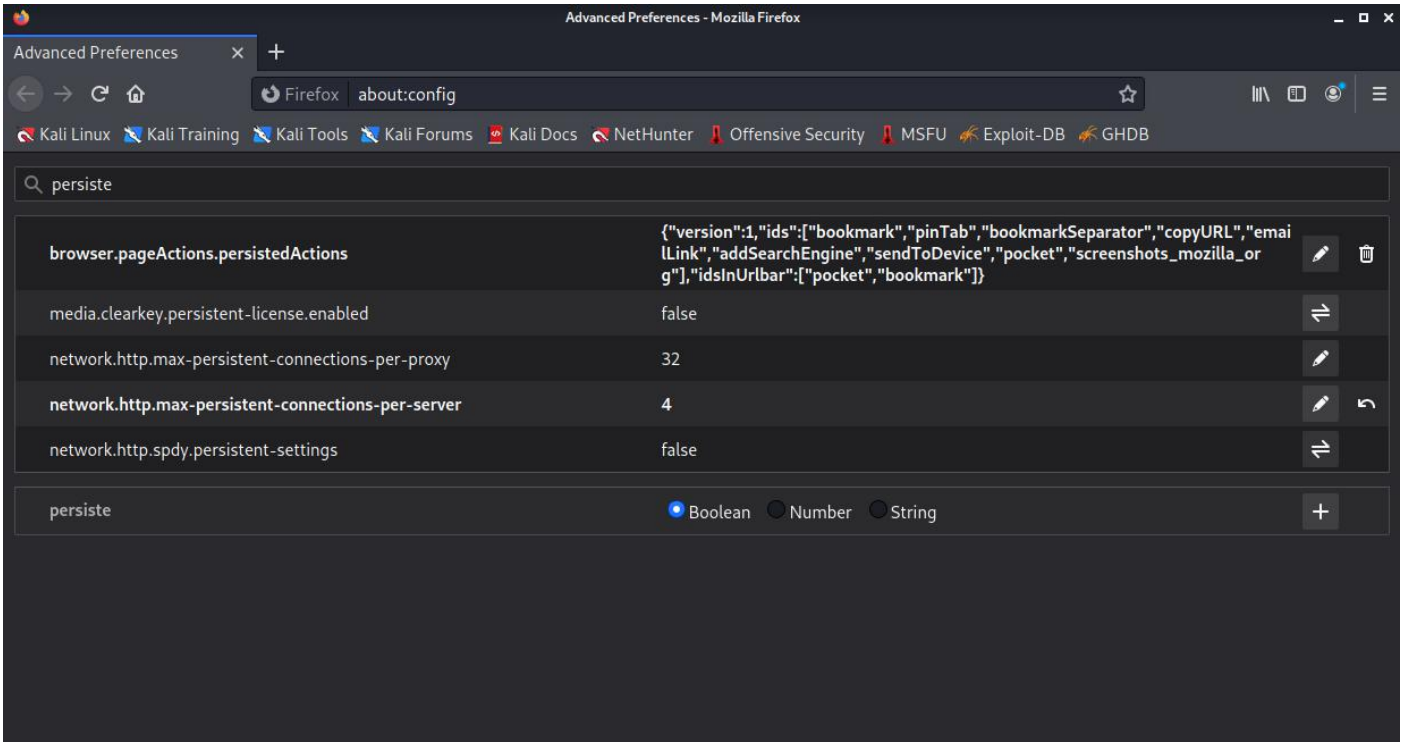


Step 2: Repeat the **steps 1-3** in the previous section.

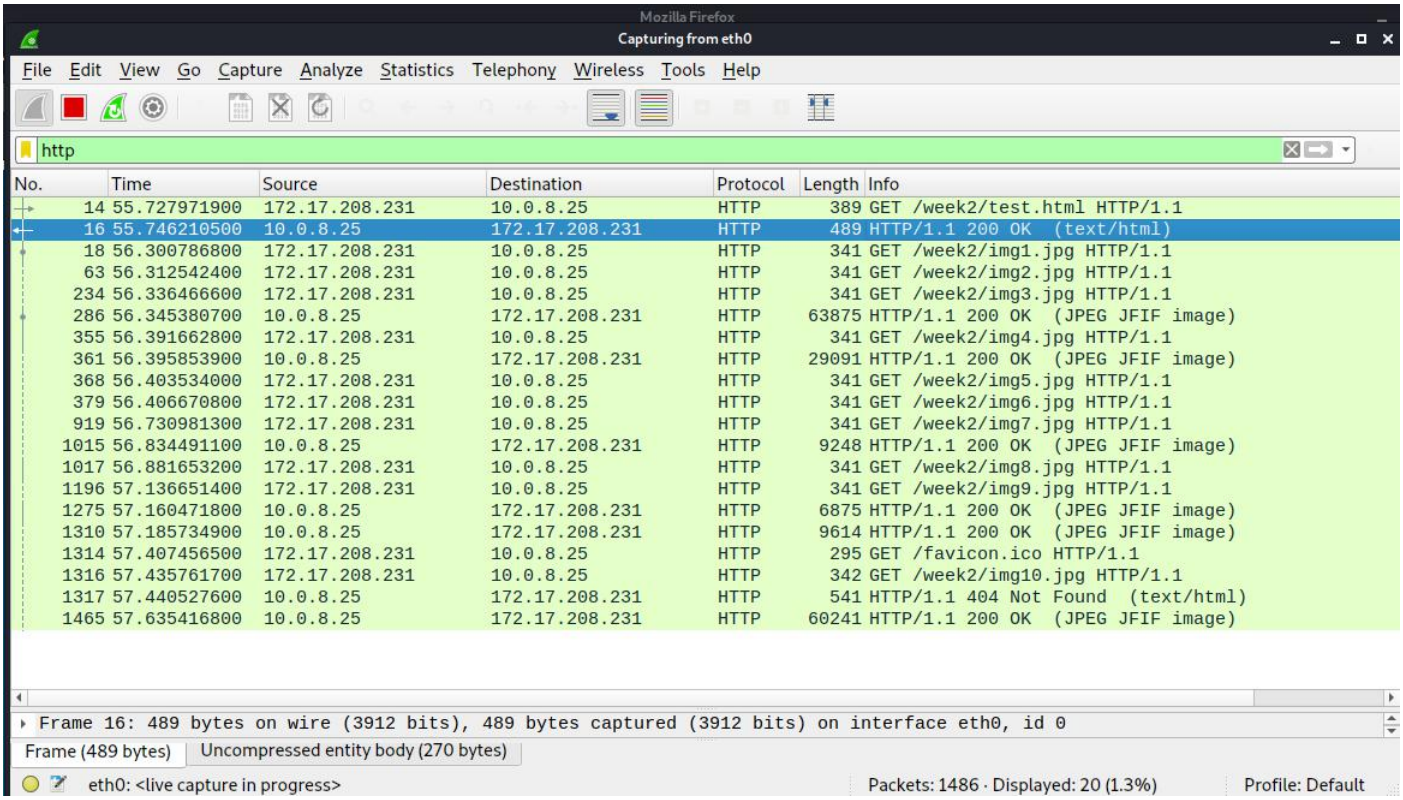


Here it is 22.158346000 – 19.361503200 = 2.7968428

Step 3: For 4 persistent connections, Set the value of **max-persistent-connection-per-server to 4** in the client computer.



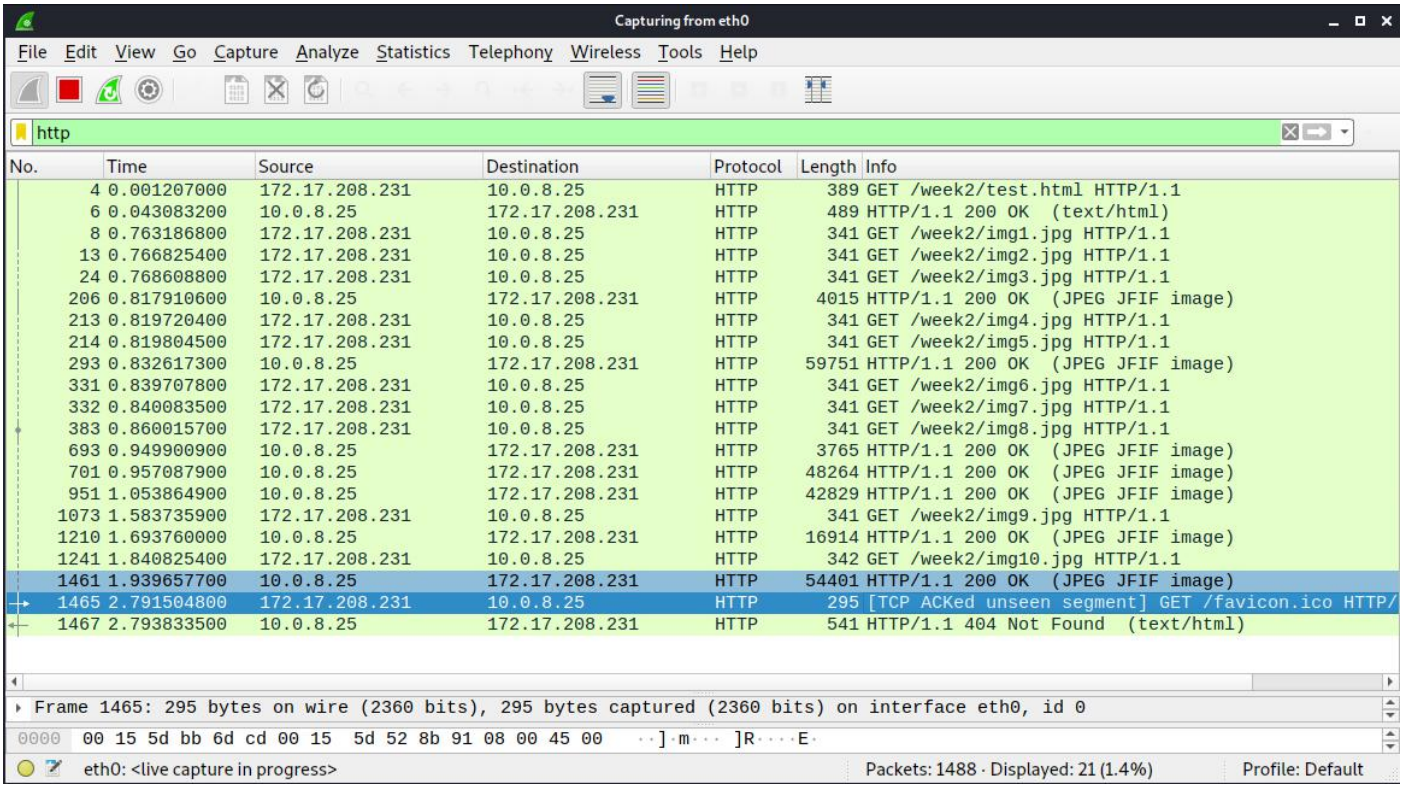
Step 4: Repeat the **steps 1-3** in the previous section



Here it is 57.635416800 – 55.746210500 = 1.8892063

Step 5: For **6 persistent connections**, set the value of **max-persistent-connection-per-server to 6** in the server computer.

