|  |  |  |
| --- | --- | --- |
| **Name: GAURAV MAHAJAN** | **SRN: PES1UG20CS150** | **Section: C** |
| **Date: 09-02-2022** | **WEEK 3** |

**3A. Understanding Working of HTTP Headers**

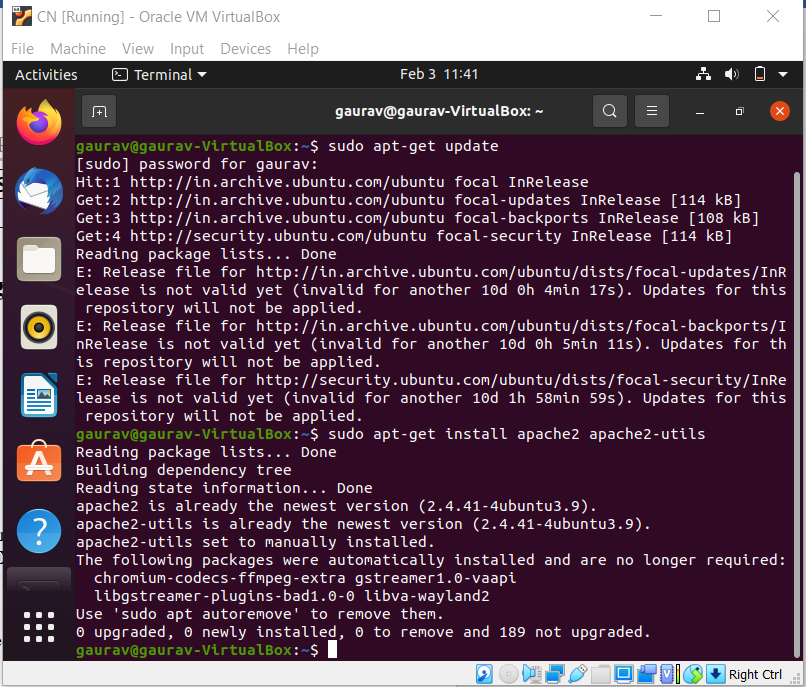
**Steps of Execution (for Password Authentication)** 1. Executing the below commands on the terminal.

--> To update and integrate the existing softwares

**sudo apt-get update**

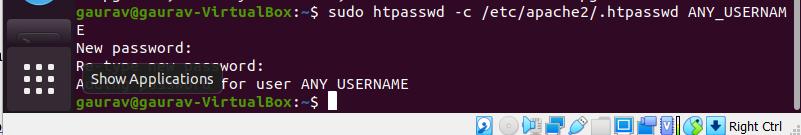
--> To install the apache utility

**sudo apt-get install apache2 apache2-utils**

****

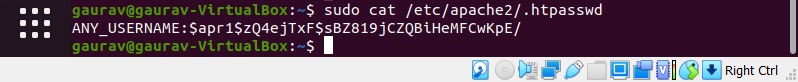
--> Provide username and password to set authentication

**sudo htpasswd -c /etc/apache2/.htpasswd ANY\_USERNAME**



--> View the authentication

**sudo cat /etc/apache2/.htpasswd**

****

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**

<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">

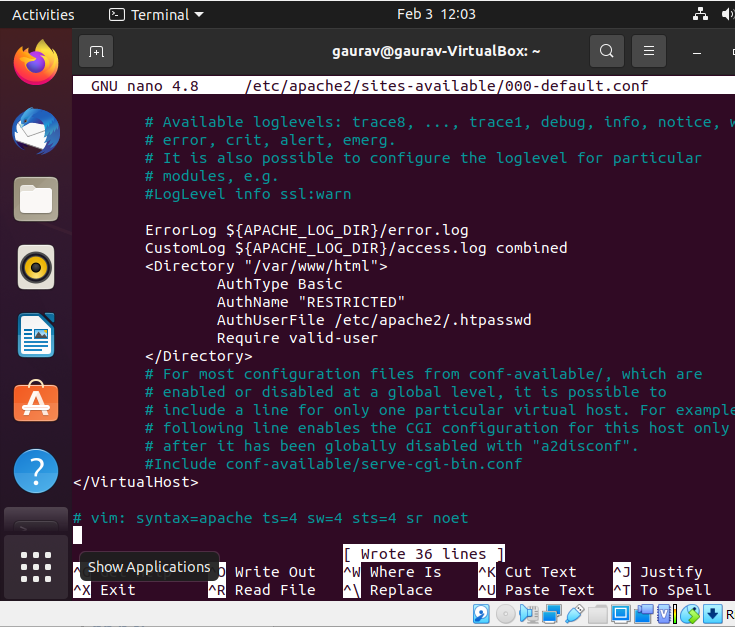
AuthType Basic

AuthName "RESTRICTED"

