

Computer Networks Lab – Week 3

SRN : PES1UG19CS542

Name: Trisha Jain

Understanding Working of HTTP Headers

I. Password Authentication

1. To generate a password file in order to enable the authentication for HTTP we use the **htpasswd** command.
2. Using the cat command we view the encrypted password (Data Encryption Standard Algorithm).

```
trisha@trisha-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd trisha
[sudo] password for trisha:
New password:
Re-type new password:
Adding password for user trisha
trisha@trisha-VirtualBox:~$ sudo cat /etc/apache2/.htpasswd
trisha:$apr1$u9hwHt1v$KGNroyhSX/P/0zR5yER3N/
trisha@trisha-VirtualBox:~$
```

sudo htpasswd -c /etc/apache2/.htpasswd

3. For enabling password authentication in the server, the Apache configuration file needs to be modified.

The authentication was added to the **/var/www/html** directory (localhost home directory).

Then the server is restarted using the command :

sudo service apache2 restart

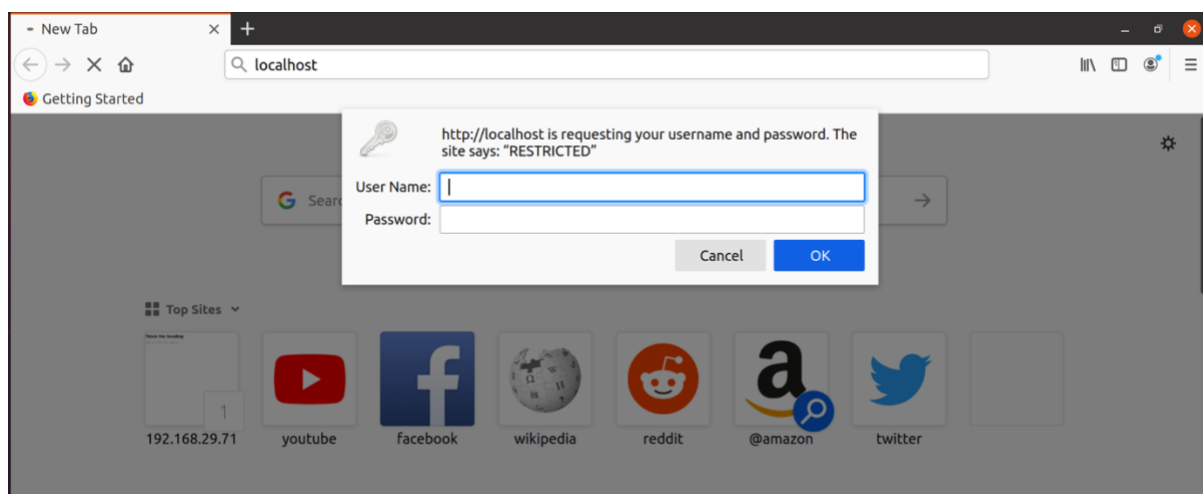
```
trisha@trisha-VirtualBox: ~  
GNU nano 4.8 /etc/apache2/sites-available/000-default.conf  
  
ErrorLog ${APACHE_LOG_DIR}/error.log  
CustomLog ${APACHE_LOG_DIR}/access.log combined  
  
# For most configuration files from conf-available/, which are  
# enabled or disabled at a global level, it is 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  
  
<Directory "/var/www/html">  
    AuthType Basic  
    AuthName "RESTRICTED"  
    AuthUserFile /etc/apache2/.htpasswd  
    Require valid-user >  
</Directory>  
  
</VirtualHost>  
|  
  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

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