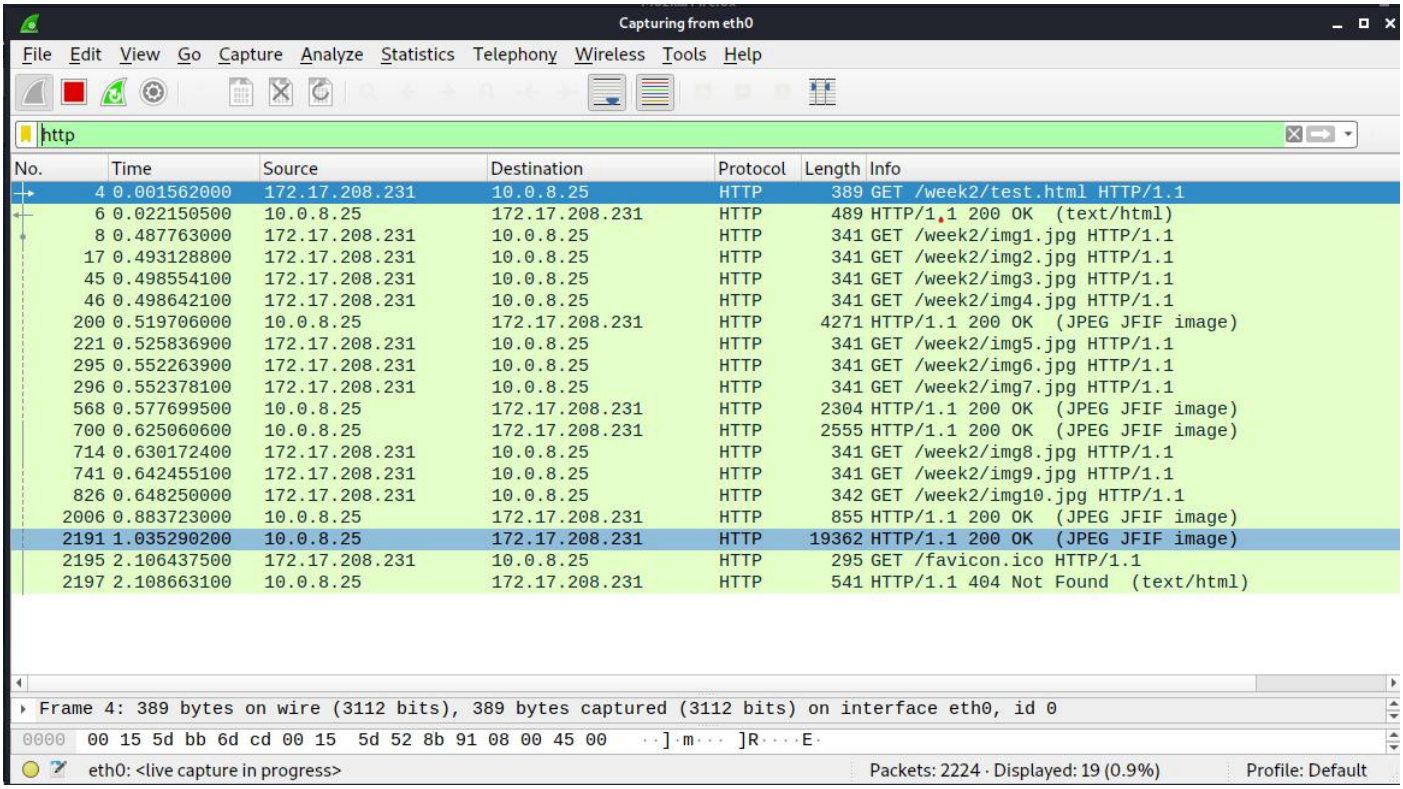
Step 6: Repeat the **steps 1-3** in the previous section



Here it is 1.939657700 – 0.043083200 = 1.8965745

Step 7: For **10 persistent connections**, set the value of **max-persistent-connection-per-server to 10** in the client computer.

Step 8: Repeat the **steps 1-3** in the previous section.



Here it is 1.035290200 – 0.0221505000 = 1.0131397

OBSERVATIONS REQUIRED ON EDMODO:

Calculate the time taken to load objects from the server for non-persistent and persistent connections (2, 4, 6 & 10).

Find out the optimal number of HTTP persistent connections based on your observations

.

Time taken to load objects from the server for non-persistent and persistent connections

- Non persistent connection

51.583000700 - 48.495926500 = 3.0870742

- 2 persistent connections

22.158346000 – 19.361503200 = 2.7968428

- 4 persistent connections

57.635416800 – 55.746210500 = 1.8892063

- 6 persistent connections

1.939657700 – 0.043083200 = 1.8965745

- 10 persistent connections

1.035290200 – 0.0221505000 = 1.0131397

The optimal number of HTTP persistent connections based observations

Ans :

10 persistent connections is the optimal because it took less time to get all the objects/images From a http server and it is not necessary to always set 10 persistent connections,
we have to set minimum of 2 to 10 persistent connections , or else Sometimes it may get overloaded.

Time to taken by 10 persistent connections

1.035290200 – 0.0221505000 = 1.0131397

B) Understand working of HTTP Headers

->Understand working of HTTP headers:

->Conditional Get: If-Modified-Since

->HTTP Cookies: Cookie and Set-Cookie

The three parts of experiment are:

1.Password Authentication

2. Cookie Setting

3. Conditional get

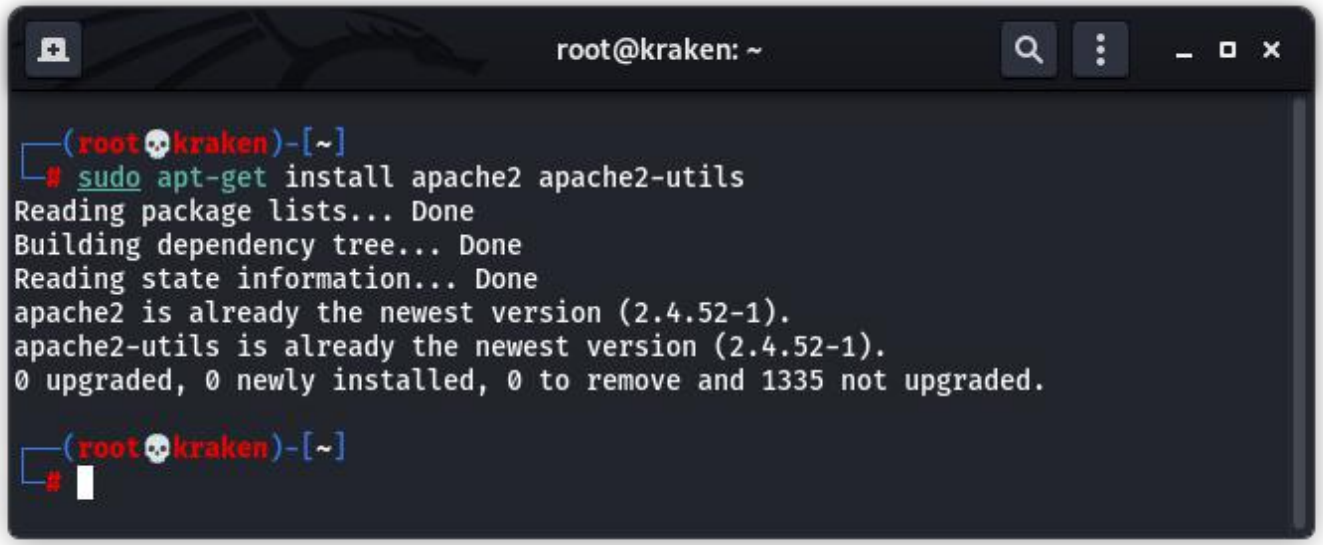
1.Password Authentication

Steps of Execution (for Password Authentication)

