

**Name:** Suhan.B.Revankar

**SRN:** PES2UG19CS412

**Week number:** 3

**Date:** 14/2/2021

**Name of the experiment:** Understand the working of HTTP Headers

---

**Objective:** The objective here is to understand the working of HTTP Headers.

**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.

## **Understanding the Working of HTTP Headers**

Question: Understand the 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. an image) and sets a cookie and enables authentication. You are required to configure the webserver (e.g. apache) with an authentication mechanism. Show the behaviour 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 Basic- Auth header using Base64 mechanism as per the password setup.

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

Solution: Analyzing Basic Authentication and Cookies

The three parts of the experiment are:

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

## Password Authentication

1.

i) The commands below are executed on the terminal.

```
sudo apt-get update
```

This command just installs any missing packages.

```
sudo apt-get install apache2 apache2-utils
```

The usage of this command suffices the need for a password authentication.

```
suhan@suhan: ~  
suhan@suhan:~$ sudo apt-get update  
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease  
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease  
Get:4 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]  
Fetched 214 kB in 25s (8,417 B/s)  
Reading package lists... Done  
suhan@suhan:~$ sudo apt-get install apache2 apache2-utils  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
apache2 is already the newest version (2.4.41-4ubuntu3.1).  
apache2-utils is already the newest version (2.4.41-4ubuntu3.1).  
0 upgraded, 0 newly installed, 0 to remove and 341 not upgraded.  
suhan@suhan:~$
```

ii) To secure the network with a password, the following command was used:

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

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

iii) The network name is now fixed to, suhanbrevenkar and is secured with a verified password.

For taking a glance at the network the following command is fruitful:

```
sudo cat /etc/apache2/.htpasswd
```

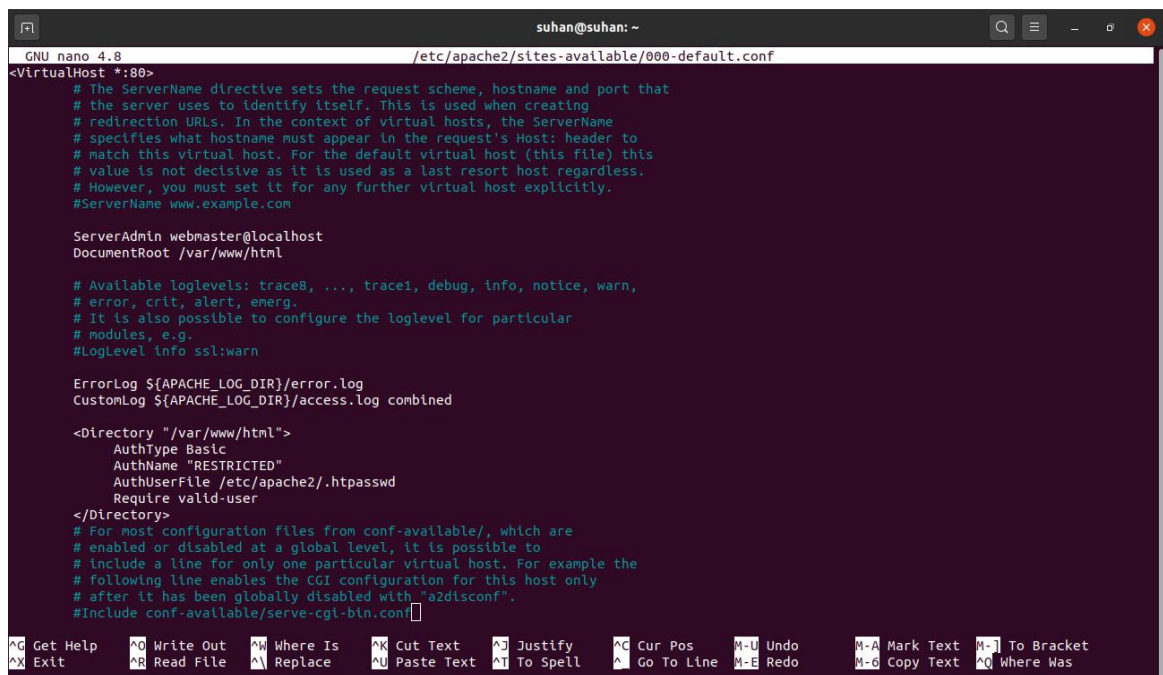


```
suhan@suhan:~$ sudo cat /etc/apache2/.htpasswd
suhanbrevankar:$apr1$D0pXfIRm$Bsuq4DEEXjy630J02njcL0
suhan@suhan:~$
```

2. To set up the authentication phase, the following commands are executed.  
Configuring access control within the Virtual Host Definition.

Since the file was changed in the text editor, the command that was used to perform this action is:

```
gedit /etc/apache2/sites-available/000-default.conf
```



```
GNU nano 4.8 /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

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
```

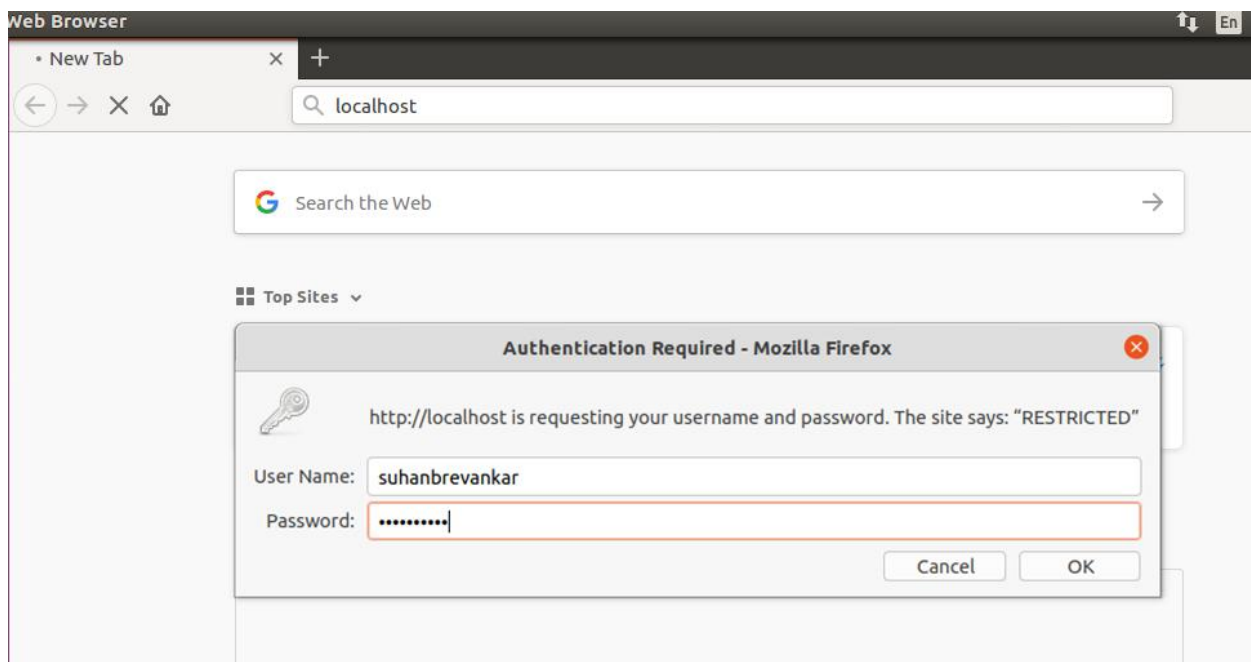
3. The password must now be implemented, henceforth, the apache service is ought to be restarted.

