Lab 2: NetNinny

User Manual

1. Development details :

The proxy has been developed, compiled on Ubuntu, a Linux distribution, thanks to the integrated development environment CodeBlocks. It has been coded in C language.
The proxy has been tested on the browser Mozilla Firefox and works also on the browser Chromium (the proxy is faster on Mozilla Firefox).

2. How to configure your browser to use the Proxy :

In Mozilla Firefox:

You have to click on the 3 parallel bands, select the “Parameters”, go in “Advanced” tab and then click on the “Network” field.
Now you have to configure the connection in this way: you have to choose “Manual configuration of the proxy”, put on the Proxy HTTP form: 0.0.0.0 (which indicates you host the proxy on your own IP address) and then you select the port you want to use for the proxy.
You have to tick the “Use this server for all the protocols” box.

In Chromium:

You have to click on the 3 parallel bands, select the “Parameters” and then click on the “Display the advanced parameters” field.
Now go on the “Network” section and click on “Configure the proxy server”. You have to tick the “Manual configuration of the proxy” box, and write in the “HTTP Server” field: 0.0.0.0 and select the port you want to use in your proxy.

3. How to use the Web proxy :

To use the proxy, you have to execute the executable file named “proxy”. To execute it, you have to open a terminal (Ctrl + Shift + T), and place yourself on the Proxy folder (with the “cd command”). Then, you have to type this line in the console to

execute the program:
./proxy
Then, you have to enter the same port number as the one you have used in your browser parameters.
You can now use the Proxy!

Proxy's Features:

1. HTTP 1.0 / HTTP 1.1

The proxy can support both HTTP versions (1.0 and 1.1).
In fact the proxy manipulates the HTTP Request sent by the browser, in different ways depending of this version of HTTP. This functionality is more detailed in the "Request_Manipulation" function contained in the proxy's source code.

2. Handling HTTP GET interactions between client and server

The proxy is able to redirect HTTP GET requests from a client to a server, and to transmit the server's response to this client. First, the server part of the proxy receives the browser's request and delivers it to the proxy's client part. Then, this part manipulates the request sent by the client to deliver it properly to the server with the "Request_Manipulation" function. The server part also receives the response of the server and transmits it to the browser's client. This functionality is more detailed in the "client_part" and "server_part" functions contained in proxy's source code.

3. Blocking requests for undesirable URLs

The proxy blocks the HTTP Requests with undesirable URLs. In fact, when the client part receives the browser's request, it verifies if the request contains an undesirable URL. If the request contains a bad word, then the proxy doesn't send anything to the server and returns to the client an error page. This functionality is more detailed in the "client part" and "word filtering" functions contained in the proxy's source code.

4. Detection of inappropriate content bytes within a Web page

The proxy concatenates every packets coming from the server into a buffer. At the end of the sending, this buffer is filtered. This filtering is smart because it is done just for the not encoded text content.
If a bad word is found in the buffer, the proxy returns to the client an error page instead of the content of the page requested. This functionality is more detailed in the "client_part"and "word_filtering" functions contained in the proxy's source code.

5. Imposes no limit on the size of the transferred HTTP data

The proxy is able to deliver all the HTTP data to the client, without imposing a size limit.

In fact, after the filtering, the proxy sends another time the client-request to the server, and sends the response of the server to the client, packet by packet. These packets are just stored into a very big buffer (100 000 bytes) which is reset every time a packet has been sent to the client. The packet-size (typically < 1400 bytes) cannot exceed the buffer-size, so the proxy is able to deliver all the content without imposing a limit size. This functionality is more detailed in the "client_part" function contained in the proxy's source code.

6. Compatibility with several browsers

The proxy can work with two browsers: Mozilla Firefox and Chromium.
In fact, the HTTP Request sent by the browser is modified differently depending upon the browser. This functionality is more detailed in the "Request_Modification" function in the proxy's source code.

7. Allowing the user to select the proxy port

The proxy allows at the beginning of its execution, the user to select its own port.
In fact, the user just has to type a number between 1000 and 65535. If the number is not between these bounds or if the content entered is not a number, the user is asked again to choose a number between those bounds.
However, if the user enters a number between the bounds followed by a character chain, this number will be chosen for the port number. This functionality is more detailed in the "select_port" and in the "server_part" functions contained in the proxy's source code.

8. Filtering depending on the content-type

The proxy filters only the non-encoded text content of the server's response.
In fact, the server looks into the first packet sent by the server containing the Headers of the total response. If the "Content-type: text" header is detected and if the "Content -encoding" header is not detected, then the proxy will filter all the server's response. On the other hand, the content will not be filter because it is either encoded or non-text content so there is no need to search into it. This functionality is more detailed in the "client_part" function contained in the proxy's source code.

Testing of the Proxy:

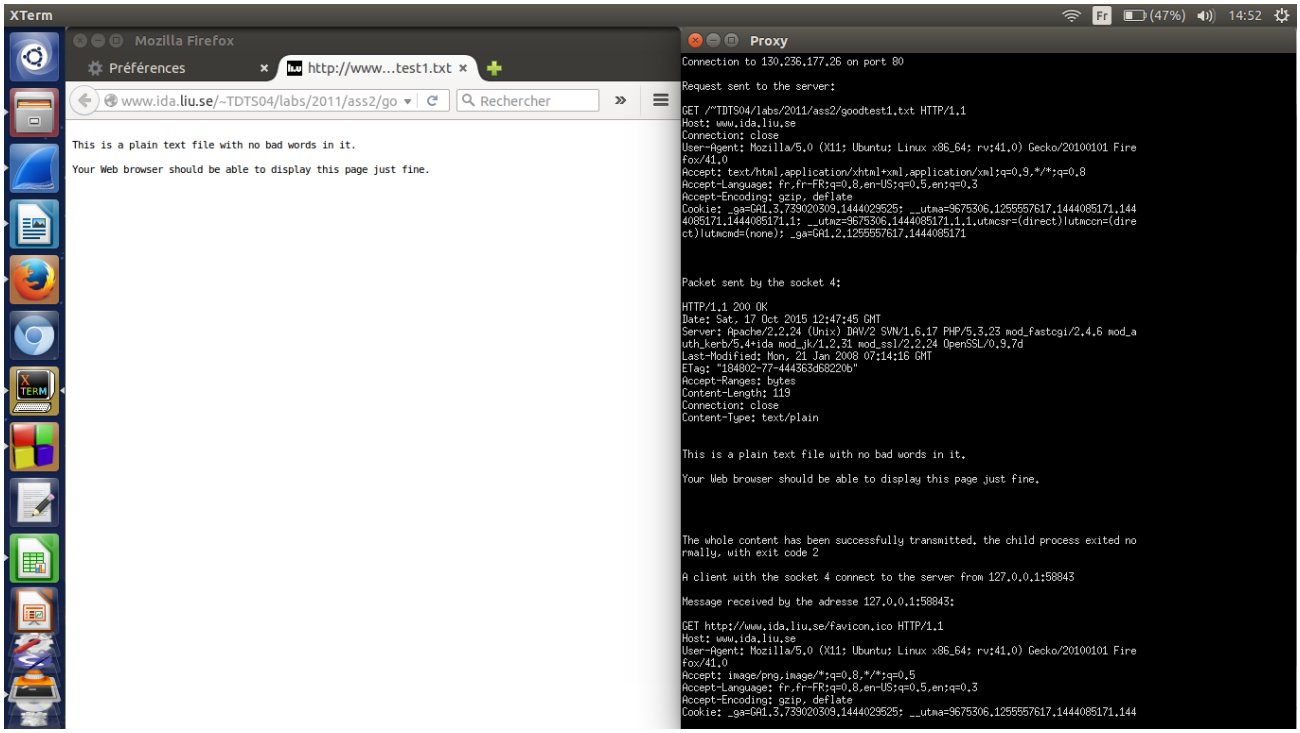
To prove you the proxy is efficient and works, a series of tests have been realized on our own computers.
You can go on the websites clicking on the bold blue text and pressing the Ctrl key at the same time.
You can zoom on the images pressing the Ctrl key and using the mouse wheel at the same time.