1. Executing the below commands on the terminal.

--> To install the apache utility

sudo apt-get install apache2 apache2-utils

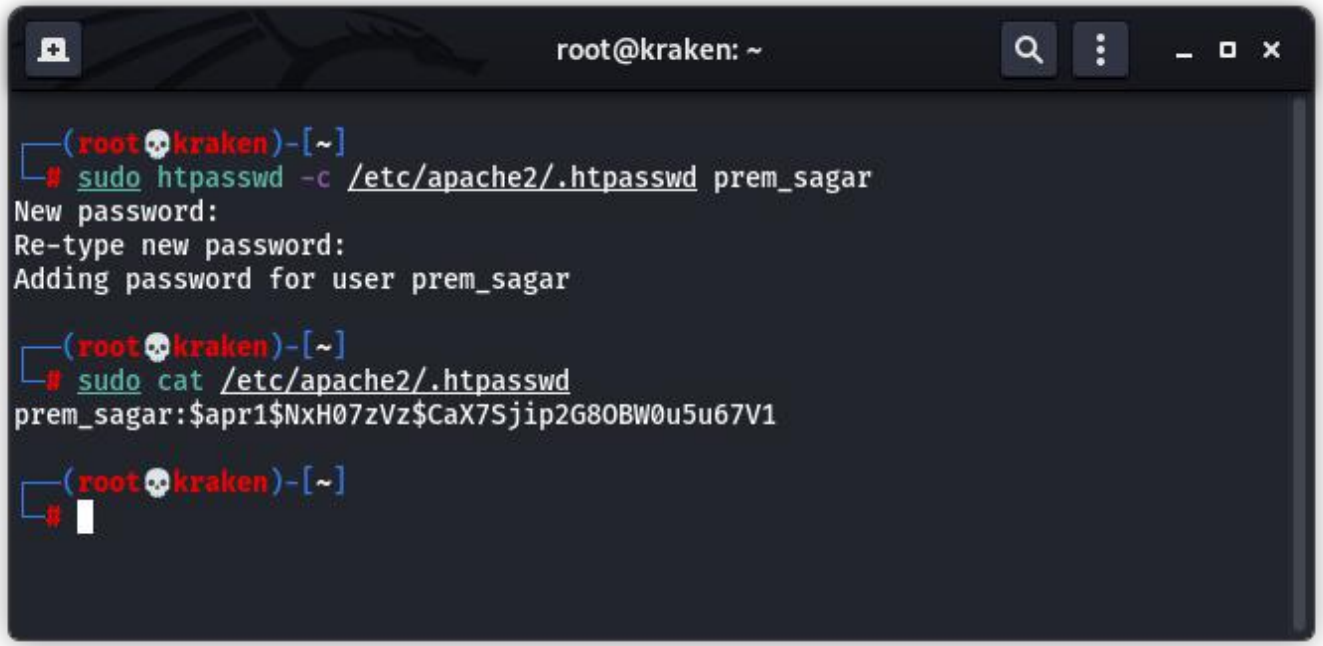


--> Provide username and password to set authentication

sudo htpasswd -c /etc/apache2/.htpasswd prem_sagar

--> View the authentication

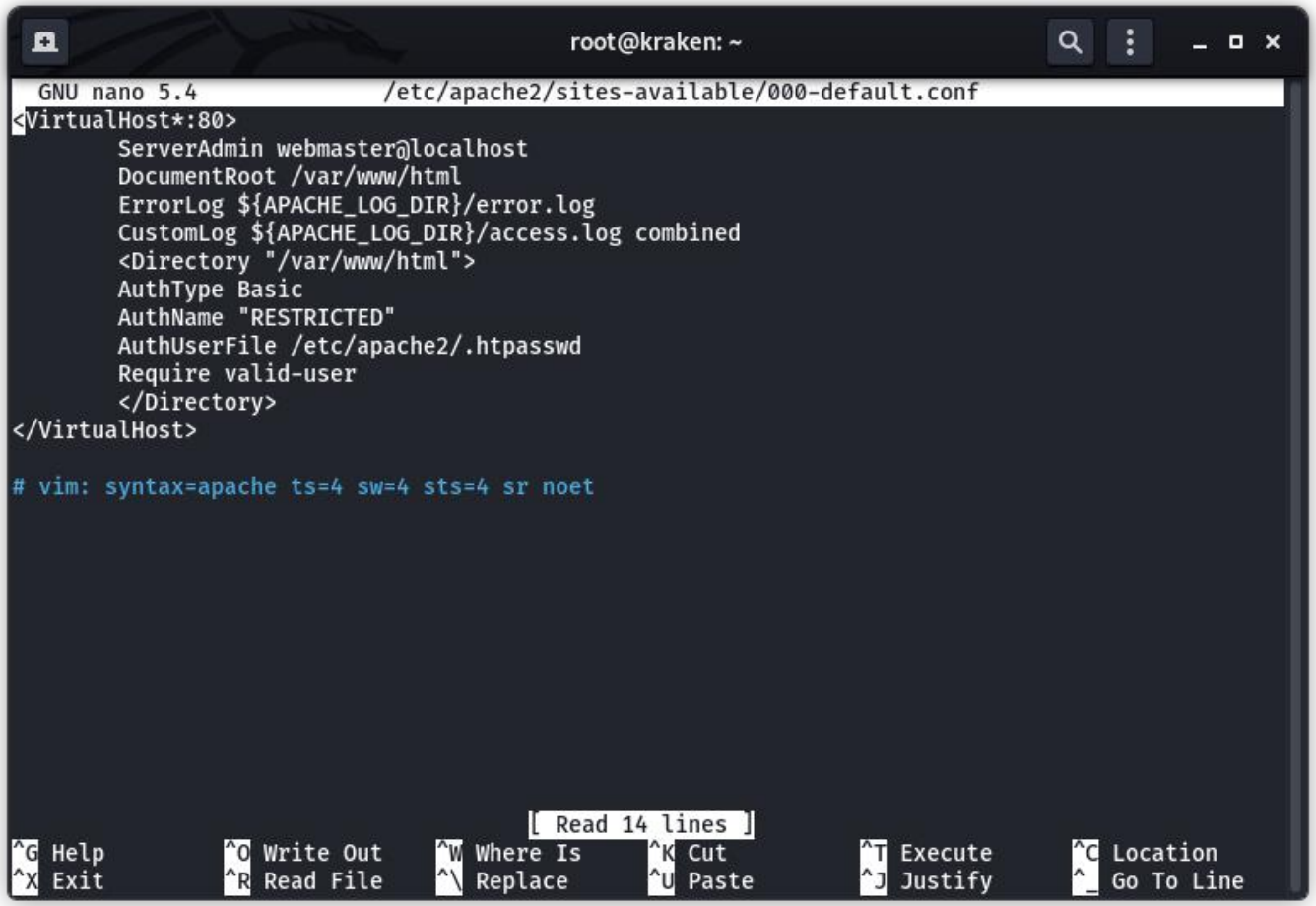
sudo cat
/etc/apache2/.htpasswd



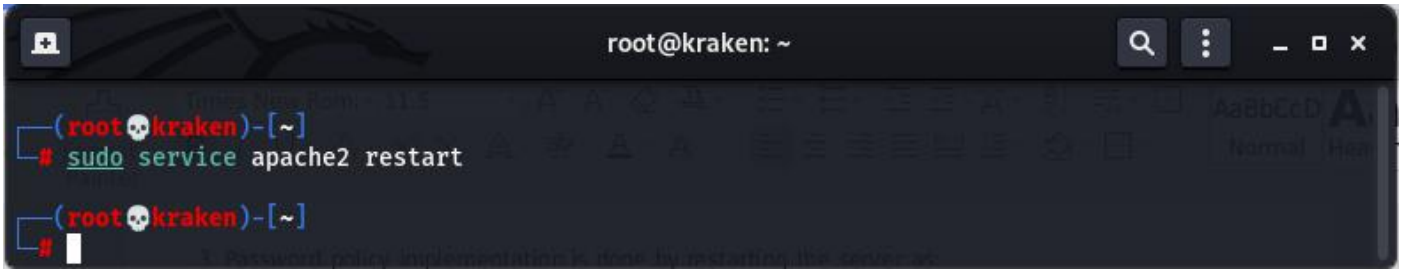
Here “prem_sagar” is the username. Also, password is entered twice.

2. To setup the authentication phase, execute the following commands. Configuring Access control within the Virtual Host Definition.

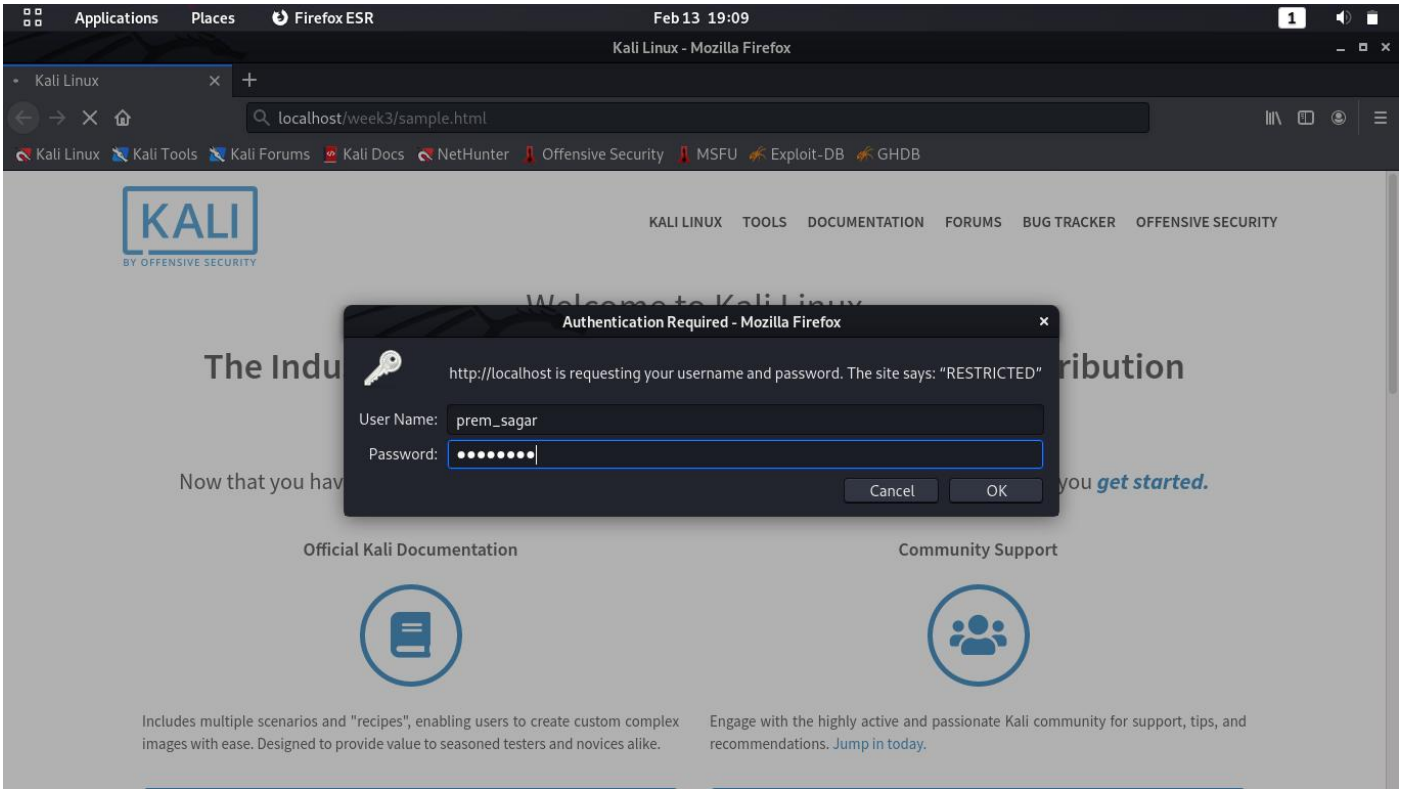
--> Opening the file for setting authentication
sudo nano /etc/apache2/sites-available/000-default.conf