AuthUserFile /etc/apache2/.htpasswd

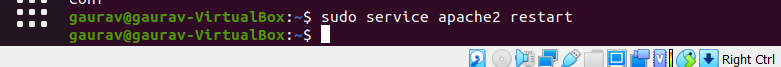
Require valid-user

</Directory>

</VirtualHost> 

1. Password policy implementation is done by restarting the server as:

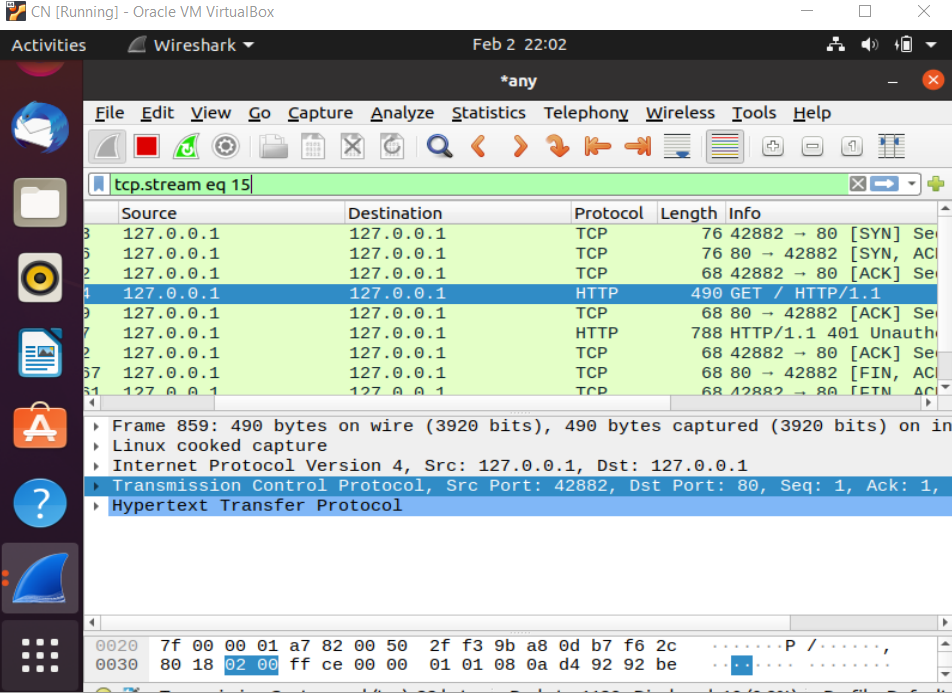
**sudo service apache2 restart**



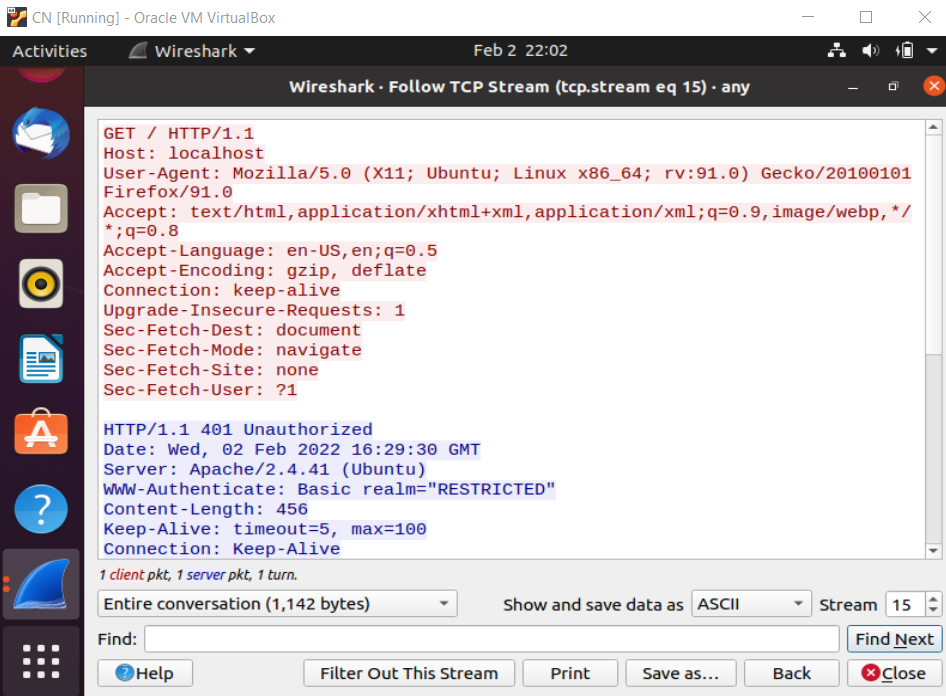
1. The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase.



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



1. 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.

