

COMPUTER NETWORKS LABORATORY

By:

Nitish S

PES2201800368

5 'A'

WEEK – 3- Understanding Working of HTTP Headers

Date: 14/09/2020

Understand working of HTTP headers:

Conditional Get: If-Modified-Since

HTTP Cookies: Cookie and Set-Cookie

Authentication: Auth-Basic

Design a web page that has one embedded page (e.g. image) and sets a cookie and enables authentication. You are required to configure the web server (e.g. apache) with authentication mechanism.

Show the behavior of conditional get when embedded objects is modified and when it

is not (you can just change the create date of the embedded object). Decode the Basic-Auth header using Base64 mechanism as per the password setup.

Observation: Show the behavior of browser when is cookie is set and when cookie is removed.

Solution: Analyzing Basic Authentication and Cookies

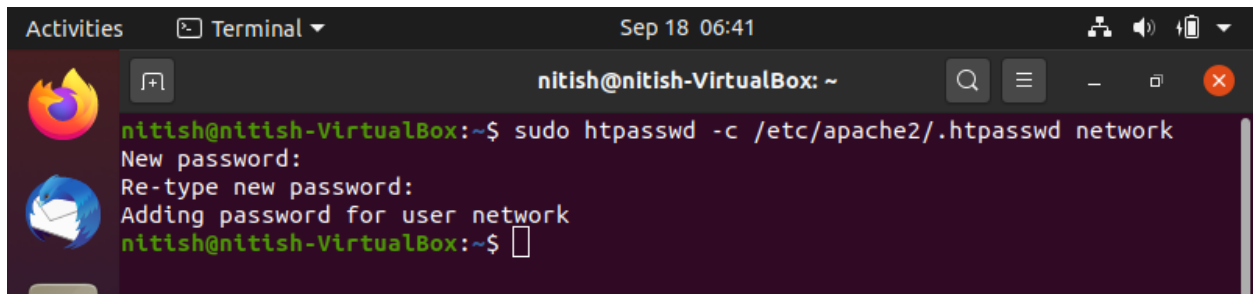
The three parts of experiment are:

1. Password Authentication
2. Cookie Setting
3. Conditional get

Steps of Execution (for Password Authentication):

- 1) `sudo apt-get update`
- 2) `sudo apt-get install apache2 apache2-utils`
- 3) **Provide username and password to set authentication :**

`sudo htpasswd -c /etc/apache2/.htpasswd network`

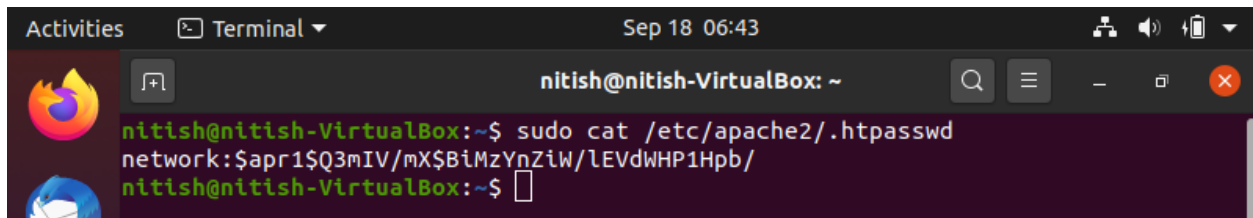


A terminal window titled 'nitish@nitish-VirtualBox: ~' showing the execution of the command `sudo htpasswd -c /etc/apache2/.htpasswd network`. The output shows the prompt for a new password, followed by the prompt to re-type the password, and then the message 'Adding password for user network'. The prompt returns to the shell.

```
nitish@nitish-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd network
New password:
Re-type new password:
Adding password for user network
nitish@nitish-VirtualBox:~$
```