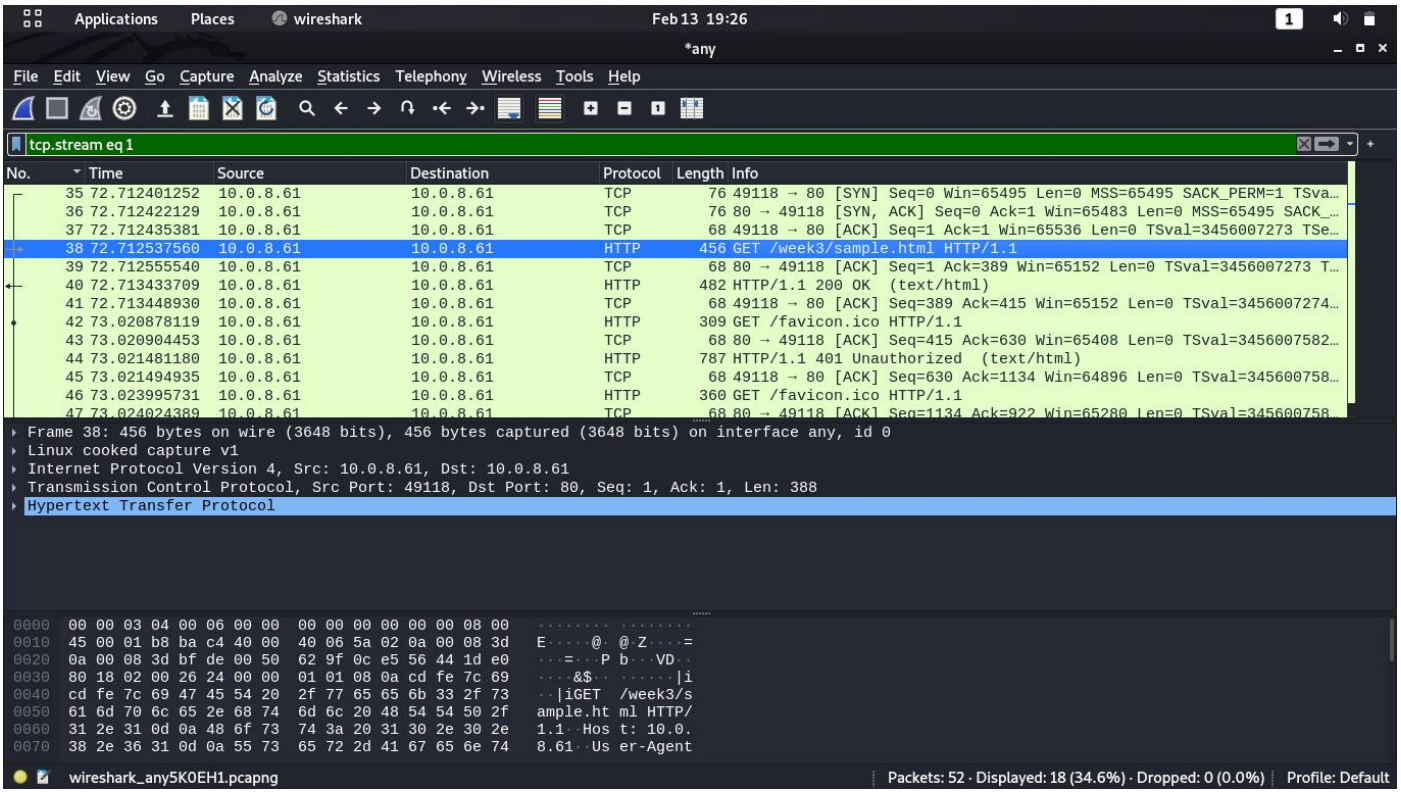
3. Password policy implementation is done by restarting the server as:
sudo service apache2 restart



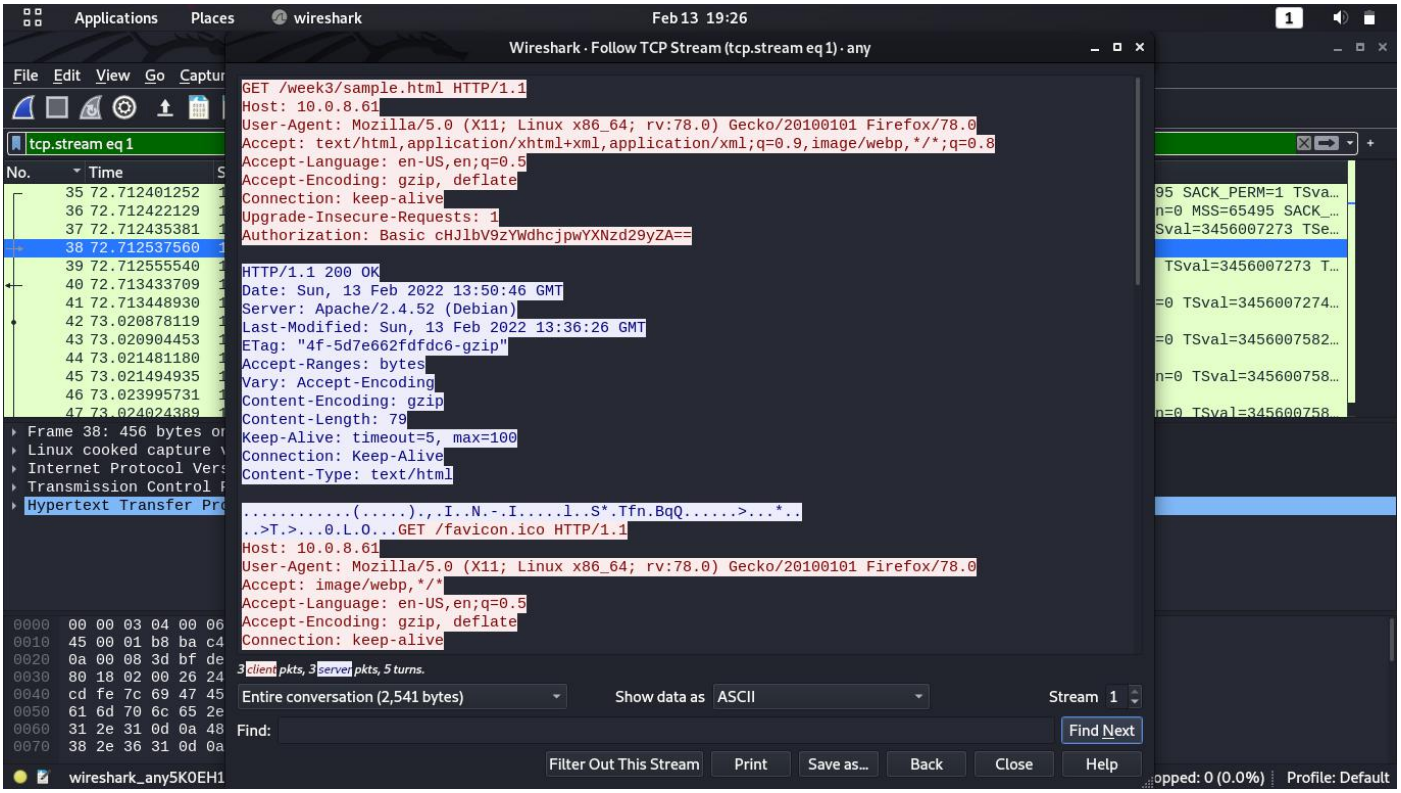
4. The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase

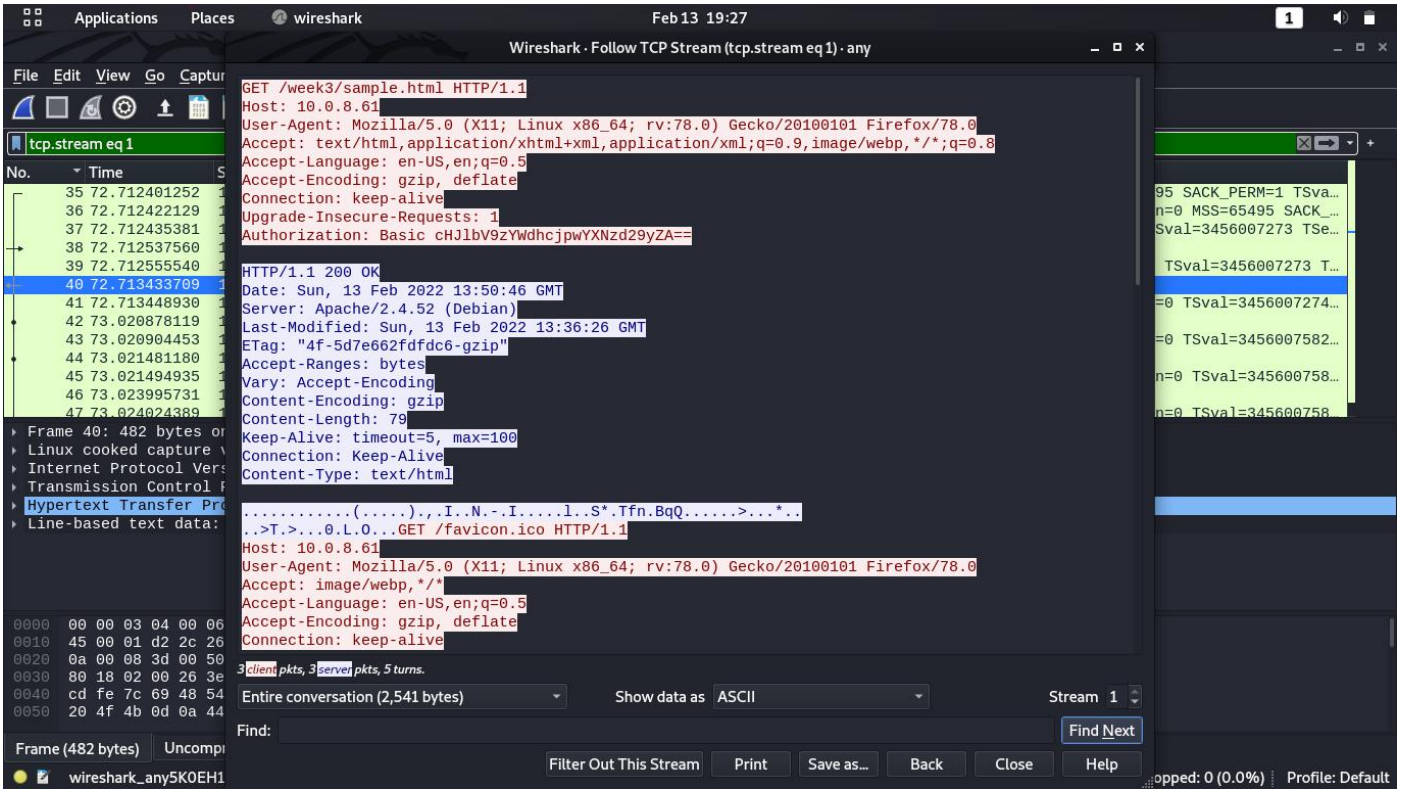


5. Wireshark is used to capture the packets sent upon the network.



6. Using the “follow TCP stream” on the HTTP message segment the password was retrieved which was encrypted by the base64 algorithm and decryption could be done with same algorithm.