This command performs the action required: `sudo service apache2 restart`

```
suhan@suhan:~$ sudo service apache2 restart
suhan@suhan:~$
```

The apache2 restart service program successfully completed its job, since there are no errors.

Now, the network can be tested for security.



The help of Wireshark is taken to capture the packets.

Since we are accessing the file, index.html over localhost, the IP address of the lo interface is reflected with no doubt at all.

Wireshark interface showing a packet capture on interface 'any'. The packet list shows a GET request for /hello.html. The packet details pane shows the request structure, including Host, User-Agent, Accept, and Authorization headers. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
358	35.128585777	127.0.0.1	127.0.0.1	TCP	76	32832 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 ...
359	35.128604236	127.0.0.1	127.0.0.1	TCP	76	80 → 32832 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 S...
360	35.128617734	127.0.0.1	127.0.0.1	TCP	60	32832 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3282216658...
361	35.128625338	127.0.0.1	127.0.0.1	HTTP	465	GET /hello.html HTTP/1.1
362	35.128712297	127.0.0.1	127.0.0.1	TCP	60	80 → 32832 [ACK] Seq=1 Ack=398 Win=65152 Len=0 TSval=32822166...
363	35.159254456	127.0.0.1	127.0.0.1	HTTP	379	HTTP/1.1 200 OK (text/html)
364	35.159289310	127.0.0.1	127.0.0.1	TCP	60	32832 → 80 [ACK] Seq=398 Ack=312 Win=65280 Len=0 TSval=328221...
372	35.435507977	127.0.0.1	127.0.0.1	HTTP	376	GET /favicon.ico HTTP/1.1
373	35.435542646	127.0.0.1	127.0.0.1	TCP	60	80 → 32832 [ACK] Seq=312 Ack=706 Win=65280 Len=0 TSval=328221...
374	35.436208432	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
375	35.436686274	127.0.0.1	127.0.0.1	TCP	60	32832 → 80 [ACK] Seq=706 Ack=799 Win=65152 Len=0 TSval=328221...
412	40.436324714	127.0.0.1	127.0.0.1	TCP	60	80 → 32832 [FIN, ACK] Seq=799 Ack=706 Win=65536 Len=0 TSval=3...

Frame 361: 465 bytes on wire (3720 bits), 465 bytes captured (3720 bits) on interface any, id 0

Ethernet II, Src: VirtualBox (08:00:00:00:00:00), Dst: VirtualBox (08:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 32832, Dst Port: 80, Seq: 1, Ack: 1, Len: 397

Hypertext Transfer Protocol

GET /hello.html HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:79.0) Gecko/20100101 Firefox/79.0

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 c3VoYW51cmV2YW5rYXI6cXdlcnR5MTIzNA==

Here, it's the ipv6 address that was reflected.

Wireshark - Follow TCP Stream (tcp.stream eq 19) - any

```

GET /hello.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
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 c3VoYW51cmV2YW5rYXI6cXdlcnR5MTIzNA==

HTTP/1.1 200 OK
Date: Thu, 11 Feb 2021 05:29:28 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Thu, 11 Feb 2021 05:24:22 GMT
ETag: "1c-5bb08bb7932f4"
Accept-Ranges: bytes
Content-Length: 28
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<html>
hello world!
</html>
GET /favicon.ico HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic c3VoYW51cmV2YW5rYXI6cXdlcnR5MTIzNA==
Connection: keep-alive

HTTP/1.1 404 Not Found
Date: Thu, 11 Feb 2021 05:29:28 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Length: 271
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
  
```

Packet 363: 2 client pkts, 2 server pkts, 3 turns. Click to select.

Entire conversation (1,503 bytes) Show and save data as ASCII Stream 19

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



Now, on a thorough glance at these results of following a TCP stream on GET /HTTP/1.1 what can be observed?

**The localhost is successfully secured with a password.**

Deeply looking at it, what's that bizarre-looking word beside the "Authorization" field?

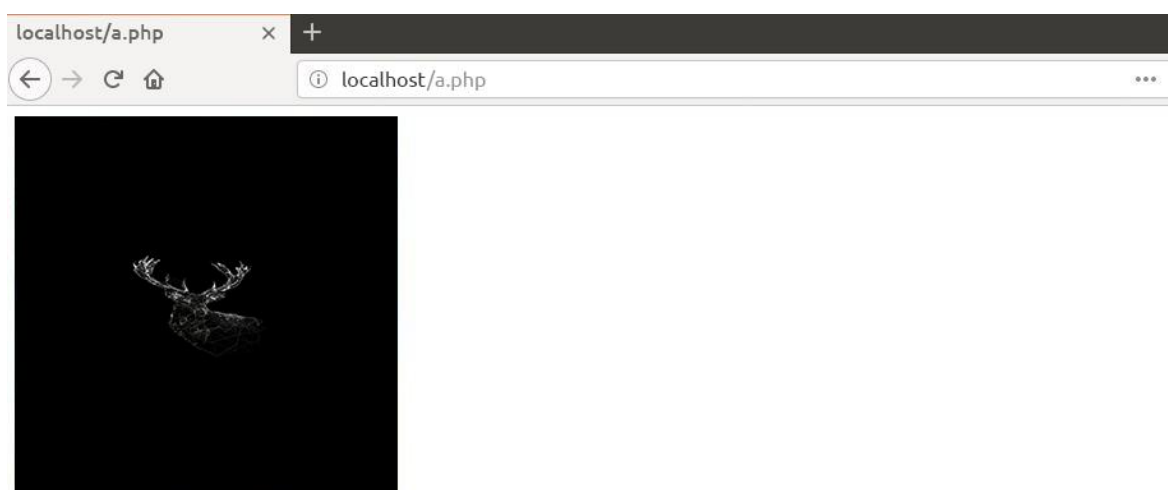
It's the password that was set to keep the network secured. It's encoded in a format coined as **Base64**. This shall be decoded in a later phase. The Base64 code word is, c3VoYW5icmV2YW5rYXl6cXdlcnR5MTIzNA==

## Cookie Setting

1. A PHP file is created to set the cookie.

```
suhan@suhan: /var/www/html
GNU nano 4.8 a.php
<html>
<?php
setcookie("suhan","suhan",time()+3600,'/');
setcookie("nickname","work",time()+36000,'/');
?>
<img src= "highres.jpg" width= "300" height= "300" title= "password" />
</html>
```

2. The file was saved under the HTML directory in the path /var/www/html



That's the webpage.