Websites tested:

1. Good text file test

The first test is the download of **a simple text file.**

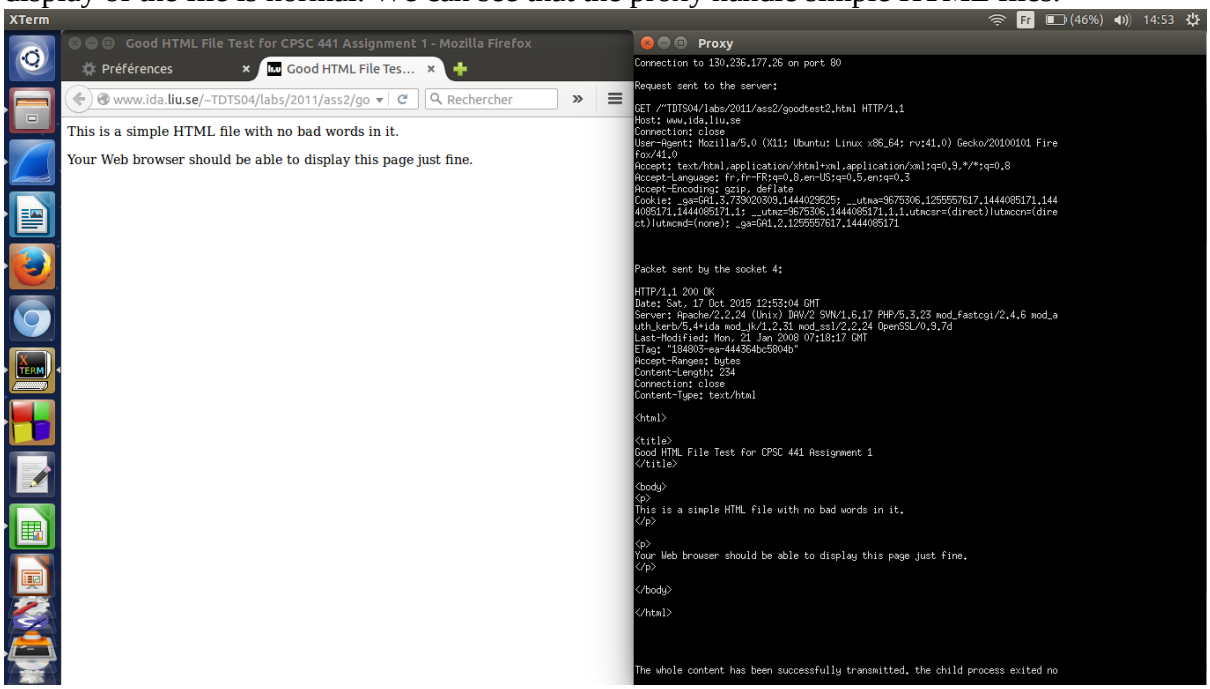
Of course, as we can see on this capture, this website is well downloaded. We can see the request send to the server (first paragraph) and then the server's response (second paragraph) with its content. We can see that the proxy handle simple text files.



2. Good HTML file test

The second test is the download of **a simple HTML file**.

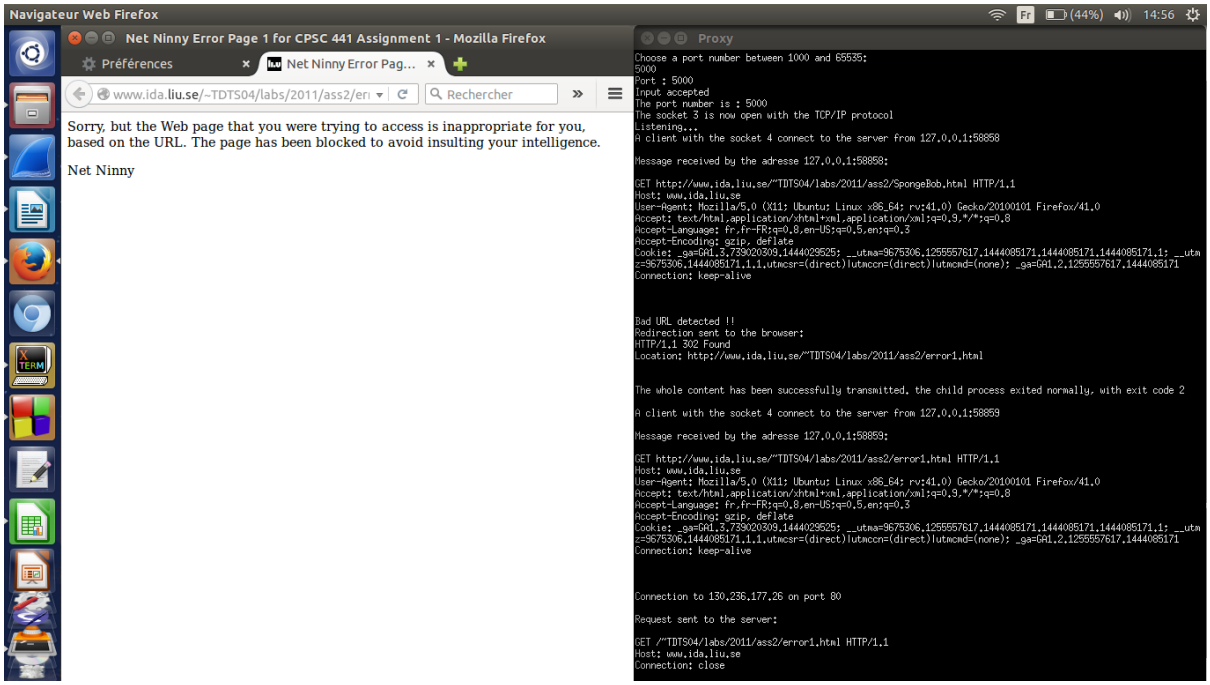
This download works and the display of the page is good. Here also, we can see the request sent to the server and its response. We can see the HTML tag in the terminal. Moreover, the display of the file is normal. We can see that the proxy handle simple HTML files.

