

# 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

Ubuntu (vm)

Client

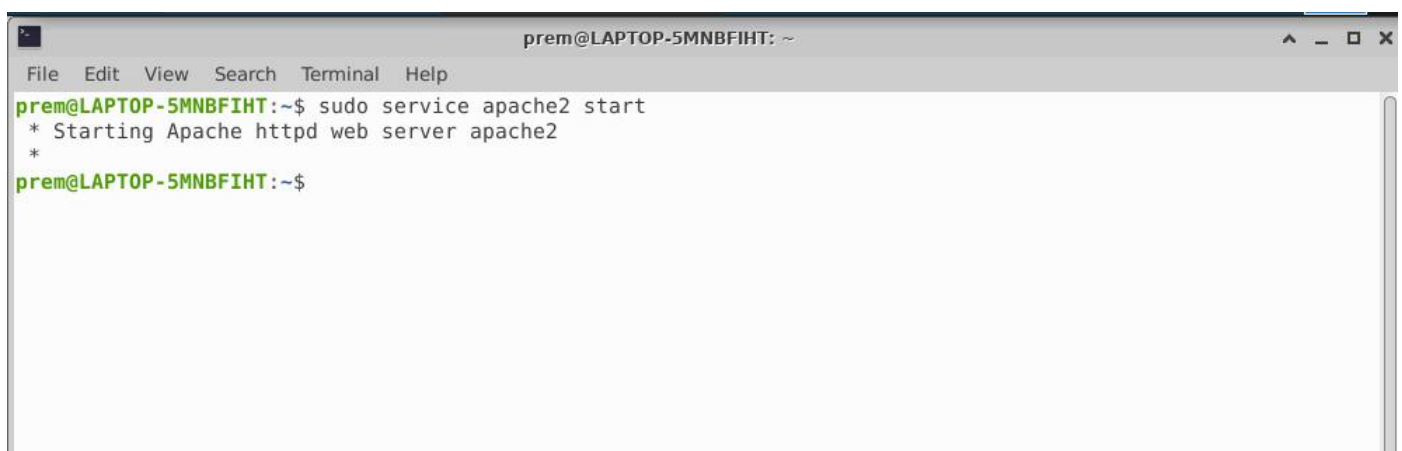
kali linux (vm)

**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.



```
prem@LAPTOP-5MNBFIHT: ~  
File Edit View Search Terminal Help  
prem@LAPTOP-5MNBFIHT:~$ sudo service apache2 start  
* Starting Apache httpd web server apache2  
*  
prem@LAPTOP-5MNBFIHT:~$
```

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

**\$sudo ip addr**

```
premi@LAPTOP-5MNBFIHT: ~  
File Edit View Search Terminal Help  
premi@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**

```
premi@LAPTOP-5MNBFIHT: ~
File Edit View Search Terminal Help
GNU nano 4.8 /etc/apache2/apache2.conf
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

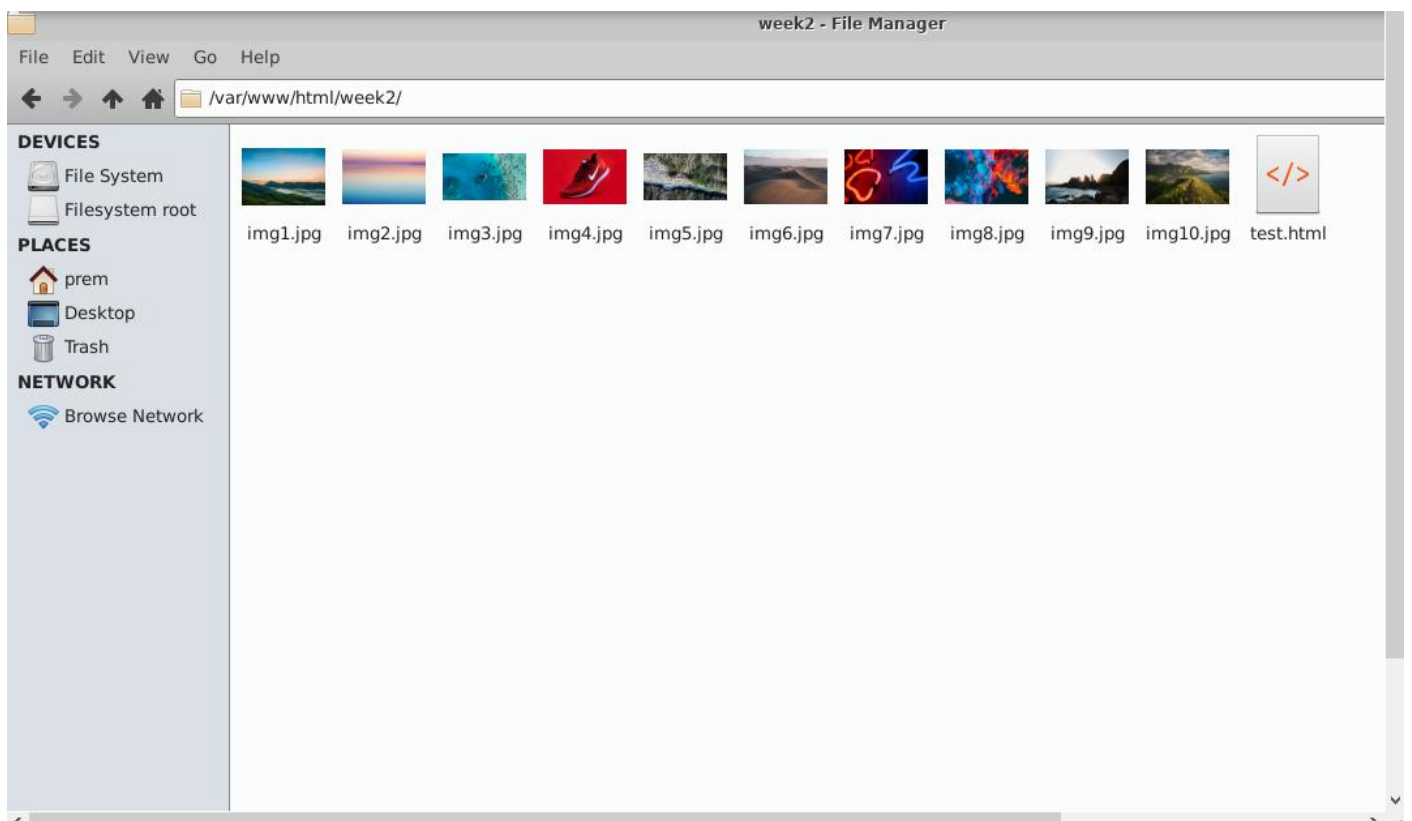
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5

# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line    M-E Redo
```

**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>










</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

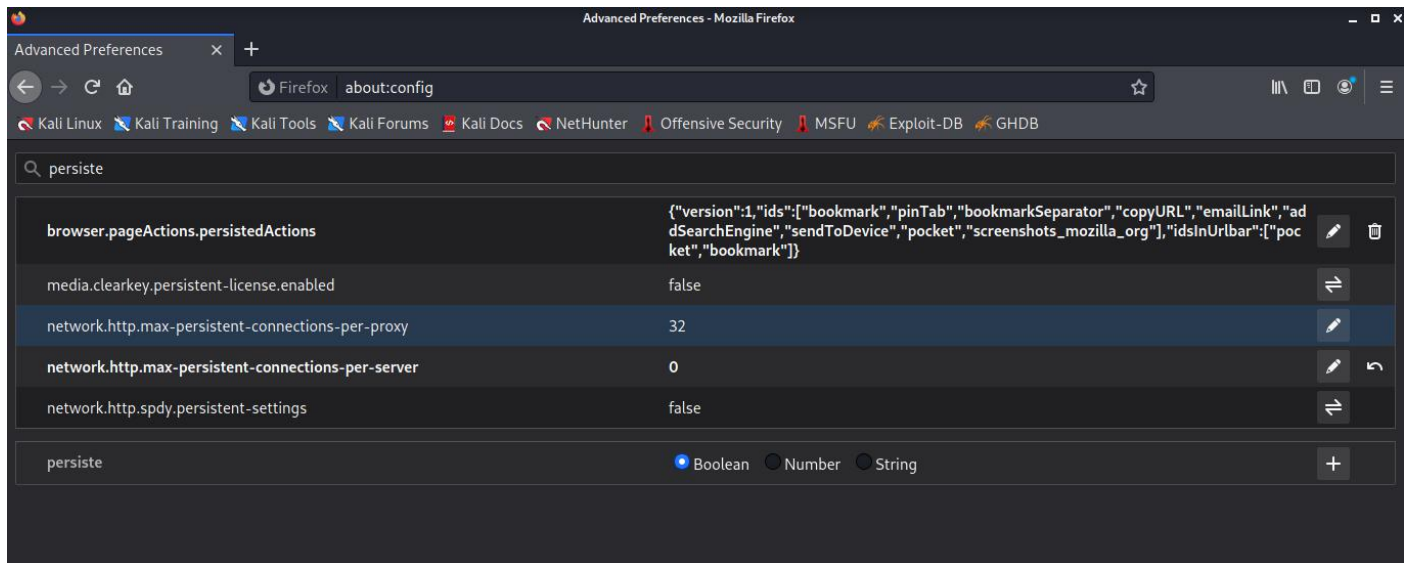
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 33391 bytes 76972933 (73.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 33391 bytes 76972933 (73.4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



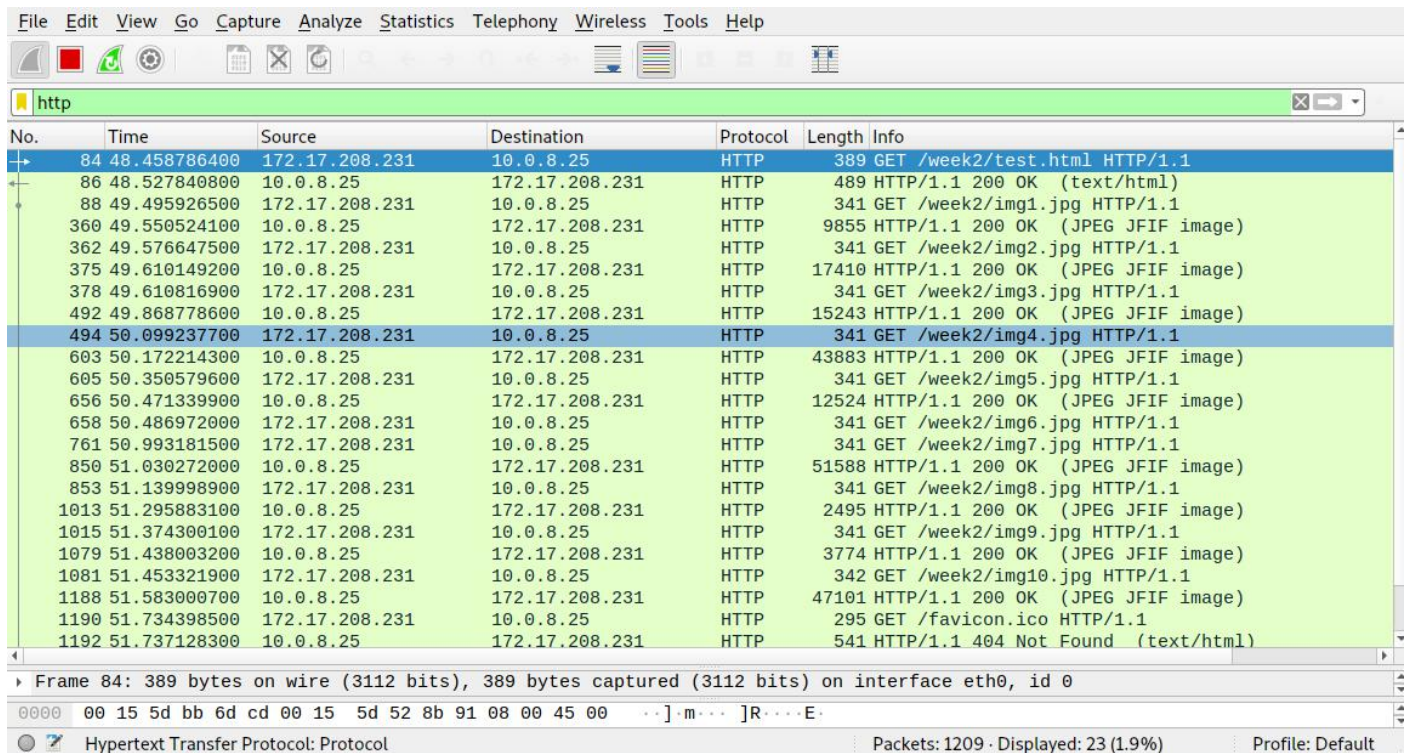
**Step 2:** Access web page on client-side browser (Firefox)

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`

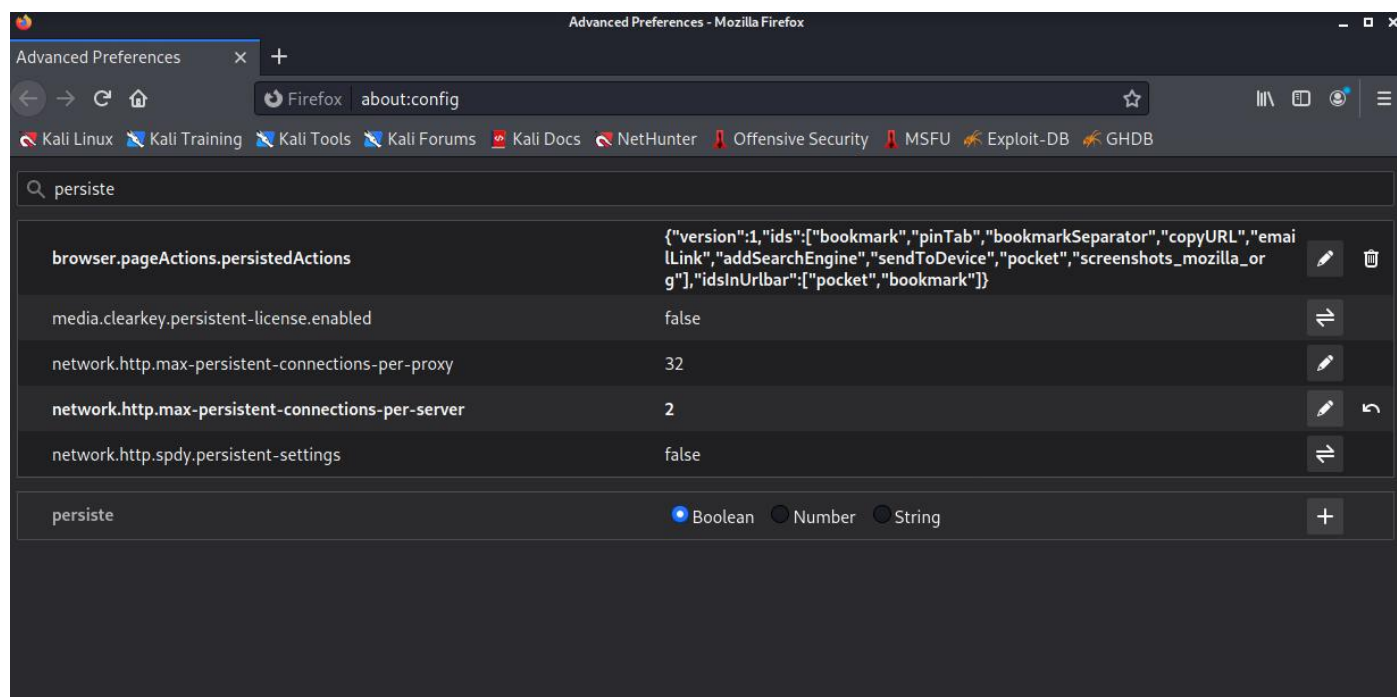
**Step 3:** Use Wireshark. Open Wireshark in the server computer while client is trying to access the server's local host webpage. Apply 'http' filter and note the time to capture all the 10 images.



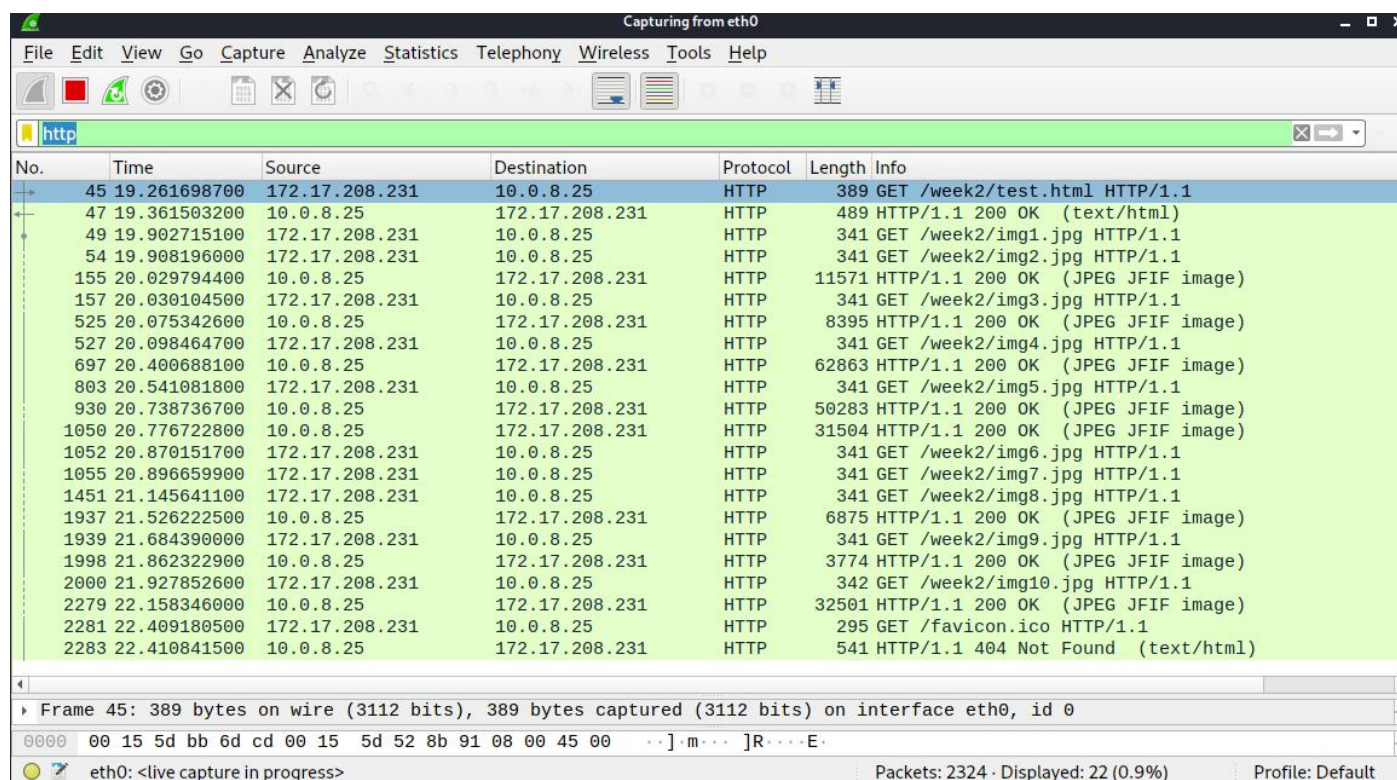
Here it is `51.583000700 - 48.495926500 = 3.0870742`

## 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.



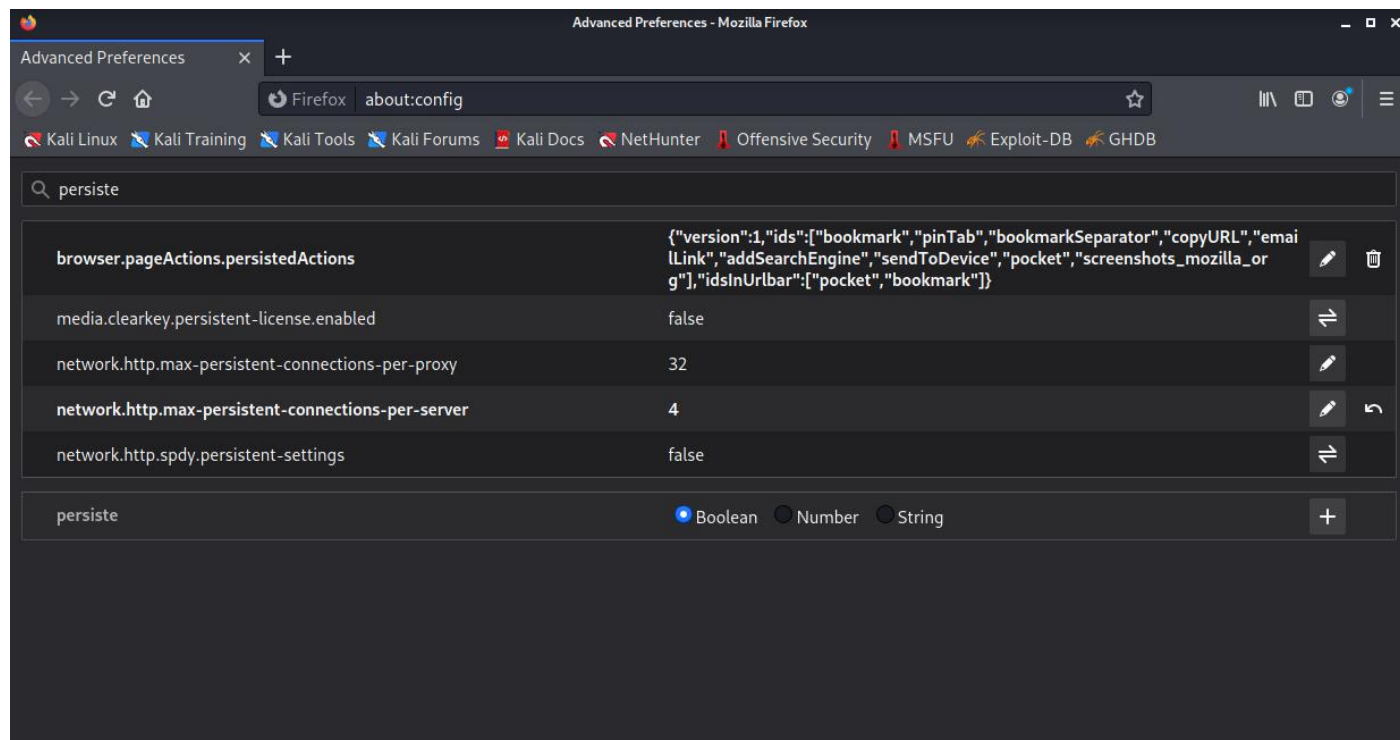
Step 2: Repeat the **steps 1-3** in the previous section.



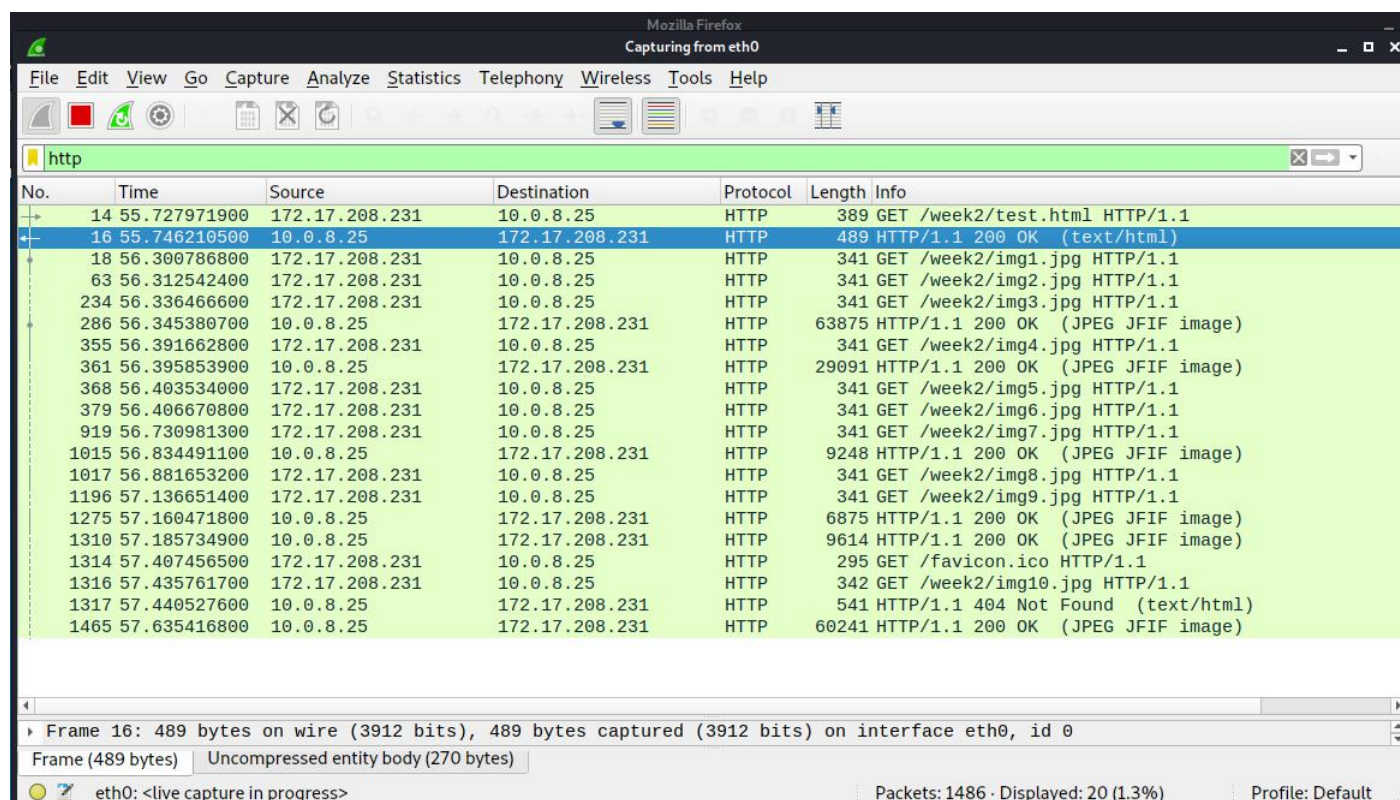
Here it is 22.158346000 – 19.361503200 = 2.7968428



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



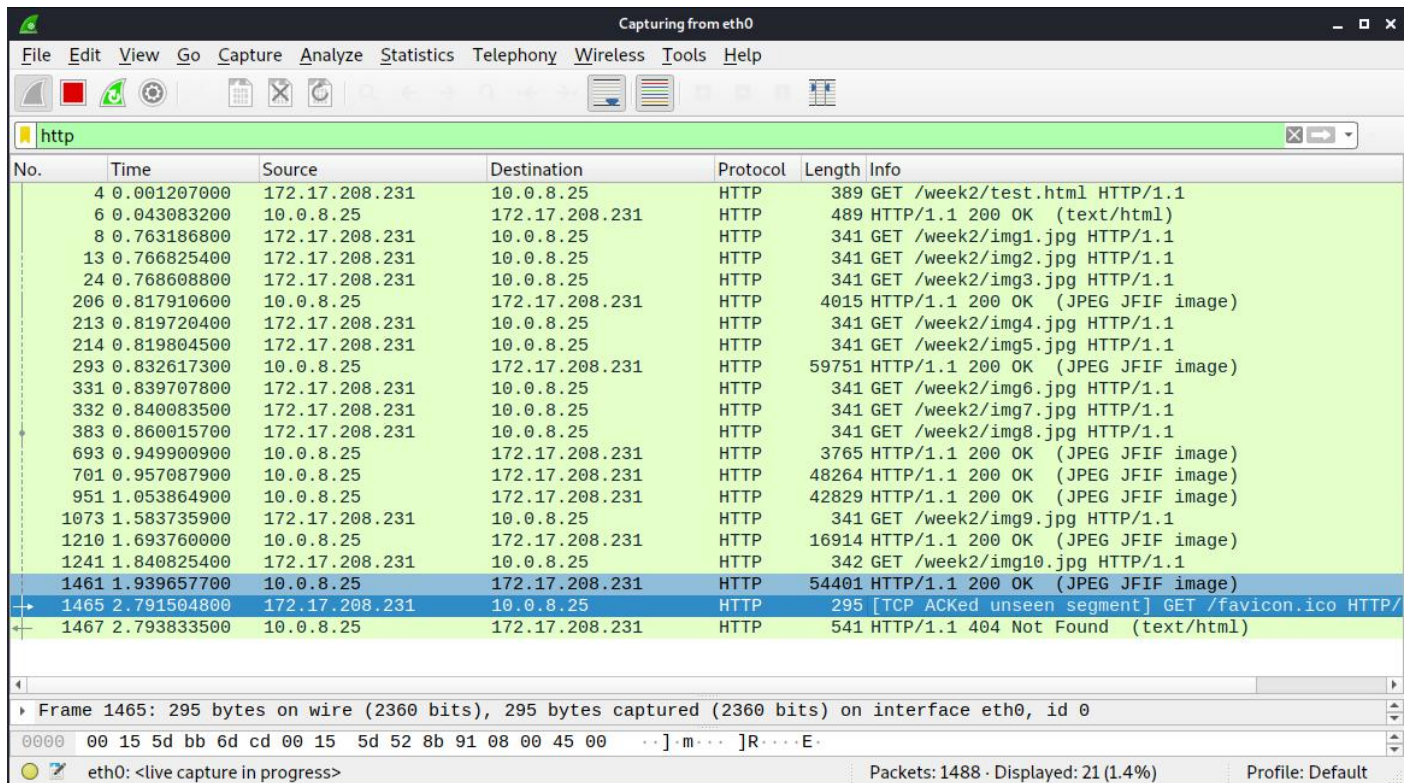
Step 4: Repeat the **steps 1-3** in the previous section



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



No.	Time	Source	Destination	Protocol	Length	Info
4	0.001207000	172.17.208.231	10.0.8.25	HTTP	389	GET /week2/test.html HTTP/1.1
6	0.043083200	10.0.8.25	172.17.208.231	HTTP	489	HTTP/1.1 200 OK (text/html)
8	0.763186800	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img1.jpg HTTP/1.1
13	0.766825400	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img2.jpg HTTP/1.1
24	0.768608800	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img3.jpg HTTP/1.1
206	0.817910600	10.0.8.25	172.17.208.231	HTTP	4015	HTTP/1.1 200 OK (JPEG JFIF image)
213	0.819720400	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img4.jpg HTTP/1.1
214	0.819804500	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img5.jpg HTTP/1.1
293	0.832617300	10.0.8.25	172.17.208.231	HTTP	59751	HTTP/1.1 200 OK (JPEG JFIF image)
331	0.839707800	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img6.jpg HTTP/1.1
332	0.840083500	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img7.jpg HTTP/1.1
383	0.860015700	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img8.jpg HTTP/1.1
693	0.949900900	10.0.8.25	172.17.208.231	HTTP	3765	HTTP/1.1 200 OK (JPEG JFIF image)
701	0.957087900	10.0.8.25	172.17.208.231	HTTP	48264	HTTP/1.1 200 OK (JPEG JFIF image)
951	1.053864900	10.0.8.25	172.17.208.231	HTTP	42829	HTTP/1.1 200 OK (JPEG JFIF image)
