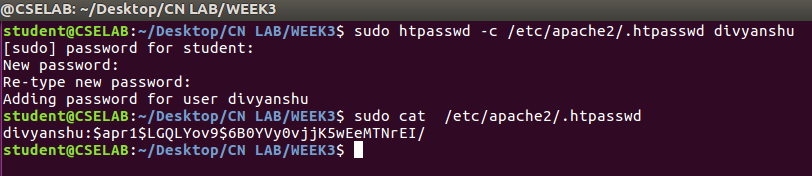
**Computer Network Lab – WEEK 3**

**PES1UG20CS806**

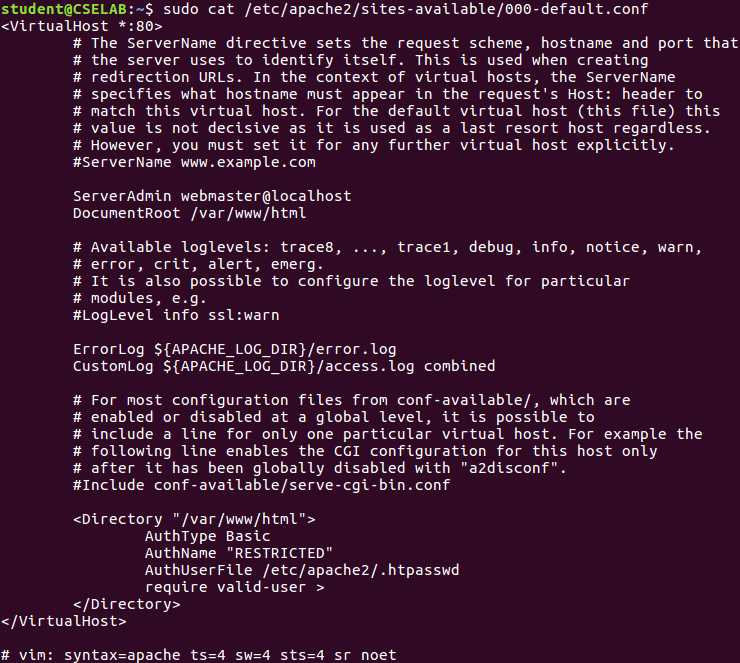
**Divyanshu Sharma**

1. **Password Authentication**
   1. **Password Generation:**

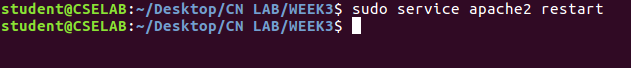
* To enable basic authentication for HTTP, we need to generate a password file. This file can be generated using the **htpasswd** command.
* Using **sudo htpasswd -c /etc/apache2/.htpasswd** **username** we can set a password for the given user username and write it into the .htpasswd configuration file
* The **cat command** can be used to view the encrypted password file, which is encrypted using the Data Encryption Standard algorithm



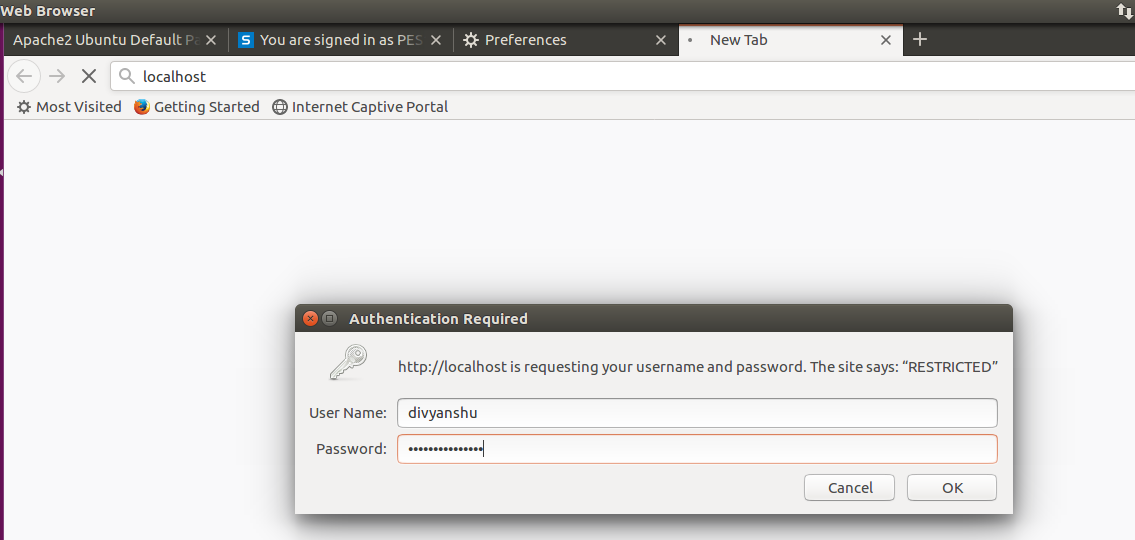
* 1. **Apache Server Authentication**
* To enable password authentication in the server, we need to modify the Apache configuration file.
* This can be done using **sudo nano /etc/apache2/sites-available/000default.conf**
* Password authentication is added to **the /var/www/html directory** which is the localhost home directory so that all files hosted here will require authentication to access.
* To activate the authentication and policy, we need to restart the server using **sudo service apache2 restart**