 **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:

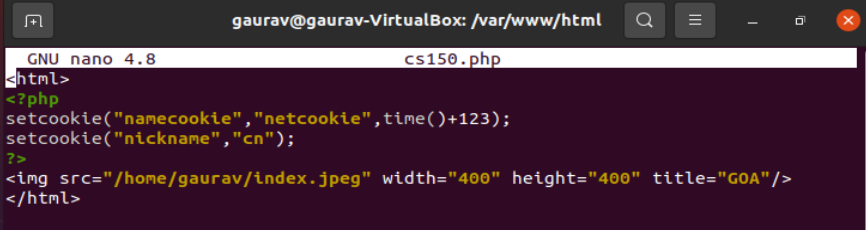
<html>

<?php

setcookie("namecookie","netqwerty",time()+123); setcookie("nickname","work"); ?>

<img src= “highres.png” width= “300” height= “300” title= “password” />

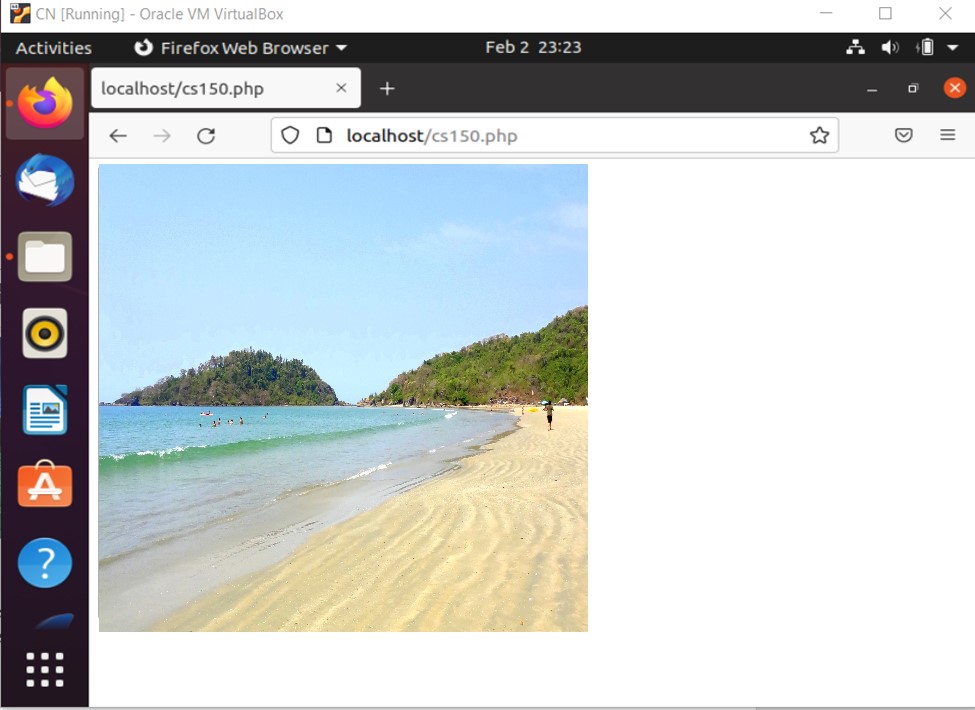
</html>

****

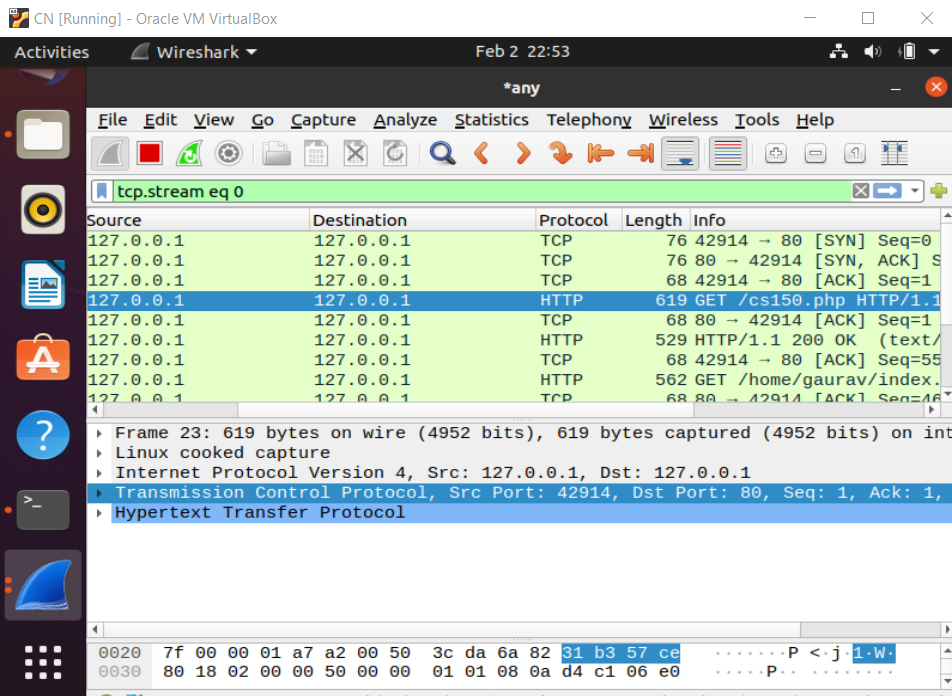
Note: Here you can add any image if required

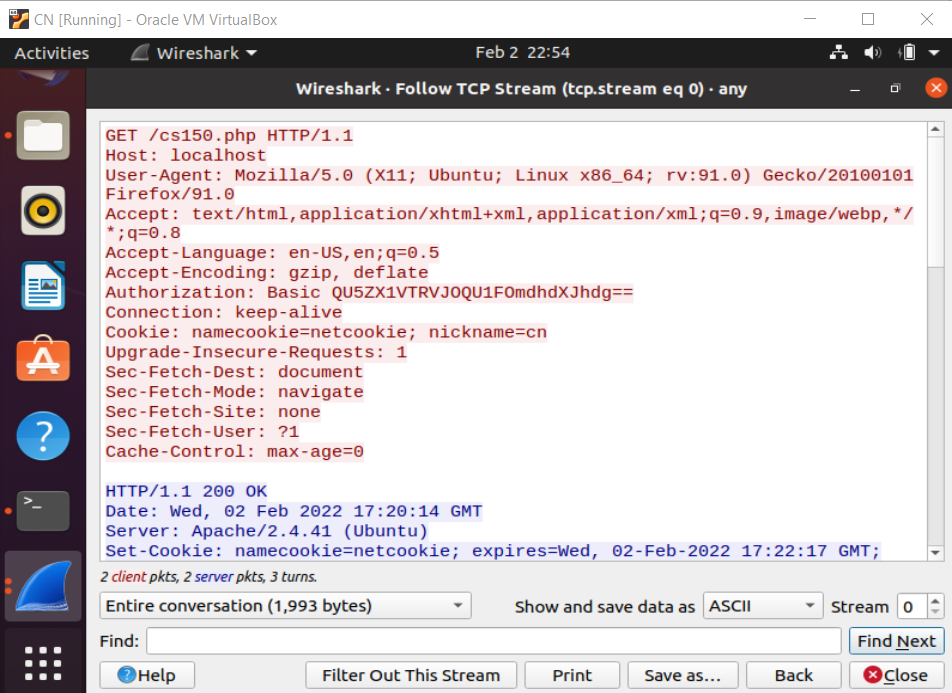
**Note: You can capture Cookies mostly during the first time of web access. Hence keep wireshark capture ready before executing the task for the first time.**

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



1. 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.