2. Cookie Setting

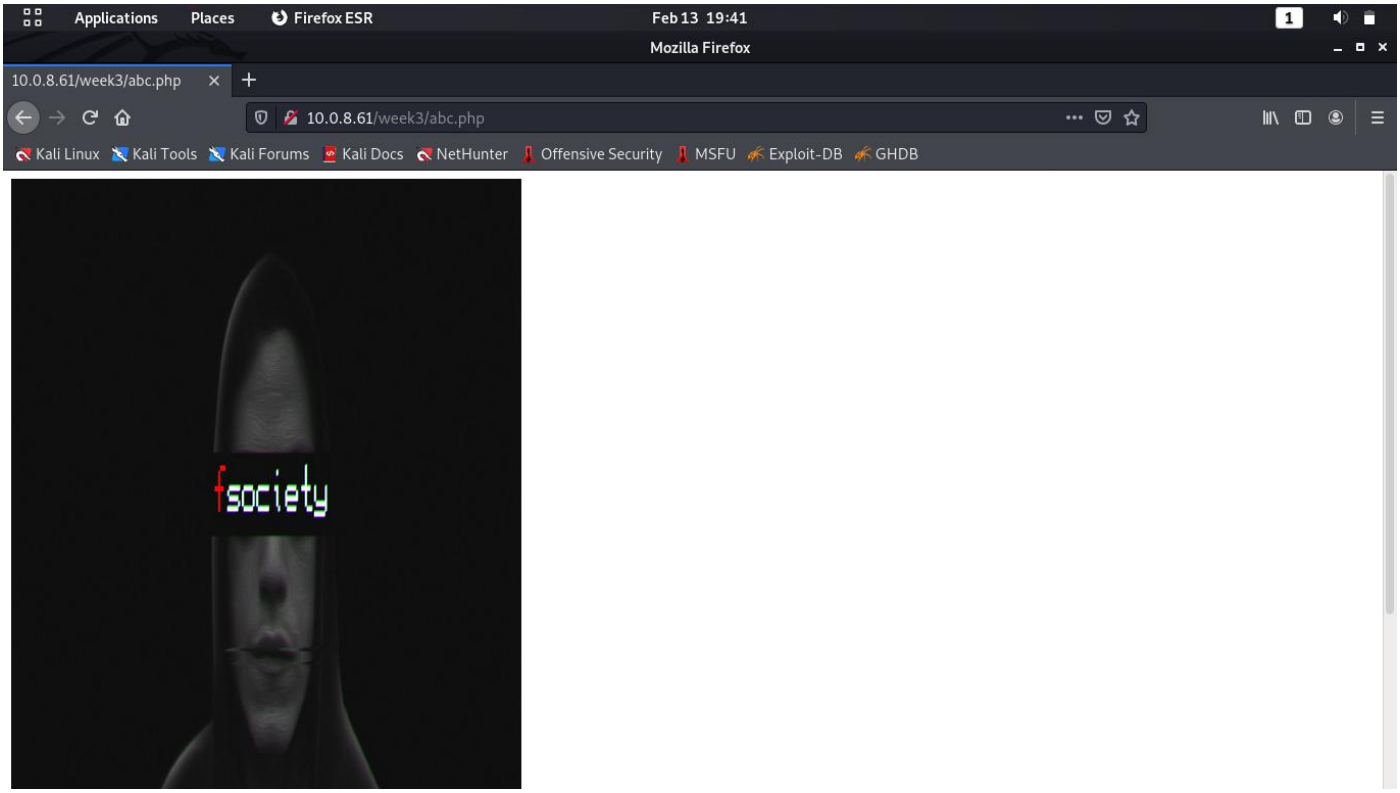
Steps of Execution (Cookie Setting)

1. A PHP file to set the cookie is created which also contains an image in it (placed under the HTML directory) to be accessed once the cookie is set. The following code helped to set the cookie:

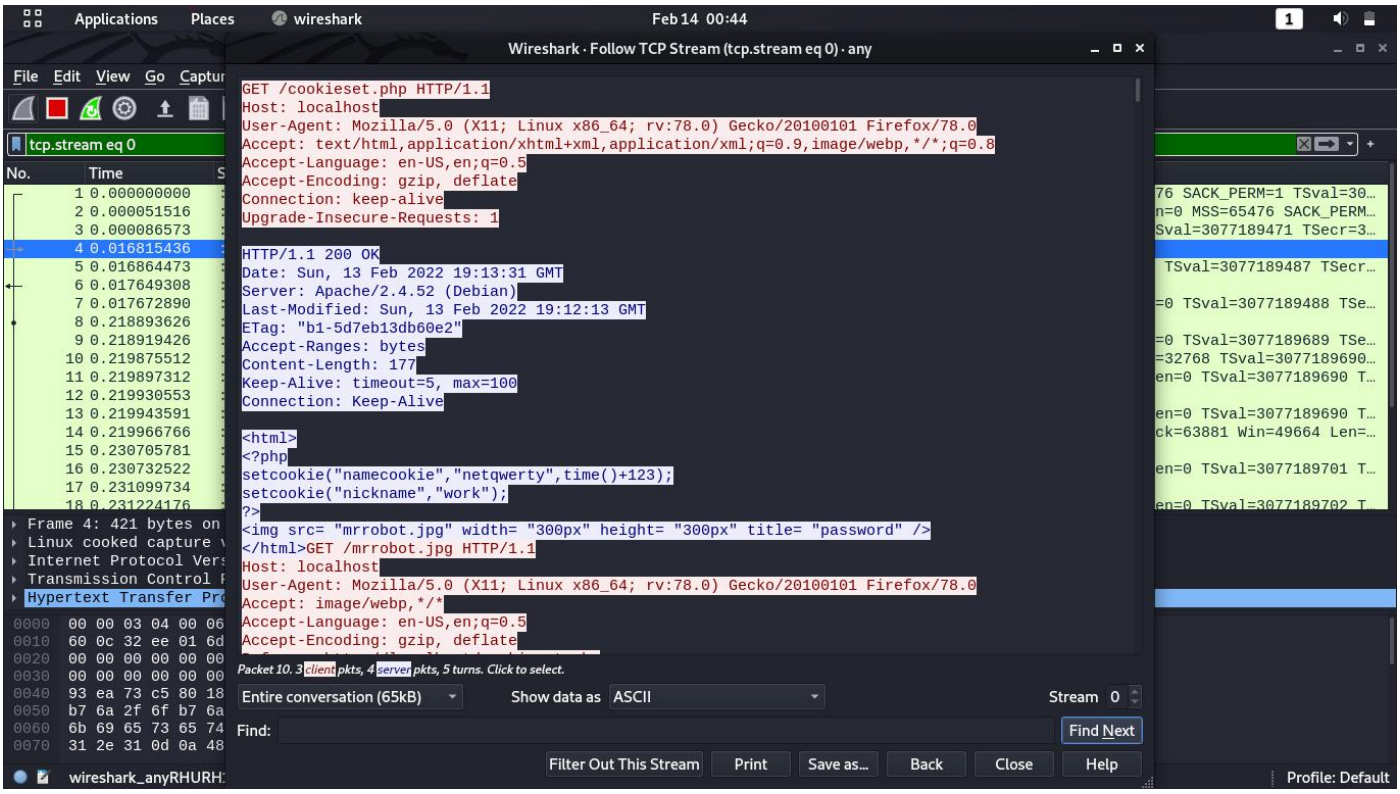
```
root@kraken: /var/www/html/week3
GNU nano 5.4                    abc.php
<html>
<?phpsetcookie("namecookie","netqwerty",time()+123);
setcookie("nickname","work");
?>
<img src= “mrrobot” width= “500” height= “800” title= “password” />
</html>

                    [ Read 6 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute     ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify     ^_ Go To Line
```

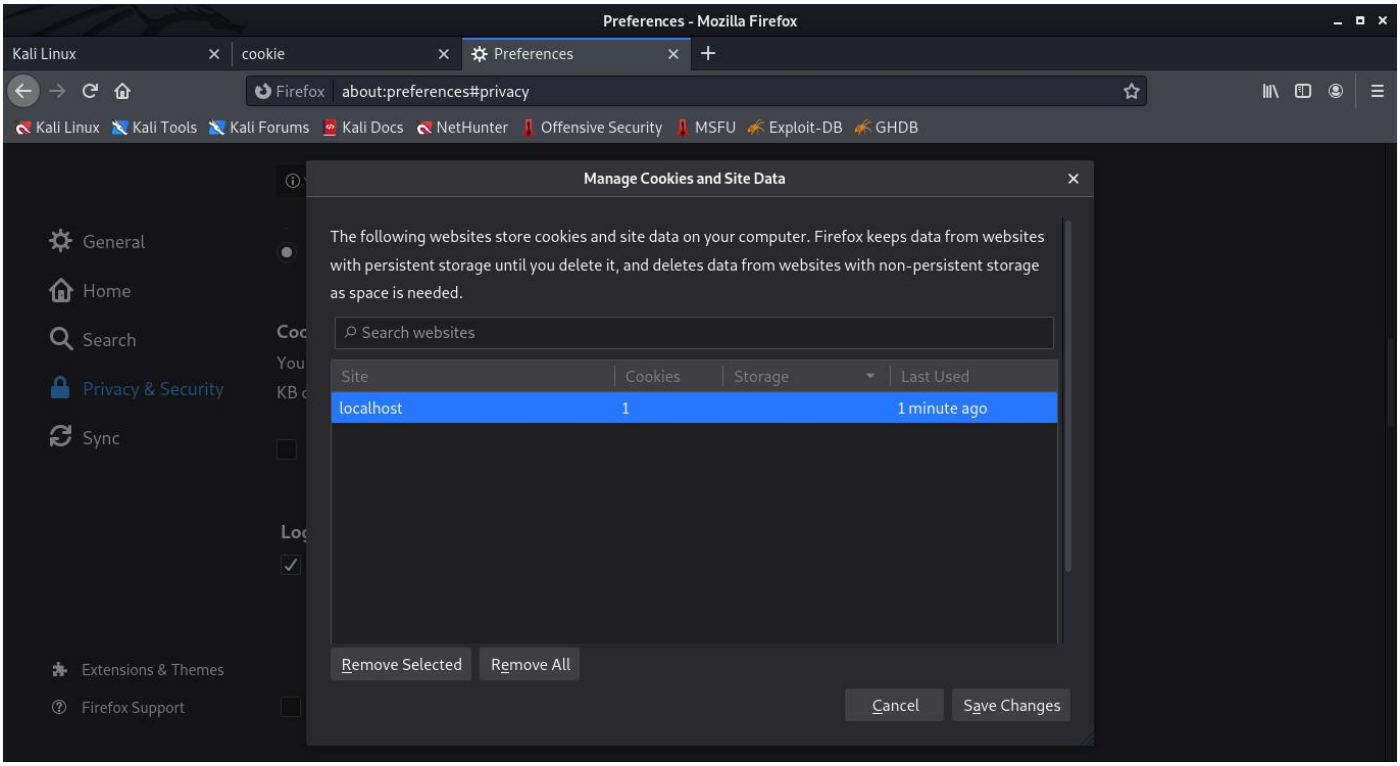
2. The combined file saved with a .php extension is placed under **/var/www/html** for accessing.