- 4) **Viewing the authentication :**

`sudo cat /etc/apache2/.htpasswd`



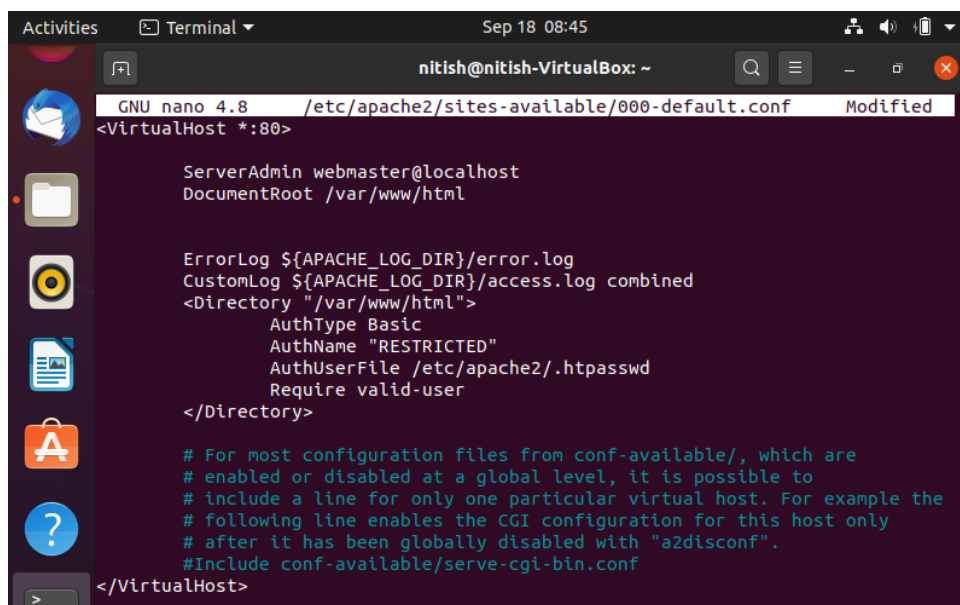
A terminal window titled 'nitish@nitish-VirtualBox: ~' showing the execution of the command `sudo cat /etc/apache2/.htpasswd`. The output displays the password entry for the user 'network' in the format 'network:\$apr1\$Q3mIV/mX\$BiMzYnZiW/LEvdWHP1Hpb/'.

```
nitish@nitish-VirtualBox:~$ sudo cat /etc/apache2/.htpasswd
network:$apr1$Q3mIV/mX$BiMzYnZiW/LEvdWHP1Hpb/
nitish@nitish-VirtualBox:~$
```

- 5) **Configuring Access control within the Virtual Host Definition:**

Opening the file for setting authentication:

`sudo nano /etc/apache2/sites-available/000-default.conf`



A screenshot of the nano text editor editing the file `/etc/apache2/sites-available/000-default.conf`. The editor shows the configuration for a virtual host, including the `<VirtualHost *:80>` block. Inside this block, the `ServerAdmin` and `DocumentRoot` are set. The `ErrorLog` and `CustomLog` are also configured. The `<Directory "/var/www/html">` block contains the authentication configuration: `AuthType Basic`, `AuthName "RESTRICTED"`, `AuthUserFile /etc/apache2/.htpasswd`, and `Require valid-user`. The editor also shows comments about including configuration files from `conf-available/`.

```
GNU nano 4.8 /etc/apache2/sites-available/000-default.conf Modified
<VirtualHost *:80>

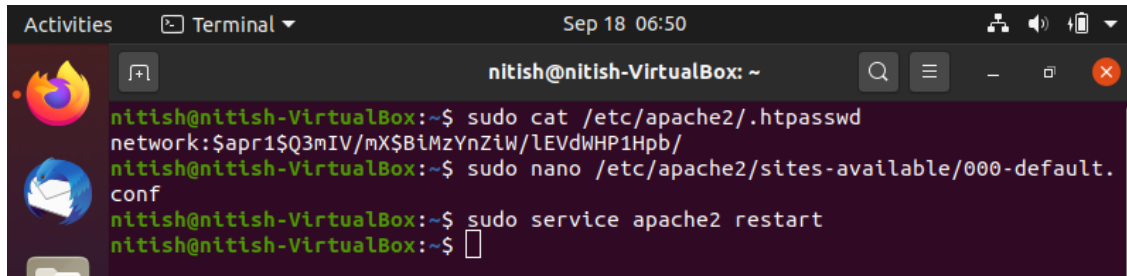
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    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>

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

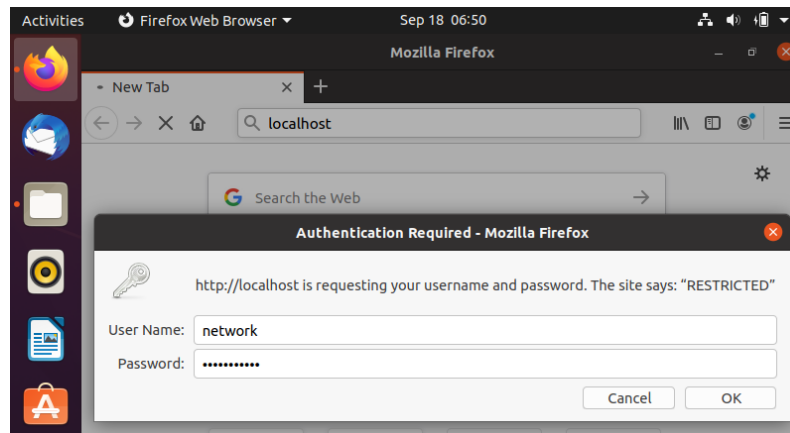
Password policy implementation is done by restarting the server as:

`sudo service apache2 restart`

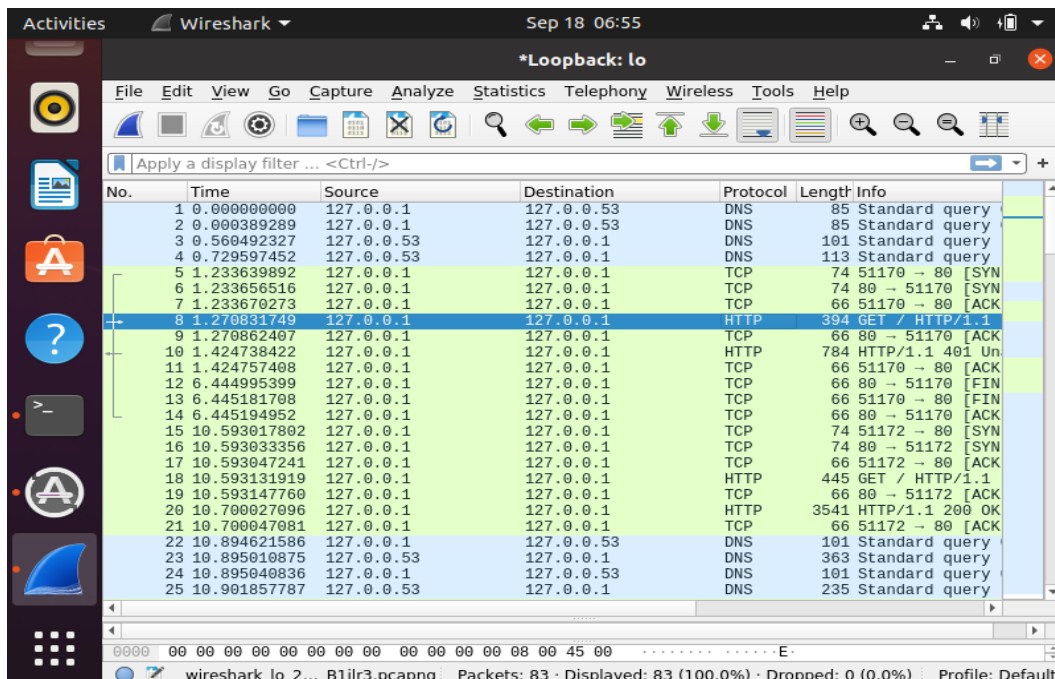


