

Week: 3

Understanding Working of HTTP Headers

Question: 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 are modified and when it is not (you can just change the create date of the embedded object). Decode the BasicAuth 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. 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**

```
itsatul@Pearl:~$ sudo apt-get install apache2.utils
[sudo] password for itsatul:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'apache2-utils' for regex 'apache2.utils'
apache2-utils is already the newest version (2.4.41-4ubuntu3.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
itsatul@Pearl:~$
```

--> Provide username and password to set authentication **sudo htpasswd -c /etc/apache2/.htpasswd ANY_USERNAME**

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

--> View the authentication **sudo cat /etc/apache2/.htpasswd**

```
itsatul@Pearl:~$ sudo htpasswd -c /etc/apache2/.htpasswd atul
New password:
Re-type new password:
Adding password for user atul
itsatul@Pearl:~$ sudo cat /etc/apache2/.htpasswd
atul:$apr1$f/os02i2$hVezen9vI4NVzZ0BLHdKs1
itsatul@Pearl:~$
```

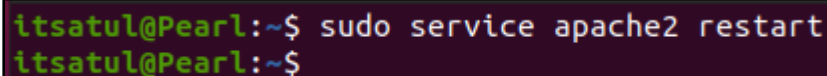
2. To setup the authentication phase, execute the following commands. Configuring Access control within the Virtual Host Definition.

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



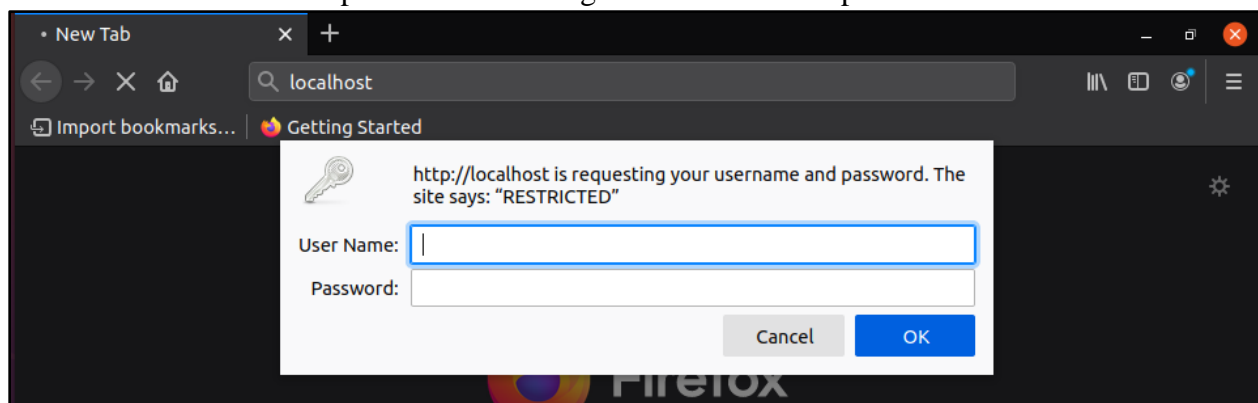
```
1 <VirtualHost *:80>
2
3     ServerAdmin webmaster@localhost
4     DocumentRoot /var/www/html
5
6     ErrorLog ${APACHE_LOG_DIR}/error.log
7     CustomLog ${APACHE_LOG_DIR}/access.log combined
8     <Directory "/var/www/html">
9         AuthType Basic
10        AuthName "RESTRICTED"
11        AuthUserFile /etc/apache2/.htpasswd
12        Require valid-user >
13    </Directory>
14
15 </VirtualHost>
16
17 # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

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