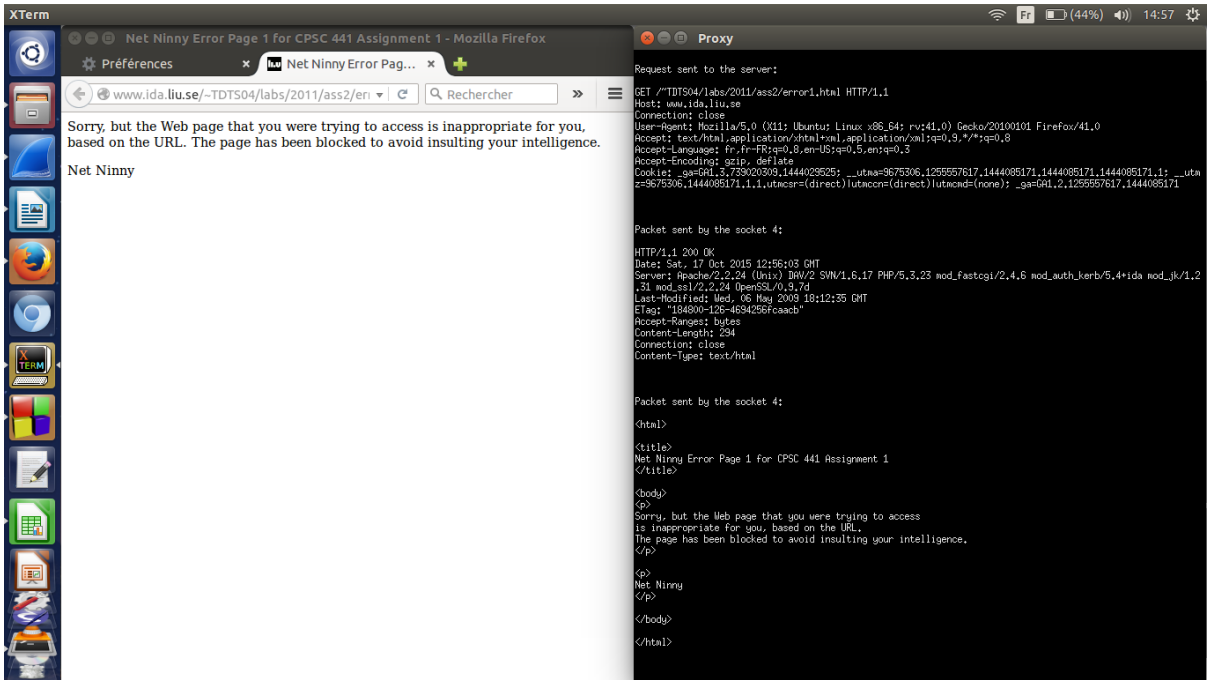


3. Bad URL HTML file test

The third test is the download of **a bad URL HTML file.**

This is the first test concerning the filtering of our proxy. Now, we want to check if the URL filtering works. So a bad URL is sent to the browser (it contains the word SpongeBob). We can see in the first capture that the proxy detects this bad URL. Then, we can see in the second capture that it will send to the browser the bad URL redirection page, which is displayed on the left of the capture. So the filtering of the URL address works.

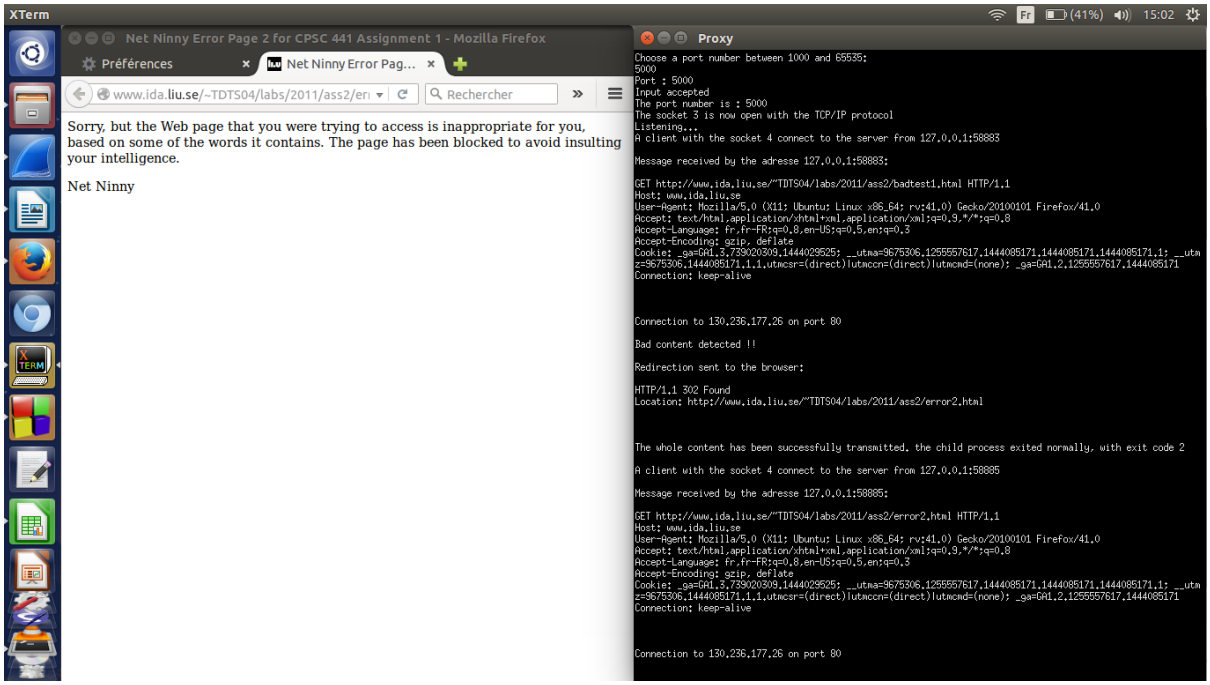


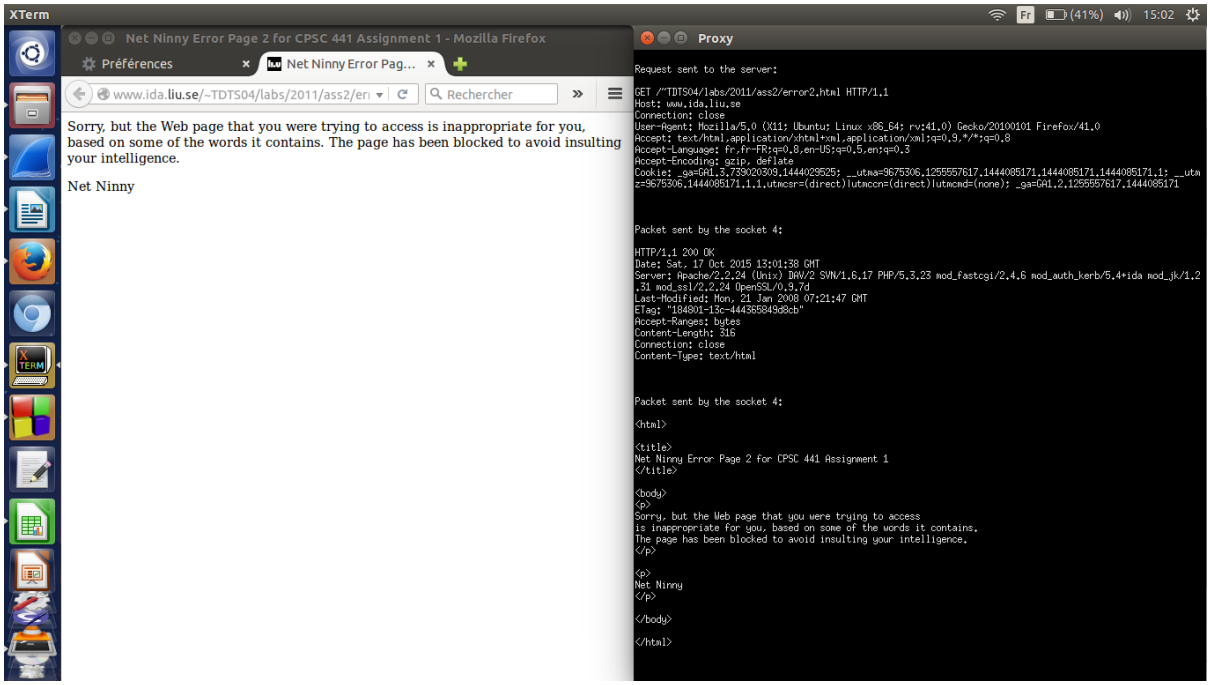


4. Bad content HTML file test

The fourth test is the download of **a bad content HTML file.**

The fourth test concerns the filtering of the content. We send to the proxy a page containing a bad word. We can see in the first capture that the bad word is detected. Then, the proxy sends directly to the browser the bad content redirection page. This page is displayed on the left of the captures. So the filtering of the content works.