```
nitish@nitish-VirtualBox: ~  
nitish@nitish-VirtualBox:~$ sudo cat /etc/apache2/.htpasswd  
network:$apr1$Q3mIV/mX$8iMzYnZiW/LEVdWHP1Hpb/  
nitish@nitish-VirtualBox:~$ sudo nano /etc/apache2/sites-available/000-default.conf  
nitish@nitish-VirtualBox:~$ sudo service apache2 restart  
nitish@nitish-VirtualBox:~$
```

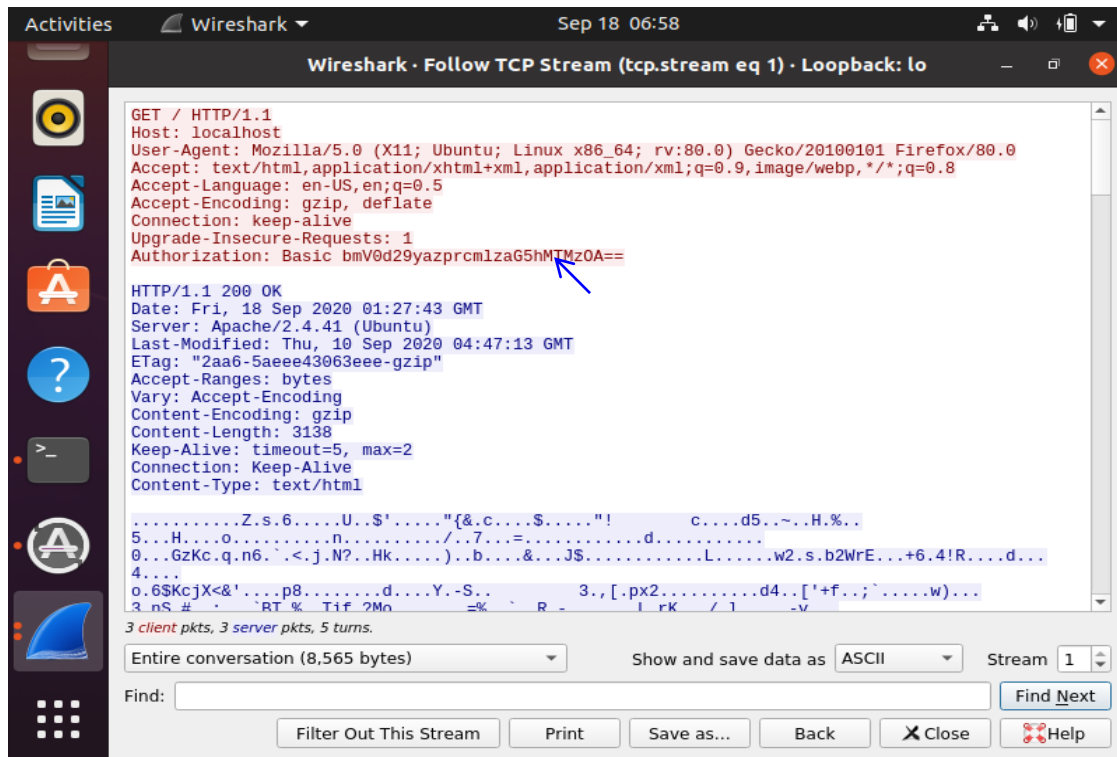
The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase.



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



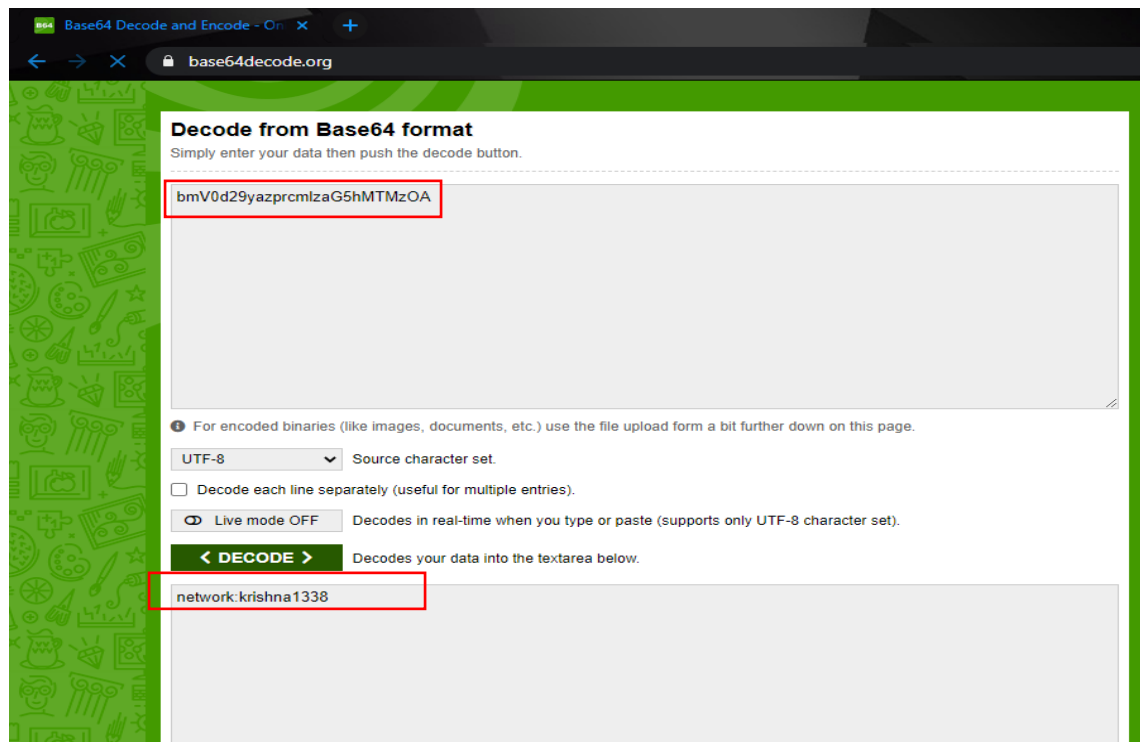
Using the “follow TCP stream” on the HTTP message segment the password was retrieved which was encrypted by the base64 algorithm and decryption could be done with same algorithm.