- 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 image shows a Wireshark packet capture window titled "\*any". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. The packet list pane on the left shows a list of captured packets. Packet 154 is selected, showing an HTTP GET request for /a.php. The packet details pane on the right shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The packet bytes pane at the bottom shows the raw data of the selected packet in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
58	14.766282934	192.168.42.187	216.58.200.131	OCSP	453	Request
60	14.889322110	216.58.200.131	192.168.42.187	OCSP	769	Response
133	29.819828143	127.0.0.1	127.0.0.1	HTTP	401	GET /a.php HTTP/1.1
135	29.969804625	127.0.0.1	127.0.0.1	HTTP	788	HTTP/1.1 401 Unauthorized (text/html)
154	44.137088518	127.0.0.1	127.0.0.1	HTTP	460	GET /a.php HTTP/1.1
156	44.204684033	127.0.0.1	127.0.0.1	HTTP	591	HTTP/1.1 200 OK (text/html)
171	44.540325575	127.0.0.1	127.0.0.1	HTTP	463	GET /%E2%80%9Chighres.jpg%E2%80%9D HTTP/1.1
173	44.540859011	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
176	44.644229602	127.0.0.1	127.0.0.1	HTTP	412	GET /favicon.ico HTTP/1.1
178	44.644761763	127.0.0.1	127.0.0.1	HTTP	555	HTTP/1.1 404 Not Found (text/html)
242	63.552614480	192.168.42.187	35.232.111.17	HTTP	155	GET / HTTP/1.1
244	63.813463886	35.232.111.17	192.168.42.187	HTTP	216	HTTP/1.1 204 No Content

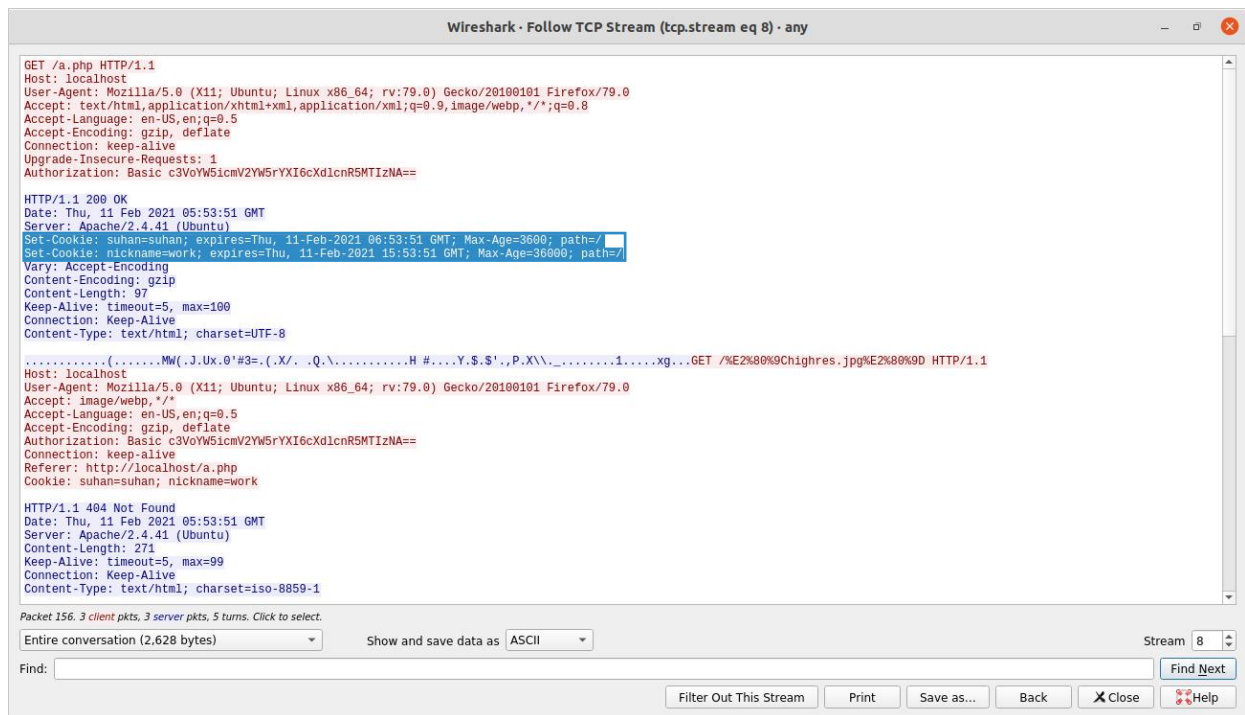
Frame 154: 460 bytes on wire (3680 bits), 460 bytes captured (3680 bits) on interface any, id 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 33162, Dst Port: 80, Seq: 1, Ack: 1, Len: 392
- Hypertext Transfer Protocol

0000 00 00 03 04 00 06 00 00 00 00 00 00 00 00 00 .....  
0010 45 00 01 bc 16 51 40 00 40 06 24 e9 7f 00 00 01 E....Q...@ \$.....  
0020 7f 00 00 01 81 8a 00 50 6f 8b 7e ca a6 04 ee 09 .....P o~.....  
0030 80 18 02 00 ff b0 00 00 01 01 08 0a c3 b8 f8 8b .....  
0040 c3 b8 f8 8b 47 45 54 20 2f 61 2e 70 68 70 20 48 ...GET /a.php H  
0050 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6c TTP/1.1 Host: l  
0060 6f 63 61 6c 68 6f 73 74 0d 0a 55 73 65 72 2d 41 ocalhost User-A  
0070 67 65 6e 74 3a 20 4d 0f 7a 09 6c 6c 61 2f 35 2e gent: Mozilla/5.  
0080 30 20 28 58 31 31 3b 20 55 62 75 6e 74 75 3b 20 0 (X11; Ubuntu;  
0090 4c 69 6e 75 78 20 78 38 36 5f 36 34 3b 20 72 76 Linux x8 6.64; rv  
00a0 3a 37 39 2e 30 29 20 47 65 63 6b 6f 2f 32 30 31 :79.0) Gecko/201  
00b0 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 37 39 00101 Firefox/79  
00c0 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 74 .0 Accept: text

Hypertext Transfer Protocol: Protocol Packets: 264 · Displayed: 14 (5.3%) · Dropped: 0 (0.0%) Profile: Default





Henceforth, with no further qualm, it's been solemnly concluded that the cookie has been set.

Now, the Base64 encrypted codeword is to be decoded.