1073	1.583735900	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img9.jpg HTTP/1.1
1210	1.693760000	10.0.8.25	172.17.208.231	HTTP	16914	HTTP/1.1 200 OK (JPEG JFIF image)
1241	1.840825400	172.17.208.231	10.0.8.25	HTTP	342	GET /week2/img10.jpg HTTP/1.1
1461	1.939657700	10.0.8.25	172.17.208.231	HTTP	54401	HTTP/1.1 200 OK (JPEG JFIF image)
1465	2.791504800	172.17.208.231	10.0.8.25	HTTP	295	[TCP ACKed unseen segment] GET /favicon.ico HTTP/1.1
1467	2.793833500	10.0.8.25	172.17.208.231	HTTP	541	HTTP/1.1 404 Not Found (text/html)

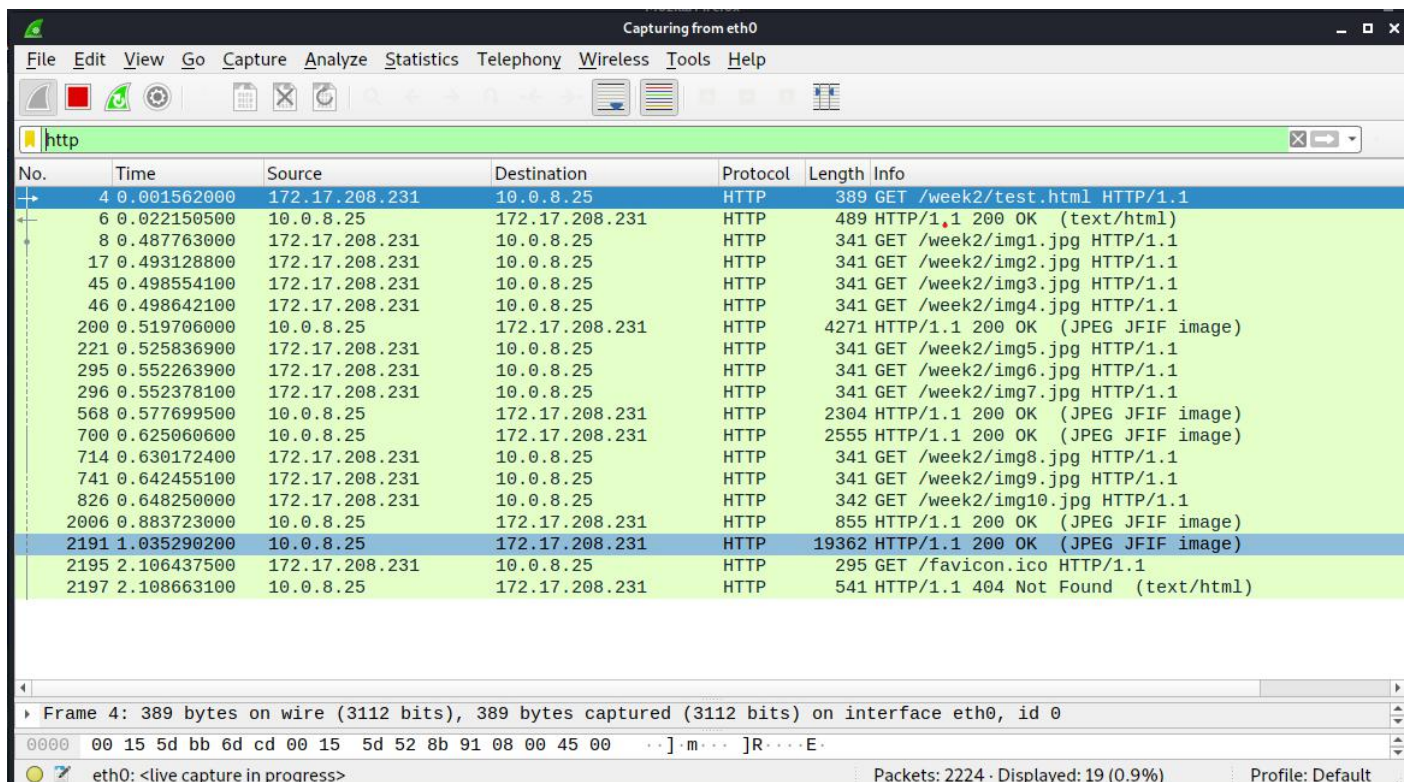
Frame 1465: 295 bytes on wire (2360 bits), 295 bytes captured (2360 bits) on interface eth0, id 0

eth0: <live capture in progress> Packets: 1488 · Displayed: 21 (1.4%) Profile: Default

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.



No.	Time	Source	Destination	Protocol	Length	Info
4	0.001562000	172.17.208.231	10.0.8.25	HTTP	389	GET /week2/test.html HTTP/1.1
6	0.022150500	10.0.8.25	172.17.208.231	HTTP	489	HTTP/1.1 200 OK (text/html)
8	0.487763000	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img1.jpg HTTP/1.1
17	0.493128800	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img2.jpg HTTP/1.1
45	0.498554100	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img3.jpg HTTP/1.1
46	0.498642100	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img4.jpg HTTP/1.1
200	0.519706000	10.0.8.25	172.17.208.231	HTTP	4271	HTTP/1.1 200 OK (JPEG JFIF image)
221	0.525836900	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img5.jpg HTTP/1.1
295	0.552263900	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img6.jpg HTTP/1.1
296	0.552378100	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img7.jpg HTTP/1.1
568	0.577699500	10.0.8.25	172.17.208.231	HTTP	2304	HTTP/1.1 200 OK (JPEG JFIF image)
700	0.625060600	10.0.8.25	172.17.208.231	HTTP	2555	HTTP/1.1 200 OK (JPEG JFIF image)
714	0.630172400	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img8.jpg HTTP/1.1
741	0.642455100	172.17.208.231	10.0.8.25	HTTP	341	GET /week2/img9.jpg HTTP/1.1
826	0.648250000	172.17.208.231	10.0.8.25	HTTP	342	GET /week2/img10.jpg HTTP/1.1
2006	0.883723000	10.0.8.25	172.17.208.231	HTTP	855	HTTP/1.1 200 OK (JPEG JFIF image)
2191	1.035290200	10.0.8.25	172.17.208.231	HTTP	19362	HTTP/1.1 200 OK (JPEG JFIF image)
2195	2.106437500	172.17.208.231	10.0.8.25	HTTP	295	GET /favicon.ico HTTP/1.1
2197	2.108663100	10.0.8.25	172.17.208.231	HTTP	541	HTTP/1.1 404 Not Found (text/html)

Frame 4: 389 bytes on wire (3112 bits), 389 bytes captured (3112 bits) on interface eth0, id 0

eth0: <live capture in progress> Packets: 2224 · Displayed: 19 (0.9%) Profile: Default



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

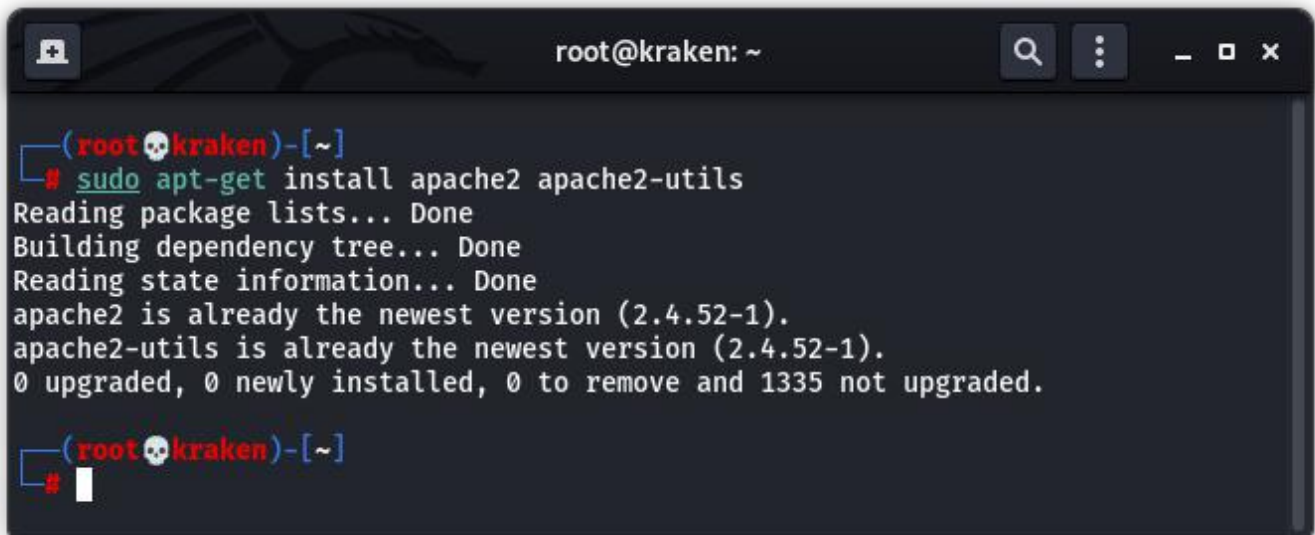
# 1.Password Authentication

## 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**



