IN423 - PW 03

HTTP POST and Network Analysis

1. Objectives and instructions

In this third practical work we will continue our previous work on TCP and HTTP. Follow the following instructions.

2. HTTP

In this part we will see how to enhance our HTTP server to make it able to send different kinds of data (encoding) and receive POST requests with data.

HTTP POST:

We are going to analyze the post requests made by the client. A post request is composed as follows: an http header ending with an empty line, followed by the data.

1 - Adapt your http server so it displays the entire requests sent by the client.

2- Start your http server

3 - Use your web browser to open the control.html page and click on one button. 4

– Take a look at the POST request (print by the server)

2.1 What is the value (data) associated with the post? Where it is located on the request body (i.e. received message) ?

```
  The data is "move=forward". It is located after a blank after the
header
```

2.2 Adapt the server so that it prints "move forward", "move backward" etc. depending on the post request.

Nb: to separate the HTTP header part from the HTTP data part that contain the post data (i.e. "move backward" etc.) you should use the sequence asciisep

= "\r\n\r\n"

as the separator.

As the request and the data are all made of ascii text you could use it as string manipulation. But it could be better to directly split the raw binary request with

binsep = b"\r\n\r\n"

2.3 Adapt your server so that replies to the post request with an HTTP 200 message and the control.html page as data.

```
See code
```

HTTP 304 Not Modified:

What is a cache?

A cache is information stored on the client side or on an intermediate server to increase the access speed to this information if it was not modified since last access.

For web content this is a common way to accelerate the web-page load time.

In the HTTP 200 response: activate cache control

To enable cache management the HTTP 200 messages from the server must contain cache control information.

There are different possible options, but here we will only add the following option in the HTTP 200 responses to activate the cache management on the client side:

`Cache-Control: max-age=<seconds>` (put

3600 for <seconds> value).

You must also indicate the expiration date for the web-resource by adding the expires field:

`Expires: <http-date>` Example:

```
Expires: Wed, 21 Oct 2015 07:28:00 GMT
```

Now use the 304 not modified message

If the client uses the

```
If-Modified-Since: <label-day>, <day> <month> <year>
<hour>:<minute>:<second> GMT
```

Example:

```
If-Modified-Since: Wed, 21 Oct 2015 07:28:00 GMT
```

Here is an example of an HTTP 304 message that tells the client that the requested resource is not modified and that it should not refresh the page.

```
HTTP/1.1 304 Not Modifed\r\n
Date : Mon, Apr 26 07:43:53 2021 GMT\r\n
Expire : Mon, Apr 26 08:43:53 2021 GMT\r\n
Etag : ipsav1
Cache-control : max-age=3600\r\n
\r\n
```

2.4 **Use this example but adapt values of Date and Expires options to respond to the image request when it has already been sent.**

/!\ For now this part is only working with google chrome /!

The date should be the current date: `time.asctime(time.gmtime())`

The Expires field should be filled with:

`time.asctime(time.gmtime(time.time()+3600)) + " GMT"`

explanation:

time.time() gives current time in seconds

time.time()+3600 gives current time in seconds + 3600 seconds

time.gmtime() translate time to GMT time.asctime() make it a

string

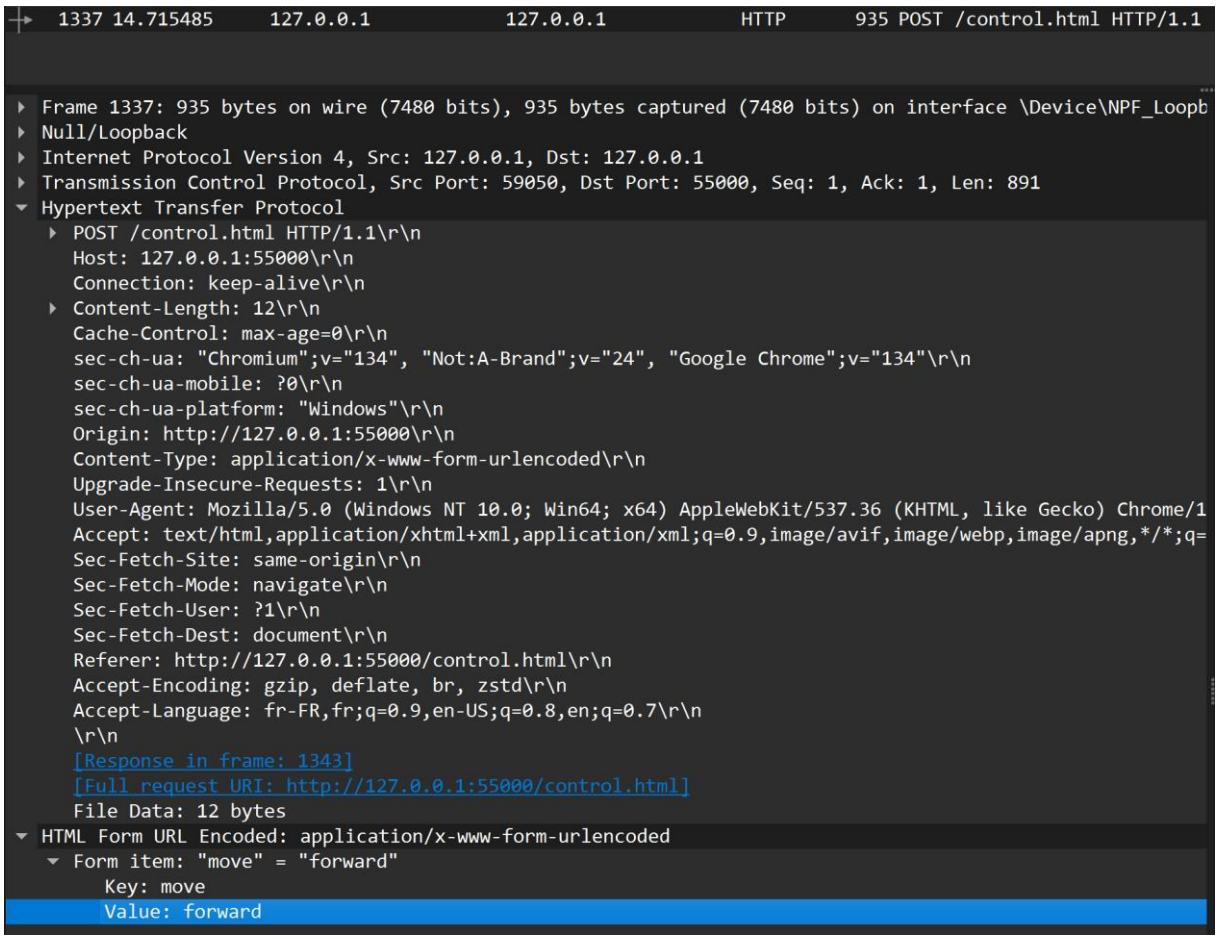
3. Network analysis

Use Wireshark to capture POST requests and responses of your server.

3.1 What is the filter you have used

```
http.request.method == "POST"
```

3.2 Copy your results here:

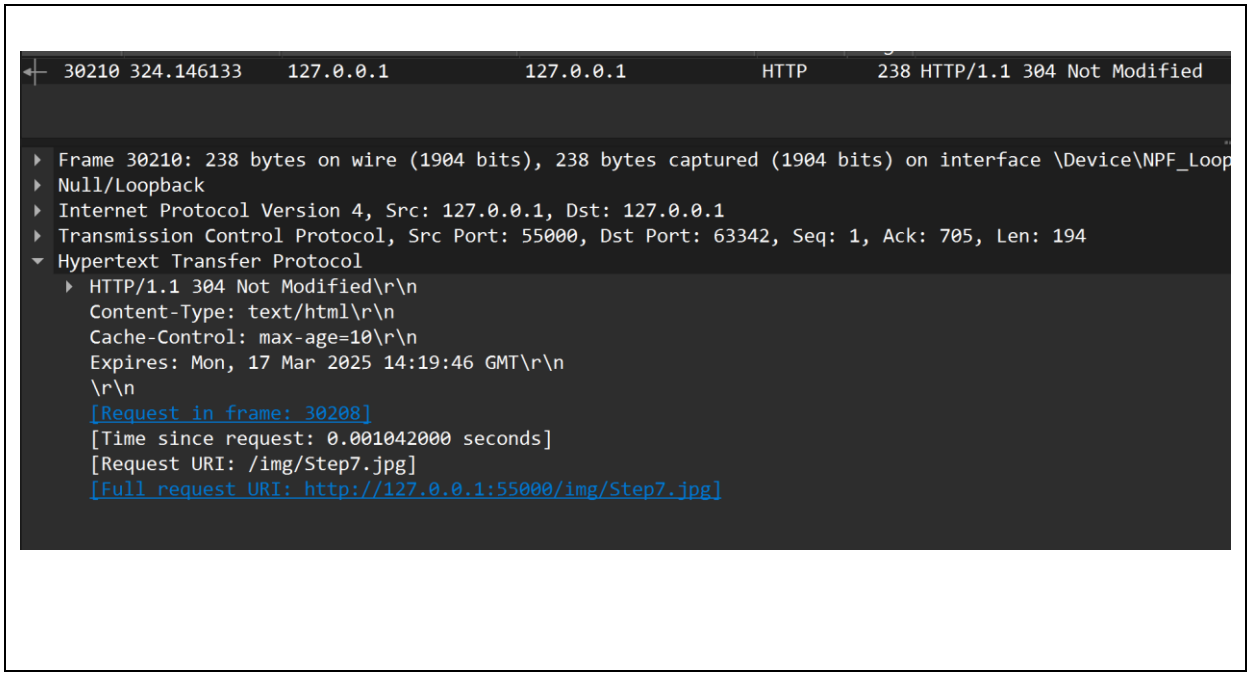


Use Wireshark to capture your HTTP 304

3.3 What filter do you have used?

```
http.response.code == 304
```

3.4 Copy your results here:


30210 324.146133 127.0.0.1 127.0.0.1 HTTP 238 HTTP/1.1 304 Not Modified
Frame 30210: 238 bytes on wire (1904 bits), 238 bytes captured (1904 bits) on interface \Device\NPF_Loop
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 55000, Dst Port: 63342, Seq: 1, Ack: 705, Len: 194
Hypertext Transfer Protocol
HTTP/1.1 304 Not Modified\r\n
Content-Type: text/html\r\n
Cache-Control: max-age=10\r\n
Expires: Mon, 17 Mar 2025 14:19:46 GMT\r\n
\r\n
[Request in frame: 30208]
[Time since request: 0.001042000 seconds]
[Request URI: /img/Step7.jpg]
[Full request URI: http://127.0.0.1:55000/img/Step7.jpg]