```
trisha@trisha-VirtualBox:~$ sudo service apache2 restart  
trisha@trisha-VirtualBox:~$
```

sudo service apache2 restart

The localhost is now accessed using the Firefox browser by entering the username and password that was set by us earlier.



Wireshark is used to capture the packets when the localhost is fetched.

*any							
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help							
tcp.stream eq 13							
No.	Time	Source	Destination	Protocol	Length	Info	
481	16.050411601	127.0.0.1	127.0.0.1	TCP	76	48558 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=	
482	16.050426308	127.0.0.1	127.0.0.1	TCP	76	80 → 48558 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495	
483	16.050438319	127.0.0.1	127.0.0.1	TCP	68	48558 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=25776376	
484	16.062748486	127.0.0.1	127.0.0.1	HTTP	396	GET / HTTP/1.1	
485	16.062777691	127.0.0.1	127.0.0.1	TCP	68	80 → 48558 [ACK] Seq=1 Ack=329 Win=65280 Len=0 TSval=257763	
486	16.063040425	127.0.0.1	127.0.0.1	HTTP	786	HTTP/1.1 401 Unauthorized (text/html)	
487	16.063149417	127.0.0.1	127.0.0.1	TCP	68	48558 → 80 [ACK] Seq=329 Ack=719 Win=64896 Len=0 TSval=2577	
488	21.068079039	127.0.0.1	127.0.0.1	TCP	68	80 → 48558 [FIN, ACK] Seq=719 Ack=329 Win=65536 Len=0 TSval	
489	21.068299196	127.0.0.1	127.0.0.1	TCP	68	48558 → 80 [FIN, ACK] Seq=329 Ack=720 Win=65536 Len=0 TSval	
490	21.068311540	127.0.0.1	127.0.0.1	TCP	68	80 → 48558 [ACK] Seq=720 Ack=330 Win=65536 Len=0 TSval=2577	
491	33.968560124	127.0.0.1	127.0.0.1	TCP	76	48560 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=	
492	33.968574032	127.0.0.1	127.0.0.1	TCP	76	80 → 48560 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495	
493	33.968585074	127.0.0.1	127.0.0.1	TCP	68	48560 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=25776555	
494	33.968649022	127.0.0.1	127.0.0.1	HTTP	439	GET / HTTP/1.1	
495	33.968662694	127.0.0.1	127.0.0.1	TCP	68	80 → 48560 [ACK] Seq=1 Ack=372 Win=65152 Len=0 TSval=257765	
496	33.969760960	127.0.0.1	127.0.0.1	HTTP	3543	HTTP/1.1 200 OK (text/html)	
497	33.969870007	127.0.0.1	127.0.0.1	TCP	68	48560 → 80 [ACK] Seq=372 Ack=3476 Win=62080 Len=0 TSval=257	

Using the follow TCP Stream on the HTTP message segment the password was retrieved (encrypted by base64 algorithm).

Wireshark · Follow TCP Stream (tcp.stream eq 7) · any

```

GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
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 dHJpc2hhOm5ldHdvcmtz

HTTP/1.1 200 OK
Date: Mon, 08 Feb 2021 11:02:59 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Mon, 01 Feb 2021 15:09:38 GMT
ETag: "2aa6-5ba47be16f454-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 3138
Keep-Alive: timeout=5, max=2
Connection: Keep-Alive
Content-Type: text/html