```
itsatul@Pearl:~$ sudo service apache2 restart
itsatul@Pearl:~$
```

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.

No.	Time	Source	Destination	Protocol	Length	Info
85	11.708151447	127.0.0.1	127.0.0.1	TCP	76	33608 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
86	11.708164081	127.0.0.1	127.0.0.1	TCP	76	80 → 33608 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS
87	11.708172688	127.0.0.1	127.0.0.1	TCP	68	33608 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=42
88	11.708216050	127.0.0.1	127.0.0.1	HTTP	431	GET / HTTP/1.1
89	11.708226230	127.0.0.1	127.0.0.1	TCP	68	80 → 33608 [ACK] Seq=1 Ack=364 Win=65152 Len=0 TSval=
90	11.709330689	127.0.0.1	127.0.0.1	HTTP	3545	HTTP/1.1 200 OK (text/html)
91	11.709351748	127.0.0.1	127.0.0.1	TCP	68	33608 → 80 [ACK] Seq=364 Ack=3478 Win=63232 Len=0 TSv
104	12.030439013	127.0.0.1	127.0.0.1	HTTP	390	GET /icons/ubuntu-logo.png HTTP/1.1
105	12.030463480	127.0.0.1	127.0.0.1	TCP	68	80 → 33608 [ACK] Seq=3478 Ack=686 Win=65280 Len=0 TSv
106	12.030638343	127.0.0.1	127.0.0.1	HTTP	3691	HTTP/1.1 200 OK (PNG)
107	12.030644926	127.0.0.1	127.0.0.1	TCP	68	33608 → 80 [ACK] Seq=686 Ack=7101 Win=63104 Len=0 TSv
116	12.067434843	127.0.0.1	127.0.0.1	HTTP	380	GET /favicon.ico HTTP/1.1
117	12.067457727	127.0.0.1	127.0.0.1	TCP	68	80 → 33608 [ACK] Seq=7101 Ack=998 Win=65280 Len=0 TSv
118	12.067814511	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
119	12.067829770	127.0.0.1	127.0.0.1	TCP	68	33608 → 80 [ACK] Seq=998 Ack=7588 Win=65152 Len=0 TSv

Frame 88: 431 bytes on wire (3448 bits), 431 bytes captured (3448 bits) on interface any, id 0
 Linux cooked capture
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 Transmission Control Protocol, Src Port: 33608, Dst Port: 80, Seq: 1, Ack: 1, Len: 363
 Hypertext Transfer Protocol

```

0000  00 00 03 04 00 06 00 00 00 00 00 00 22 52 08 00  .....R...
0010  45 00 01 9f 17 9b 40 00 40 06 23 00 7f 00 00 01  E....@.0.0...
0020  7f 00 00 01 83 48 00 50 1e 24 c3 0f f6 4b 1e 2b  ....H.P..$..K.+
0030  80 18 02 00 ff 93 00 00 01 01 08 0a 19 64 4d 4a  .........dMJ
0040  19 64 4d 4a 47 45 54 20 2f 20 48 54 54 50 2f 31  -dMJGET / HTTP/1
0050  2e 31 0d 0a 48 6f 73 74 3a 20 6c 6f 63 61 6c 68  .1..Host : localh
  
```

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.

```

Wireshark · Follow TCP Stream (tcp.stream eq 17) · any
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101
Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Authorization: Basic YXR1bDppY2U=

HTTP/1.1 200 OK
Date: Sat, 24 Apr 2021 12:10:59 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Sun, 18 Apr 2021 12:43:05 GMT
ETag: "2aa6-5c03e8dedf8b3-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 3138
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