XTerm
Net Ninny Error Page 2 for CPSC 441 Assignment 1 - Mozilla Firefox
Préférences
Net Ninny Error Pag...
www.ida.liu.se/~TDTS04/labs/2011/ass2/er
Rechercher
Sorry, but the Web page that you were trying to access is inappropriate for you, based on some of the words it contains. The page has been blocked to avoid insulting your intelligence.
Net Ninny
Proxy
Choose a port number between 1000 and 65535:
5000
Port : 5000
Input accepted
The port number is : 5000
The socket 3 is now open with the TCP/IP protocol
Listening...
A client with the socket 4 connect to the server from 127.0.0.1:58883
Message received by the adresse 127.0.0.1:58883:
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/badtest1.html HTTP/1.1
Host: www.ida.liu.se
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: _ga=GA1.3.739020309.1444029525; __utma=9675306.1255557617.1444085171.1444085171.1444085171.1; __utmz=9675306.1444085171.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); _ga=GA1.2.1255557617.1444085171
Connection: keep-alive
Connection to 130.236.177.26 on port 80
Bad content detected !!
Redirection sent to the browser:
HTTP/1.1 302 Found
Location: http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error2.html
The whole content has been successfully transmitted. the child process exited normally, with exit code 2
A client with the socket 4 connect to the server from 127.0.0.1:58885
Message received by the adresse 127.0.0.1:58885:
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error2.html HTTP/1.1
Host: www.ida.liu.se
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: _ga=GA1.3.739020309.1444029525; __utma=9675306.1255557617.1444085171.1444085171.1444085171.1; __utmz=9675306.1444085171.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); _ga=GA1.2.1255557617.1444085171
Connection: keep-alive
Connection to 130.236.177.26 on port 80


XTerm
Net Ninny Error Page 2 for CPSC 441 Assignment 1 - Mozilla Firefox
Préférences
Net Ninny Error Pag...
www.ida.liu.se/~TDTS04/labs/2011/ass2/er
Rechercher
Sorry, but the Web page that you were trying to access is inappropriate for you, based on some of the words it contains. The page has been blocked to avoid insulting your intelligence.
Net Ninny
Proxy
Request sent to the server:
GET /~TDTS04/labs/2011/ass2/error2.html HTTP/1.1
Host: www.ida.liu.se
Connection: close
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: _ga=GA1.3.739020309.1444029525; __utma=9675306.1255557617.1444085171.1444085171.1444085171.1; __utmz=9675306.1444085171.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); _ga=GA1.2.1255557617.1444085171
Packet sent by the socket 4:
HTTP/1.1 200 OK
Date: Sat, 17 Oct 2015 13:01:38 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6 mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Mon, 21 Jan 2008 07:21:47 GMT
ETag: "184801-13c-444365849d8cb"
Accept-Ranges: bytes
Content-Length: 316
Connection: close
Content-Type: text/html
Packet sent by the socket 4:
<html>
<title>
Net Ninny Error Page 2 for CPSC 441 Assignment 1
</title>
<body>

Sorry, but the Web page that you were trying to access
is inappropriate for you, based on some of the words it contains.
The page has been blocked to avoid insulting your intelligence.

Net Ninny

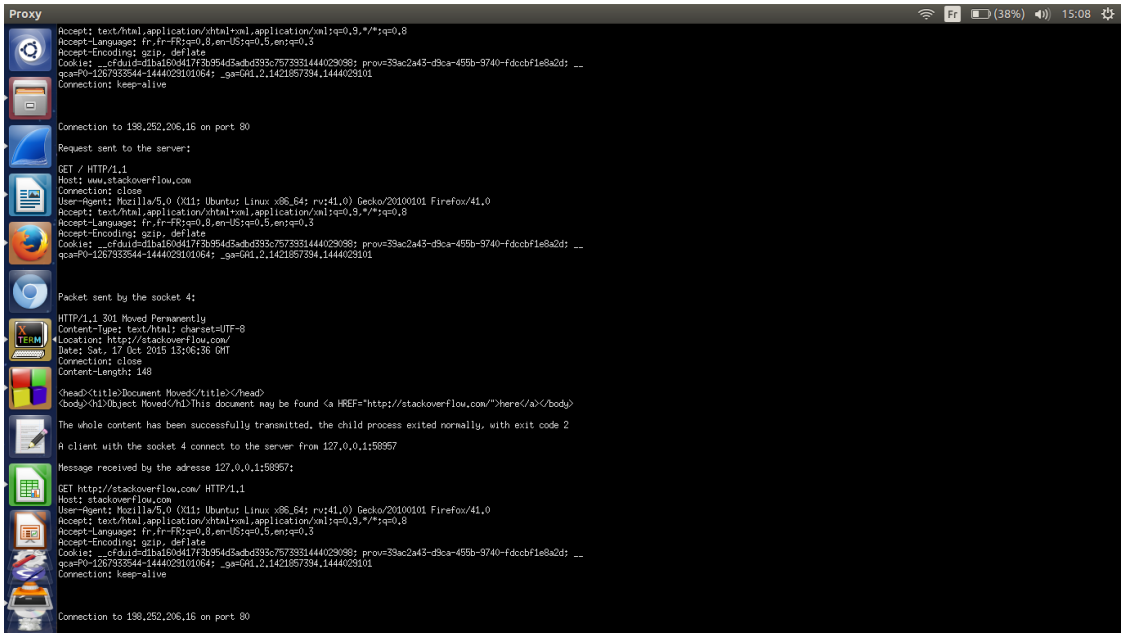
</body>
</html>

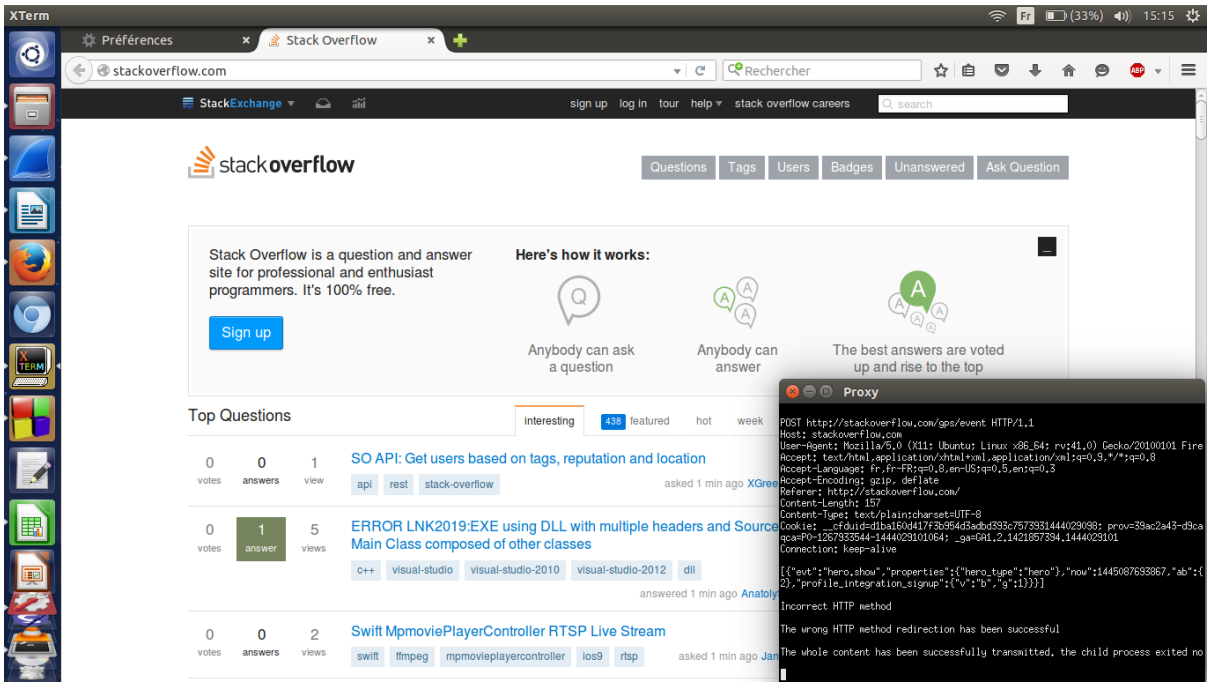
Download of complex websites.