.....Z.s.6.....U..$'....."&.c....$......"! c....d5...~..H.%..
5...H...o.....n...../..7...=.....d.....
0...GzKc.q.n6.`.<.j.N?...Hk.....).b....&...J$.....L.....w2.s.b2WrE...+6.4!
R....d....4....
o.6$KcjX<&'....p8.....d....Y.-S.. 3.,[.px2.....d4..['+f...;`.....w)...
3.n$.#.....`BT.%.Tif.?Mo... =%..`..R.-.....L.rK.. /..l.....-v...
F...l".{d.BN.{:R.%.z..g.aE..hL....K....y.h}....7.....su.sXY.
{yd..ht.P2K.A$.Tc.....>..
.t..vL..TL..'<e.....U...F..S.n...*.....L.R.|..(O.R0\..t.7&. j..921.....`^,
..rLF...&.n..4F..N..}.fLhfd.68..r.....x.'9..;7.....#..c....~.....~..h.;.u..
..isv.....)Dn.7.?.....:e?..!.....6.."......5F.K|
^..Oa...H.....l$.m.....p./.....xu.v.
[...y...l/...l..c.jF.c.....-IsZ;...N..#.....F.V...<e,>v^..)U....[f.
.#.dT..
5...x<...X.
B.

```

Decrypting the password using the base64 algorithm

Encryption technique used by this algorithm :-

- 1) Each character is converted into 8-bit binary ASCII representation and grouped into chunks of 6-bits.
- 2) These chunks are converted into their decimal equivalent and assigned the corresponding Base64 character.

Decryption technique to be used :-

- 1) Each character's 6-bit binary equivalent is found.