3. The packets are captured using Wireshark and using the “follow TCP stream” which checks for the set-cookie field whether the cookie is set or not set.



Could not able to capture cookie on wireshark.

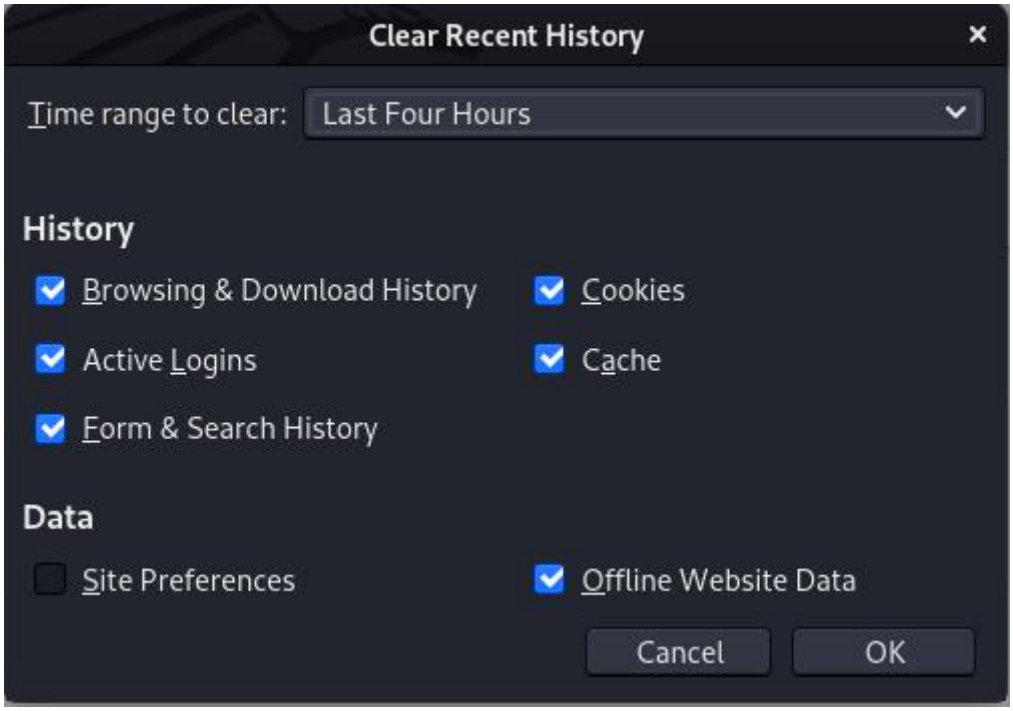


The cookie is set as shown in the above screenshot.

3. Conditional get

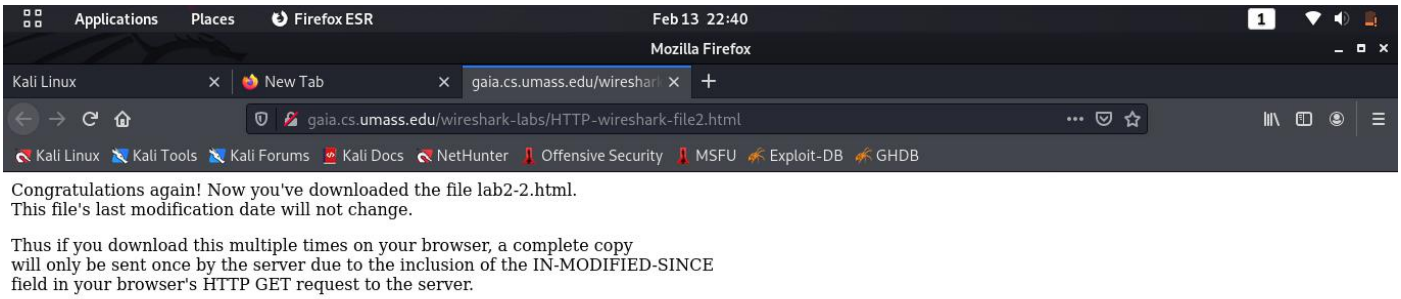
Conditional Get: If-Modified-Since

Before performing the steps below, make sure your browser's cache is empty. (To do this under Firefox, select Tools -> Clear Recent History and check the Cache box). Now do the following:



Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

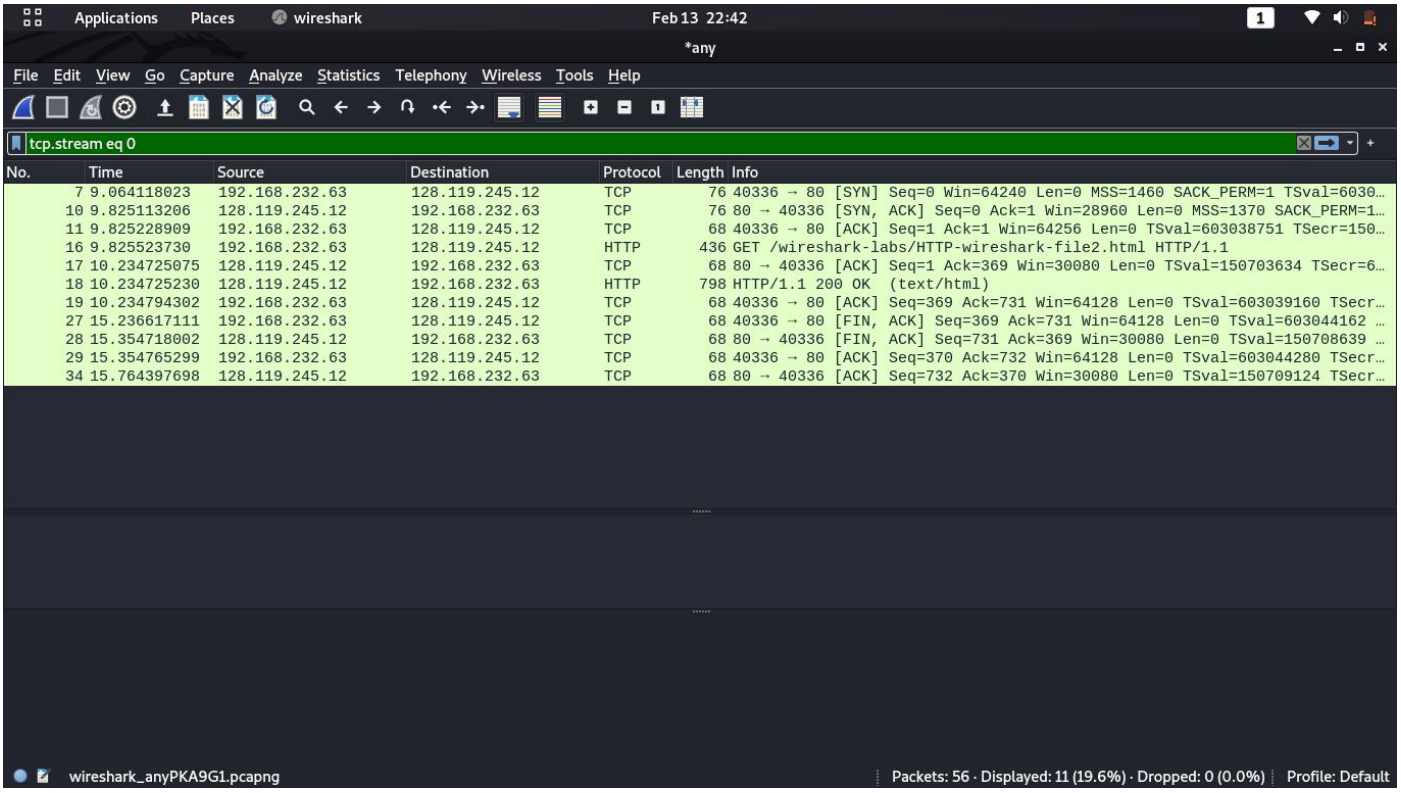
➢ Your browser should display a very simple five-line HTML file.



Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)

(Refreshed)

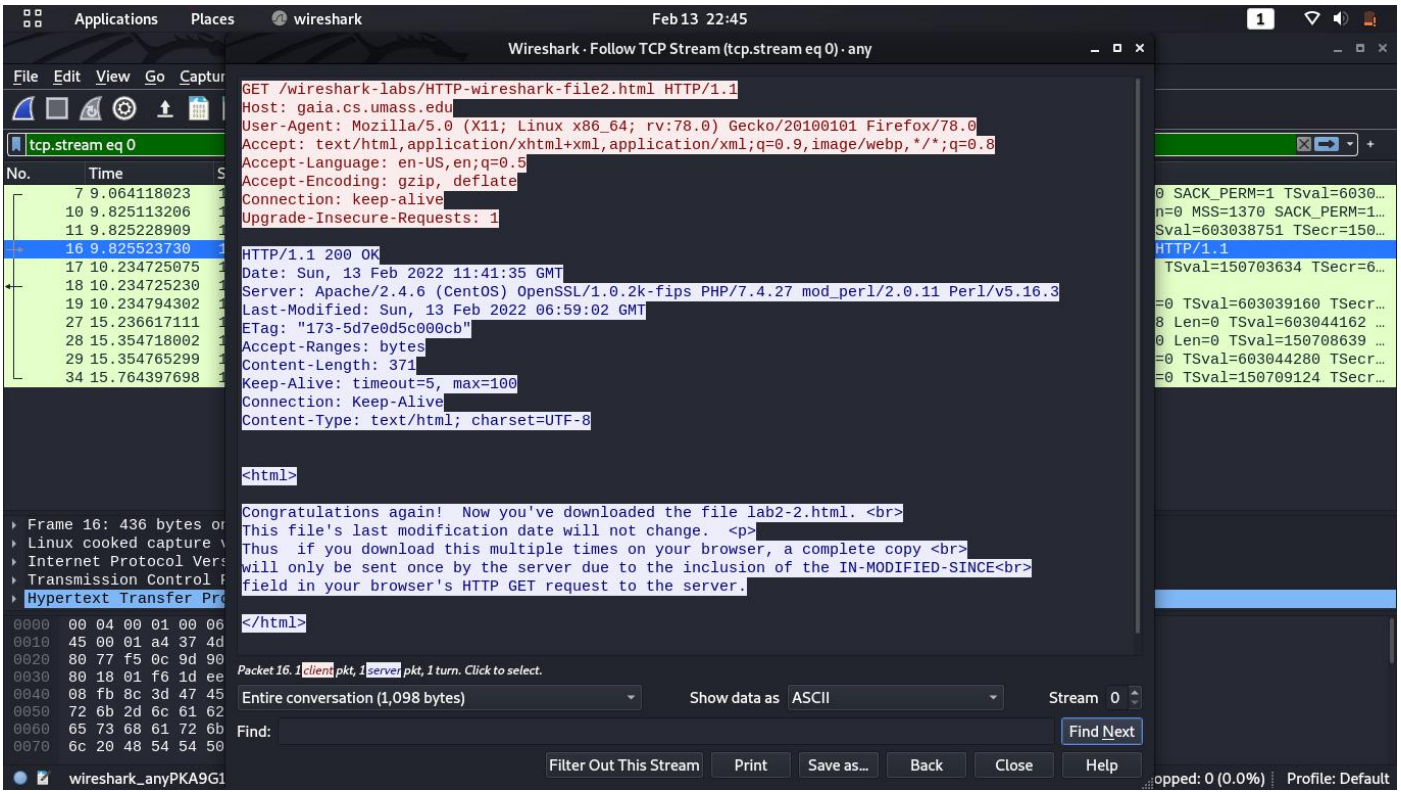
Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.



Observations:

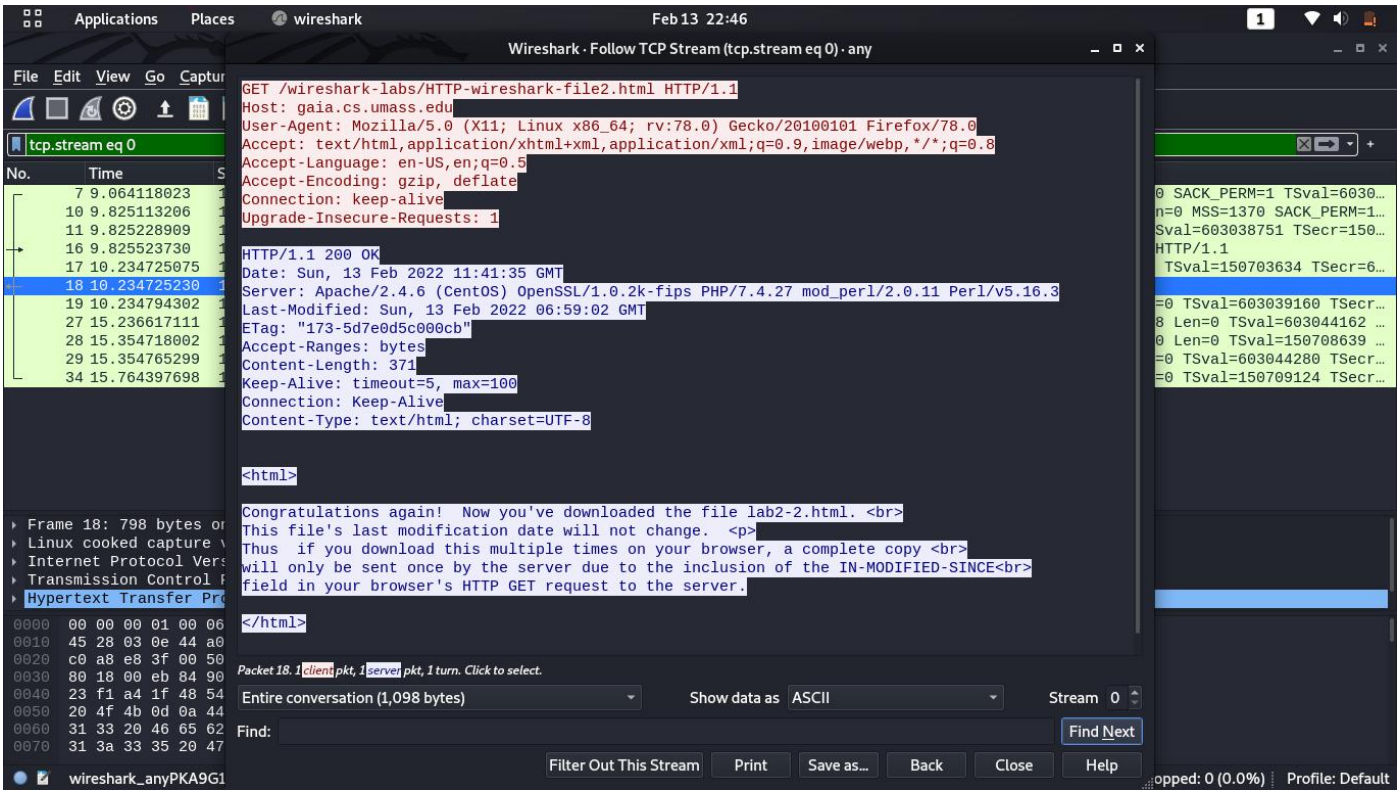
✓ Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

Ans : Yes I see an "IF-MODIFIED-SINCE:" line in the HTTP GET



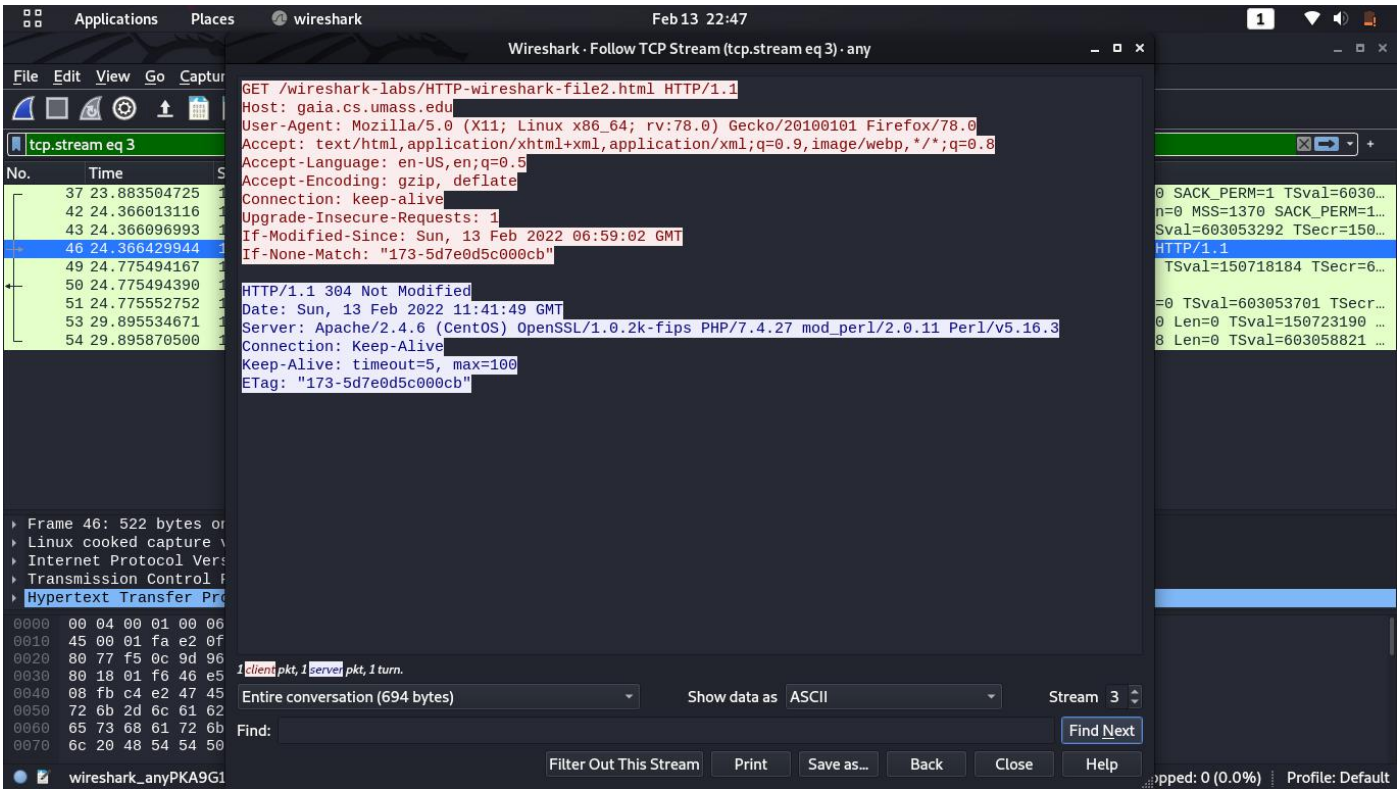
✓ Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

Ans : the server explicitly return the contents of the file , because client is receiving some text data



✓ 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?