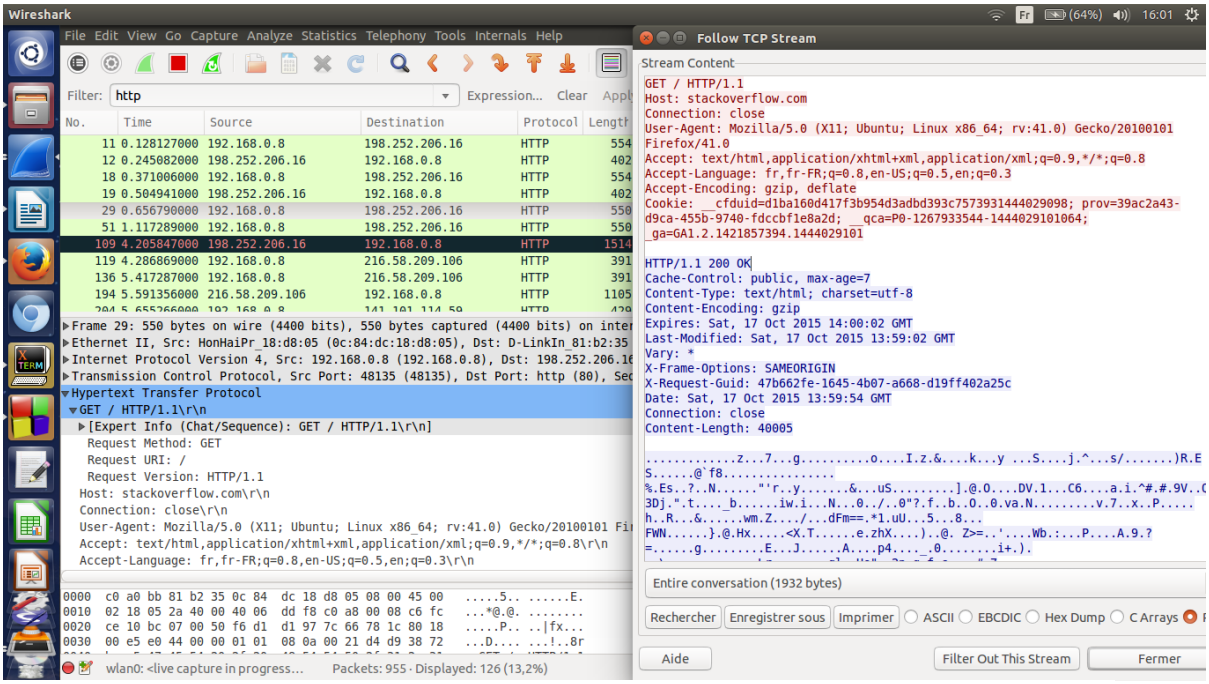
1. Stackoverflow

The first more complex website tested is the stackoverflow website. In the first capture, the request is well modified and sent to the server. Then, we can see the display of the site is good with the proxy: the text is correctly displayed, the HTML has been taken into account and some pictures appear. The writings have some layout which is handled by the CSS.

Finally, in the last capture, we can see in Wireshark that the request, the proxy sends to the server, gets a good answer: we can see the content in the Wireshark window. The status code of the answer is HTTP 200 OK, which means the transmission was successful.