- 2) Eight bit chunks are made and then decoded to ASCII.
- 3) This reveals the password that was encrypted by using Base64 algorithm.

ENCRYPTED PASSWORD = dHJpc2hhOm5ldHdvcmtz

Converting to 6-bit binary equivalent :-

d	011101
H	000111
J	001001
p	101001
c	011100
2	110110
h	100001
h	100001
O	001110
m	100110
5	111001
l	100101
d	011101
H	000111
d	011101

v	101111
c	011100
m	100110
t	101101
z	110011

Grouping these binary equivalents to get the ASCII code :-

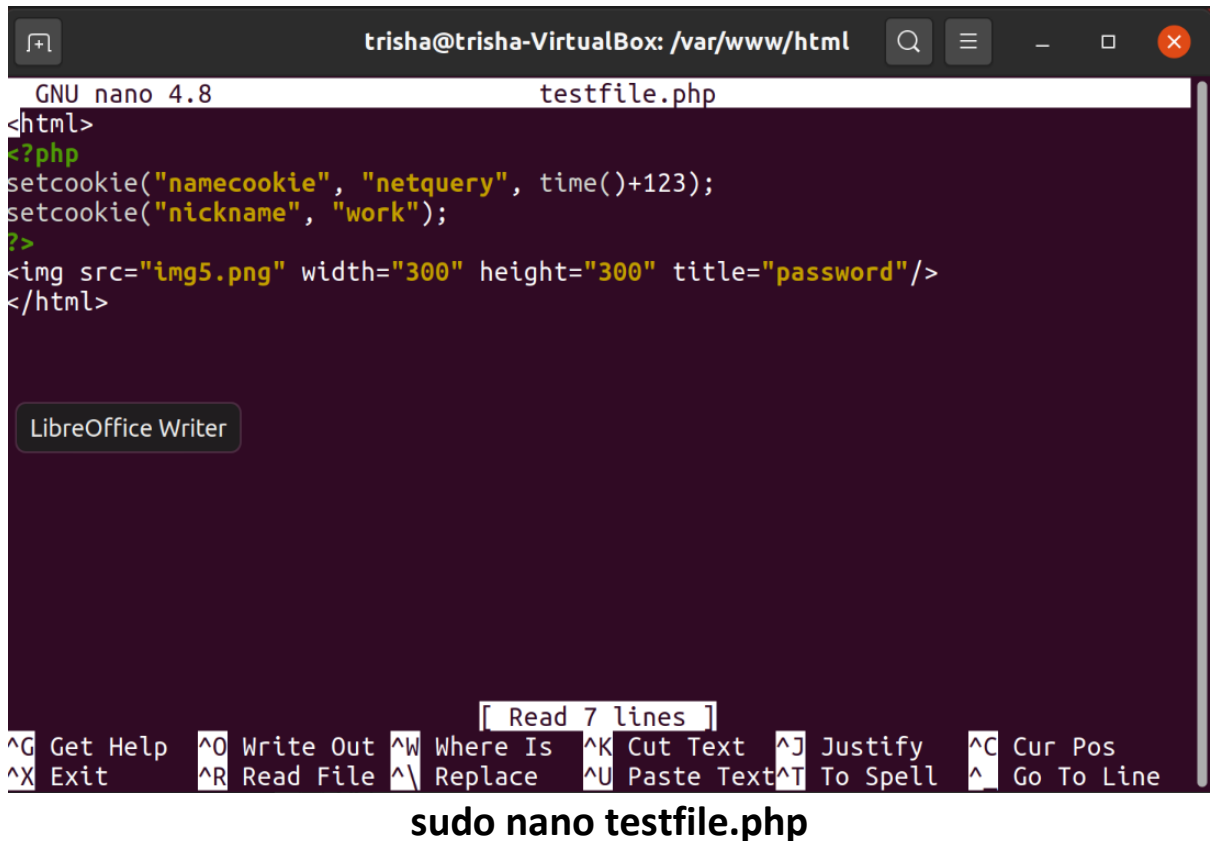
01110100	t
01110010	r
01101001	i
01110011	s
01101000	h
01100001	a
00111010	:
01101110	n
01100101	e
01110100	t
01110111	w
01101111	o
01110010	r
01101011	k
01110011	s

Username : trisha

Password : networks

II. Cookie Setting

We set cookies using a PHP file and the setcookie command as follows :-



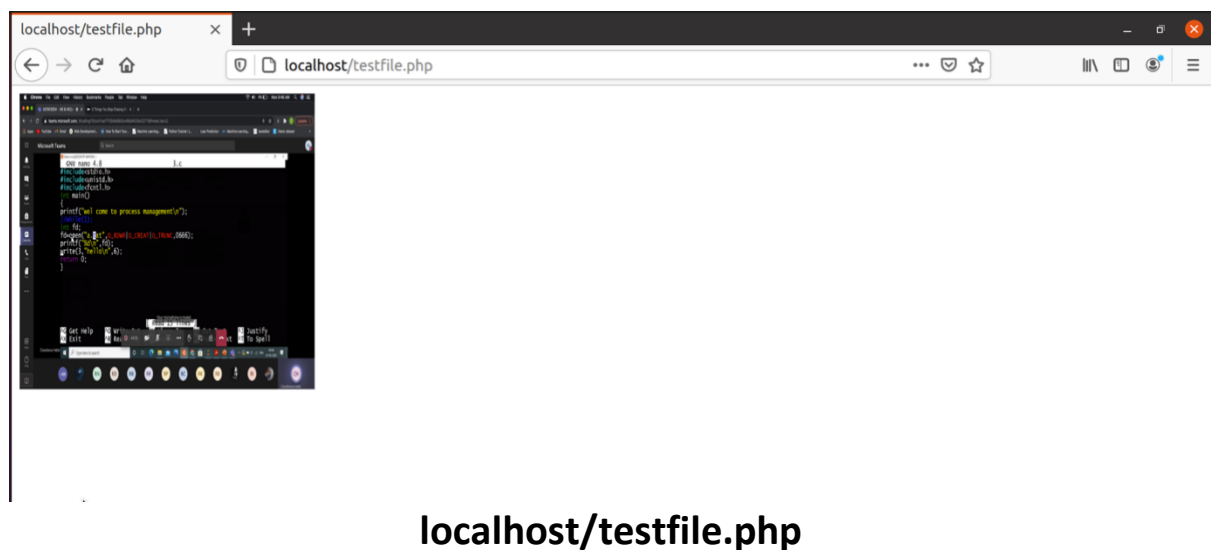
The screenshot shows a terminal window titled 'trisha@trisha-VirtualBox: /var/www/html'. Inside, the GNU nano 4.8 editor is open, editing 'testfile.php'. The code in the file is:

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

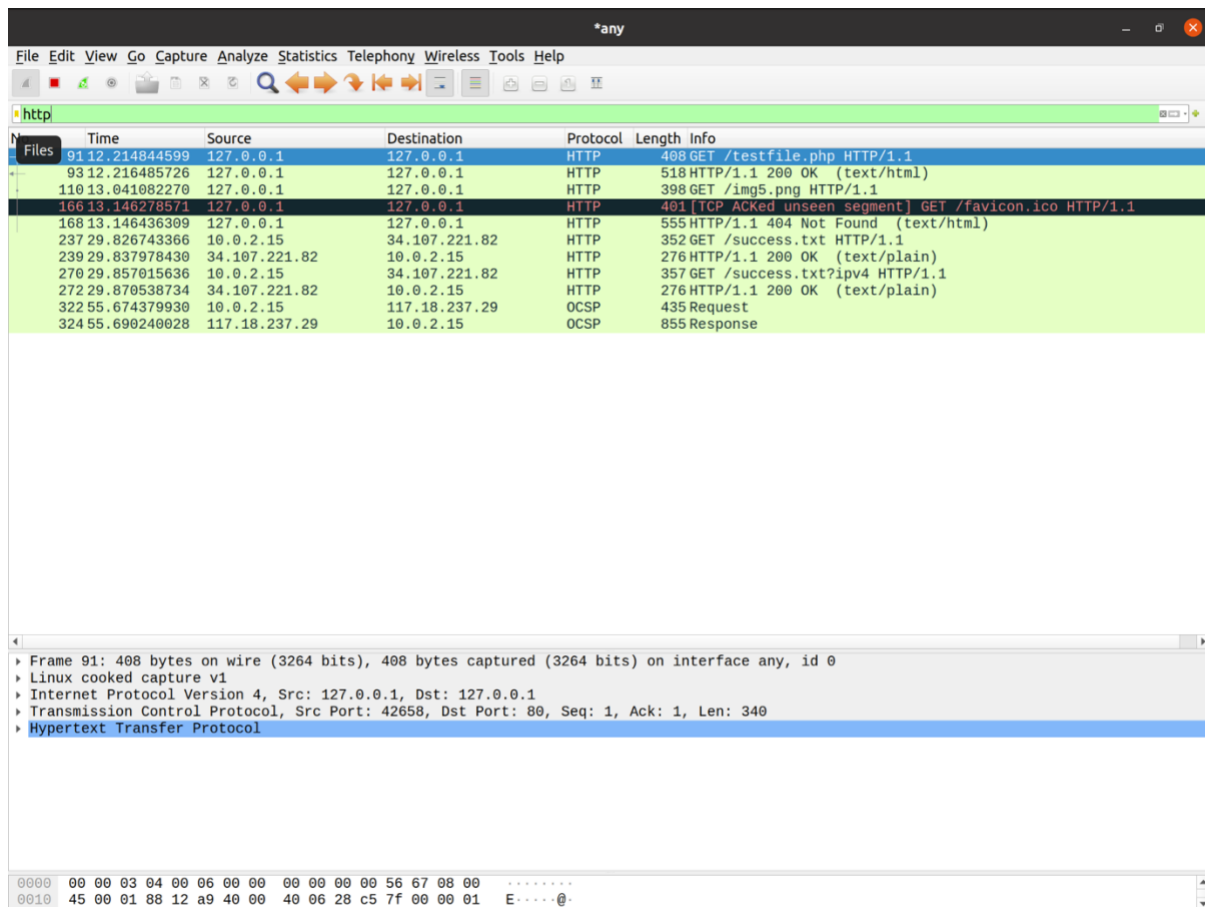
</html>
```