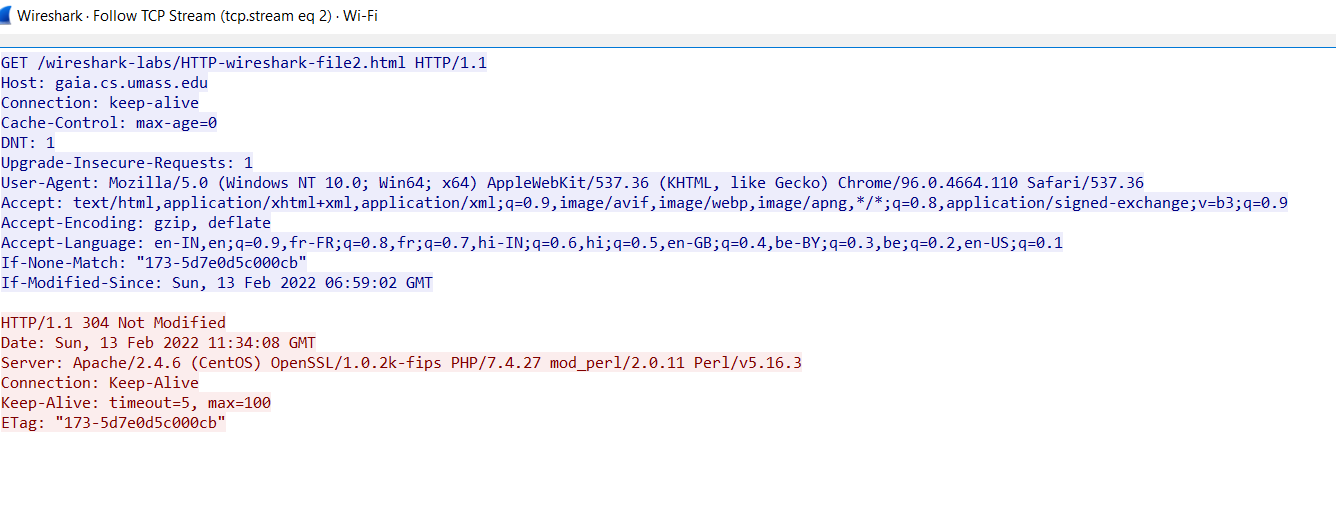
The cookie is set as shown in the above screenshot.

**Observation:** Understand and work out base 64 algorithm and write in your observation. Observe various parameters associated with Cookie in the wireshark capture.

**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: ➢ Start up your web browser, and make sure your browser’s cache is cleared, as discussed above.

* + Start up the Wireshark packet sniffer.
  + Enter the following URL into your browser [http://gaia.cs.umass.edu/wiresharklabs/HTTP-wireshark-file2.html](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)
  + 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 packetlisting 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?

🡪 YES

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

🡪YES BECAUSE WE CAN SEE THE CONTENTS IN THE LINE-BASED TEXT DATA FIELD

* + 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?

🡪 YES. THE INFORMATION FOLLOWED IS: SUN, 13 FEB 2022 06:59:02 GMT\R\N WHICH IS THE DATE OF THE LAST MODIFICATION OF THE FILE FROM THE PREVIOUS GET REQUEST.

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.

🡪THE STATUS CODE AND PHRASE RETURNED FROM THE SERVER IS HTTP/1.1 304 NOT MODIFIED. THE SERVER DIDN’T RETURN THE CONTENTS OF THE FILE SINCE THE BROWSER LOADED IT FROM ITS CACHE.

**3B) Understanding Persistent and Non-persistent HTTP Connections**

**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**

172.16.10.1/24

172.16.10.2/24



**Server Side:**

**Step 2:** Check your Web Server

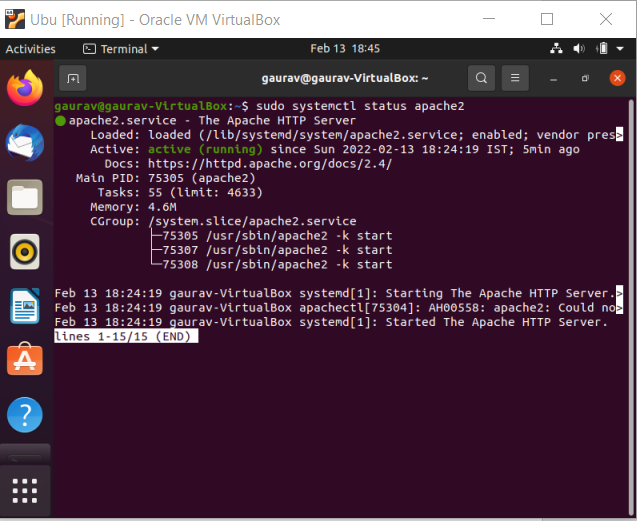
At the end of the installation process, Ubuntu 16.04 starts Apache. The web server should already be up and running. We can check with the systemctl command to make sure the service is running by typing:

**sudo systemctl status apache2**

**or**

**sudo service apache2 status**

As you can see above, the service appears to have started successfully. However, the best way to test this is to actually request a page from Apache. You can access the default Apache landing page to confirm that the software is running properly. You can access this through your server's domain name or IP address.