This is the algorithm:

1. Take down the codeword as it is.
2. Write down the appropriate corresponding number of each character.

Use this table for reference:

Upper case alphabets		Lower case alphabets		Digits		Symbols	
Index Number	Character	Index Number	Character	Index Number	Digit	Index Number	Symbol
0	A	26	a	52	0	62	+
1	B	27	b	53	1	63	/
2	C	28	c	54	2		
3	D	29	d	55	3		
4	E	30	e	56	4		
5	F	31	f	57	5		
6	G	32	g	58	6		
7	H	33	h	59	7		
8	I	34	i	60	8		
9	J	35	j	61	9		
10	K	36	k				
11	L	37	l				
12	M	38	m				
13	N	39	n				
14	O	40	o				
15	P	41	p				
16	Q	42	q				
17	R	43	r				
18	S	44	s				
19	T	45	t				
20	U	46	u				
21	V	47	v				
22	W	48	w				
23	X	49	x				
24	Y	50	y				
25	Z	51	z				

= (equals) symbol shall not be assigned an index number

3. Convert each value to its corresponding binary value, and extend up till six bits.
4. Combine all the bits and split them into eight bits (one byte).
5. Convert each byte to it's corresponding ASCII value. That's the decoded result.

Here's how c3VoYW5icmV2YW5rYXI6cXdlcnR5MTIzNA== was decoded.

	A	B	C	D	E	F	G
1	<u>BASE WORD</u>	<u>DECIMAL VALUE</u>	<u>BINARY VALUES (6 bits)</u>	<u>BINARY VALUE (combined to 1 byte)</u>	<u>ASCII CODE</u>	<u>CORRESPONDING CHARACTER</u>	
2	c	28	011100	01110011	115	s	
3	3	55	110111	01110101	117	u	
4	V	21	010101	01101000	104	h	
5	o	40	101000	01100001	97	a	
6	Y	24	011000	01101110	110	n	
7	W	22	010110	01100010	98	b	
8	5	57	111001	01110010	114	r	
9	i	34	100010	01100101	101	e	
10	c	28	011100	01110110	118	v	
11	m	38	100110	01100001	101	e	
12	V	21	010101	01101110	110	n	
13	2	54	110110	01110101	107	k	
14	Y	24	011000	01100001	97	a	
15	W	22	010110	01110010	114	r	
16	5	57	111001	00111010	58	:	
17	r	53	110101	01110001	113	q	
18	Y	24	011000	01110111	119	w	
19	X	23	010111	01100101	101	e	
20	I	8	001000	01110010	114	r	
21	6	58	111010	01110100	116	t	
22	c	28	011100	01111001	121	y	
23	X	23	010111	00110001	49	1	
24	d	29	011101	00110010	50	2	
25	l	37	100101	00110011	51	3	
26	c	28	011100	00110100	52	4	
27	n	39	100111	000000			
28	R	17	010001				
29	5	57	111001				
30	M	12	001100				
31	T	19	010011				
32	I	8	001000				
33	z	51	110011				
34	N	13	001101				
35	A	0	000000				
36	=						
37	=			FINAL RESULT = suhanbrevankar:qwerty1234			
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							
55							
56							

When the bits are combined to form a byte, the last row which shows four zeroes shall be isolated from the 132-bit digit and would remain solitary as it was used to extend the total number of bits, making 128 bits divisible by six.

Hence the codeword was successfully decoded to, suhanbrevankar:qwerty1234.

The image below shows a small verification of this:

## Decode from Base64 format


Simply enter your data then push the decode button.


```
c3VoYW5ldmV2YW5rYXI6cXdlbnR5MTIzNA==
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE** > Decodes your data into the area below.