Below the code, a 'LibreOffice Writer' button is visible. At the bottom of the terminal, a status bar shows various keyboard shortcuts. Below the terminal window, the command **sudo nano testfile.php** is written.

Accessing the page from Firefox :-



Capturing the packets using Wireshark :-



No.	Time	Source	Destination	Protocol	Length	Info
91	12.214844599	127.0.0.1	127.0.0.1	HTTP	408	GET /testfile.php HTTP/1.1
93	12.216485726	127.0.0.1	127.0.0.1	HTTP	518	HTTP/1.1 200 OK (text/html)
110	13.041082270	127.0.0.1	127.0.0.1	HTTP	398	GET /img5.png HTTP/1.1
166	13.146278571	127.0.0.1	127.0.0.1	HTTP	401	[TCP ACKed unseen segment] GET /favicon.ico HTTP/1.1
168	13.146436309	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
237	29.826743366	10.0.2.15	34.107.221.82	HTTP	352	GET /success.txt HTTP/1.1
239	29.837978430	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (text/plain)
270	29.857015636	10.0.2.15	34.107.221.82	HTTP	357	GET /success.txt?ipv4 HTTP/1.1
272	29.870538734	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (text/plain)
322	55.674379930	10.0.2.15	117.18.237.29	OCSP	435	Request
324	55.699240028	117.18.237.29	10.0.2.15	OCSP	855	Response

Frame 91: 408 bytes on wire (3264 bits), 408 bytes captured (3264 bits) on interface any, id 0	
Linux cooked capture v1	
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	
Transmission Control Protocol, Src Port: 42658, Dst Port: 80, Seq: 1, Ack: 1, Len: 340	
Hypertext Transfer Protocol	

0000	00 00 03 04 00 06 00 00 00 00 00 00 56 67 08 00
0010	45 00 01 88 12 a9 40 00 40 06 28 c5 7f 00 00 01	E.....@.

In the TCP capture of the wireshark packet highlighted above, we notice that there are two additional fields in the HTTP response section named setcookie. This demonstrates that the cookie was set successfully.

