

## Week: #3

## Understand working of HTTP Headers

	<p><b>Understand working of HTTP headers:</b></p> <p>Conditional Get: If-Modified-Since</p> <p>HTTP Cookies: Cookie and Set-Cookie</p> <p>Authentication: Auth-Basic</p> <p>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.</p> <p>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.</p> <p><b>Observation:</b> Show the behavior of browser when is cookie is set and when cookie is removed.</p>
--	---

## Week: 5

### 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 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. 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**

```

osboxes@osboxes: ~
osboxes@osboxes:~$ sudo apt-get install apache2-utils
[sudo] password for osboxes:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  apache2-utils
1 upgraded, 0 newly installed, 0 to remove and 247 not upgraded.
Need to get 81.7 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-utils amd64 2.4.18-2ubuntu3.10 [81.7 kB]
Fetched 81.7 kB in 1s (45.5 kB/s)
(Reading database ... 217540 files and directories currently installed.)
Preparing to unpack ../apache2-utils_2.4.18-2ubuntu3.10_amd64.deb ...
Unpacking apache2-utils (2.4.18-2ubuntu3.10) over (2.4.18-2ubuntu3.9) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up apache2-utils (2.4.18-2ubuntu3.10) ...
osboxes@osboxes:~$

```

--> Provide username and password to set authentication

**sudo htpasswd -c /etc/apache2/.htpasswd ANY\_USERNAME**

```

osboxes@osboxes:~$ sudo htpasswd -c /etc/apache2/.htpasswd netwo
New password:
Re-type new password:
Adding password for user netwo
osboxes@osboxes:~$ sudo cat /etc/apache2/.htpasswd
netwo:$apr1$6YdDa0Ti$ELrUa0lQ/jun9TTU1PYKu/
osboxes@osboxes:~$

```

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

--> View the authentication

**sudo cat /etc/apache2/.htpasswd**

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**

```

<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>
</VirtualHost>

```

```

GNU nano 2.5.3      File: /etc/apache2/sites-available/000-default.conf

<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>
</VirtualHost>

```

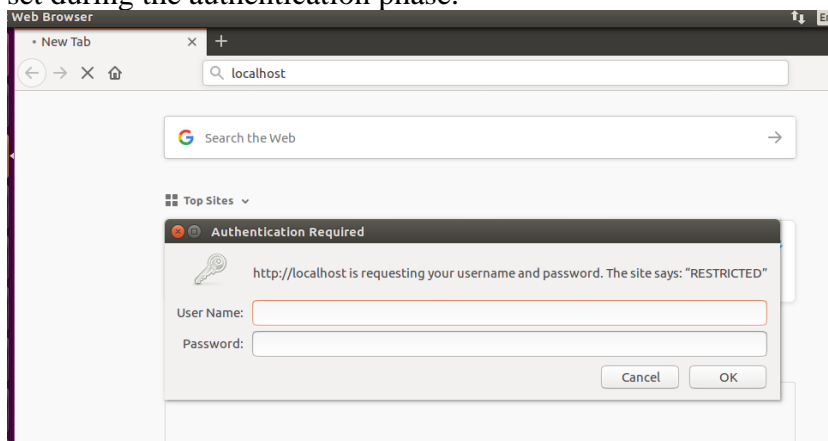
3. Password policy implementation is done by restarting the server as:  
**sudo service apache2 restart**

```

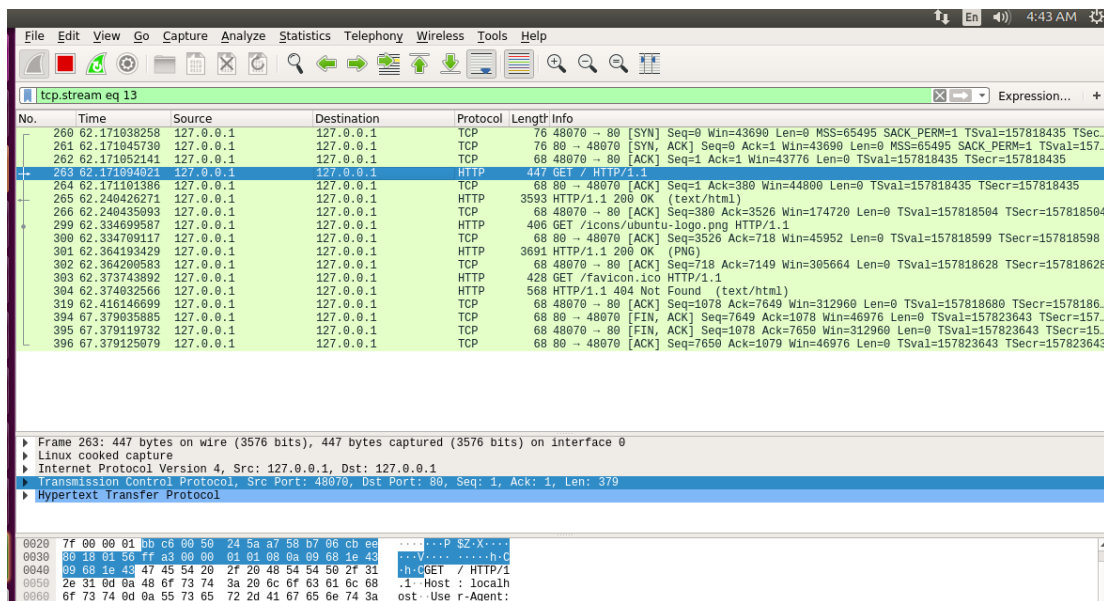
osboxes@osboxes:~$ sudo service apache2 restart
osboxes@osboxes:~$