```
root@kraken: ~  
(root@kraken)-[~]  
# sudo apt-get install apache2 apache2-utils  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
apache2 is already the newest version (2.4.52-1).  
apache2-utils is already the newest version (2.4.52-1).  
0 upgraded, 0 newly installed, 0 to remove and 1335 not upgraded.  
(root@kraken)-[~]  
#
```

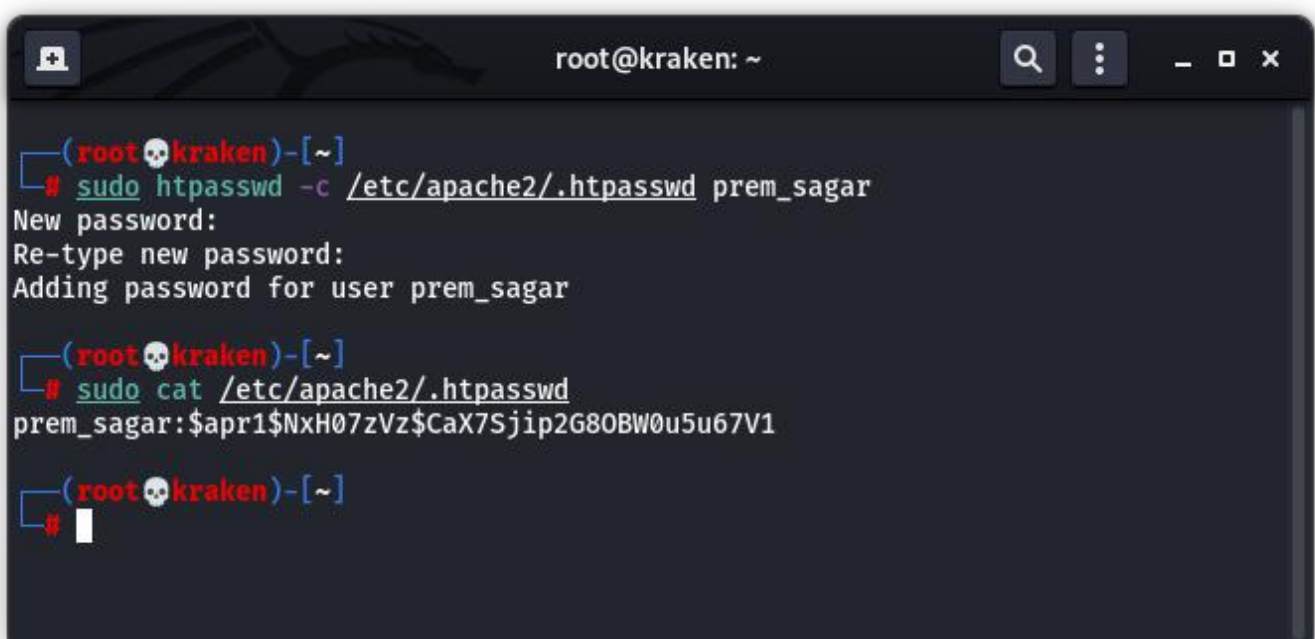
--> Provide username and password to set authentication

**sudo htpasswd -c /etc/apache2/.htpasswd prem\_sagar**

--> View the authentication

**sudo cat**

**/etc/apache2/.htpasswd**



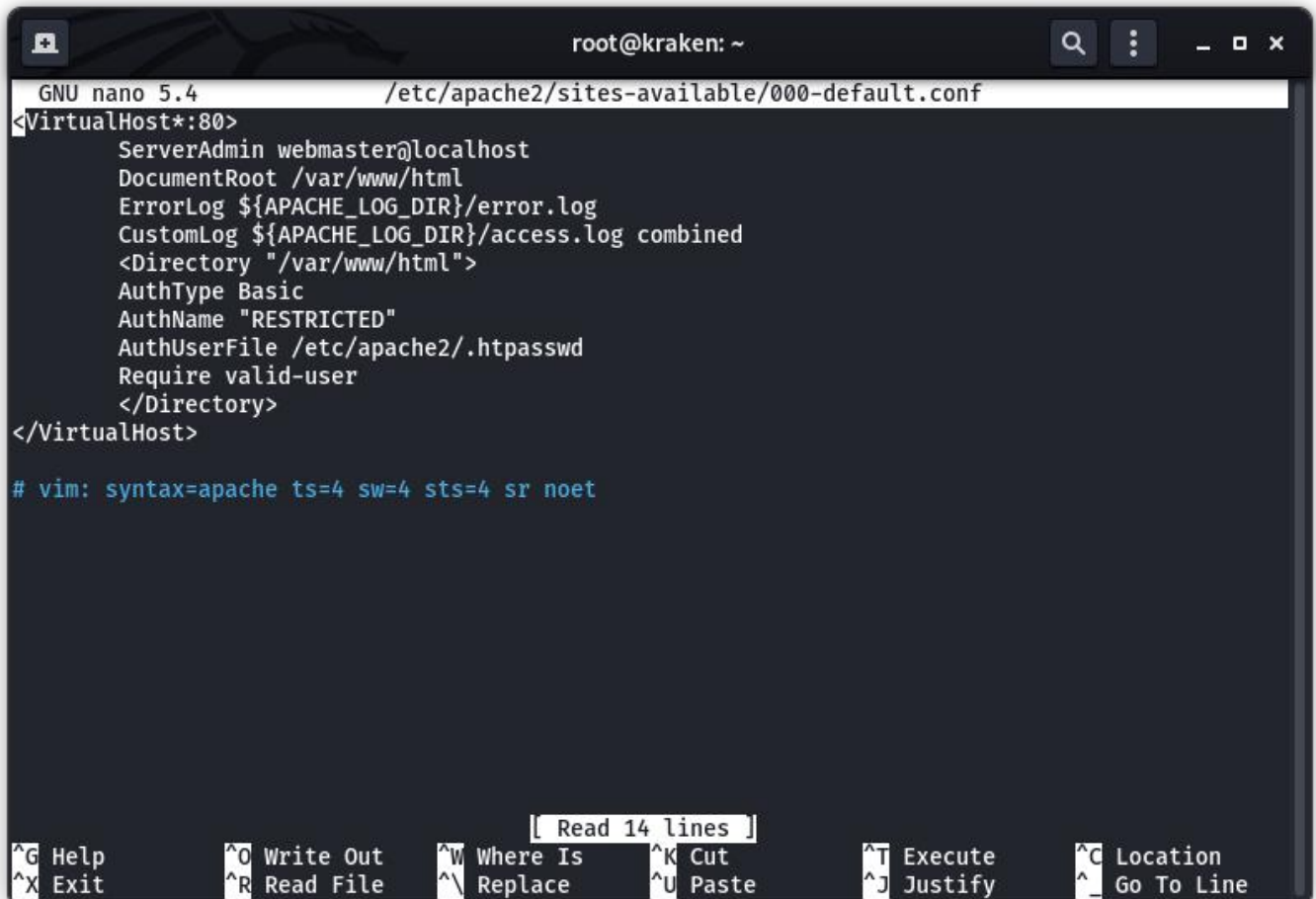
```
root@kraken: ~  
(root@kraken)-[~]  
# sudo htpasswd -c /etc/apache2/.htpasswd prem_sagar  
New password:  
Re-type new password:  
Adding password for user prem_sagar  
(root@kraken)-[~]  
# sudo cat /etc/apache2/.htpasswd  
prem_sagar:$apr1$NxBH07zVz$CaX7Sjip2G80BW0u5u67V1  
(root@kraken)-[~]  
#
```



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**

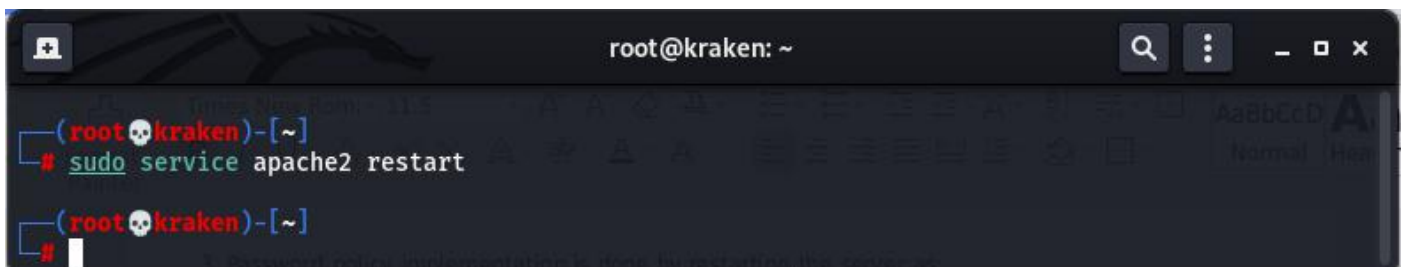