```
suhanbrevankar:qwerty1234
```

In a reversed fashion (Base64 → Plain text i.e., encoding):

If we reckon the same algorithm in an upside-down fashion, there are slight alterations to be done.

	F	G	H	I	J	K	L
1	CORRESPONDING CHARACTER	ASCII CODE	BINARY VALUE (combined to 1 byte)	BINARY VALUES (6 bits)	DECIMAL VALUE	BASE WORD	
2	s	115	01110011	011100	28	c	
3	u	117	01110101	110111	55	3	
4	h	104	01101000	010101	21	V	
5	a	97	01100001	101000	40	o	
6	n	110	01101110	011000	24	Y	
7	b	98	01100010	010110	22	W	
8	r	114	01110010	111001	57	s	
9	e	101	01100101	100010	34	i	
10	v	118	01110110	011100	28	c	
11	e	101	01100001	100110	38	m	
12	n	110	01101110	010101	21	V	
13	k	107	01110101	110110	54	2	
14	a	97	01100001	011000	24	Y	
15	r	114	01110010	010110	22	W	
16	:	58	00111010	111001	57	s	
17	q	113	01110001	110101	53	r	
18	w	119	01110111	011000	24	Y	
19	e	101	01100101	010111	23	X	
20	r	114	01110010	001000	8	I	
21	t	116	01110100	111010	58	6	
22	y	121	01111001	011100	28	c	
23	1	49	00110001	010111	23	X	
24	2	50	00110010	011101	29	d	
25	3	51	00110011	100101	37	l	
26	4	52	00110100	011100	28	c	
27			000000	100111	39	n	
28				010001	17	R	
29				111001	57	s	
30				001100	12	M	
31				010011	19	T	
32				001000	8	I	
33				110011	51	z	
34				001101	13	N	
35				000000	0	A	
36						=	
37			FINAL RESULT = c3VoYW5icmV2YW5yYXI6cXdlcnR5MTIzNA==				=
38							
39							
40							