```

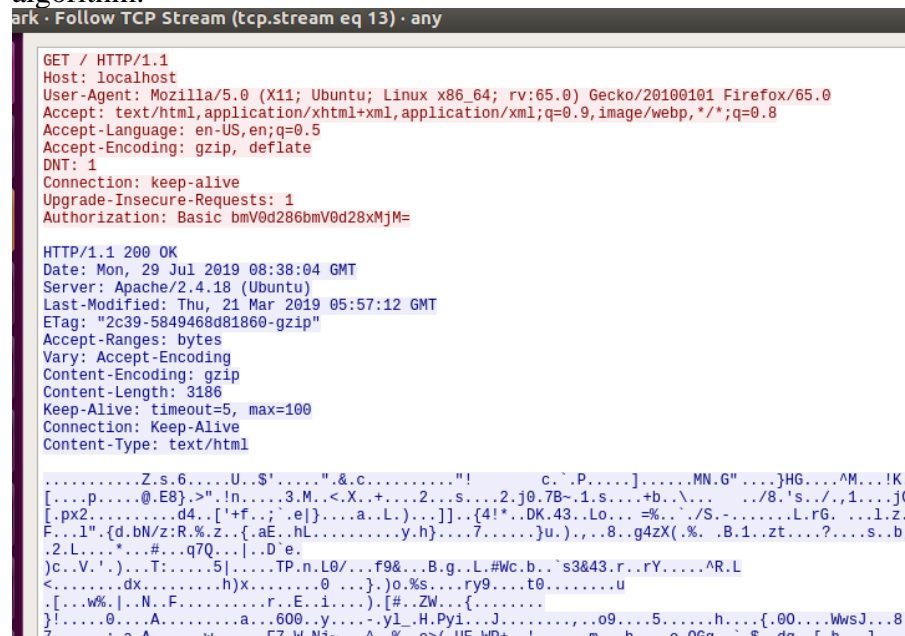
4. The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase.



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



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.



## 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:

```
<html>
<?php
```

```
        setcookie("namecookie","netqwerty",time()+123);
        setcookie("nickname","work");
?>
<img src= "highres.png" width= "300" height= "300" title= "password" />
</html>
```

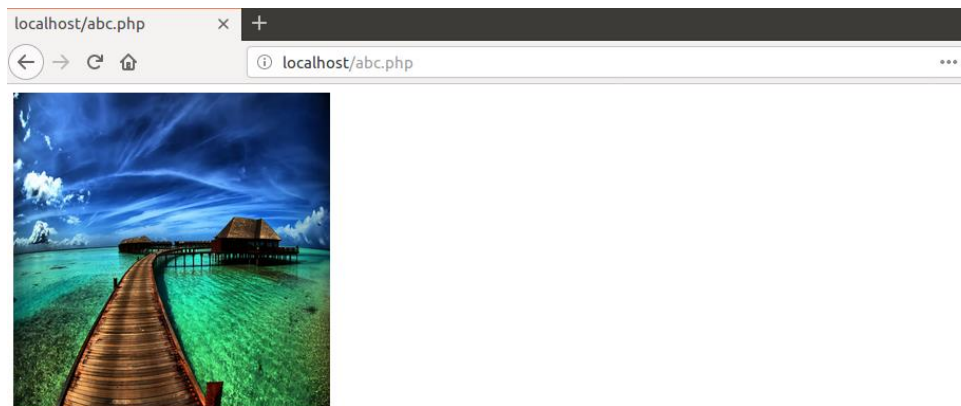
```
GNU nano 2.5.3      File: abc.php
<html>
<?php
setcookie("namecookie","netqwerty",time()+123);
setcookie("nickname","work");
?>

</html>
```