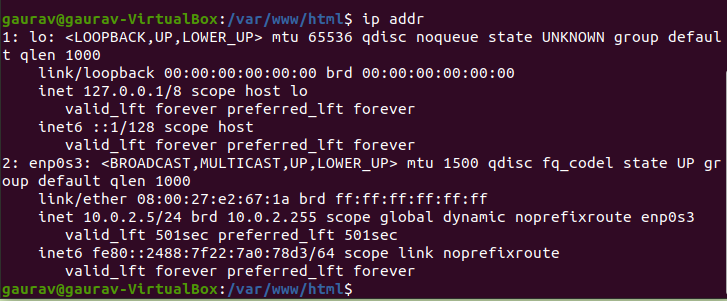


**Step 3:** Server IP address can be set by the following command

**$sudo ip addr add 172.16.10.1/24 dev enps0**

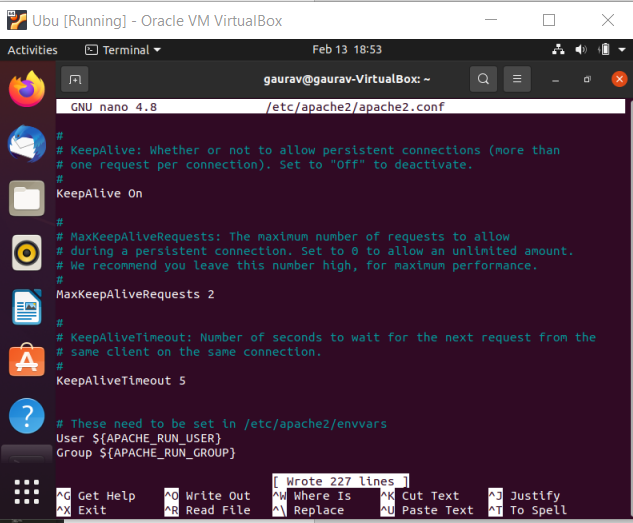
**$sudo ip addr**

Note: If IP address fluctuates, kindly setup the IP address manually using ‘Edit connections’.



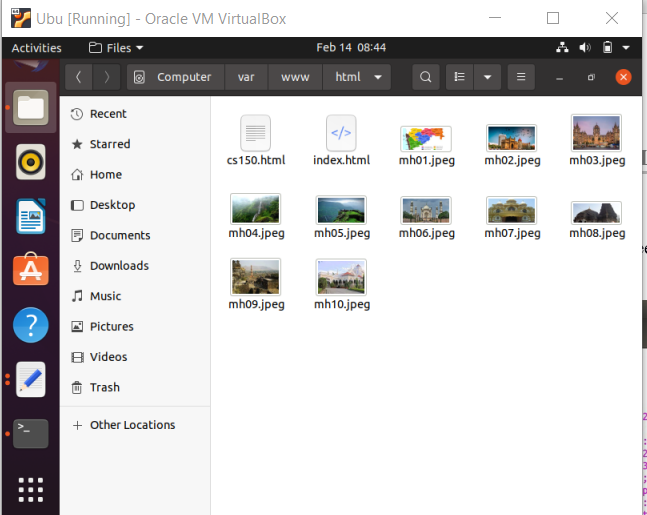
**Step 4:** The **apache2.conf** file present in the **etc/apache2** directory is modified as:

1. The **keep-alive** option was set (i.e. value was made **ON**)
2. The **MaximumKeepAliveRequests** were set to **2**

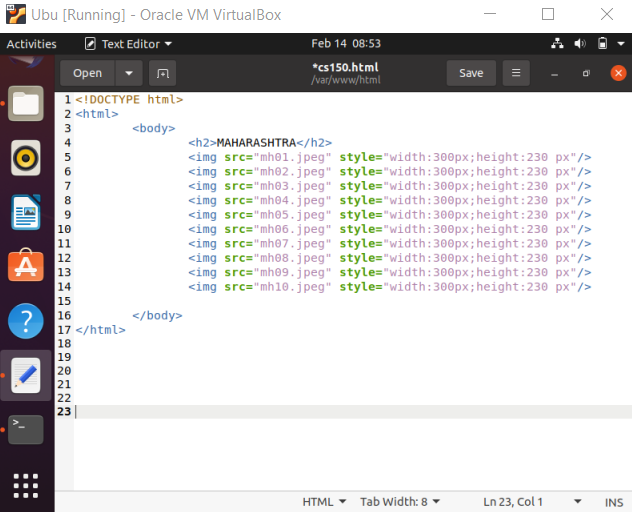


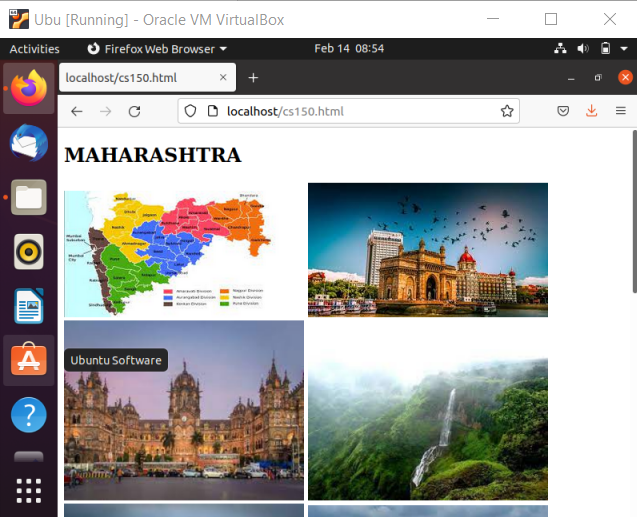
**Step 5:** Store images in the server path. A html page consisting of 10 images having size >

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)





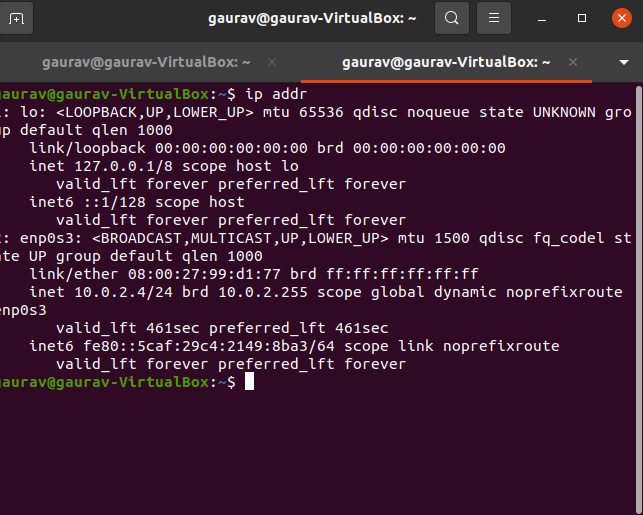
**Client side:**

Client IP address can be set by the following command.

