Instituto Superior de Engenharia de Lisboa

Computer Networks

Phase 1 - Web Server

Projeto ISEL 2023/24 — LEETC

Coordination

General: Carlos Meneses Course: Nuno Cruz

Grupo LP-07

Supervisor: Luís Pires

Student

Nuno Brito <A46948@alunos.isel.pt>

April 14th 2024

Contents

List of Figures

List of Tables

Listings

Acronyms list

API	Application Programming Interface
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
OS	Operating System
OSS	openSUSE
PHP	PHP: Hypertext Preprocessor
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TUI	Terminal User Interface
UDP	User Datagram Protocol
VPN	Virtual Private Network
WWW	World Wide Web
XAMPP	Cross-Platform, Apache, MySQL, PHP, and Perl

Glossary

Apache2
An opensource HTTP web server.

Browser
A browser is a internet navigation software. It comes in multiple flavours, nowadays the big three are Microsoft Edge, Mozilla Firefox and Google Chrome.

Firewall
A barrier between networks. Controls inbound and outbound traffic.

LibreWolf
An internet browser based on Mozilla's Firefox. It's primary purpose is to allow privacy, and with it comes security. It achieves this by removing telemetry and data collection.

MariaDB
A community-developed fork of MySQL database server.

openSUSE Tumbleweed
An openSUSE (OSS) is an open-source community driven Linux-based distribuition sponsored by SUSE Software Solutions. Tumbleweed is a rolling release version allowing for up-to-date software releases.

Operating system
A program that manages a computer's resources from software to hardware.

Python
Python is a high-level programming language, object-oriented.

Perl
A high-level, general-purpose, interpreted, dynamic programming language

Rolling release distribuition
A distribuition where it's software release cycle is more frequent than those of Long Term Support (LTS). It's up to the Linux-based distribuitor to guarantee the testing of a package.

Socket
A network socket serves as an endpoint for sending and receiving data across the network.

VPN
A private network creating a secure connection between a device and a network.

Windows
Microsoft's operating system. First released in 1985 as a Graphical User Interface (GUI) for MS-DOS, continued to evolve with it's latest version being 11. Due to it's nature, it's not recommended for server production environment.

Wireshark

Wireshark is a network protocol analyser software. Allows traffic capture between a computer and a network.

XAMPP

A software package environment collection containing Apache2 webserver, MariaDB database, PHP and Perl.

Chapter 1

Introduction

The project consists in building a computer network through four phases. First with a webserver, then simulating a local area network (LAN) with two computers and a switch. By the end of the journey, this project will develop into something similar to a corporate network.

Chapter 2

Phase 1

2.1 Milestones

- Setup apache2 web server in localhost
- Access web server locally (http://127.0.0.1/ or http://localhost)
- Access web server from a remote computer (http://172.24.1.12)
- Use wireshark in a remote host to capture packages from the server
- Compare the HTTP headers sent by the client and the server
- Develop a simple barebones HTTP webclient
- Establish a TCP connection to the server
- Request the base webpage

2.2 WebClient requirements

- HTTP library forbidden
- Establish TCP connection using available sockets library - send/receive the HTTP request/reply
- Output HTTP reply to the user
- - Optional - act to the various HTTP replies
- Text-only application