TCP Capture :-

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

```
GET /testfile.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
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: Wed, 10 Feb 2021 16:21:35 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: namecookie=netquery; expires=Wed, 10-Feb-2021 16:23:38 GMT; Max-Age=123
Set-Cookie: nickname=work
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 87
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

.....(.....MW(.J.U.2L.
.....3SJ2l..
...2R3.3J.....T[.....%}.~}.9.h.X.O...GET /img5.png HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost/testfile.php
Cookie: namecookie=netquery; nickname=work

HTTP/1.1 200 OK
Date: Wed, 10 Feb 2021 16:21:36 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Mon, 01 Feb 2021 15:25:37 GMT
ETag: "165ad2-5ba47f73f23d7"
Accept-Ranges: bytes
Content-Length: 1465042
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: image/png

.PNG
3 client pkts, 31 server pkts, 5 turns.
```

Entire conversation (918kB) Show data as ASCII Stream 2

Find: Find Next

Help Filter Out This Stream Print Save as... Back Close

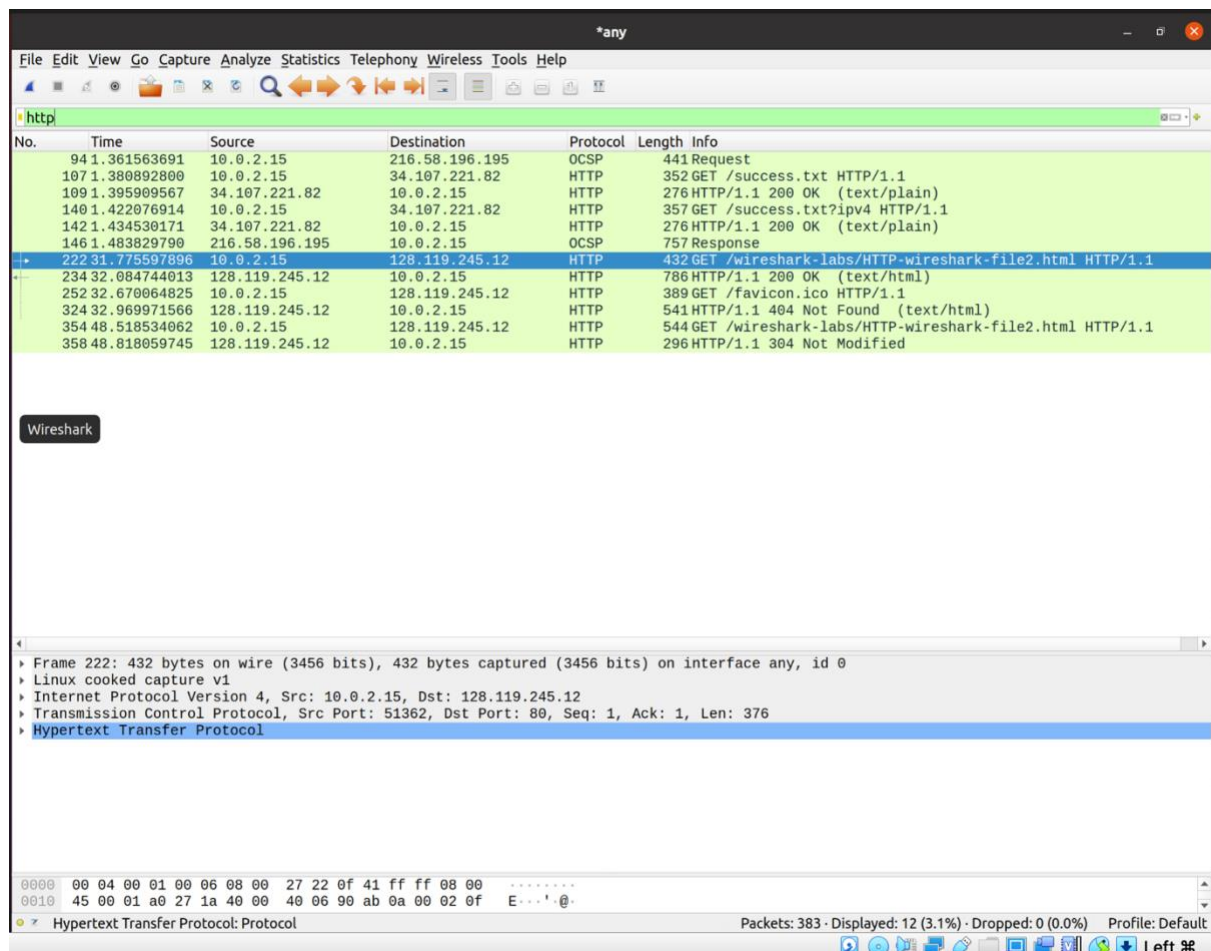
III. Conditional get

Conditional GET response can be implemented by using the **If-Modified-Since** which is checked by the server and then the resource that is requested is resent only if the resource has been modified since the timestamp in the header.