**$sudo ip addr add 172.16.10.2/24 dev enps0**

**$sudo ip addr**

Note: If IP address fluctuates, kindly setup the IP address manually using ‘Edit connections’.



There are broadly two parts of execution:

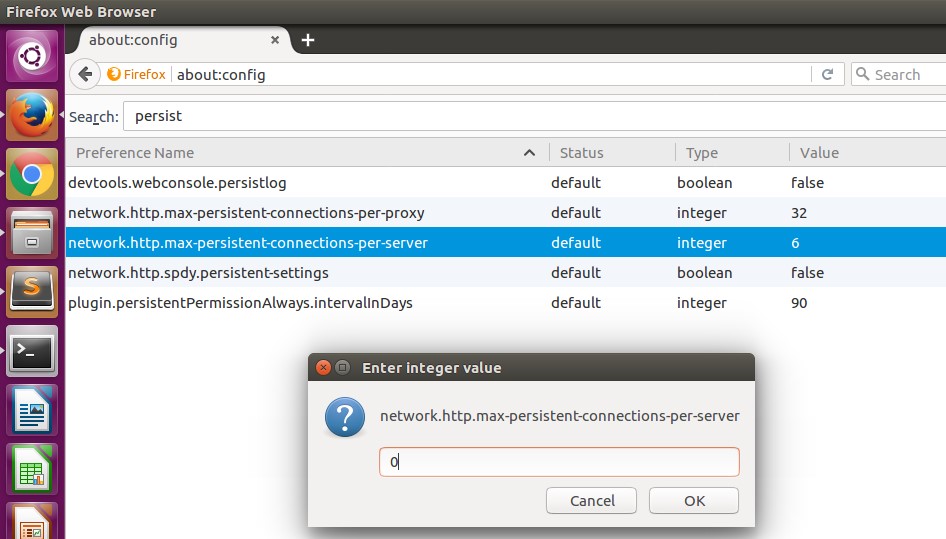
1. Dealing with non-persistent connections
2. Dealing with persistent connections

Open Firefox browser to configure for persistent option. Go to browser andtype **about:config** and search for the term **‘persistent’**

* While using non-persistent connection experiment, the **max-persistent-connectionsper-server** has the value set to **0** and **persistent-settings** value set to false.
* While using persistent connection experiment, the **max-persistent-connections-perserver** should have value greater than 0 (depending on the number of persistent connections needed) and **persistent-settings** value set to true.

**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:

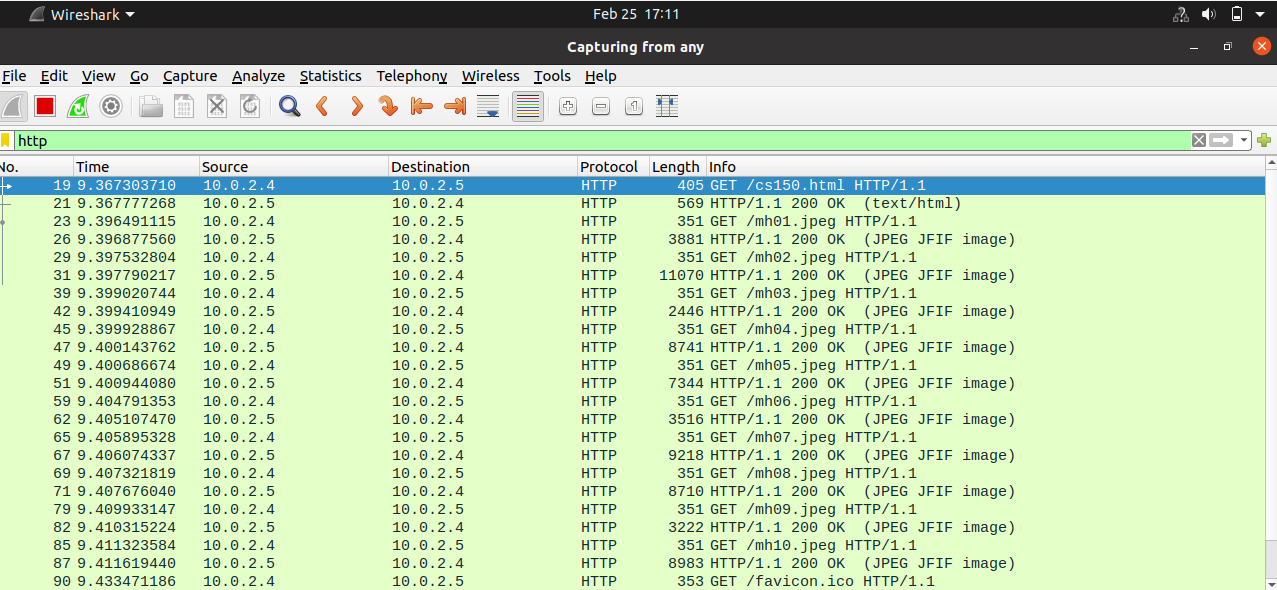
**172.16.10.1/file\_name.html** where**--> 172.16.10.1** is Server’s IP

Here the file name is **a.html** present in server. So, by tying **172.16.10.1/a.html** in client browser, we will be able to open the requested web page.