```
root@kraken: ~
GNU nano 5.4 /etc/apache2/sites-available/000-default.conf
<VirtualHost*:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    <Directory "/var/www/html">
        AuthType Basic
        AuthName "RESTRICTED"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
    </Directory>
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

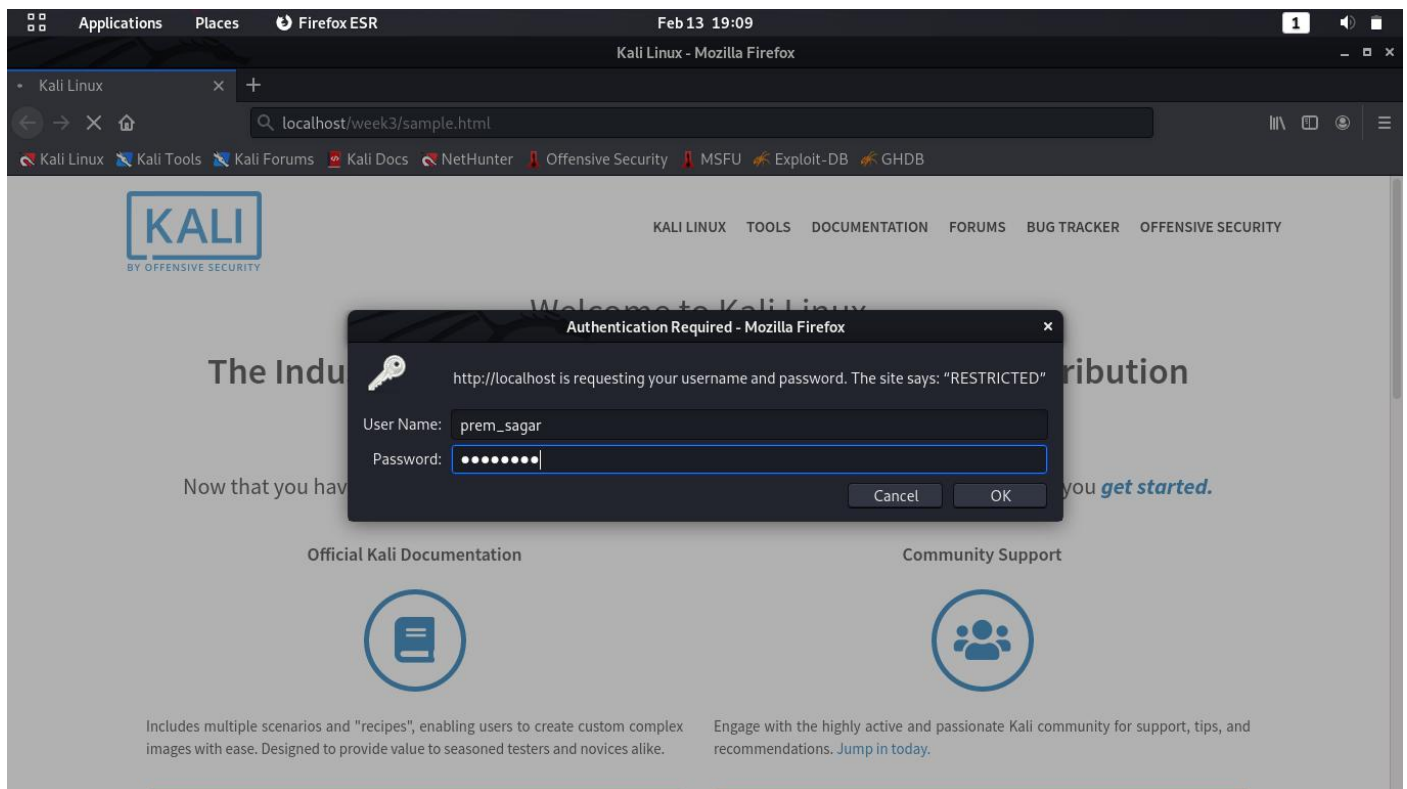
[ Read 14 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

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

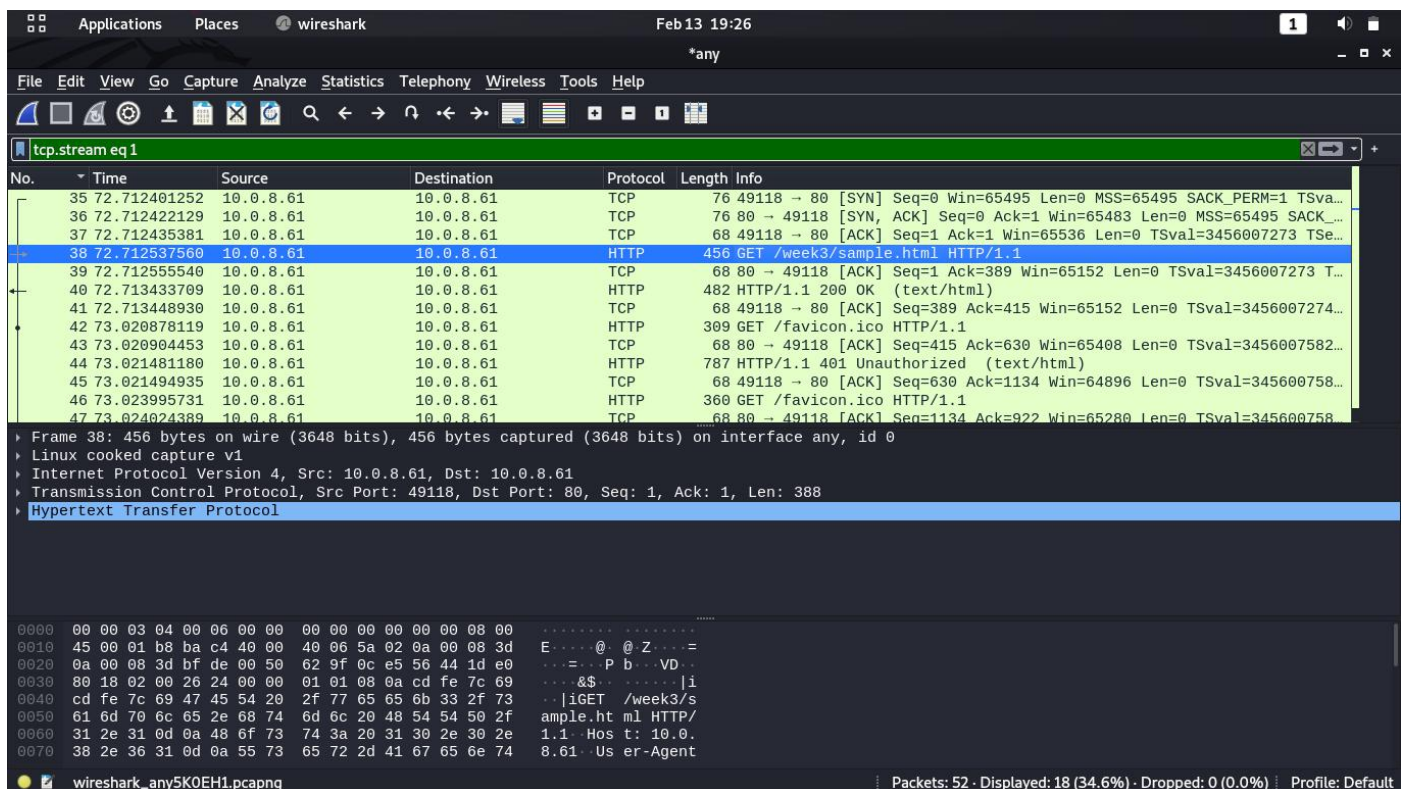


```
root@kraken: ~
(root@kraken)~[~]
# sudo service apache2 restart
(root@kraken)~[~]
#
```

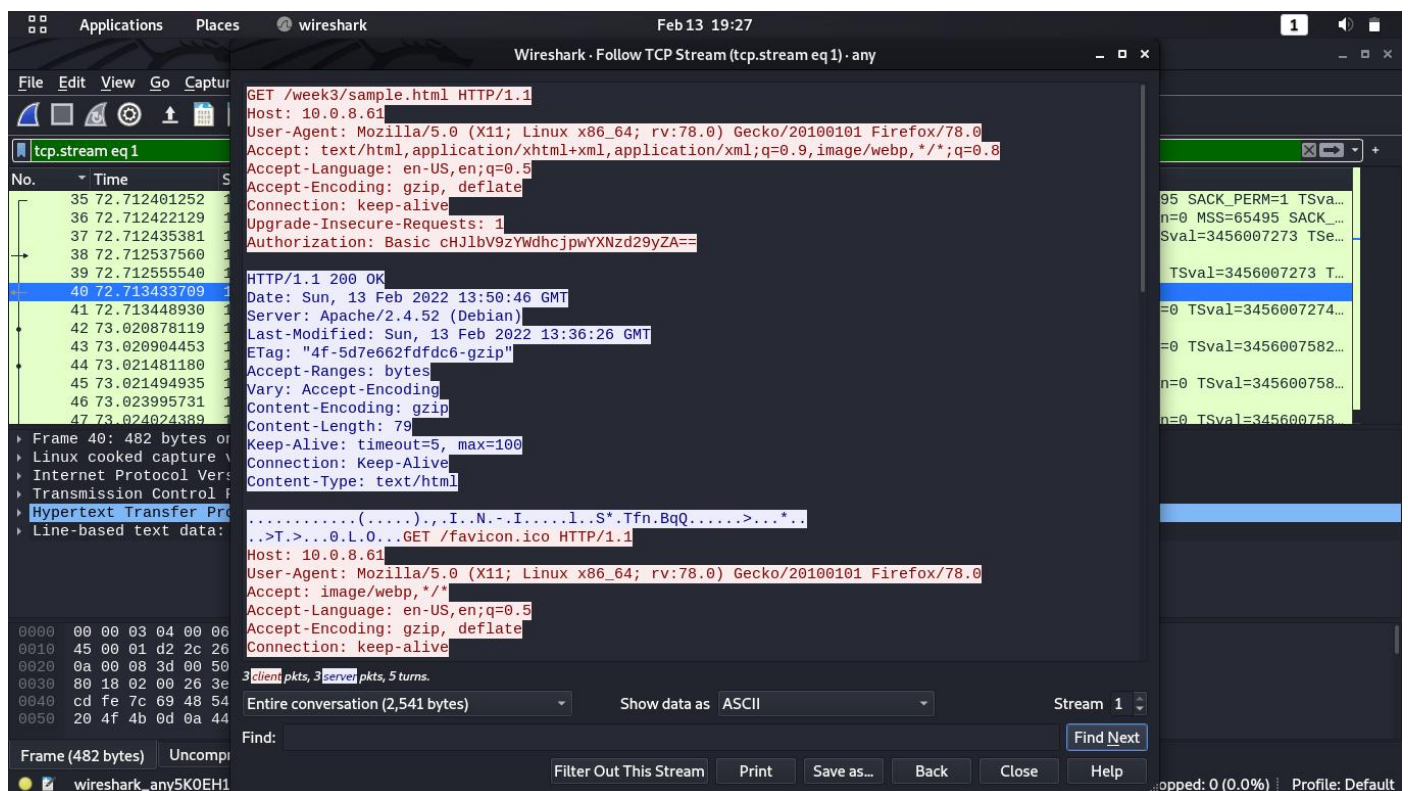
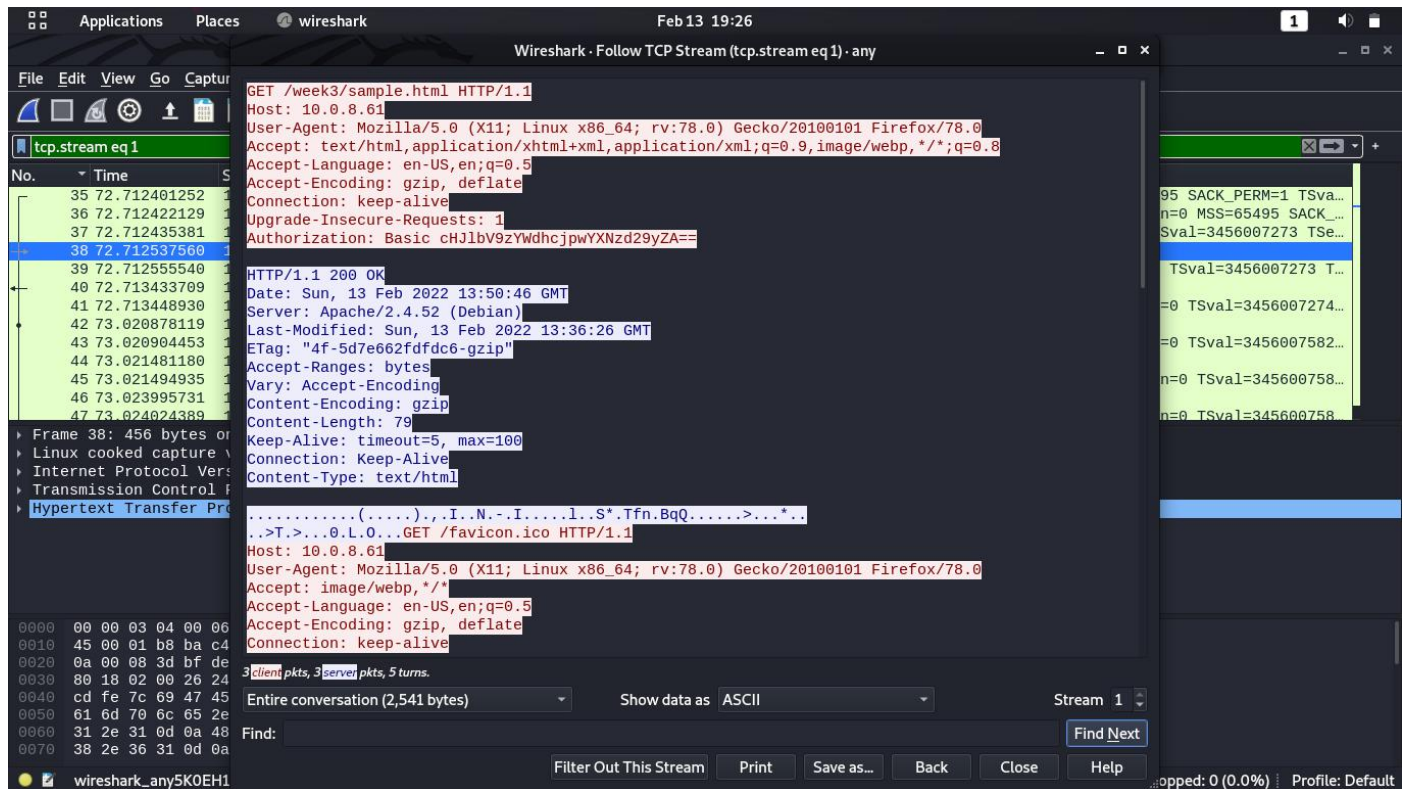
4. The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase



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.

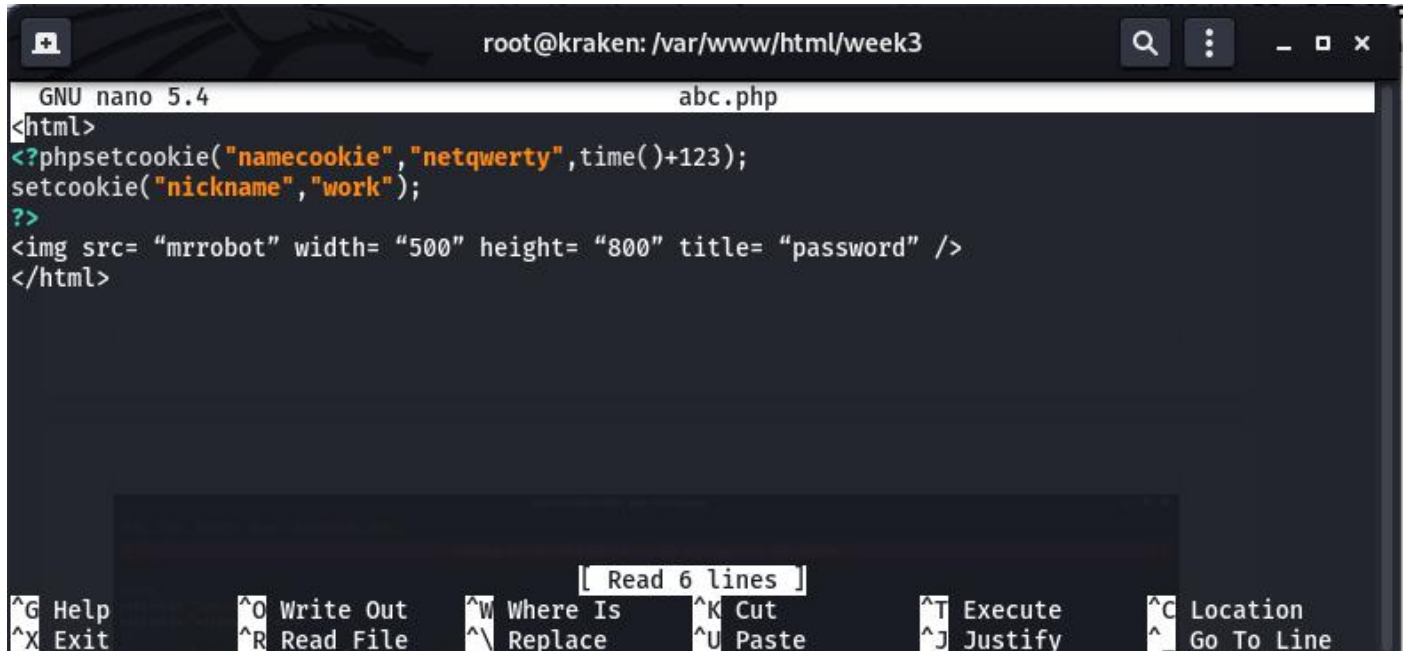




## 2. Cookie Setting

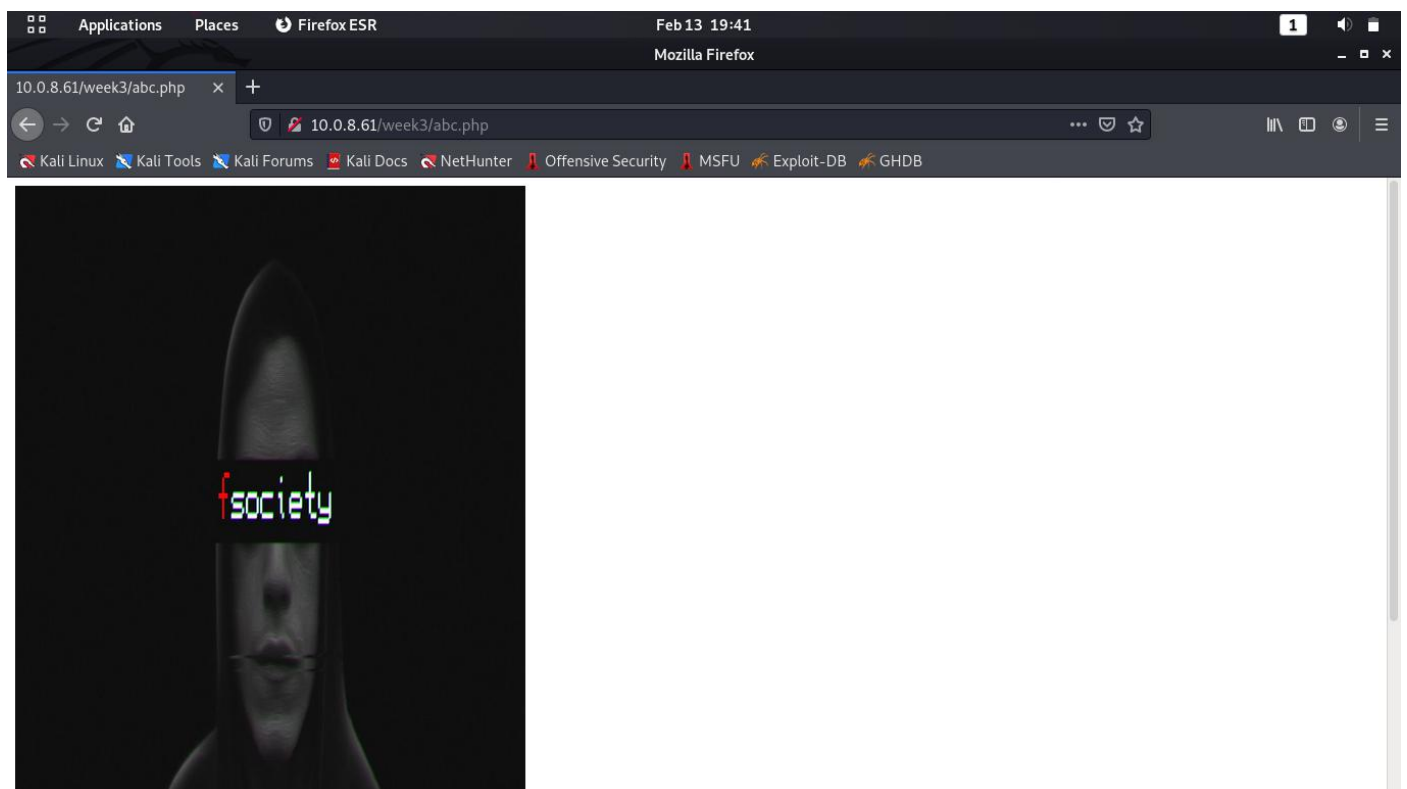
### Steps of Execution (Cookie Setting)

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:

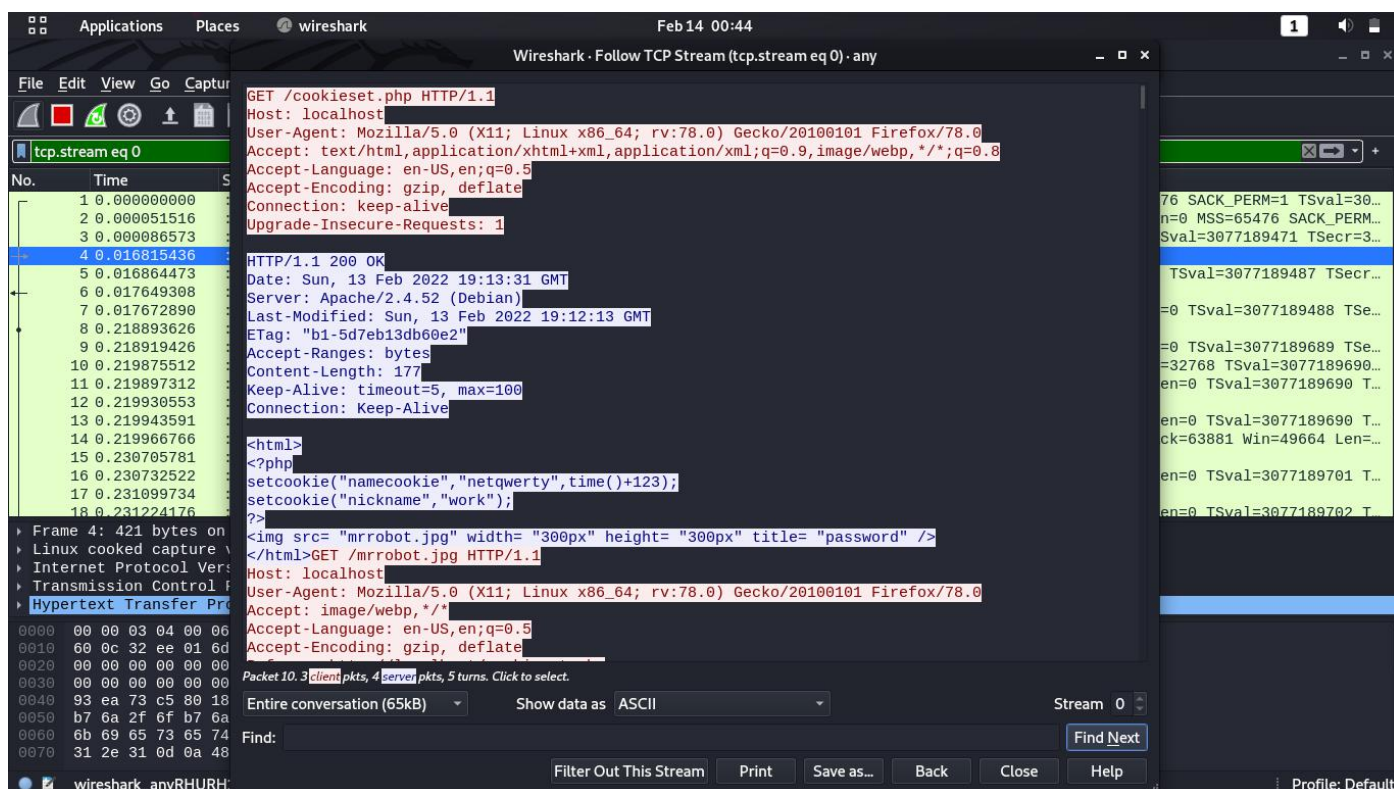


```
root@kraken: /var/www/html/week3
GNU nano 5.4 abc.php
<html>
<?phpsetcookie("namecookie","netqwerty",time()+123);
setcookie("nickname","work");
?>
<img src= "mrrobot" width= "500" height= "800" title= "password" />
</html>
```