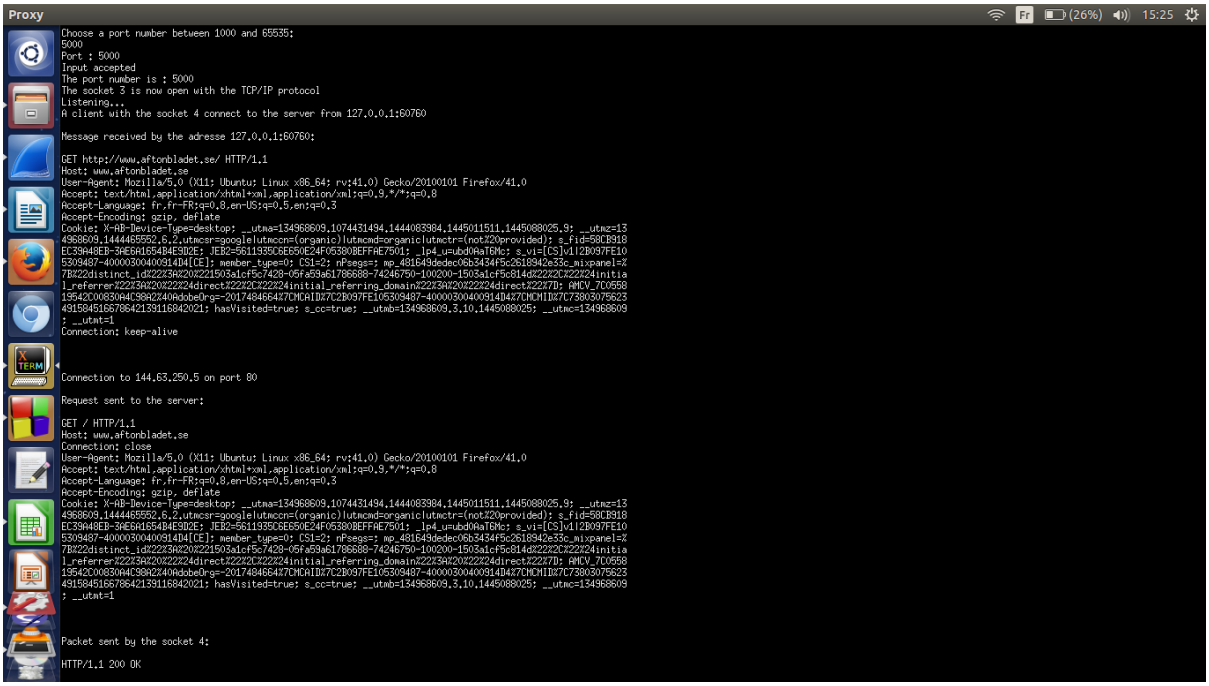






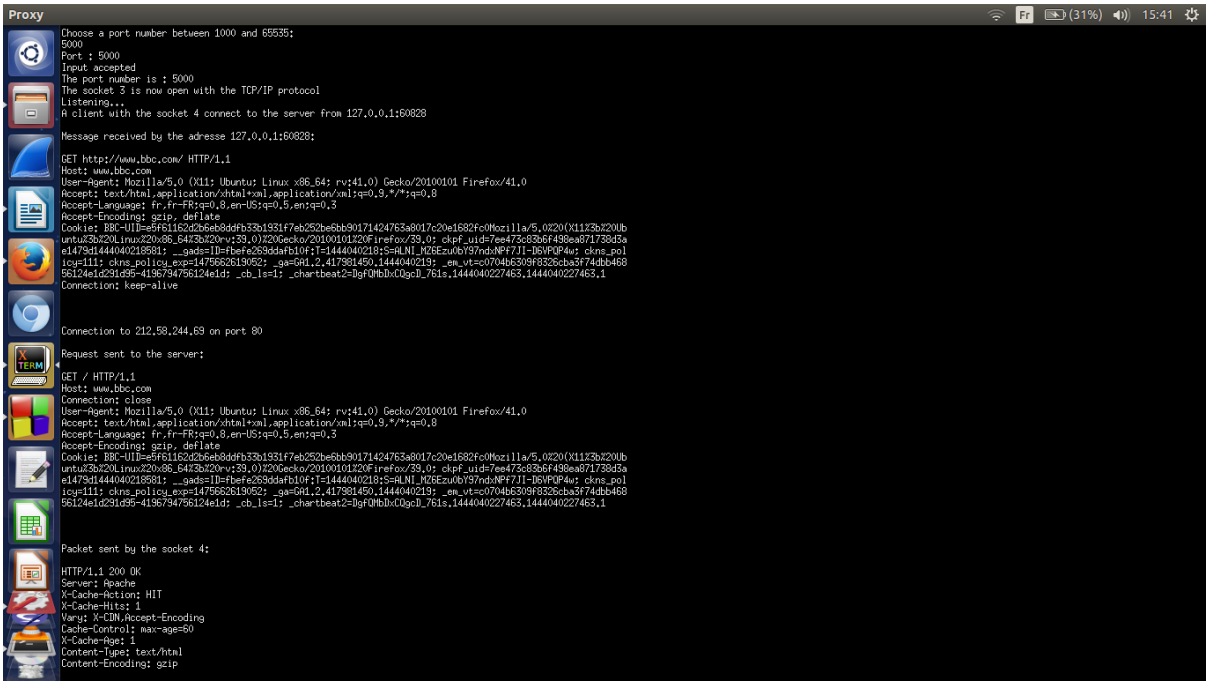
2. Aftonbladet

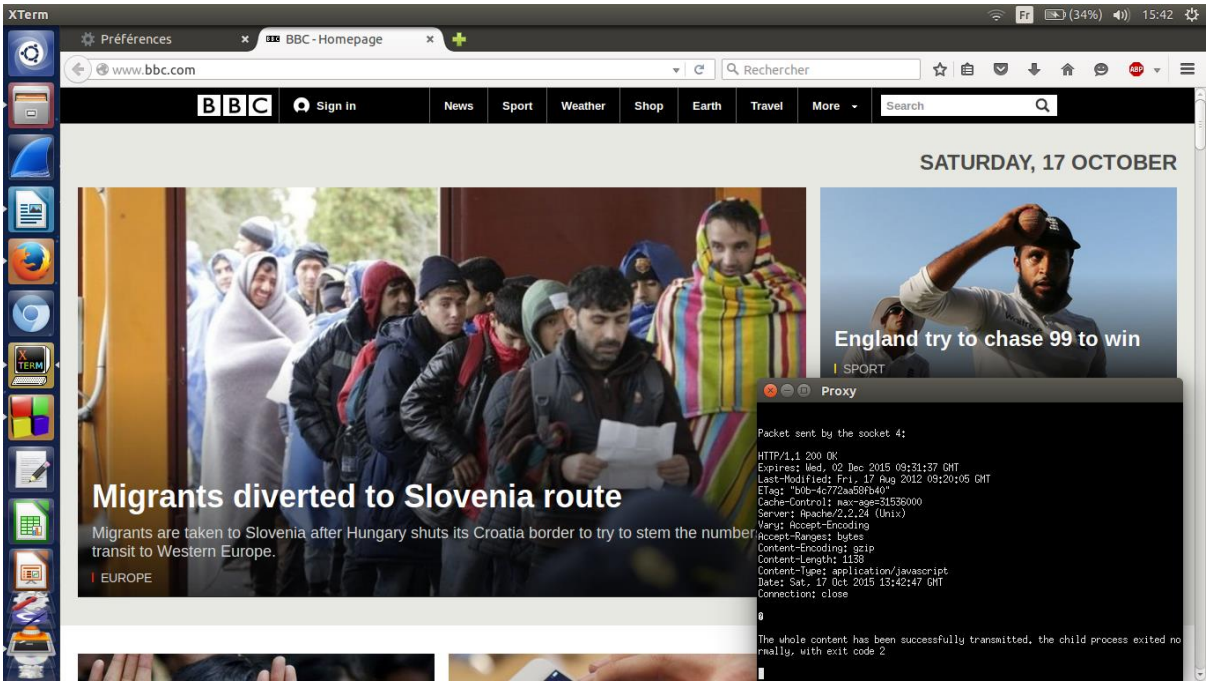
The second complex test is aftonbladet.se. In the first capture, we can see the request performed by the client and the request the proxy modifies and sends to the proxy.
Then, in the second picture, we can see the good display of the website. There are several images on this website which are displayed. So we can see that our proxy handles basic pictures. The layout is also well display. Our proxy handle the CSS.



3. BBC.

The third complex webpage tested is the **BBC** website. As before, we can see the request made by the client is modified and sent to the server. In the second capture, we can also see that the website is well displayed. Here also, there are several images.