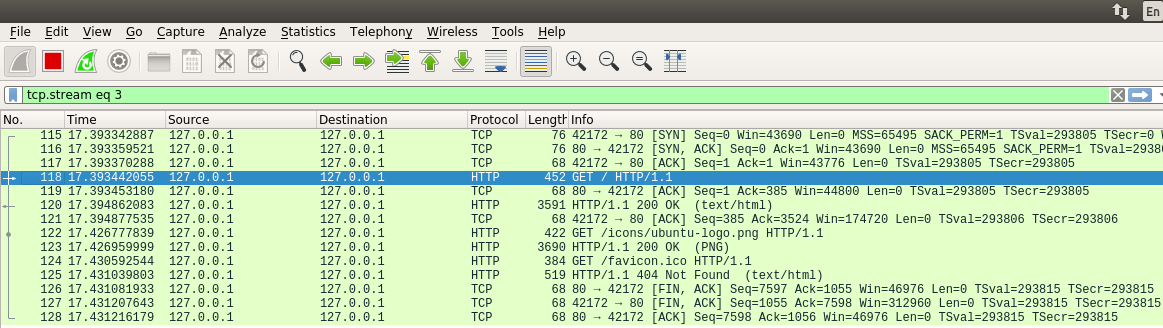
* Password policy implementation is done by restarting the server as: sudo service apache2 restart

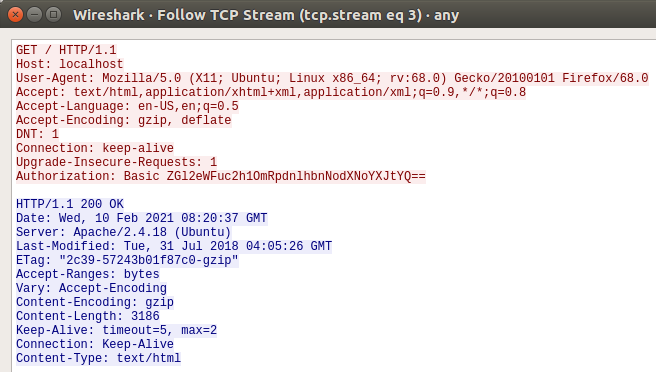


* 1. **Accessing Localhost:**
* We can now access localhost only after entering the username and password set earlier
* These credentials are entered on the browser window.



* 1. **Wireshark Packet Capture:**
* Wireshark can be used to capture the packets sent on the network. The f**irst GET request** corresponding to the HTML file is analyzed and its TCP Stream is expanded, and parameters examined.





* 1. **Decrypting Base64 Encryption:**
* We can observe that the Authorization field stores the password we had entered to access localhost.
* This password is encrypted using the Base64 algorithm before it is transmitted along the network.
  + Each character is **converted into 8-bit binary ASCII representation**
  + Group these bits into **chunks of 6-bits**.
  + **Convert these chunks into their decimal equivalent** and assign the corresponding Base64 character
  + The Base64 algorithm supports the use of lowercase as well as uppercase alphabets, all digits from 0 to 9 and the special characters + and / only.
    - Similarly, Base64 is decoded by obtaining the 6-bit binary chunks for each character, grouping them into chunks of 8-bits and then converting into their corresponding character.