In the second row, the last column, we see additional four zeroes. This is for the very reason that the binary digit must be divisible by six (which are made solitary in decoding). In the last column, there's additional two equals' symbol. To make the length of the string divisible by four, the two equals were added.

And, just for a bijou verification:

### Encode to Base64 format

Simply enter your data then push the encode button.

suhanbrevenkar:qwerty1234

UTF-8

Destination character set.

LF (Unix)

Destination newline separator.

☐

Encode each line separately (useful for when you have multiple entries).

☐

Split lines into 76 character wide chunks (useful for MIME).

☐

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF

Encodes in real-time as you type or paste (supports only the UTF-8 character set).

ENCODE

Encodes your data into the area below.

c3VoYW5icmV2ZW5rYXI6cXdlcnR5MTIzNA==

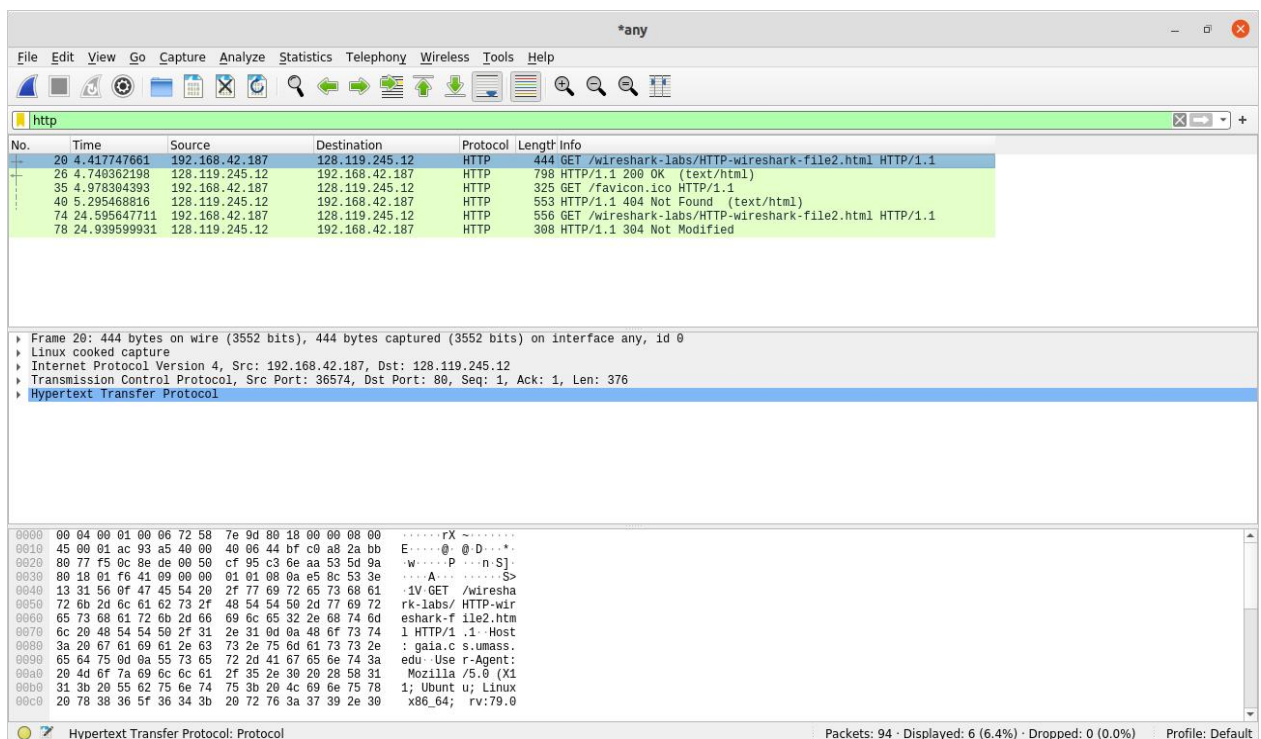
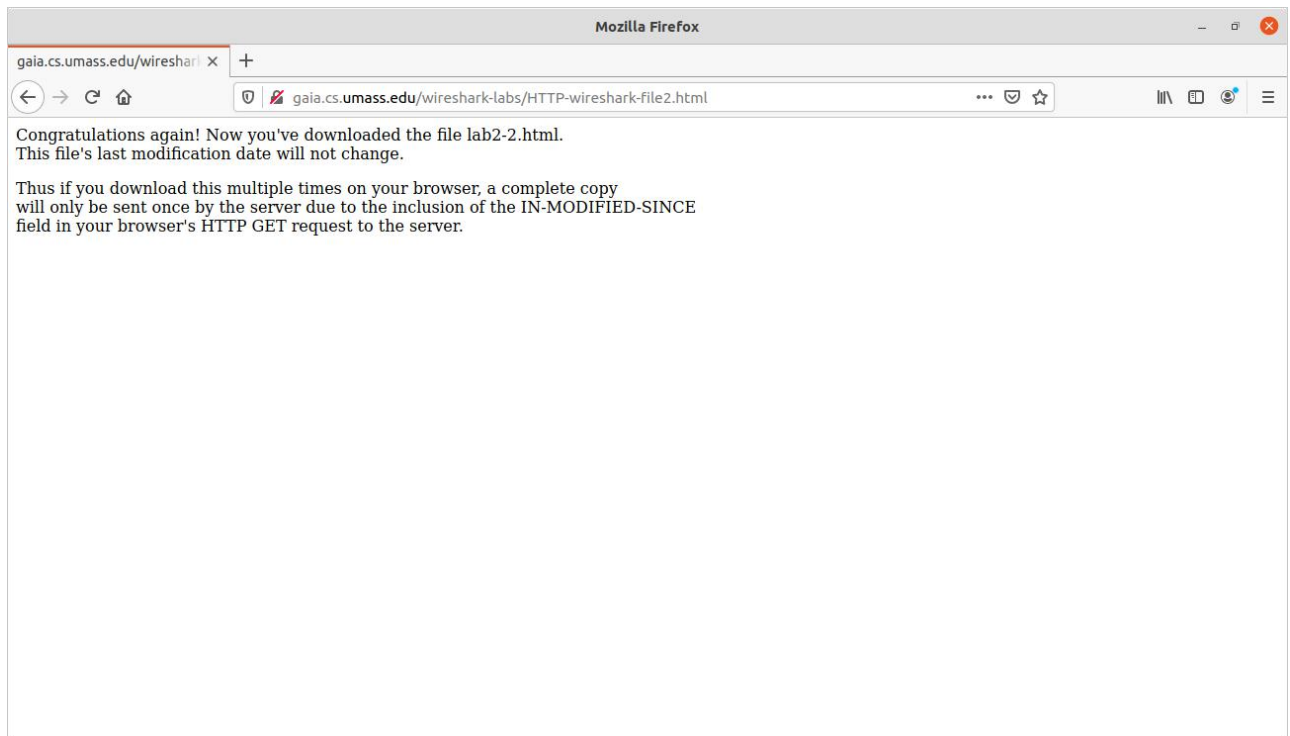
There are not many exclamatory results from the Wireshark capture. All that could be observed is that the cookies are set successfully. The set-cookie attribute under the HTTP header in the TCP stream manifests this.

## Conditional Get: If-Modified-Since

Here, an attempt is being made to look for any modification in a file, which is monitored with the Wireshark tool.

First, the history cache of the browser was expunged. And then, the website was accessed.



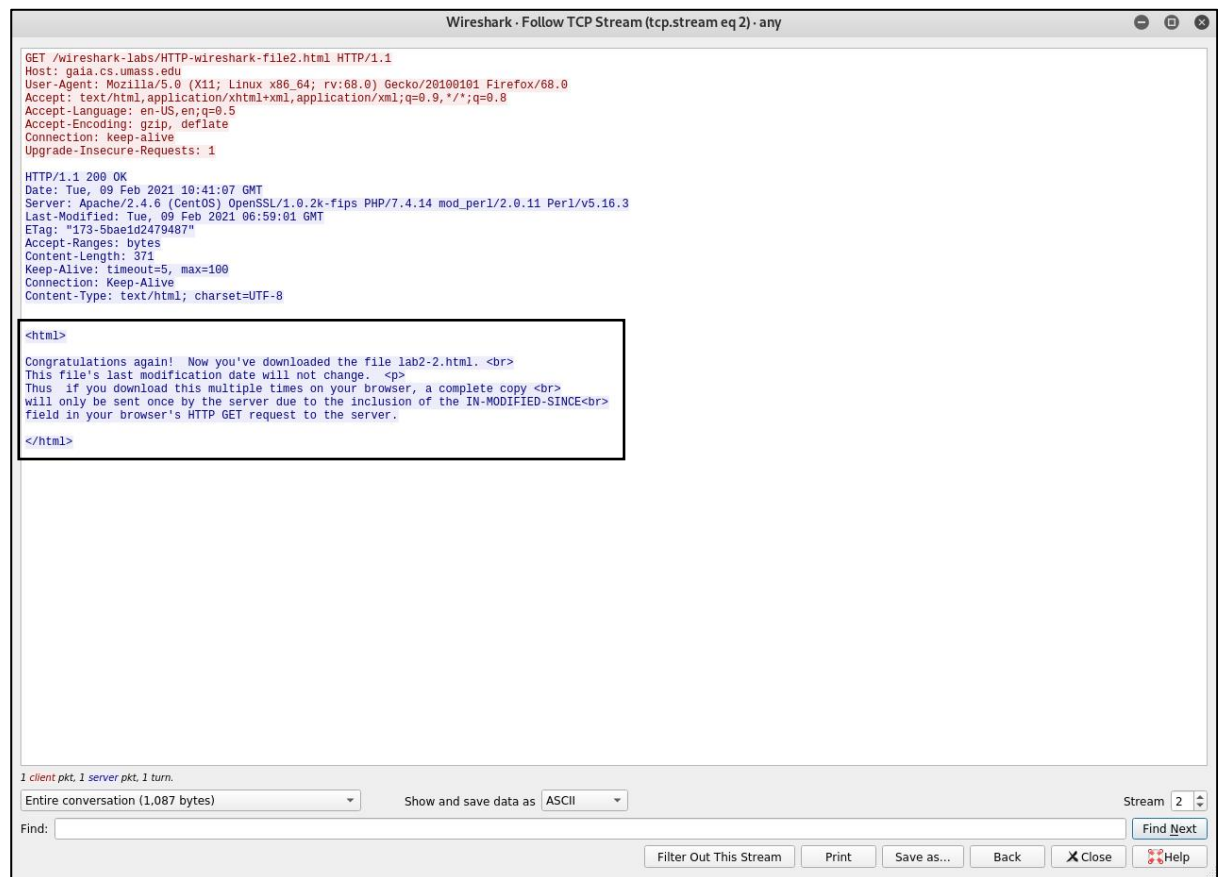


These were the results provided by the Wireshark tool.