2.3 Software

- Local server side

  Operating system: Windows 11 x64

  WebServer: XAMPP x64 8.2.12-0-VS16 for windows

- Client side

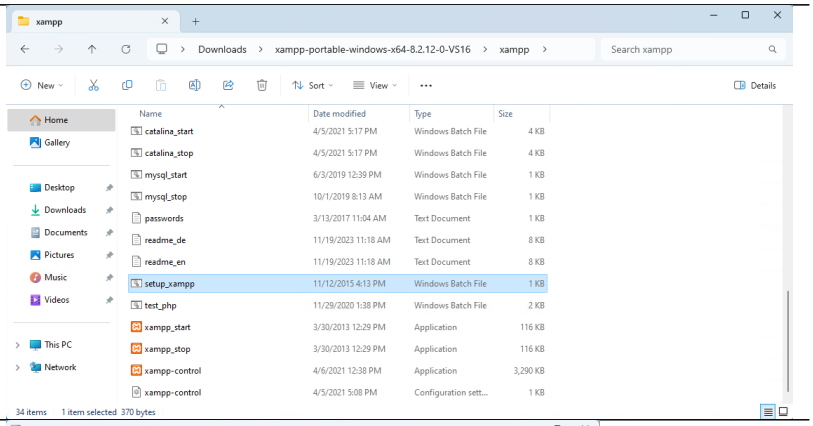
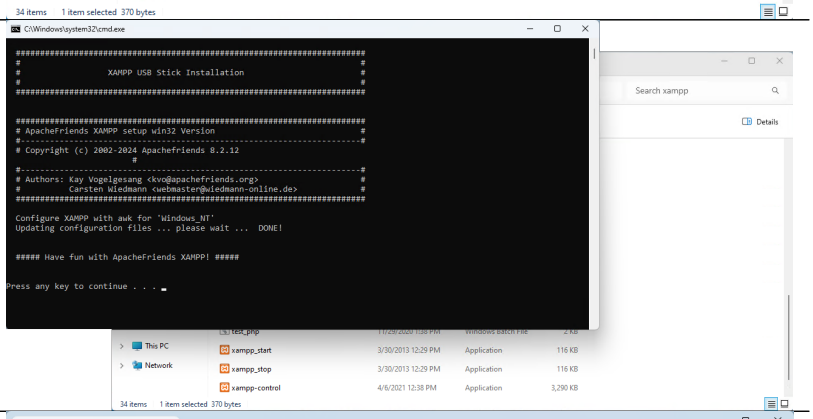
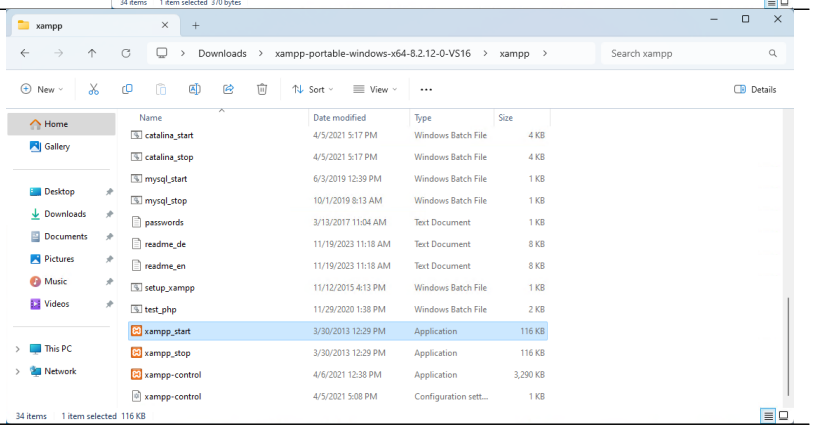
  Operating system: openSUSE Tumbleweed

  Browser: LibreWolf version 123.0-1

  Package monitor: Wireshark version 4.2.3 (Git commit b0da86c196d1).

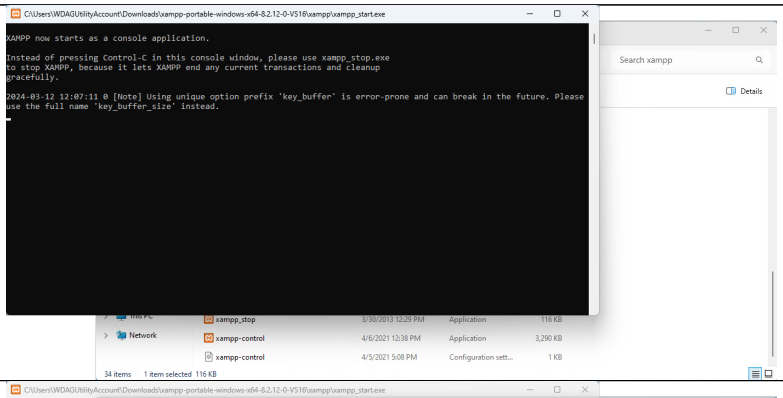
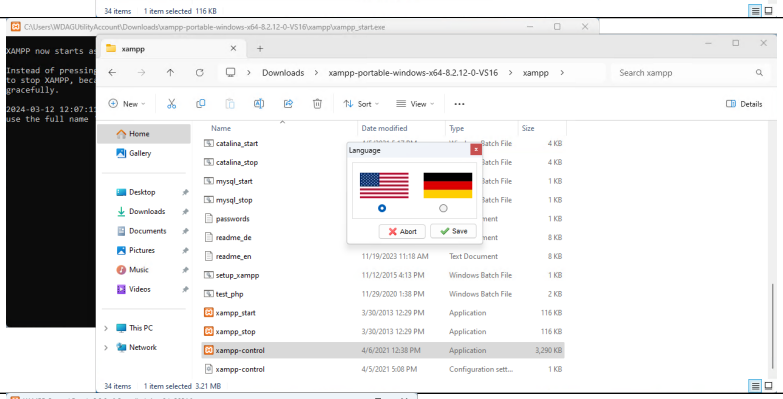
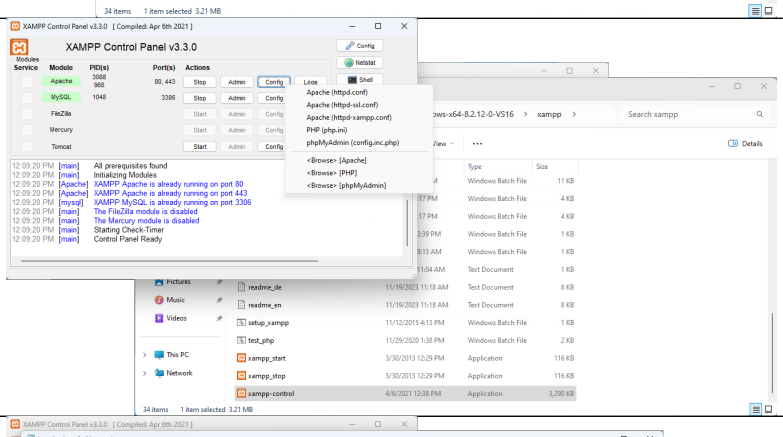
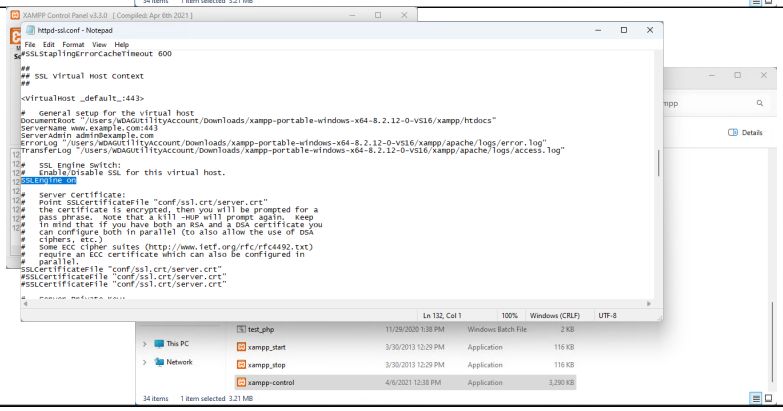
2.4 Software install