304 Not Modified status code is sent back if the resource has not been modified.

This is called conditional HTTP response because it only resends the resource if it has been modified since the last GET request by the client.

Accessing the HTML page : <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html> for the **first time** :-



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:84.0) Gecko/20100101 Firefox/84.0

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: Mon, 08 Feb 2021 12:18:25 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: Mon, 08 Feb 2021 06:59:01 GMT

ETag: "173-5bacdb4665369"

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.

This file's last modification date will not change. <p>

Thus if you download this multiple times on your browser, a complete copy
will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE
field in your browser's HTTP GET request to the server.

</html>

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (1,106 bytes)

Show data as ASCII

Stream 8

Find:

Find Next

Help

Filter Out This Stream

Print

Save as...

Back

Close

Accessing the HTML page : <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html> for the **second time** :-

```
Wireshark · Follow TCP Stream (tcp.stream eq 11) · any

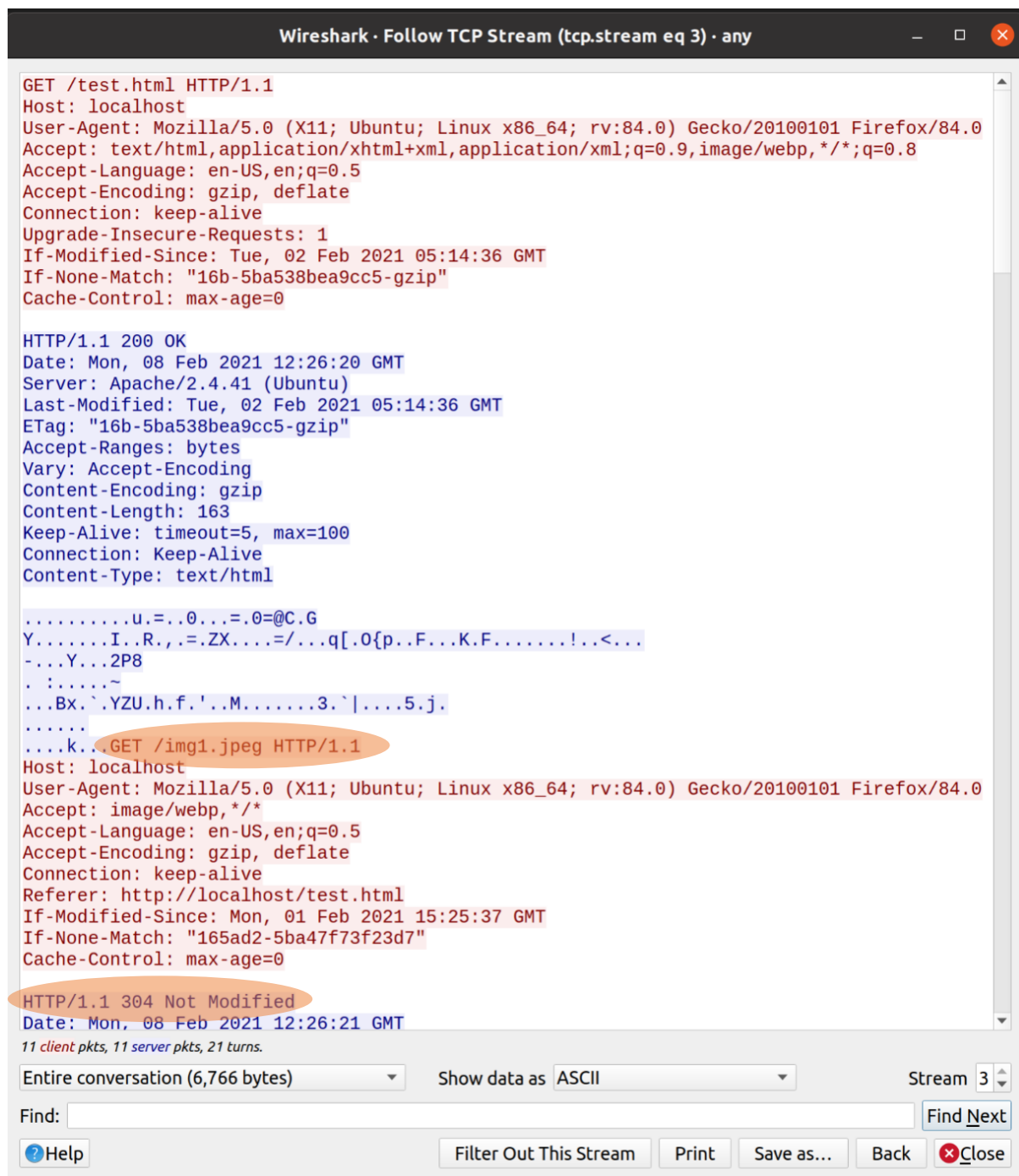
GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 08 Feb 2021 06:59:01 GMT
If-None-Match: "173-5bacdb4665369"
Cache-Control: max-age=0

HTTP/1.1 304 Not Modified
Date: Mon, 08 Feb 2021 12:18:42 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
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
ETag: "173-5bacdb4665369"
```