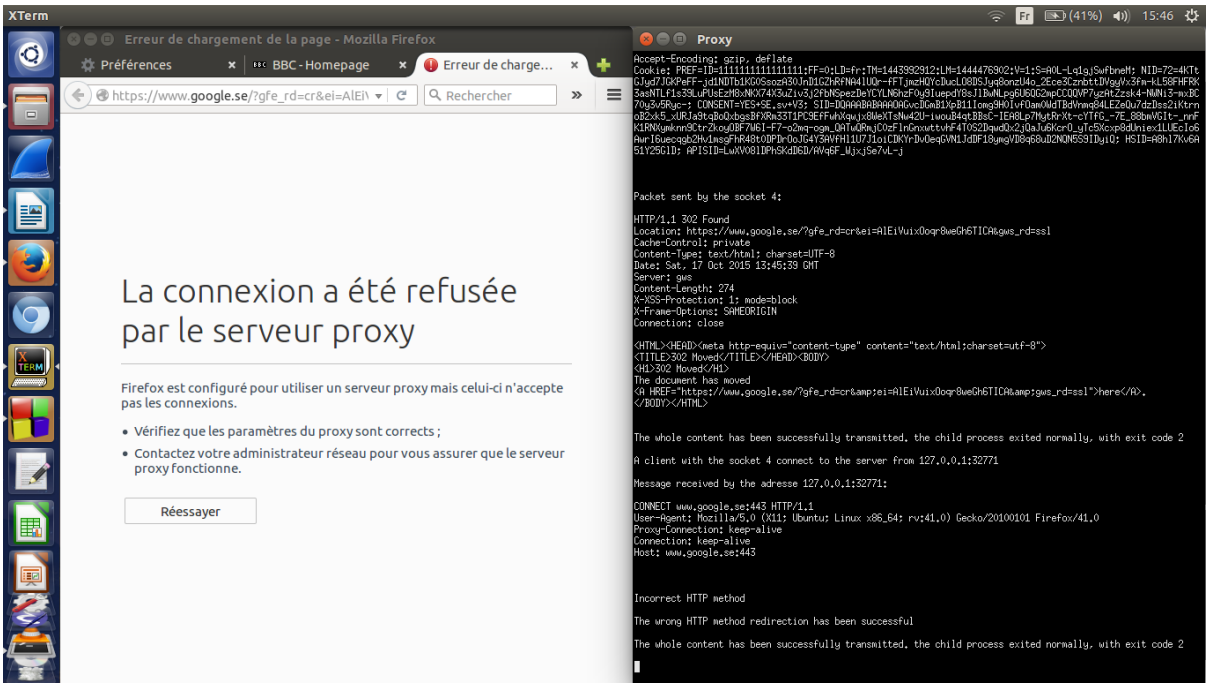




HTTPs pages

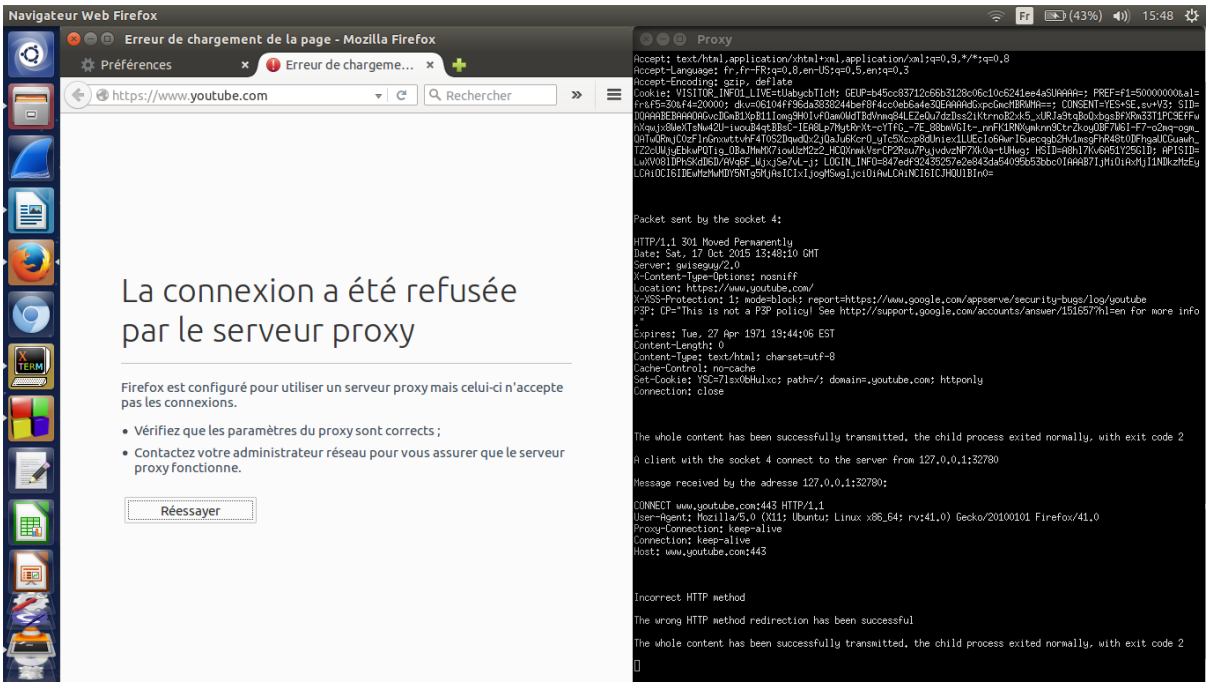
1. Simple google search

When someone realizes a research on Google, HTTPS protocol is used. The proxy does not have to handle this protocol. However, it must not crash when an HTTP request is received. In the capture below, we can see that when an HTTPS request arrived, the proxy sends an error message to the browser. It uses the “405 Method Not Allowed” status code and the browser reacts as we can see below. We can still launch another request: the proxy is still listening and wait for new connections to accept.



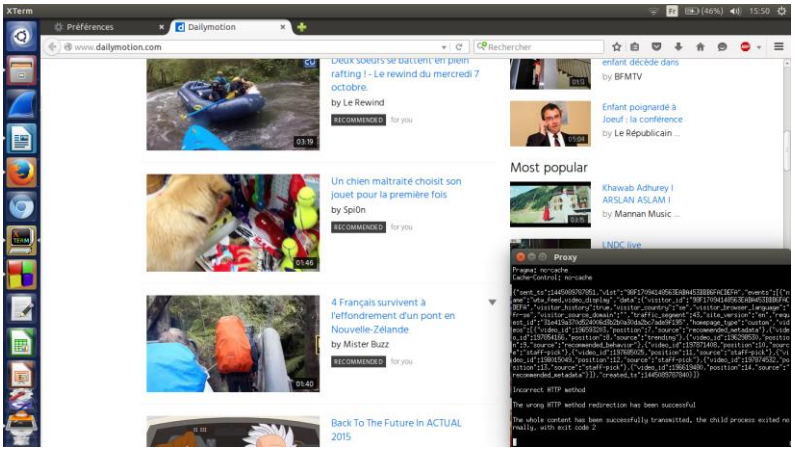
2. Youtube.com

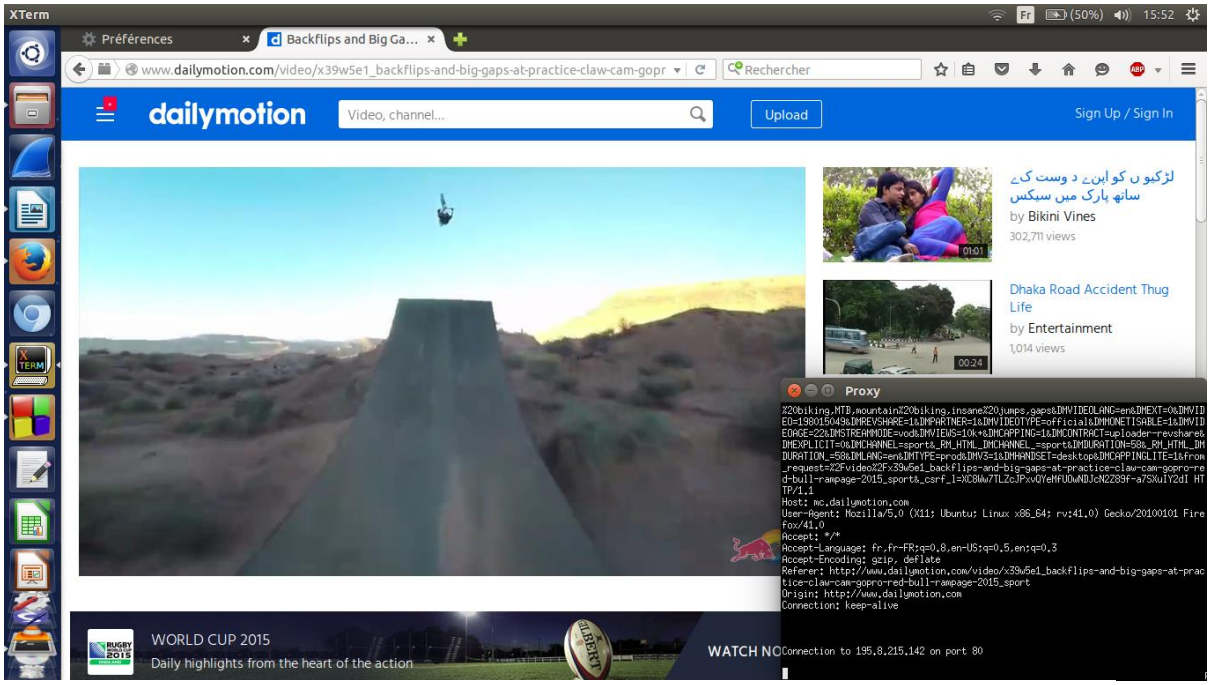
Then, we have tried to go on Youtube.com. In the capture below, the proxy sends an error message to the browser. This means that the HTTPS protocol is used. Indeed, we can see in the URL bar of the browser that the website uses the HTTPS Protocol.
We still can use the proxy after this error message.