**ZGl2eWFuc2h10mRpdnlhbnNodXNoYXJtYQ**== can be first converted to a 6-bit binary equivalent

Z 011001

G 000110

l 100101

2 110110

e 011110

W 010110

F 000101

u 101110

c 011100

2 110110

h 100001

1 110101

0 110100

m 100110

R 010001

p 101001

d 011101

n 100111

l 100101

h 100001

b 011011

n 100111

N 001101

o 101000

d 011101

X 010111

N 001101

o 101000

Y 011000

X 010111

J 001001

t 101101

Y 011000

Q 010000

* **These binary equivalents can then be grouped together and then decoded to ASCI**

**01100100 d**

**01101001 i**

**01110110 v**

**01111001 y**

**01100001 a**

**01101110 n**

**01110011 s**

**01101000 h**

**01110101 u**

**01110011 s**

**01101000 h**

**01100001 a**

**01110010 r**

**01101101 m**

**01100001 a**

1. **Setting Cookies**
   1. **Setting Cookies with PHP:**

* We can set cookies using a PHP script and the **setcookie(name, value, expire\_time) function**
* When this file is requested by the browser a cookie will be set

<html>

<?php

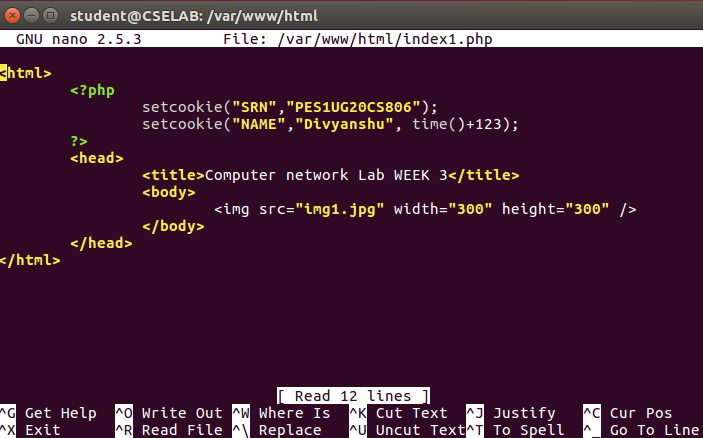
setcookie("SRN","PES1UG20CS806”);

setcookie("NAME","Divyanshu", time()+123);