Decoding the password from Base64 format: Base64 format obtained from Wireshark capture and decoded using the website <https://www.base64decode.org>.

Base64 format: bmV0d29yazprcmlzaG5hMTMzOA

Decoded password: krishna1338



COOKIE SETTING:-

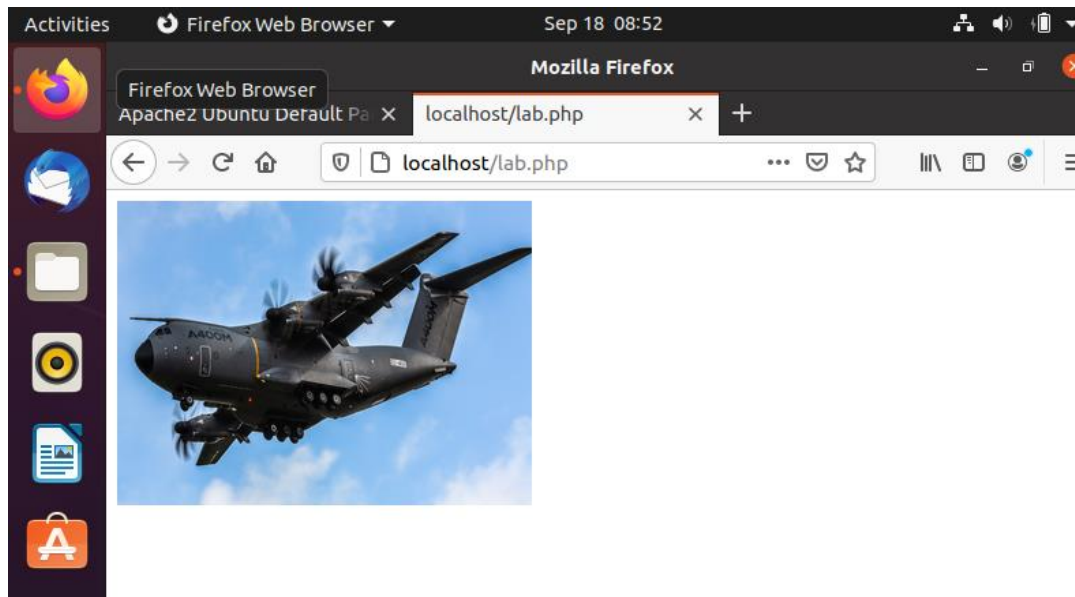
1) Creation of Php file:

A screenshot of a text editor window titled 'lab.php' with the path '/var/www/html'. The code is as follows:

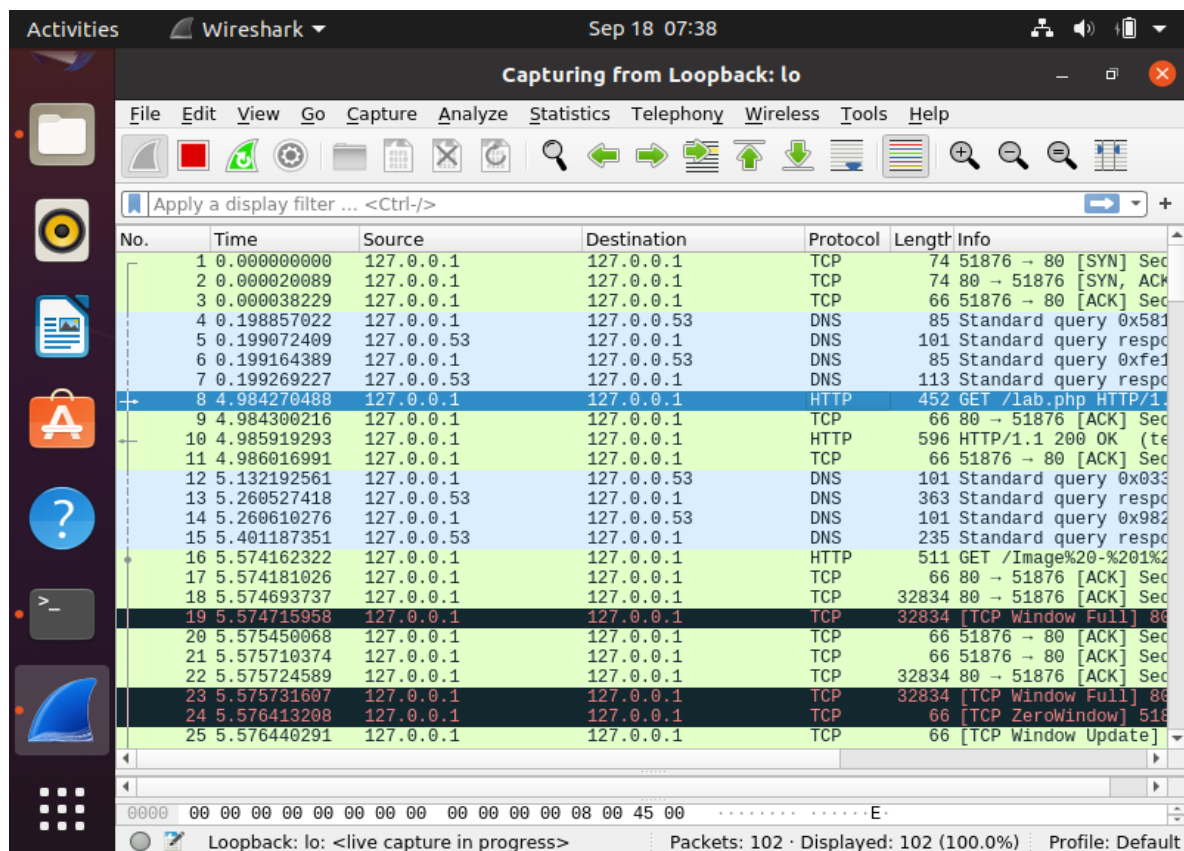
```
1 <html>
2 <?php
3 setcookie("namecookie","netqwerty",time()+123);
4 setcookie("nickname","work");
5 ?>
6 
7 </html>