This leads to some questions:

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?

Here's the image of the TCP stream:



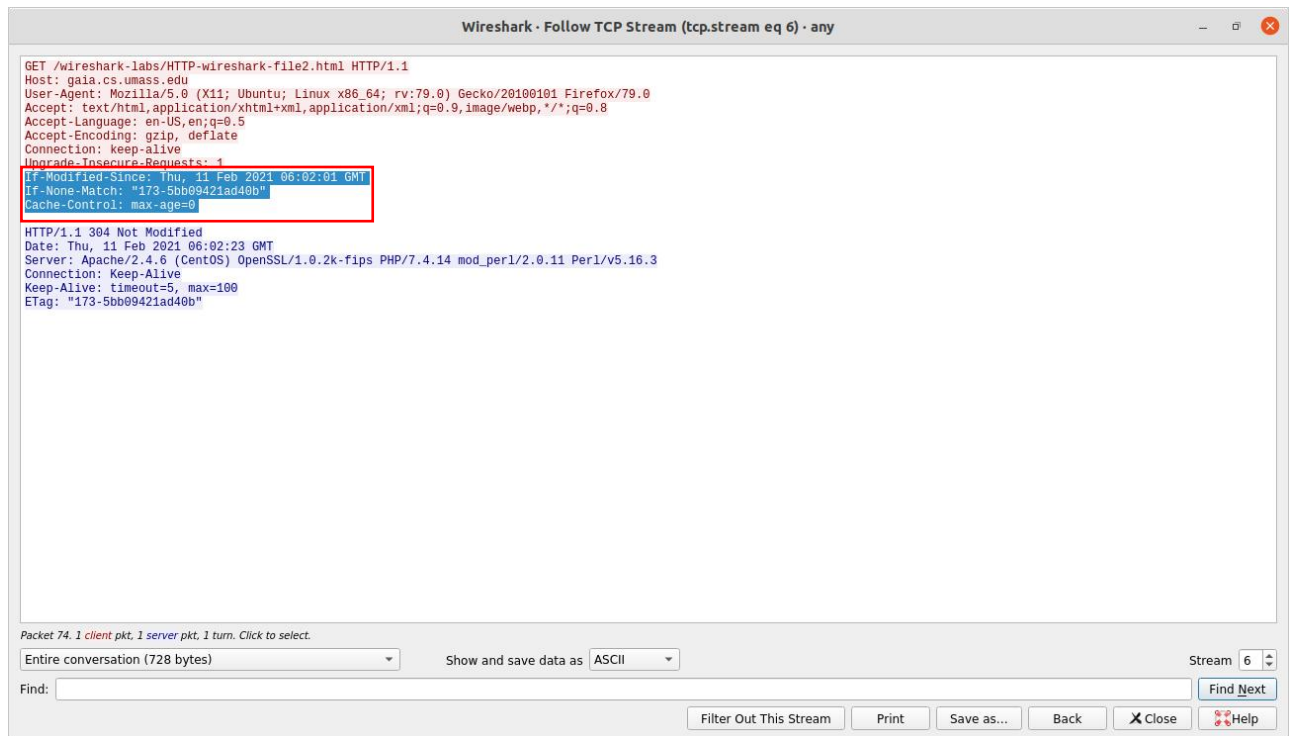
Alas, it's a no. There's no `If-modified-since` in the first HTTP GET request.

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

Undoubtedly, yes. The server did return the contents of the file. Under the HTTP response, we can see a `bijou` code, which manifests that the server explicitly returned the contents of the file.

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?

Taking a rapid glance at this image,



It's undoubtedly claimed that the "IF-MODIFIED-SINCE:" line is in the HTTP GET request. It shows the latest date and time that the user has visited the website. This is the information followed by the "IF-MODIFIED-SINCE:" header.

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.

In response to the second HTTP GET request, the server returned a status code of 304 signifying the phrase, "Not modified". No, the server did not return the contents of the file. The file has been by no means modified and hence retrieved the old file from the browser's cache memory. The scenario would have been altered if the file had been modified. The server would have of course returned the contents of the file.

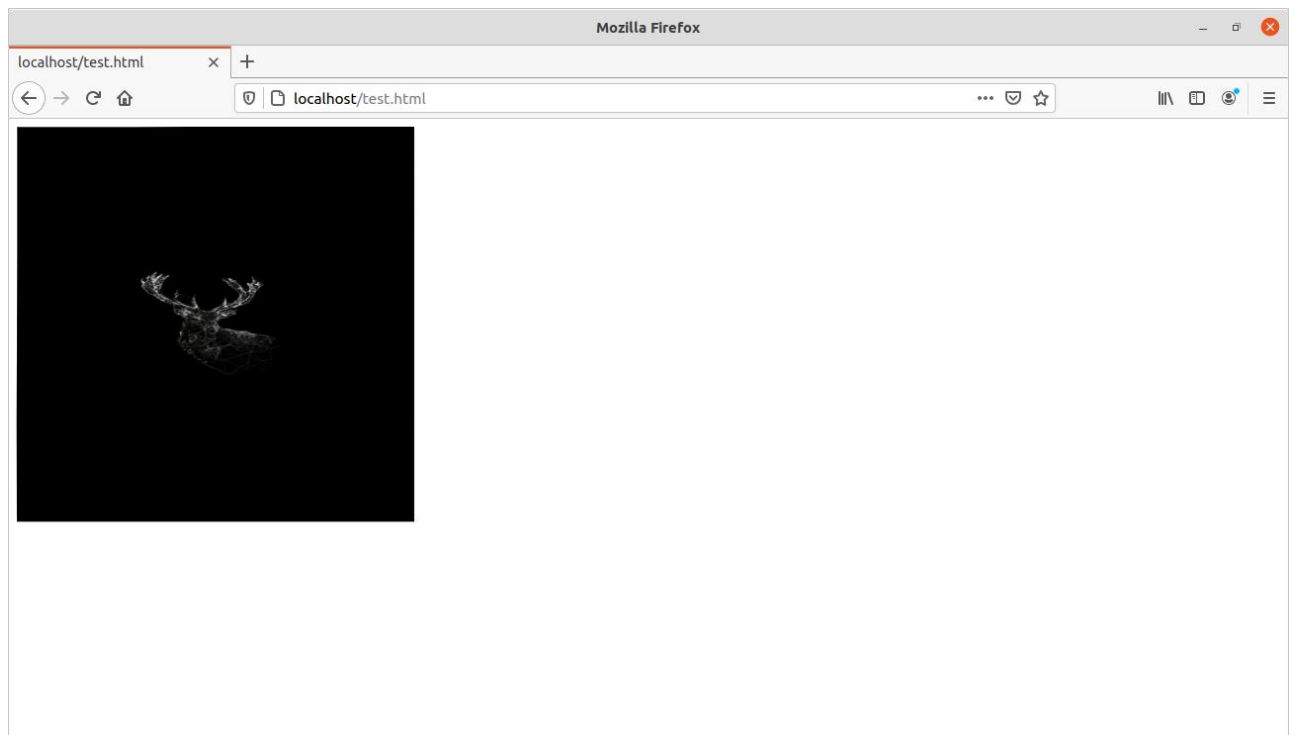
If the same experiment is repeated with a webpage that contains some images in it:

The HTML program:

```
Open  [icon] *test.html /var/www/html Save [icon] [icon] [icon]
1 <!DOCTYPE html>
2 <html>
3 
4 </html>
5
```

HTML ▾ Tab Width: 8 ▾ Ln 5, Col 1 ▾ INS

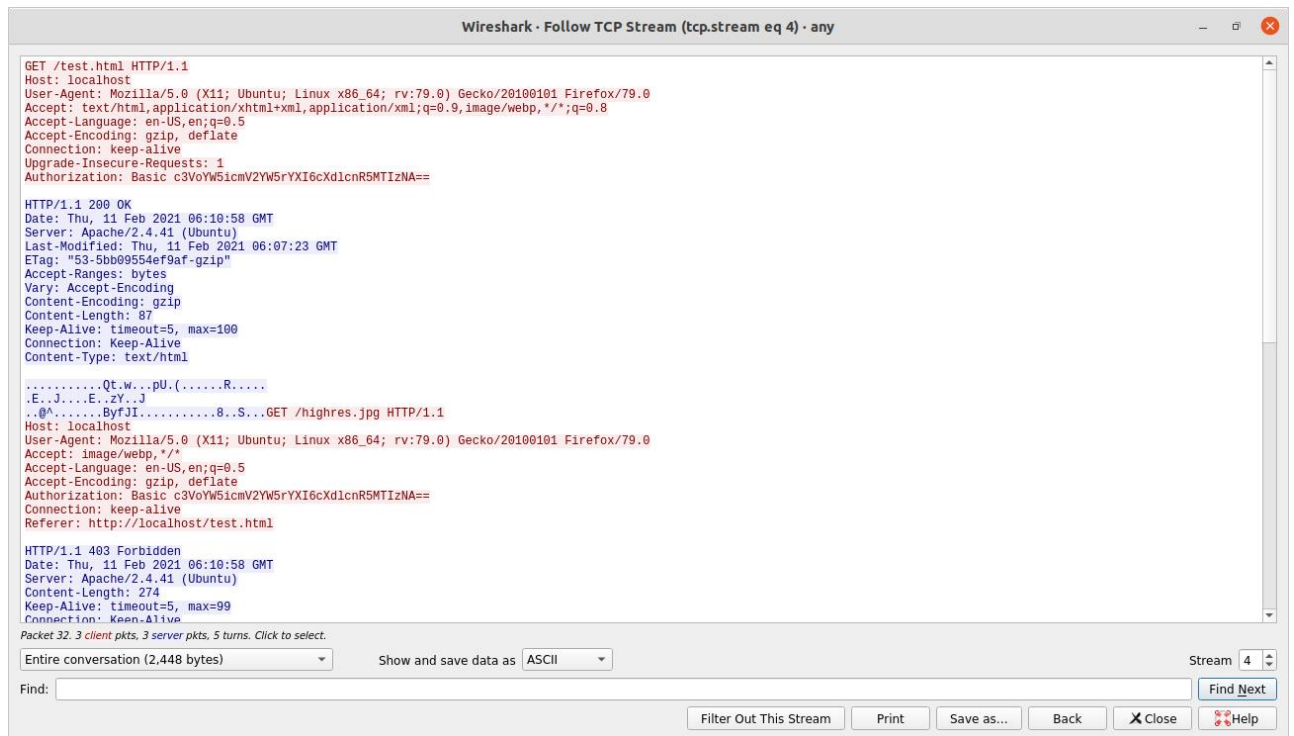
The webpage:



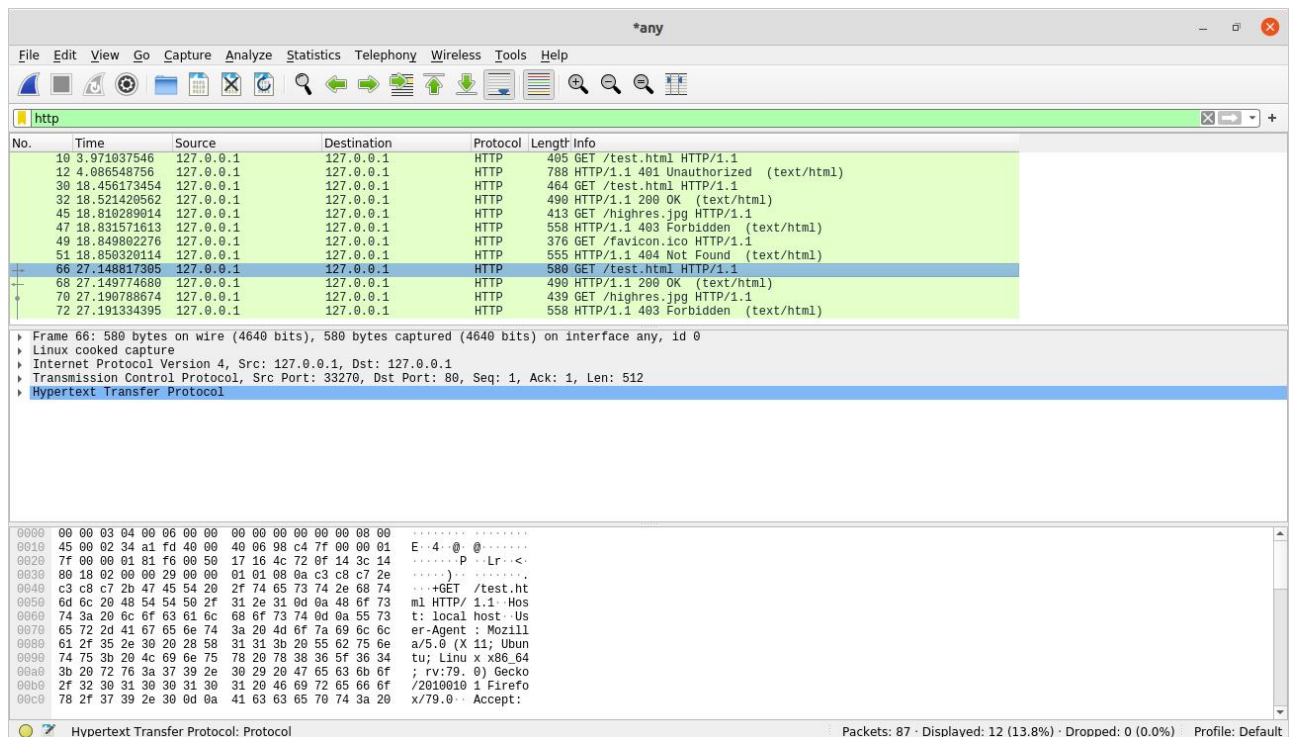
# Wireshark results:

*any					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help					
http					
No.	Time	Source	Destination	Protocol	Length Info
10	3.971037546	127.0.0.1	127.0.0.1	HTTP	405 GET /test.html HTTP/1.1
12	4.086548756	127.0.0.1	127.0.0.1	HTTP	788 HTTP/1.1 401 Unauthorized (text/html)
30	18.456173454	127.0.0.1	127.0.0.1	HTTP	464 GET /test.html HTTP/1.1
32	18.521420552	127.0.0.1	127.0.0.1	HTTP	490 HTTP/1.1 200 OK (text/html)
45	18.810289014	127.0.0.1	127.0.0.1	HTTP	413 GET /highres.jpg HTTP/1.1
47	18.831571613	127.0.0.1	127.0.0.1	HTTP	558 HTTP/1.1 403 Forbidden (text/html)
49	18.849802276	127.0.0.1	127.0.0.1	HTTP	376 GET /favicon.ico HTTP/1.1
51	18.850320114	127.0.0.1	127.0.0.1	HTTP	555 HTTP/1.1 404 Not Found (text/html)
66	27.148817305	127.0.0.1	127.0.0.1	HTTP	580 GET /test.html HTTP/1.1
68	27.149774680	127.0.0.1	127.0.0.1	HTTP	490 HTTP/1.1 200 OK (text/html)
70	27.190788674	127.0.0.1	127.0.0.1	HTTP	439 GET /highres.jpg HTTP/1.1
72	27.191334395	127.0.0.1	127.0.0.1	HTTP	558 HTTP/1.1 403 Forbidden (text/html)
Frame 30: 464 bytes on wire (3712 bits), 464 bytes captured (3712 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: 33268, Dst Port: 80, Seq: 1, Ack: 1, Len: 396					
Hypertext Transfer Protocol					
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 08 00 ..... 0010 45 00 01 c0 14 f0 40 00 40 06 26 46 7f 00 00 01 E.....@ @&F.... 0020 7f 00 00 01 81 f4 00 50 0b 63 7a 6c 17 35 3b 48 .....P czl 5;H 0030 80 18 02 00 ff b4 00 00 01 01 08 0a c3 c8 a5 3a ..... 0040 c3 c8 a5 3a 47 45 54 20 2f 74 65 73 74 2e 68 74 ...:GET /test.ht 0050 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 m1 HTTP/ 1.1 Hos 0060 74 3a 20 6c 6f 63 61 6c 68 6f 73 74 0d 0a 55 73 t: local host -Us 0070 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c er-Agent : Mozill 0080 61 2f 35 2e 30 20 28 58 31 31 3b 20 55 62 75 6e a/5.0 (X 11; Ubun 0090 74 75 3b 20 4c 69 6e 75 78 20 78 38 36 5f 36 34 tu; Linu x x86.64 00a0 3b 20 72 76 3a 37 39 2e 30 29 20 47 65 63 6b 6f ; rv:79. 0) Gecko 00b0 2f 32 30 31 30 30 31 30 31 20 46 69 72 65 66 6f /2010010 1 Firefo 00c0 78 2f 37 39 2e 30 0d 0a 41 63 63 65 70 74 3a 20 x/79.0 Accept:					
Hypertext Transfer Protocol: Protocol					
Packets: 87 · Displayed: 12 (13.8%) · Dropped: 0 (0.0%) · Profile: Default					

# The first response:



There is an If-modified-since line and returned the details of the images.





Wireshark · Follow TCP Stream (tcp.stream eq 5) · any

```
GET /test.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
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 c3VoYW51cmV2YW5rYXI6cXdlcnR5MTIzNA==
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Thu, 11 Feb 2021 06:07:23 GMT
If-None-Match: "53-5bb09554ef9af-gzip"
Cache-Control: max-age=0

HTTP/1.1 200 OK
Date: Thu, 11 Feb 2021 06:11:07 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Thu, 11 Feb 2021 06:07:23 GMT
ETag: "53-5bb09554ef9af-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 87
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

.....Qt.w...pU.(.....R.....
.E..J....E..zY..J
..0^.....ByfJl.....8..S...GET /highres.jpg HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic c3VoYW51cmV2YW5rYXI6cXdlcnR5MTIzNA==
Connection: keep-alive
Referer: http://localhost/test.html
Cache-Control: max-age=0

HTTP/1.1 403 Forbidden
Date: Thu, 11 Feb 2021 06:11:07 GMT
Packet 66: 2 client pkts, 2 server pkts, 3 turns. Click to select.
```

Entire conversation (1,795 bytes)      Show and save data as ASCII      Stream 5

Find:

\*\*\*\*\*