.....Z.s.6....U..$'....."(&.c....$......"! c....d5...~..H.%..
5...H...0....n...../..7...F.....d.....
0...GzKc.q.n6...<.j.N?...Hk.....).b...&...J$......L.....w2.s.b2WrE...
+6.4!R...d....4....
o.6$Kc]X<&'...p8.....d....Y..-S... 3.,[.px2.....d4..
[!+!...;...w)...3.nS.#.....BT.%..Tif.?Mo.....=%....R...L.rK.
./..1...-v...
F...l*.{d.bN.{:R.%..z..g.aE..hL....K....y.h)...7.....su.sXY.
{yd..ht.P2K.A$.Tc.....>..
.t..vL..TL..'.<e.....U...F..S.n...*.....L.R..|(O.R0\..t.7&. j..
921.....^A.
..rLF...&.n..4F..N...}.fLhfd.
68..r.....x.'9...;7.....#..c.....~...h.;u..
..1sv.....)Dn.7.?...:e?.!.....6..."5F.K|
^..0a...H.....1$.m.....p./.....xu.v.
f v 1/ 1 c i F c -Ts? N # h * E V <e>vA \II f f
3 client pkts, 3 server pkts, 5 turns.
Entire conversation (8,584 bytes) Show and save data as ASCII Stream 17
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help
  
```

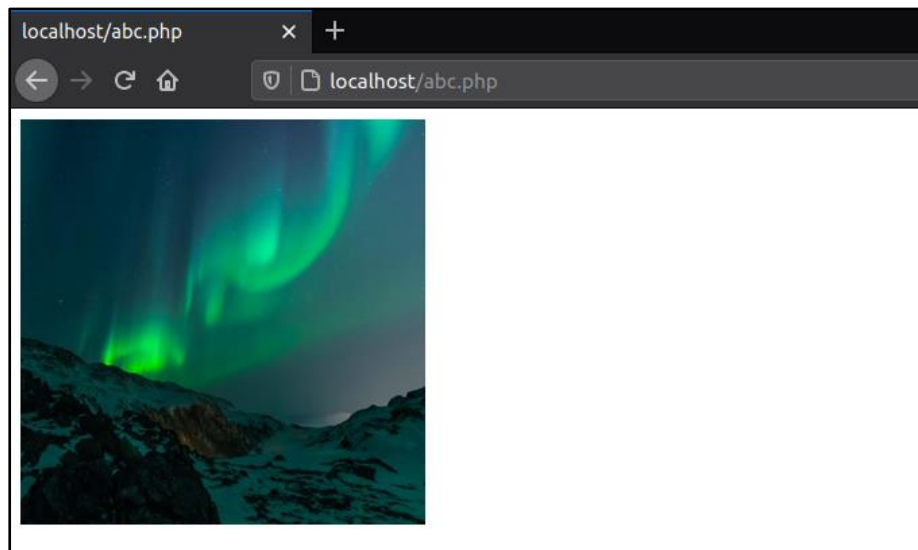
Steps of Execution (Cookie Setting)

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:

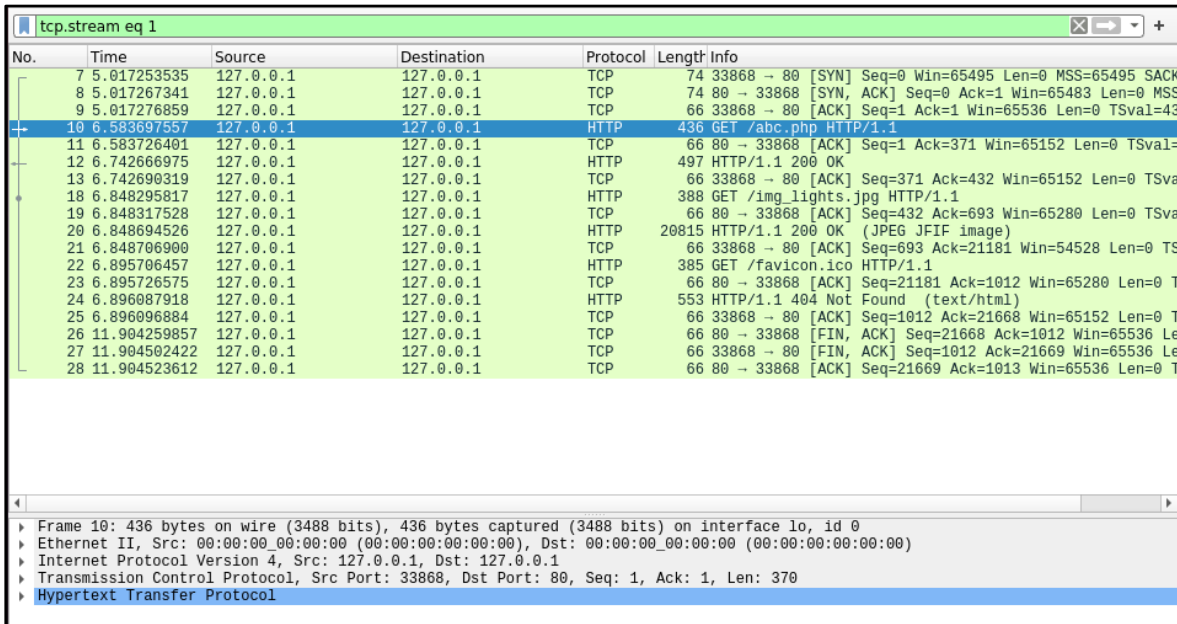
```
Open  abc.php /var/www/html
1 <html>
2 <?php
3 setcookie("namecookie","netqwerty",time()+123);
4 setcookie("nickname","work");
5 ?>
6 
7 </html>
```

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.

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

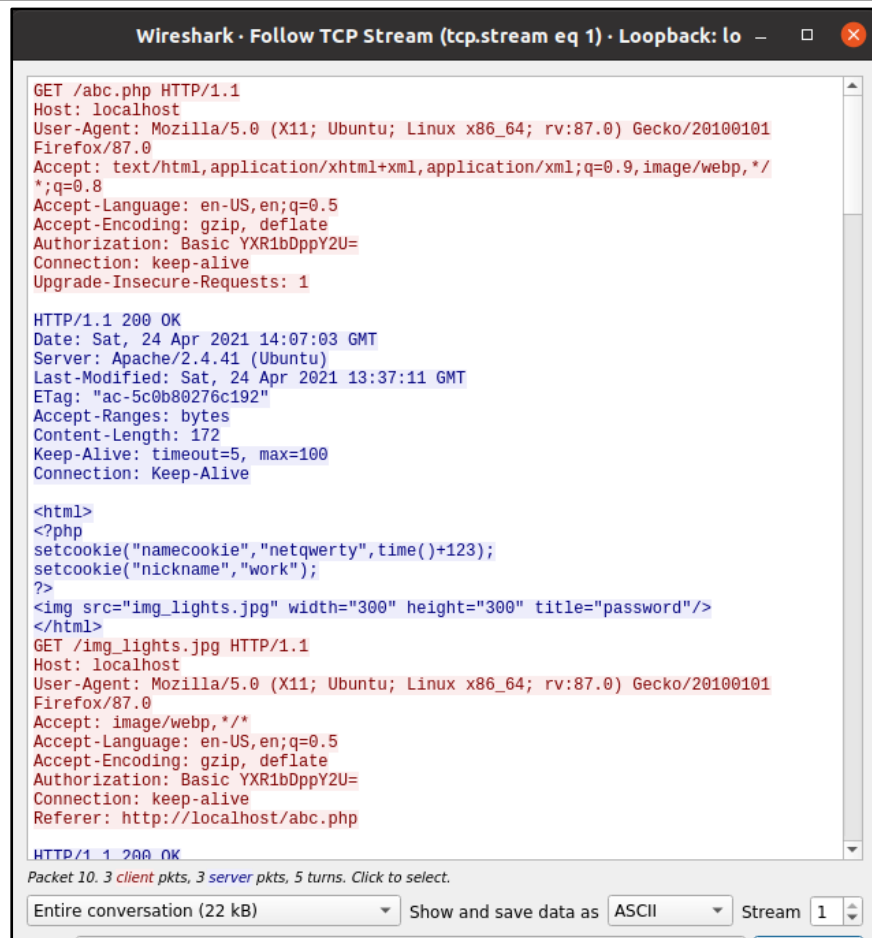


3. The packets are captured using Wireshark and using the “follow TCP stream” which checks for the set-cookie field whether the cookie is set or not set.



No.	Time	Source	Destination	Protocol	Length	Info
7	5.017253535	127.0.0.1	127.0.0.1	TCP	74	33868 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