Accessing a local host file :- An HTML file with ten images was created and saved in the localhost home directory. The screenshots below demonstrate the conditional get requests and responses.

Wireshark packet capture =>

No.	Time	Source	Destination	Protocol	Length	Info
124	893229955	127.0.0.1	127.0.0.1	HTTP	405	GET /test.html HTTP/1.1
144	893644037	127.0.0.1	127.0.0.1	HTTP	568	HTTP/1.1 200 OK (text/html)
59	5.444790101	127.0.0.1	127.0.0.1	HTTP	352	GET /img1.jpeg HTTP/1.1
101	5.530446421	127.0.0.1	127.0.0.1	HTTP	351	[TCP ACKed unseen segment] GET /img2.png HTTP/1.1
143	5.541373263	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
145	5.574164692	127.0.0.1	127.0.0.1	HTTP	351	GET /img3.png HTTP/1.1
174	5.640042091	127.0.0.1	127.0.0.1	HTTP	351	[TCP ACKed unseen segment] GET /img4.png HTTP/1.1
202	5.651181001	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
260	5.701671551	127.0.0.1	127.0.0.1	HTTP	351	GET /img5.png HTTP/1.1
280	5.779428413	127.0.0.1	127.0.0.1	HTTP	351	GET /img6.png HTTP/1.1
307	5.790180753	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
310	5.831792535	127.0.0.1	127.0.0.1	HTTP	351	GET /img7.png HTTP/1.1
348	5.840478995	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
351	5.877102389	127.0.0.1	127.0.0.1	HTTP	351	GET /img8.png HTTP/1.1
386	5.912113572	127.0.0.1	127.0.0.1	HTTP	351	[TCP ACKed unseen segment] GET /img9.png HTTP/1.1
417	5.923090969	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
419	5.945841803	127.0.0.1	127.0.0.1	HTTP	352	GET /img10.png HTTP/1.1
449	5.956206328	127.0.0.1	127.0.0.1	HTTP	24775	HTTP/1.1 200 OK (PNG)
452	6.435277448	127.0.0.1	127.0.0.1	HTTP	354	GET /favicon.ico HTTP/1.1
453	6.435424201	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
461	15.210845375	127.0.0.1	127.0.0.1	HTTP	522	GET /test.html HTTP/1.1
463	15.211674640	127.0.0.1	127.0.0.1	HTTP	568	HTTP/1.1 200 OK (text/html)
465	15.343828887	127.0.0.1	127.0.0.1	HTTP	407	GET /img1.jpeg HTTP/1.1
467	15.344043099	127.0.0.1	127.0.0.1	HTTP	251	HTTP/1.1 304 Not Modified
469	15.354055798	127.0.0.1	127.0.0.1	HTTP	466	GET /img2.png HTTP/1.1
471	15.354256964	127.0.0.1	127.0.0.1	HTTP	251	HTTP/1.1 304 Not Modified
473	15.357239813	127.0.0.1	127.0.0.1	HTTP	466	GET /img3.png HTTP/1.1
475	15.357448170	127.0.0.1	127.0.0.1	HTTP	251	HTTP/1.1 304 Not Modified
477	15.359227499	127.0.0.1	127.0.0.1	HTTP	466	GET /img4.png HTTP/1.1



It can be observed that when the file is reloaded the images are not resent by the HTTP client. This is because the files were not modified since the last time that we requested them. (img1.jpeg is highlighted in the screenshots as an example)