First we install, start and configure XAMPP. Using the following link, https://www.apachefriends.org/download.html, we can choose our prefered method, for this project the portable version was the best choice since no installation was needed. After uncompressing our downloaded file, we can start the process.

Steps	Example
The read me file available in the root directory states that we need to run *setup_xampp* batch file first to populate the registry it's directory.	
After completing, just press any button to continue.	
Next click in the *xampp_start* executable, windows will prompt some firewall permissions which will gladly accept.	

Continued on next page

Table 2.1 – continued

Steps	Example
	
Pick a language for the program.	
Apache2 comes with SSL on by default. Since we're studying the HTTP protocol it's convenient to disable HTTPS. Pressing the *config* button shows a list of configuration files. Select *Apache (httpd-ssl.conf)*.	
Search and comment the line with *SSLEngine on*.	

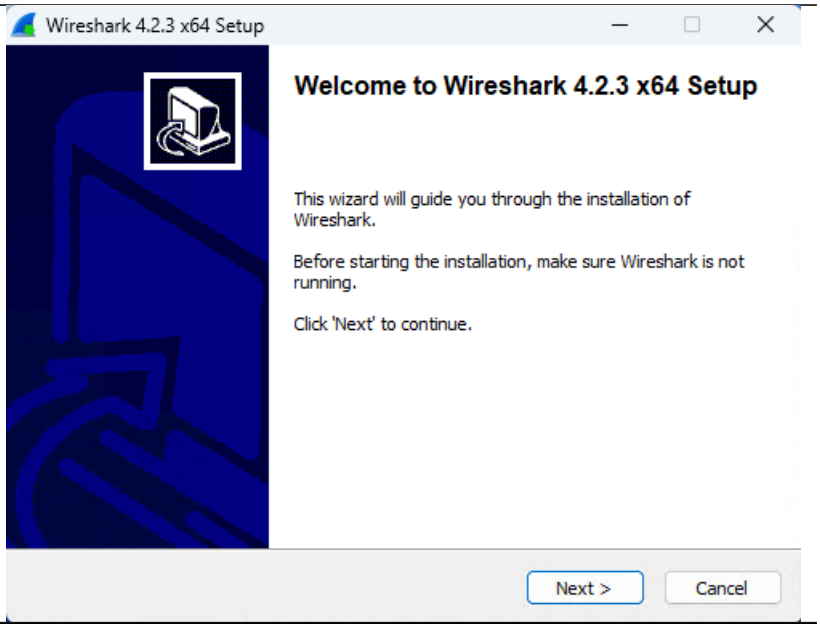
Continued on next page

Table 2.1 – continued

Steps	Example
Previously if we went to *http://localhost* it would redirect to the HTTPS version. After completing the above step it'll no longer redirect, showing us the non-secure version.	