Note 1: The wireshark should capture the packets between the client and the server while the file is accessed.

Note 2: The images in the HTML page should have all the permissions specified through the server for the proper access.

**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.

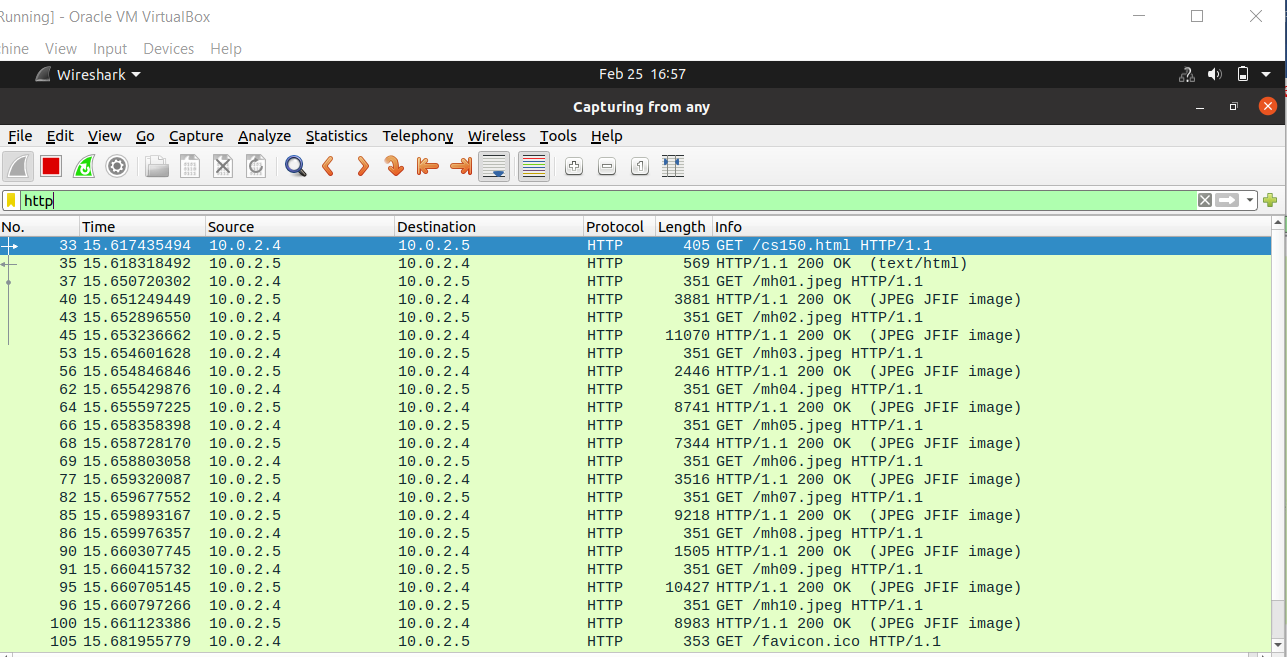


Here it is 9.411619440- 9.367303710 = 0.04431573

**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.

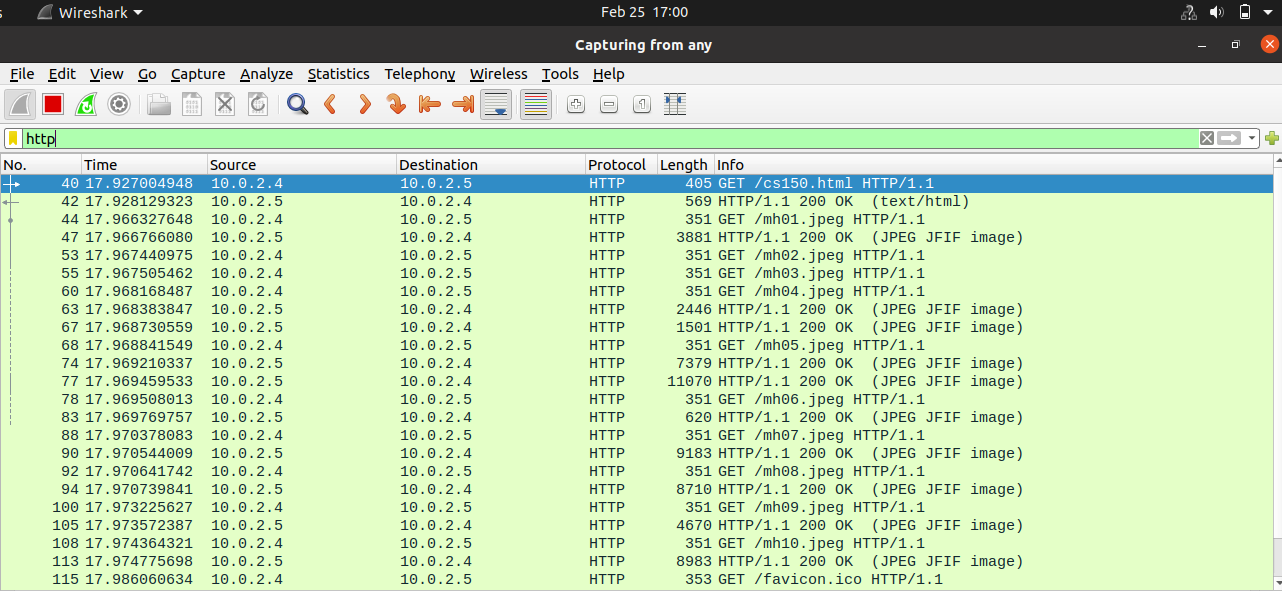
Step 2: Repeat the **steps 1-3** in the previous section.