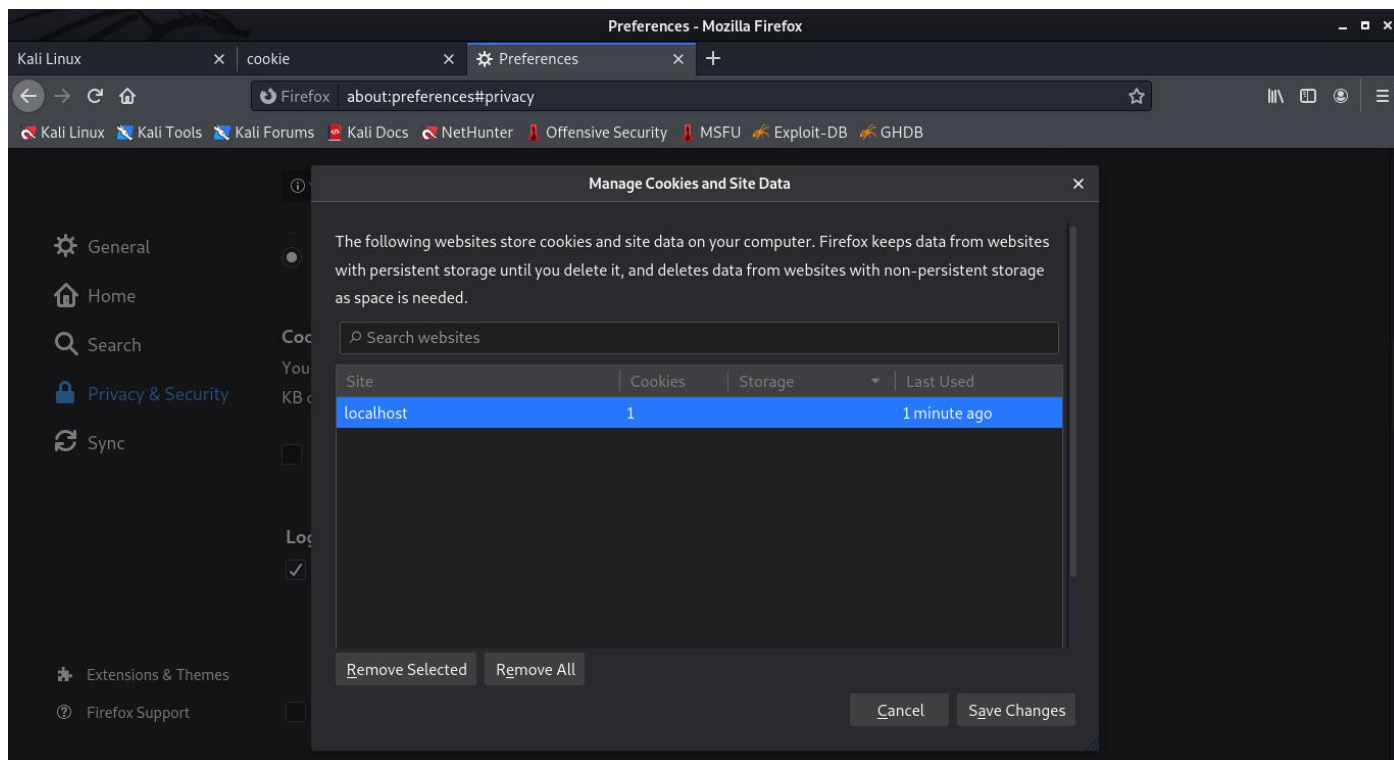
2. The combined file saved with a .php extension is placed under **/var/www/html** for accessing.



- 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.

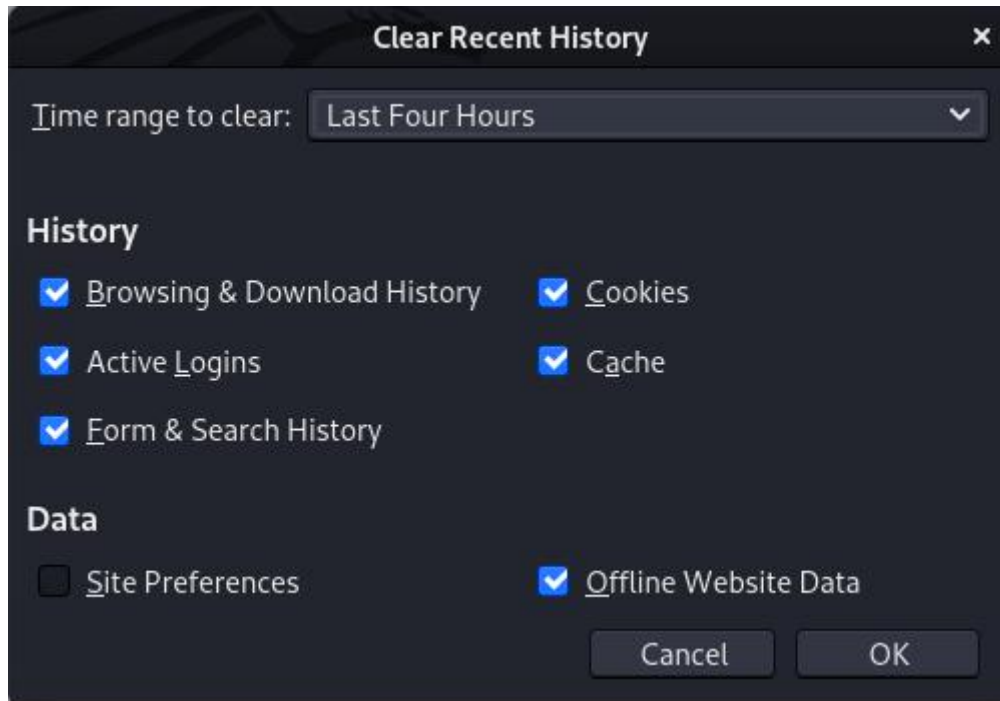


The cookie is set as shown in the above screenshot.

### 3. Conditional get

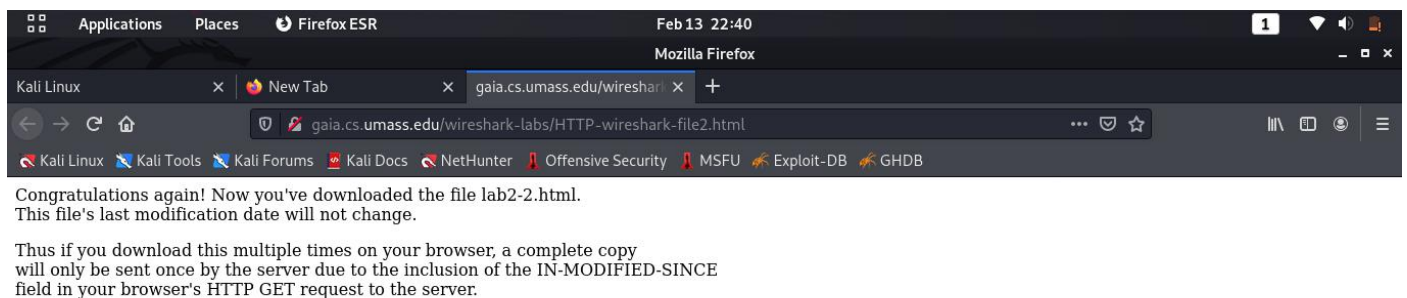
#### Conditional Get: If-Modified-Since

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.

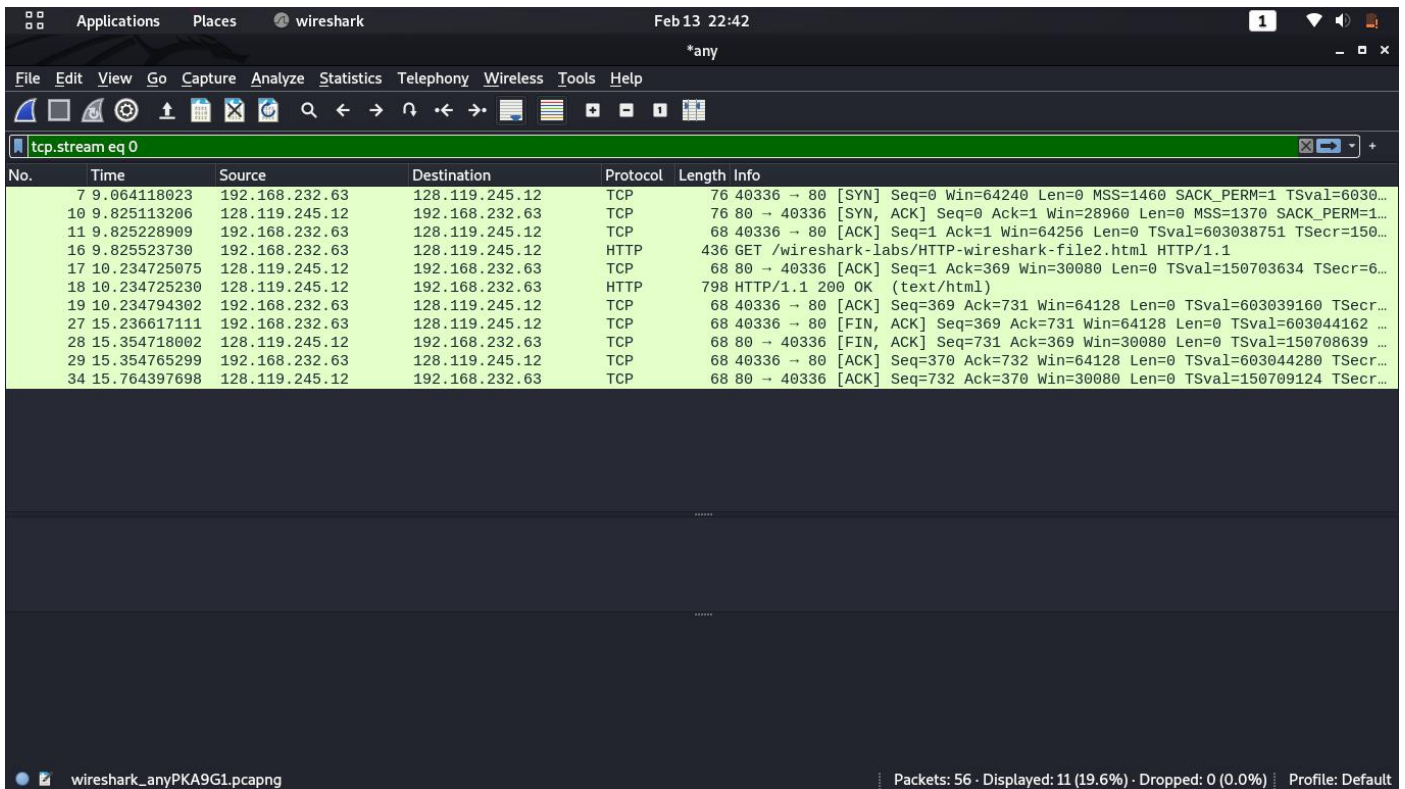


Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)

(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.