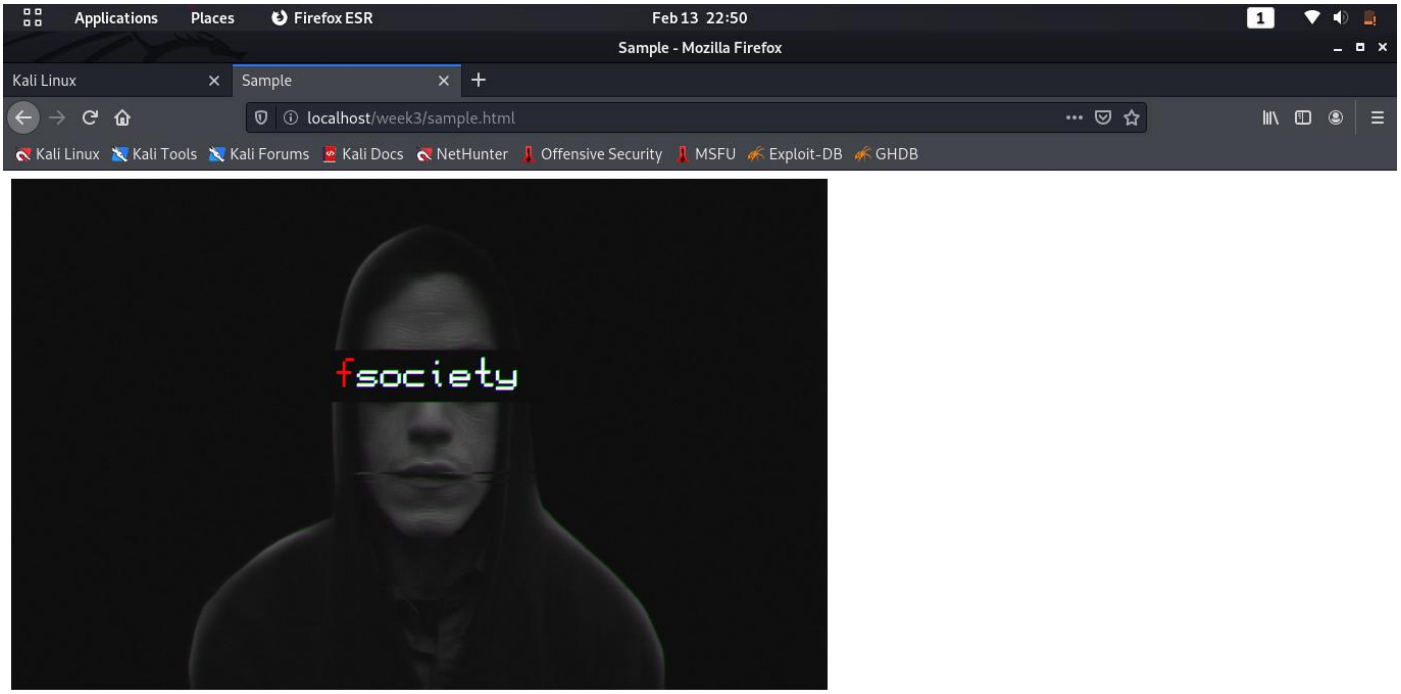
Ans: No



✓ 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.

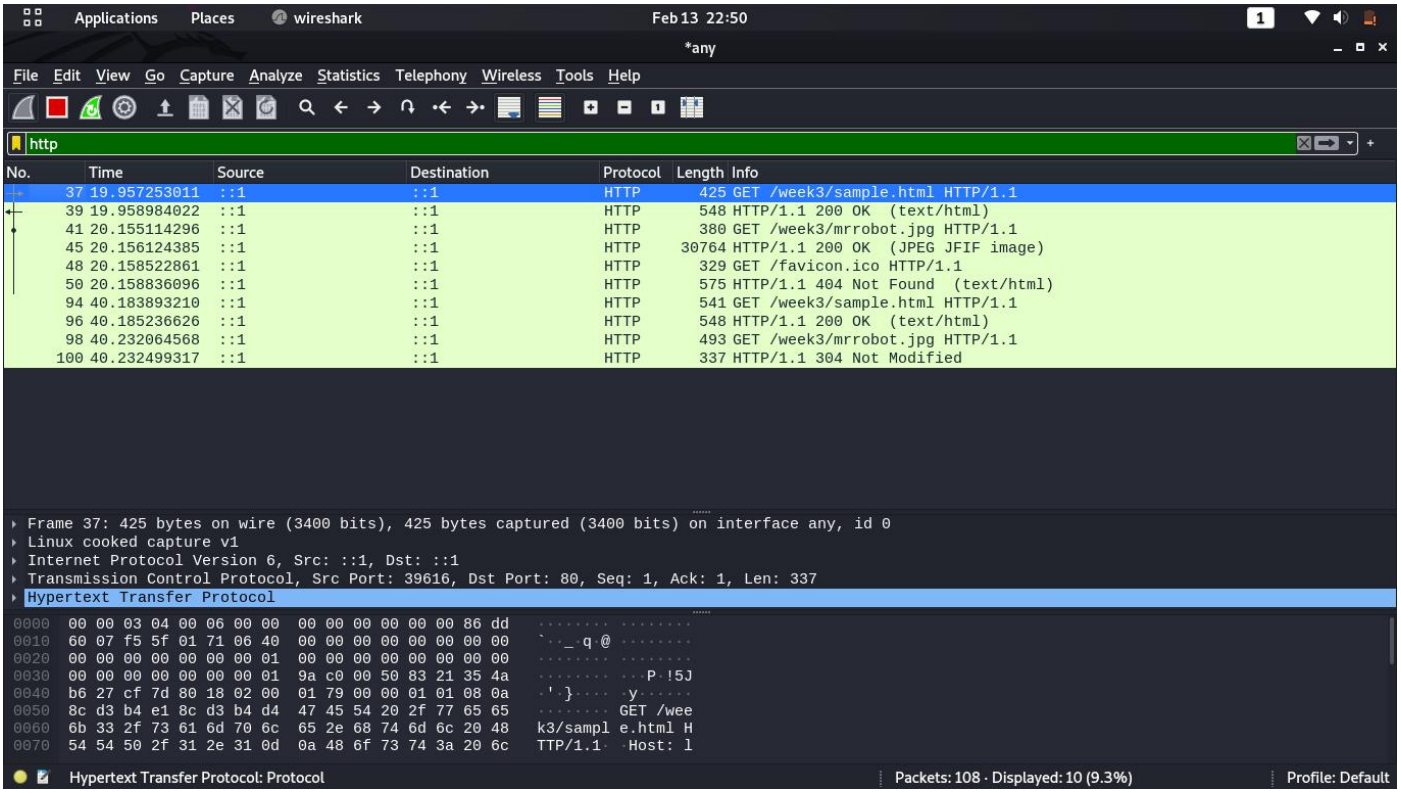
Ans : NO , because the contents are already downloaded in the browsers cache , so it'll just get the data from cache If it hasn't modified . (status code : 304 NOT MODIFIED)

Repeat the above task with some images on the server.

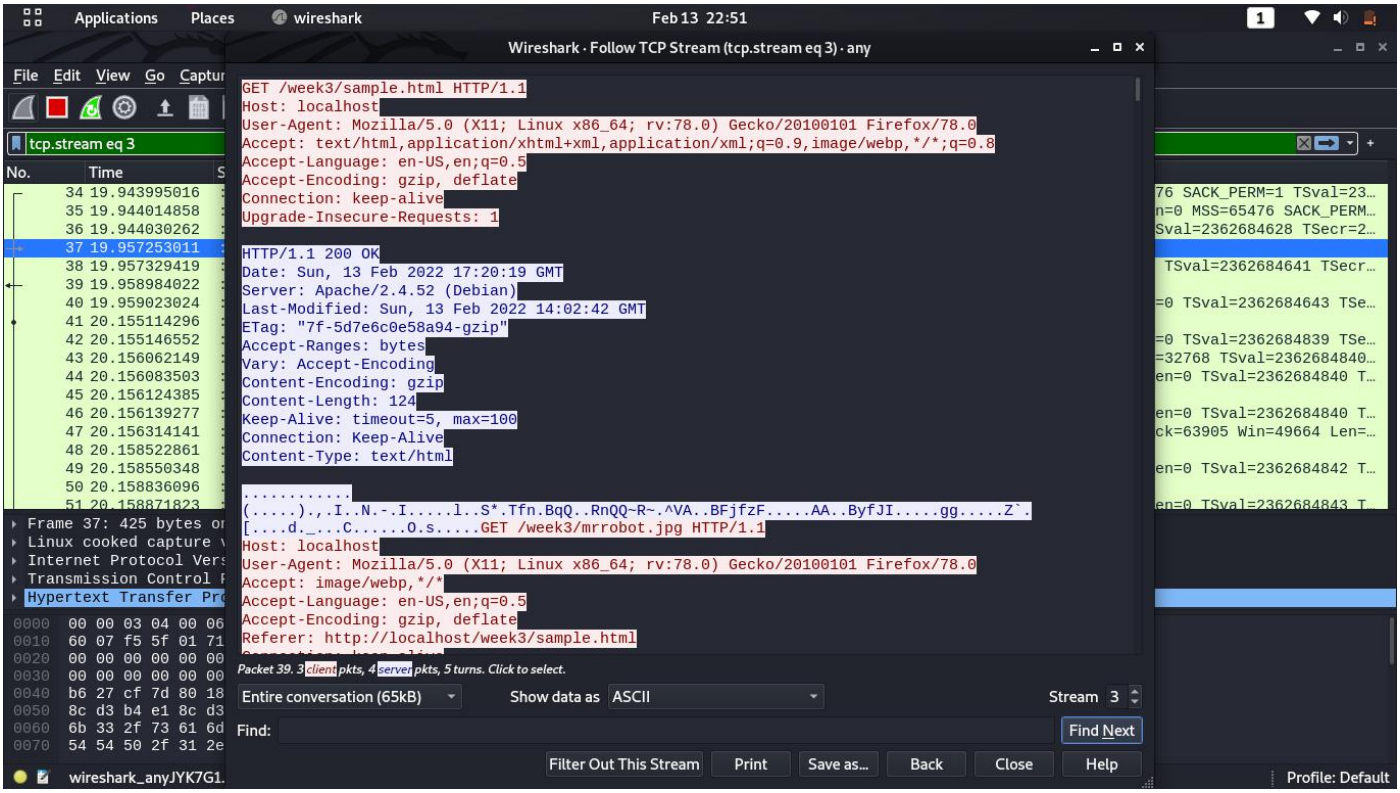


Mr.Robot

Stop Wireshark packet capture, and enter "http" in the display-filter-specification window



Inspect the contents of the first HTTP GET request from your browser to the server and
Inspect the contents of the server response



Now inspect the contents of the second HTTP GET request from your browser to the server
And
What is the HTTP status code and phrase returned from the server in response to this second HTTP GET