```

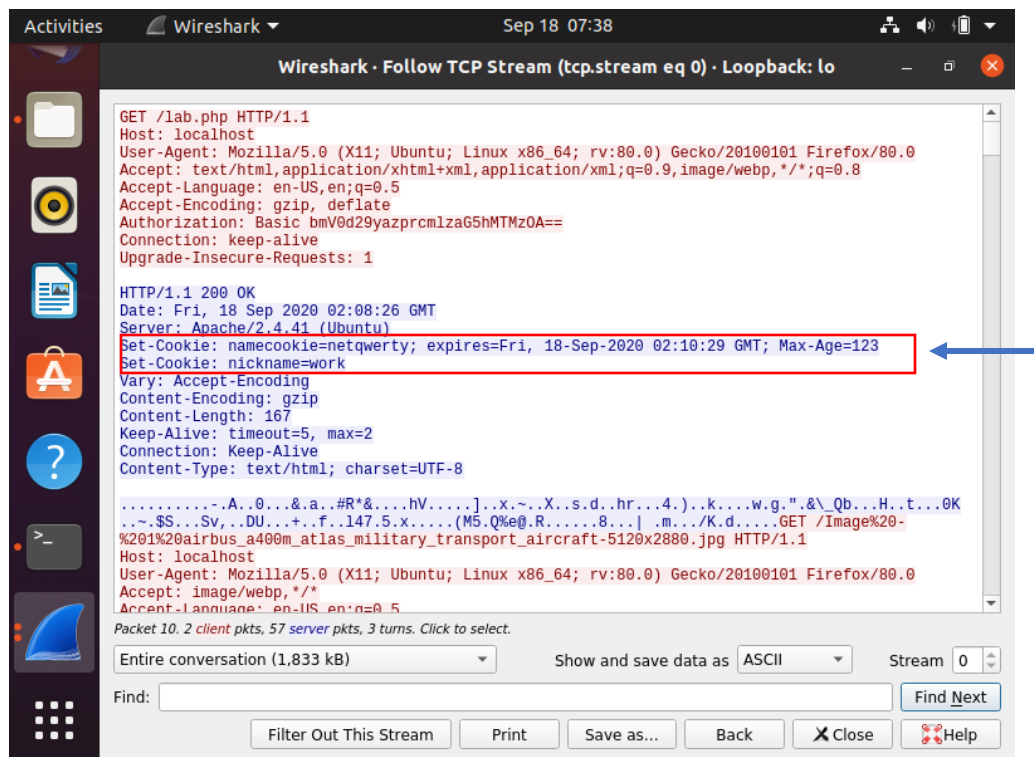
Note: You can capture Cookies mostly during the first time of web access. Hence keep wireshark capture ready before executing the task for the first time.

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.



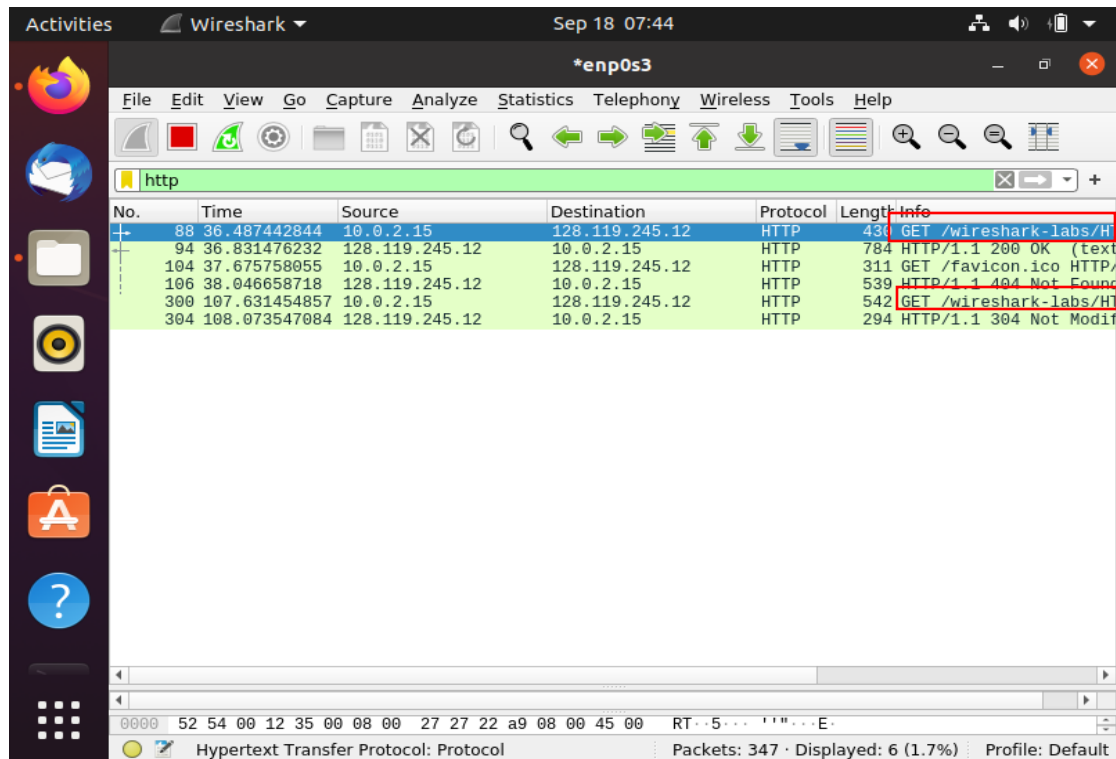


The cookie is set as shown in the above screenshot.

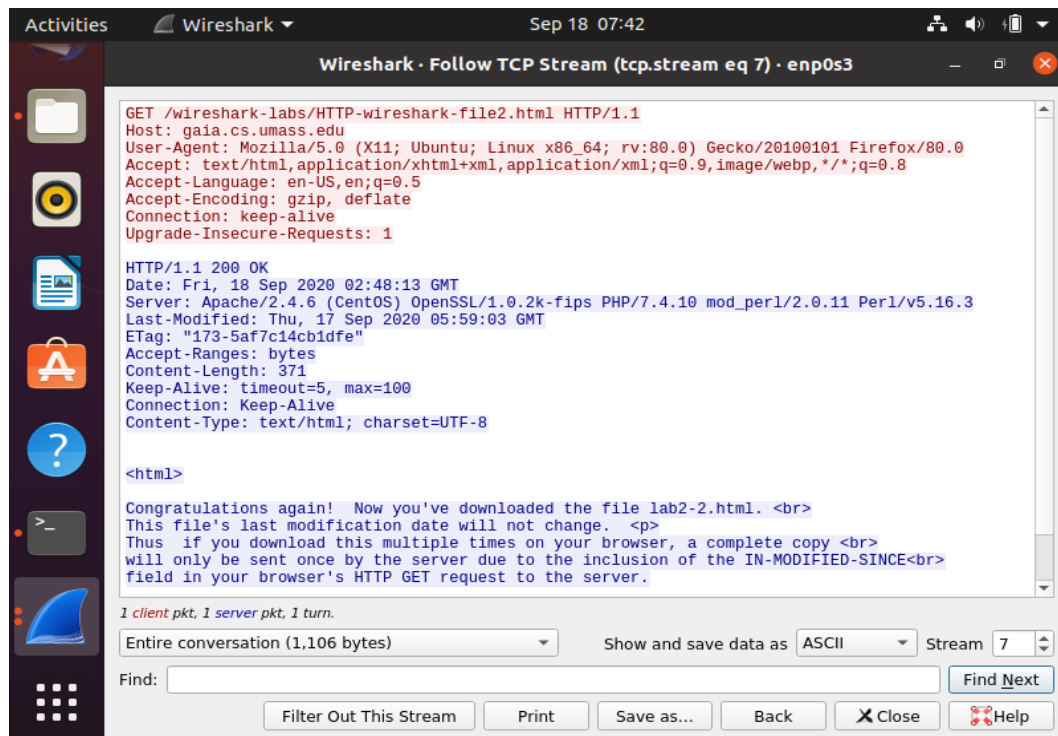
Conditional Get: If-Modified-Since:-

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser <http://gaia.cs.umass.edu/wiresharklabs/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 packet-listing window.

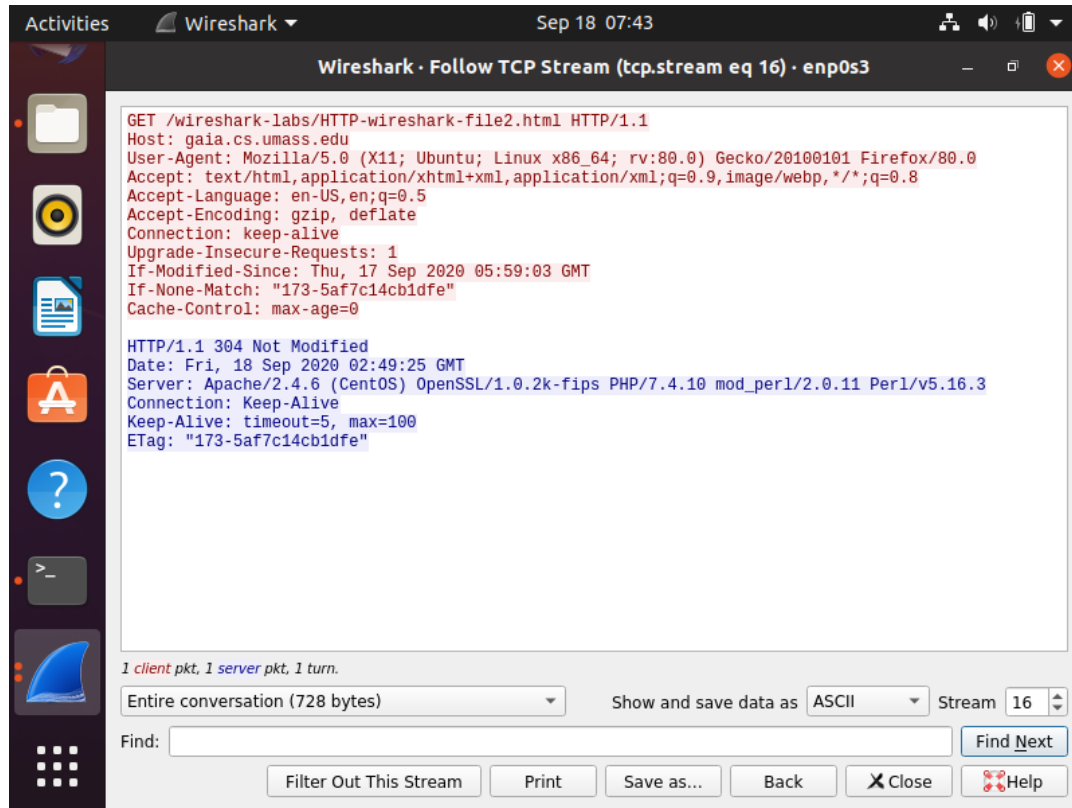
OBSERVATIONS:



First HTTP GET request:-



Second HTTP GET request:-



1. 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?

There won't be an 'IF-MODIFIED-SINCE' line in the HTTP GET request because the HTTP page is being requested for the first time, it is not accessed before and hence is not stored and retrieved from the proxy server and so is directly retrieved from the origin server.

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

The server did explicitly return the contents of the file. Wireshark includes a section titled “Line-Based Text Data” which shows what the server sent back to the browser which is specifically what the website showed it was requested.

3. 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, in the second HTTP message the IF-MODIFIED-SINCE line is included. The information that follows is the date and time that the webpage was accessed previously.

4. 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 HTTP status code and phrase is “304: Not Modified”
- The server did not return the contents of the file but the browser simply retrieved the contents from its cache/proxy server. Had the file been modified since it was last accessed, it would have returned the contents of the file retrieved from the origin server.