Note: Here you can add any image if required

**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.

tcp.stream eq 11						
No.	Time	Source	Destination	Protocol	Length	Info
309	23.481763156	127.0.0.1	127.0.0.1	TCP	76	48404 → 80 [SYN] Seq=0 Win=43690 Len=0 MSS=65495
310	23.481770358	127.0.0.1	127.0.0.1	TCP	76	80 → 48404 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0
311	23.481776479	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSV
312	23.481809760	127.0.0.1	127.0.0.1	HTTP	454	GET /abc.php HTTP/1.1
313	23.481816171	127.0.0.1	127.0.0.1	TCP	68	80 → 48404 [ACK] Seq=1 Ack=387 Win=44800 Len=0
314	23.482261152	127.0.0.1	127.0.0.1	HTTP	524	HTTP/1.1 200 OK (text/html)
315	23.482299412	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=387 Ack=457 Win=44800 Len=0
327	23.581439939	127.0.0.1	127.0.0.1	HTTP	448	GET /highres.jpg HTTP/1.1
328	23.581712586	127.0.0.1	127.0.0.1	TCP	22468	80 → 48404 [ACK] Seq=457 Ack=767 Win=45952 Len=2
329	23.581718698	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=22857 Win=175744 Len=0
330	23.581735122	127.0.0.1	127.0.0.1	TCP	65551	80 → 48404 [PSH, ACK] Seq=22857 Ack=767 Win=45952 Len=0
331	23.581740280	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=88340 Win=306816 Len=0
332	23.581756133	127.0.0.1	127.0.0.1	TCP	65551	80 → 48404 [ACK] Seq=88340 Ack=767 Win=45952 Len=0
333	23.581764076	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=153823 Win=437760 Len=0
334	23.581780285	127.0.0.1	127.0.0.1	TCP	65551	80 → 48404 [ACK] Seq=153823 Ack=767 Win=45952 Len=0
335	23.581785728	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=219306 Win=419840 Len=0
336	23.581801828	127.0.0.1	127.0.0.1	TCP	65551	80 → 48404 [ACK] Seq=219306 Ack=767 Win=45952 Len=0
337	23.581806654	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=284789 Win=386560 Len=0
338	23.581820136	127.0.0.1	127.0.0.1	HTTP	61937	HTTP/1.1 200 OK (JPEG JFIF image)
339	23.581824816	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=767 Ack=346658 Win=355200 Len=0
340	23.607036646	127.0.0.1	127.0.0.1	HTTP	473	GET /favicon.ico HTTP/1.1
341	23.607286489	127.0.0.1	127.0.0.1	HTTP	568	HTTP/1.1 404 Not Found (text/html)
342	23.641216059	127.0.0.1	127.0.0.1	TCP	68	48404 → 80 [ACK] Seq=1172 Ack=347158 Win=568704

▶ Frame 312: 454 bytes on wire (3632 bits), 454 bytes captured (3632 bits) on interface 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 48404, Dst Port: 80, Seq: 1, Ack: 1, Len: 386  
 ▶ **Hypertext Transfer Protocol**

```

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

GET /abc.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
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
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Authorization: Basic bmV0d286bmV0d28xMjM=

HTTP/1.1 200 OK
Date: Mon, 29 Jul 2019 09:10:23 GMT
Server: Apache/2.4.18 (Ubuntu)
Set-Cookie: namecookie=netqwert; expires=Mon, 29-Jul-2019 09:12:26 GMT; Max-Age=123
Set-Cookie: nickname=work
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 92
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

.....Mw(.J.U..L.(J-..*HwR(.L)..U260PR.H..%9.%9..J.....E)J.0#.!Fq....S...GET /highres.jpg HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/abc.php
DNT: 1
Authorization: Basic bmV0d286bmV0d28xMjM=
Connection: keep-alive
Cookie: namecookie=netqwert; nickname=work

```

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/wireshark-labs/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:**

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

**Repeat the above task with some images on the server.**

**Attach screenshots wherever necessary.**