8	5.017267341	127.0.0.1	127.0.0.1	TCP	74	80 → 33868 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS
9	5.017276859	127.0.0.1	127.0.0.1	TCP	66	33868 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=43
10	6.583697557	127.0.0.1	127.0.0.1	HTTP	436	GET /abc.php HTTP/1.1
11	6.583726401	127.0.0.1	127.0.0.1	TCP	66	80 → 33868 [ACK] Seq=1 Ack=371 Win=65152 Len=0 TSval=
12	6.742666975	127.0.0.1	127.0.0.1	HTTP	497	HTTP/1.1 200 OK
13	6.742690319	127.0.0.1	127.0.0.1	TCP	66	33868 → 80 [ACK] Seq=371 Ack=432 Win=65152 Len=0 TSva
18	6.848295817	127.0.0.1	127.0.0.1	HTTP	388	GET /img_lights.jpg HTTP/1.1
19	6.848317528	127.0.0.1	127.0.0.1	TCP	66	80 → 33868 [ACK] Seq=432 Ack=693 Win=65280 Len=0 TSva
20	6.848694526	127.0.0.1	127.0.0.1	HTTP	20815	HTTP/1.1 200 OK (JPEG JFIF image)
21	6.848706900	127.0.0.1	127.0.0.1	TCP	66	33868 → 80 [ACK] Seq=693 Ack=21181 Win=54528 Len=0 TS
22	6.895706457	127.0.0.1	127.0.0.1	HTTP	385	GET /favicon.ico HTTP/1.1
23	6.895726575	127.0.0.1	127.0.0.1	TCP	66	80 → 33868 [ACK] Seq=21181 Ack=1012 Win=65280 Len=0 T
24	6.896087918	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 404 Not Found (text/html)
25	6.896096884	127.0.0.1	127.0.0.1	TCP	66	33868 → 80 [ACK] Seq=1012 Ack=21668 Win=65152 Len=0 T
26	11.904259857	127.0.0.1	127.0.0.1	TCP	66	80 → 33868 [FIN, ACK] Seq=21668 Ack=1012 Win=65536 Le
27	11.904502422	127.0.0.1	127.0.0.1	TCP	66	33868 → 80 [FIN, ACK] Seq=1012 Ack=21669 Win=65536 Le
28	11.904523612	127.0.0.1	127.0.0.1	TCP	66	80 → 33868 [ACK] Seq=21669 Ack=1013 Win=65536 Len=0 T

Frame 10: 436 bytes on wire (3488 bits), 436 bytes captured (3488 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 33868, Dst Port: 80, Seq: 1, Ack: 1, Len: 370
Hypertext Transfer Protocol