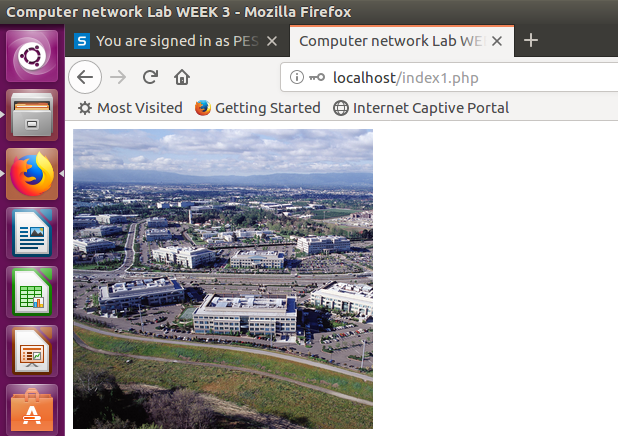
?>

<img src= “img1.jpg” width= “300” height= “300” />

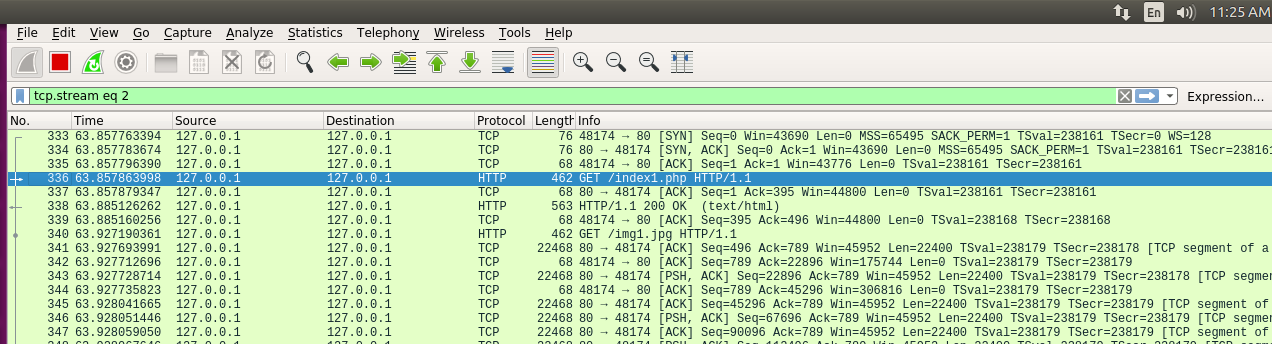
</html>

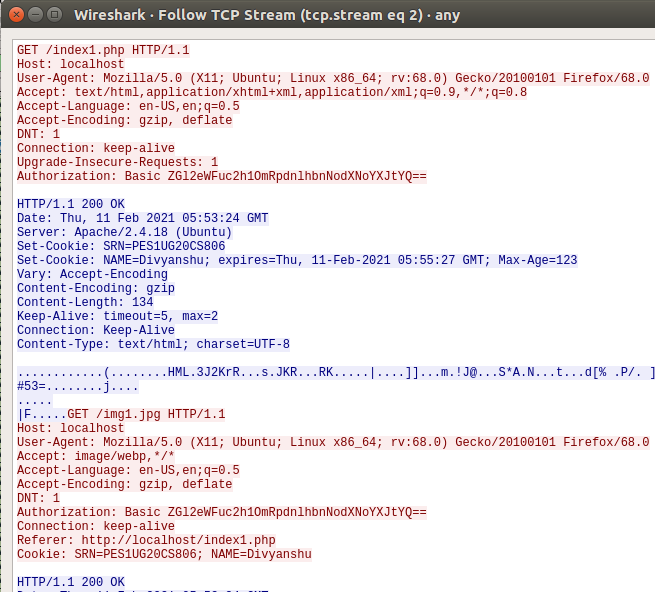


* The combined file saved with a .php extension is placed under /var/www/html for accessing.



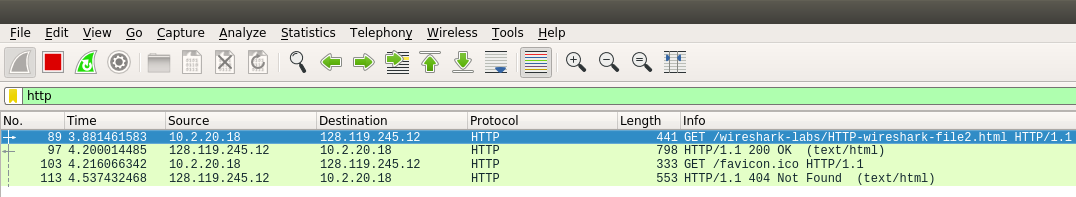
* 1. **Wireshark Capture**
* Wireshark can be used to capture the packets sent on the network. The first GET request corresponding to the PHP file is analyzed and its TCP Stream is expanded and examined.
* The Cookie name, value and the associated parameters can be viewed under the HTTP header Set-Cookie.
* We can observe the name, value, and the expiry time of the set cookie, if the cookie has not already expired.

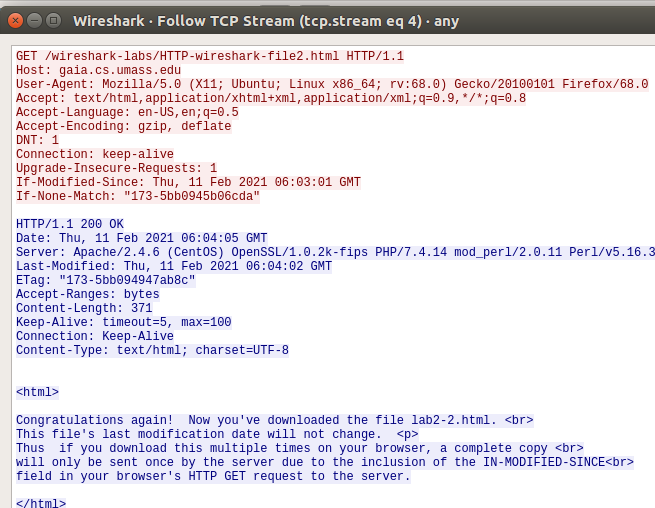




1. **Conditional GET**

* A conditional HTTP response is one that carries the resource only it had been modified since the last GET request by the client.
* The HTTP header **If-Modified**-Since is one way to implement Conditional GET
* The server checks the If-Modified-Since header value and resends the resource only if it has been modified since the timestamp in the header
* If it has not been modified, **a 304 Not Modified** status code is sent back.
  1. **Repeat Requests for HTML Page**
* An HTML page is requested by the client and the HTML file is obtained along with a 200 OK response status
* Immediately, the request is made again either by refreshing or accessing it via a browser tab
* The second response from the server is obtained as 304 Not Modified since the resource has not been modified since the last GET.





* 1. **Conditional GET on Localhost**
* A simple HTML file with 2 images is placed in the localhost home directory.
* From a browser, a request is made for the file, which receives a response of 200 OK with both images being sent by the server.
* When the request is sent again, the 304 Not Modified status code is sent and images are not sent back.