Here it is 15.661123386-15.617435494= 0.043687892

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

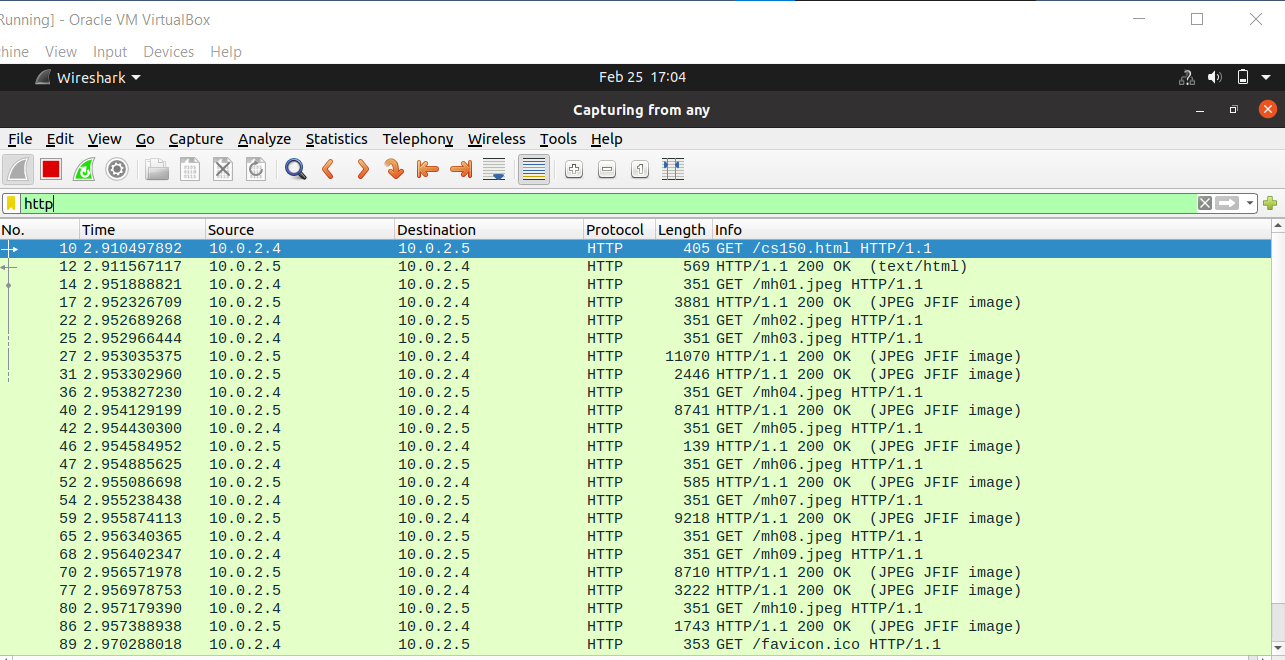
Step 4: Repeat the **steps 1-3** in the previous section.



Here is it 17.974775698 – 17.927004948 = 0.04777075

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

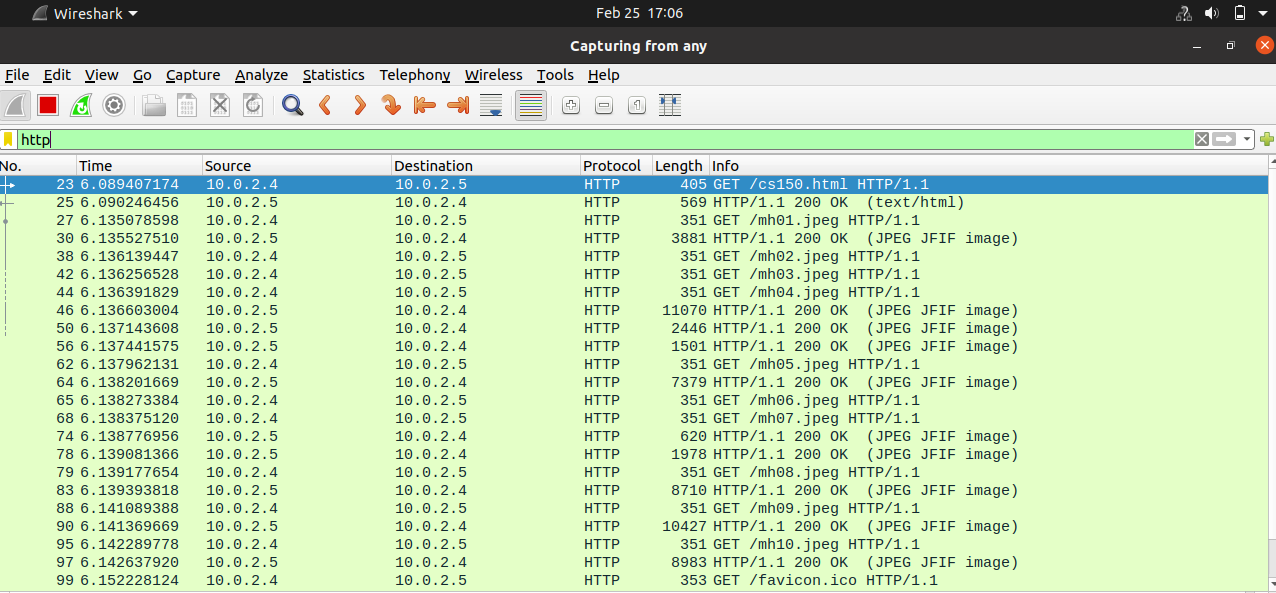
Step 6: Repeat the **steps 1-3** in the previous section.



Here it is 2.957388938- 2.910497892= 0.046891046

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

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



Here it is 6.142637920-6.089407174= 0.053230746