```
GET /abc.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic YXR1bDppY2U=
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Sat, 24 Apr 2021 14:07:03 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Sat, 24 Apr 2021 13:37:11 GMT
ETag: "ac-5c0b80276c192"
Accept-Ranges: bytes
Content-Length: 172
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

<html>
<?php
setcookie("namecookie","netqwerly",time()+123);
setcookie("nickname","work");
?>

</html>
GET /img_lights.jpg HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic YXR1bDppY2U=
Connection: keep-alive
Referer: http://localhost/abc.php

HTTP/1.1 200 OK
```

Packet 10. 3 client pkts, 3 server pkts, 5 turns. Click to select.

Entire conversation (22 kB) Show and save data as ASCII Stream 1

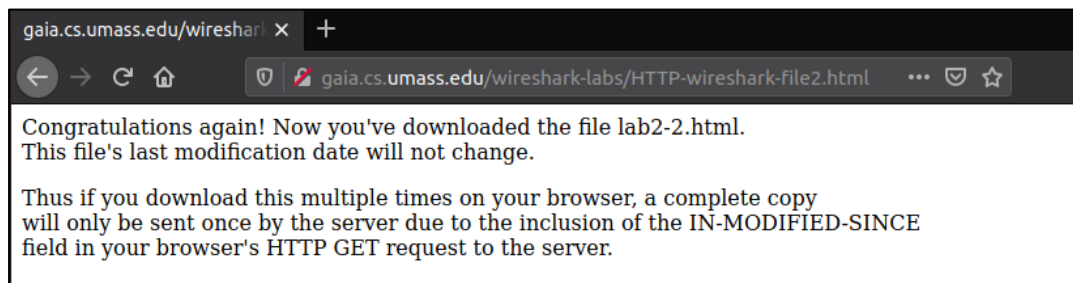
The cookie is set as shown in the above screenshot.

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>
- 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.



The screenshot shows the Wireshark packet capture interface. The top pane displays a list of captured packets, filtered by 'http'. The bottom pane shows the details of the selected packet (No. 38).

No.	Time	Source	Destination	Protocol	Length	Info
12	2.969924783	10.0.2.15	128.119.245.12	HTTP	432	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
18	3.331530132	128.119.245.12	10.0.2.15	HTTP	786	HTTP/1.1 200 OK (text/html)
28	3.511708162	10.0.2.15	128.119.245.12	HTTP	389	GET /favicon.ico HTTP/1.1
32	3.834872489	128.119.245.12	10.0.2.15	HTTP	541	HTTP/1.1 404 Not Found (text/html)
38	4.250215136	10.0.2.15	128.119.245.12	HTTP	544	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
42	4.581596515	128.119.245.12	10.0.2.15	HTTP	295	HTTP/1.1 304 Not Modified

Frame 38: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 45922, Dst Port: 80, Seq: 377, Ack: 731, Len: 488
Hypertext Transfer Protocol

```
Wireshark · Follow TCP Stream (tcp.stream eq 2) · any

GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101
Firefox/87.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Sat, 24 Apr 2021 14:33:49 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.14 mod_perl/2.0.11
Perl/v5.16.3
Last-Modified: Sat, 24 Apr 2021 05:59:01 GMT
ETag: "173-5c0b19bf7a23b"
Accept-Ranges: bytes
Content-Length: 371
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html>
Congratulations again! Now you've downloaded the file lab2-2.html. <br>
This file's last modification date will not change. <p>
Thus if you download this multiple times on your browser, a complete copy <br>
will only be sent once by the server due to the inclusion of the IN-MODIFIED-
SINCE<br>
field in your browser's HTTP GET request to the server.

</html>
```