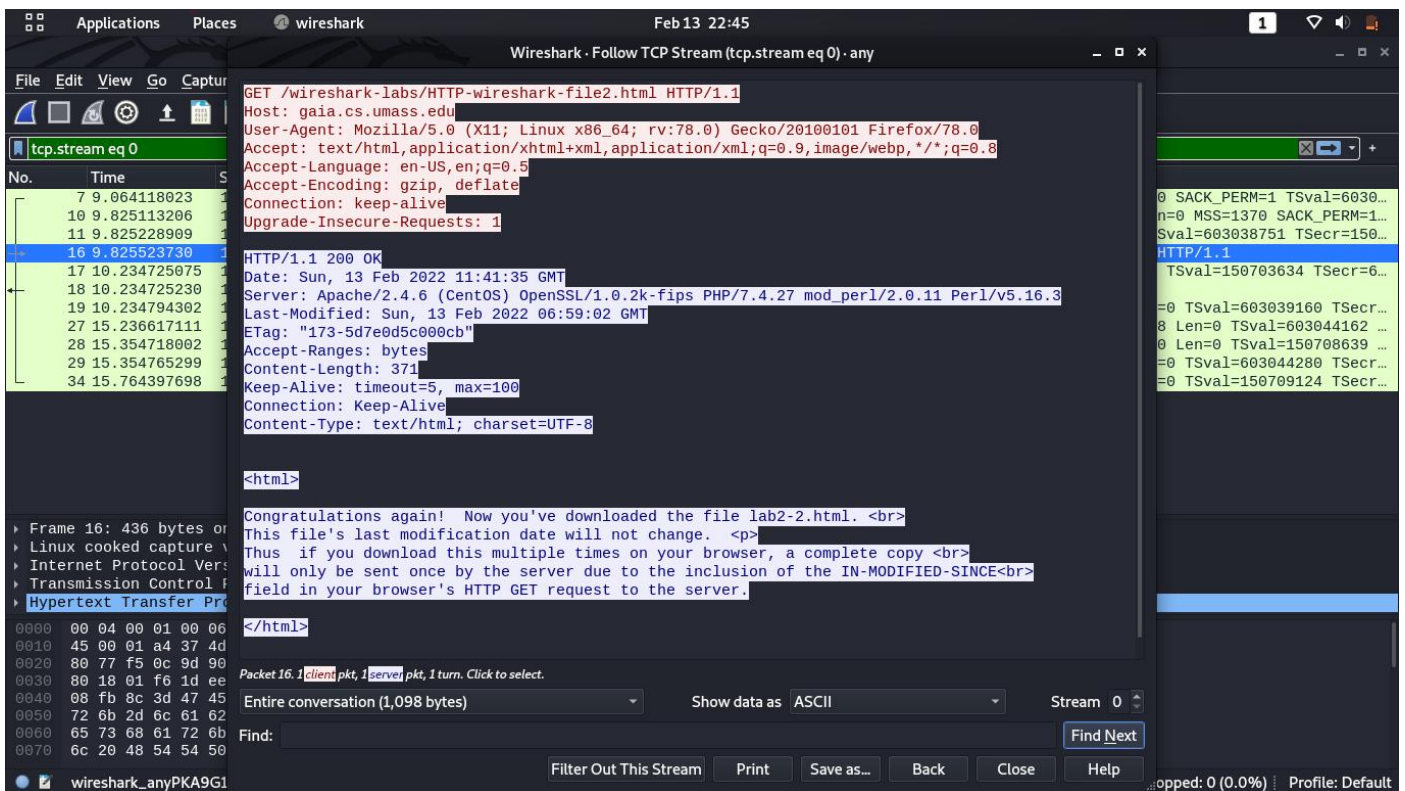




## Observations:

✓ 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

The screenshot shows the Wireshark interface with the 'Follow TCP Stream' window open for 'tcp.stream eq 0'. The packet list on the left shows a GET request (No. 18) and its corresponding OK response (No. 19). The packet details pane for packet 19 shows the response structure: HTTP/1.1 200 OK, Date: Sun, 13 Feb 2022 11:41:35 GMT, Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.27 mod\_perl/2.0.11 Perl/v5.16.3, Last-Modified: Sun, 13 Feb 2022 06:59:02 GMT, ETag: "173-5d7e0d5c000cb", Accept-Ranges: bytes, Content-Length: 371, Keep-Alive: timeout=5, max=100, Connection: Keep-Alive, Content-Type: text/html; charset=UTF-8. The packet bytes pane shows the raw data, and the packet contents pane displays the HTML content, which includes a congratulatory message and a link to download the file.

✓ 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

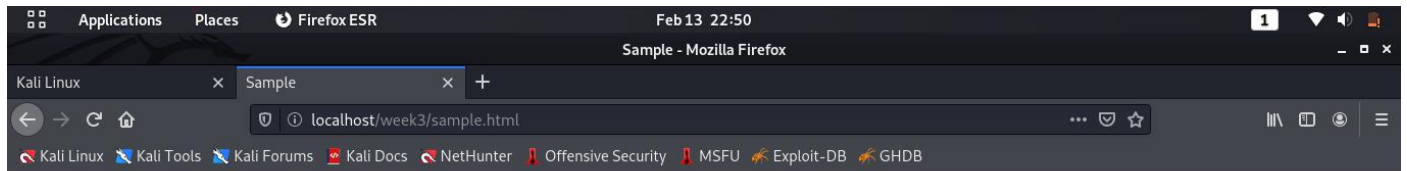
The screenshot shows the Wireshark interface with the 'Follow TCP Stream' window open for 'tcp.stream eq 3'. The packet list on the left shows a GET request (No. 46) and its corresponding 304 Not Modified response (No. 47). The packet details pane for packet 47 shows the response structure: HTTP/1.1 304 Not Modified, Date: Sun, 13 Feb 2022 11:41:49 GMT, Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.27 mod\_perl/2.0.11 Perl/v5.16.3, Connection: Keep-Alive, Keep-Alive: timeout=5, max=100, ETag: "173-5d7e0d5c000cb". The packet bytes pane shows the raw data, and the packet contents pane displays the response structure, which includes the 'If-Modified-Since' header and the 'If-None-Match' header.



✓ 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.

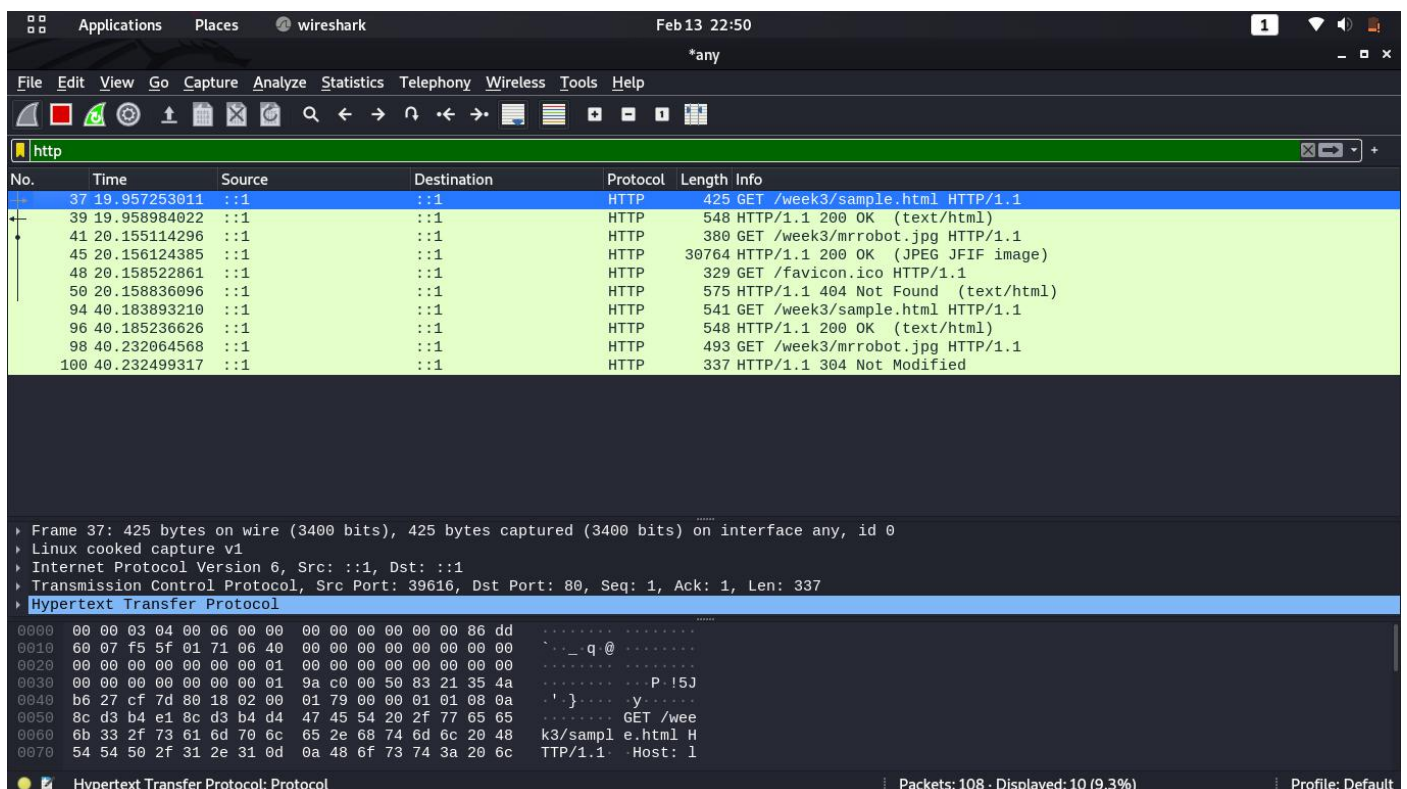
Ans : NO , because the contents are already downloaded in the browsers cache , so it'll just get the data from cache If it hasn't modified . (status code : 304 NOT MODIFIED)

Repeat the above task with some images on the server.



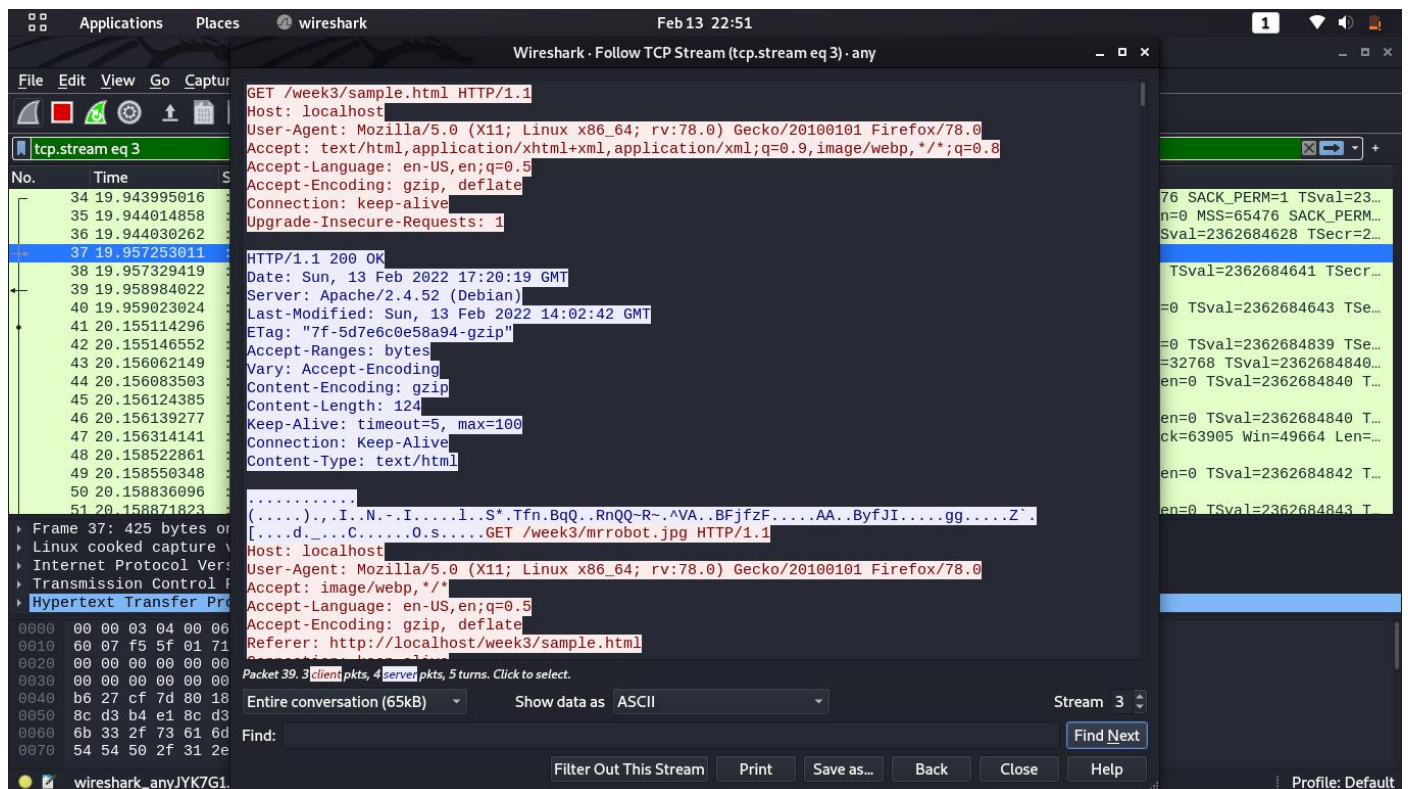
Mr.Robot

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