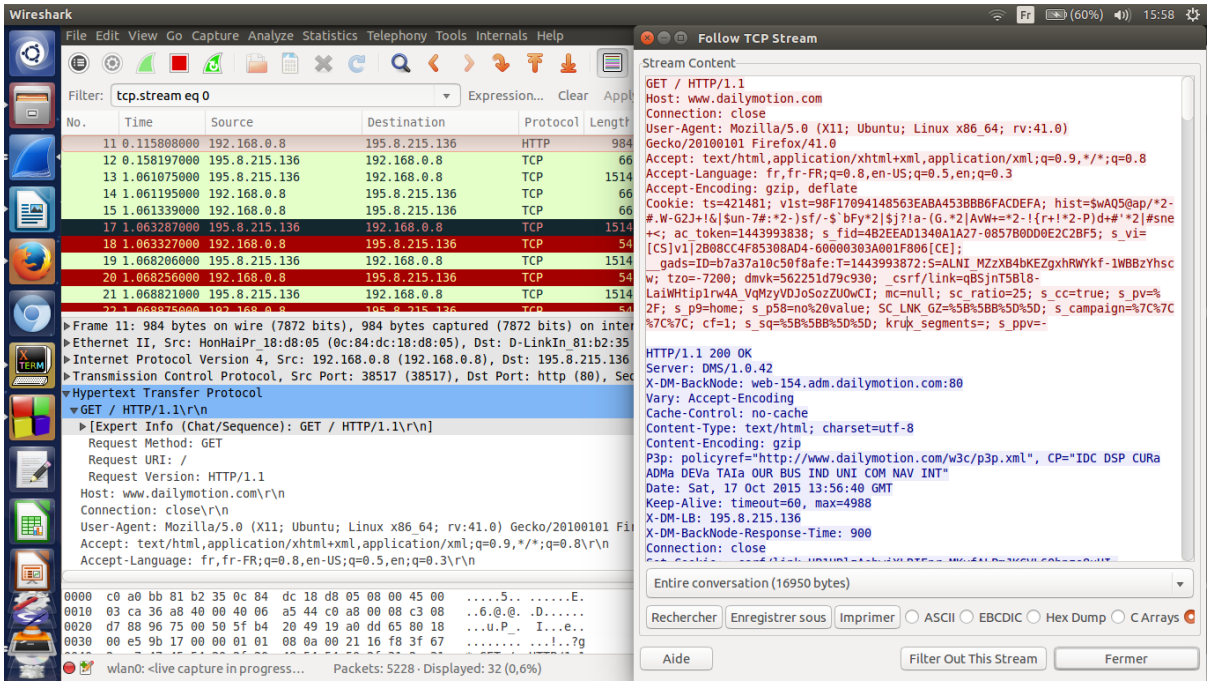


3. Dailymotion

Finally, we have tried one streaming website which does not use HTTPS Protocol: Dailymotion. First, we can see that we can normally go to the homepage. The whole page is well displayed. Then, we have tried to watch a video. Even if the download is quite long, we achieve to watch the video with a quite good quality. So our proxy handles also streaming files! We have also generated a trace of the packets with Wireshark to prove that our proxy was really used during this test. We can see that the request sent to the server is the one that the proxy modifies.




XTerm
Préférences
Backflips and Big Ga...
www.dailymotion.com/video/x39w5e1_backflips-and-big-gaps-at-practice-claw-cam-gopr
Rechercher
dailymotion
Video, channel...
Upload
Sign Up / Sign In
لڑکیو ں کو اپنے د وست ک ے ساتھ پارک میں سیکس
by Bikini Vines
302,711 views
Dhaka Road Accident Thug Life
by Entertainment
1,014 views
Proxy
WORLD CUP 2015
Daily highlights from the heart of the action
WATCH NO
Connection to 195.8.215.142 on port 80


Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 0
Expression... Clear
No. Time Source Destination Protocol Length
11 0.115808000 192.168.0.8 195.8.215.136 HTTP 984
12 0.158197000 195.8.215.136 192.168.0.8 TCP 66
13 1.061075000 195.8.215.136 192.168.0.8 TCP 1514
14 1.061195000 192.168.0.8 195.8.215.136 TCP 66
15 1.061339000 192.168.0.8 195.8.215.136 TCP 66
17 1.063287000 195.8.215.136 192.168.0.8 TCP 1514
18 1.063327000 192.168.0.8 195.8.215.136 TCP 54
19 1.068206000 195.8.215.136 192.168.0.8 TCP 1514
20 1.068256000 192.168.0.8 195.8.215.136 TCP 54
21 1.068821000 195.8.215.136 192.168.0.8 TCP 1514
Frame 11: 984 bytes on wire (7872 bits), 984 bytes captured (7872 bits) on inter
Ethernet II, Src: HonHaiPr_18:d8:05 (0c:84:dc:18:d8:05), Dst: D-LinkIn_81:b2:35
Internet Protocol Version 4, Src: 192.168.0.8 (192.168.0.8), Dst: 195.8.215.136
Transmission Control Protocol, Src Port: 38517 (38517), Dst Port: http (80), Sec
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
Request Method: GET
Request URI: /
Request Version: HTTP/1.1
Host: www.dailymotion.com\r\n
Connection: close\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0) Gecko/20100101 Fi
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3\r\n
0000 c0 a0 bb 81 b2 35 0c 84 dc 18 d8 05 08 00 45 00 .....5.. ......E.
0010 03 ca 36 a8 40 00 40 06 a5 44 c0 a8 00 08 c3 08 ..6.@.@. .D......
0020 d7 88 96 75 00 50 5f b4 20 49 19 a0 dd 65 80 18 ...u.P_. I...e..
0030 00 e5 9b 17 00 00 01 01 08 0a 00 21 16 f8 3f 67 ....!..?g
wlan0: <live capture in progress... Packets: 5228 · Displayed: 32 (0,6%)
Follow TCP Stream
Stream Content
GET / HTTP/1.1
Host: www.dailymotion.com
Connection: close
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0)
Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ts=421481; v1st=98F17094148563EABA453BBB6FACDEFA; hist=$wAQ5@ap/*2-
#.W-G2J+!&|$un-7#:*2-)sf/-$`bFy*2|$j?!a-(G.*2|AvW+=*2-!{r+!*2-P)d+#'*2|#sne
+<; ac_token=1443993838; s_fid=4B2EEAD1340A1A27-0857B0DD0E2C2BF5; s_vi=
[CS]v1|2B08CC4F85308AD4-60000303A001F806[CE];
__gads=ID=b7a37a10c50f8afe:T=1443993872:S=ALNI_MZzXB4bKEZgxhRWYkf-1WBBzYhsc
w; tzo=-7200; dmvk=562251d79c930; _csrf/link=qBSjnT5Bl8-
LaiWHtip1rw4A_VqMzyVDJoSozZUOwCI; mc=null; sc_ratio=25; s_cc=true; s_pv=%
2F; s_p9=home; s_p58=no%20value; SC_LNK_GZ=%5B%5BB%5D%5D; s_campaign=%7C%7C
%7C%7C; cf=1; s_sq=%5B%5BB%5D%5D; kruk_segments=; s_ppv=-
HTTP/1.1 200 OK
Server: DMS/1.0.42
X-DM-BackNode: web-154.adm.dailymotion.com:80
Vary: Accept-Encoding
Cache-Control: no-cache
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
P3p: policyref="http://www.dailymotion.com/w3c/p3p.xml", CP="IDC DSP CURa
ADMa DEVa TAIa OUR BUS IND UNI COM NAV INT"
Date: Sat, 17 Oct 2015 13:56:40 GMT
Keep-Alive: timeout=60, max=4988
X-DM-LB: 195.8.215.136
X-DM-BackNode-Response-Time: 900
Connection: close
Entire conversation (16950 bytes)
Rechercher
Enregistrer sous
Imprimer
ASCII
EBCDIC
Hex Dump
C Arrays
Aide
Filter Out This Stream
Fermer

Capabilities of the Proxy:

HTTP and HTTPS

The proxy can handle every website that use only HTTP requests (for instance: Dailymotion, Stackoverflow…), but it cannot handle every website which uses HTTPS Requests.
In fact, if the connection to the website server is done directly via a HTTPS connection (for instance : Google, YouTube, Vimeo, Facebook, Twitter…), the website is not accessible because the proxy does not relay the HTTPS requests and returns to the client a 405 status code ("Method not allowed"). But, if the website just display HTTPs protected content, the site is accessible except this resource (for instance, YouTube videos on the LiU website or on BBC website).
To conclude, the proxy cannot handle a lot of popular websites such as Social Networks or some streaming-websites which use HTTPS, but can handle every other type of websites.

Encoding

Encoding has no effect on the proxy.
In fact, the proxy sends packet by packet the response's server to the client: so it doesn't concatenate the server's response before sending it to the client. This is this concatenation of encoding content that can be problematic and disturb a proxy, but the problem is avoided here.

Filtering

The proxy can filter the requested page's content of page's URL: if it contains a bad word the website is blocked. The filtering works on every websites which use HTTP Requests (for instance: on Dailymotion or Stackoverflow the content is blocked when we research a bad word on the website).
The filtering cannot work on the HTTPs Websites (for instance: Wikipedia, Google or YouTube), because the filtering is done in the client_part function of the proxy: but the proxy transmits the browser's request to the client_part only if it is an HTTP Request. So instead of filtering the content of the URL, a "Method Not Allowed" error page is displayed by the browser.