Table 2.1: XAMPP install

Next up is wireshark, the powerful network analyser. We'll download the installer from https://www.wireshark.org/download.html.

Steps	Example
Wireshark setup. Select everything available and continue by clicking *next*.	

Continued on next page

Table 2.2 – continued

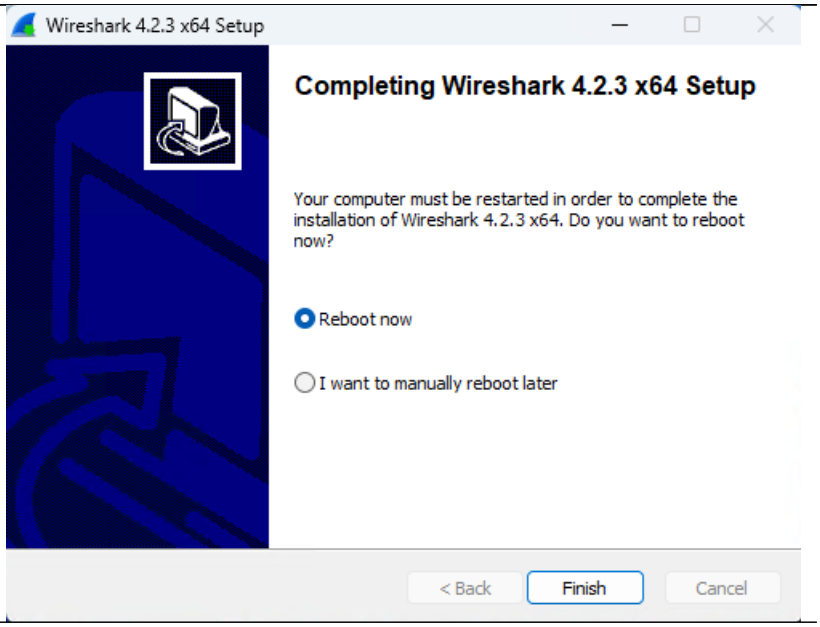
Steps	Example
After completing the installation, reboot the computer.	

Table 2.2: Wireshark install

2.5 WebClient - Python Code

```
"""
Name: Python TCP Client
Description: Simple TCP client using sockets
Original code by: Luis Pires
Source: Chapter 2, slide 104

Commented and adapted by: Nuno Brito
"""

# Import from everything from the socket library
from socket import *

# Specify servername and port destination
serverName = "172.24.1.12"
serverPort = 80

# GET list
httpTestMessages = [
        "GET /dashboard/ HTTP/1.1\r\n",             # 200 OK
        "GET /dashboard HTTP/1.1\r\n",              # 301 Moved Permanently
        "PUT / HTTP/1.1\r\n",                       # 302 Found
        "GET /dashboard HTTP/1.\r\n",               # 400 Bad Request
        "GET /dashboard/index.htm HTTP/1.1\r\n",    # 404 Not Found
        "PUT /d HTTP/1.1\r\n",                      # 405 Method Not Allowed
        "BREW /coffee/ HTTP/1.1\r\n",               # 501 Not implemented
        ]

# Cycle through predefined messages
```

```
for sentence in httpTestMessages:

    # Socket open and connect
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect((serverName,serverPort))

    # Join serverName to the current sentence
    sentence += "Host:" + serverName + "\r\n\r\n"

    # Socket encode message and send
    clientSocket.send(sentence.encode())

    # Receive and out the response message
    modifiedSentence = clientSocket.recv(1024)

    # Close socket connection
    clientSocket.close()

    # Print the requested message
    print ("-"*60)
    print ("From Server:", modifiedSentence.decode())
```

Listing 2.1: Simple HTTP WebClient using sockets in python

The code listed in 2.1 was adapted to be simple and cycle through the various request messages without any user input.
Modifiable variables include *serverName*, *serverPort* and the *httpTestMessages* list.

The webclient produces the following output:

```
------------------------------------------------------------
From Server: HTTP/1.1 200 OK
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Last-Modified: Sun, 19 Nov 2023 11:18:30 GMT
ETag: "1443-60a7f87754d80"
Accept-Ranges: bytes
Content-Length: 5187
Content-Type: text/html

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <!-- Always force latest IE rendering engine or request Chrome Frame -->
    <meta content="IE=edge,chrome=1" http-equiv="X-UA-Compatible">
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- Use title if it's in the page YAML frontmatter -->
    <title>Welcome to XAMPP</title>

    <meta name="description" content="XAMPP is an easy to install Apache
    distribution containing MariaDB, PHP and Perl." />
    <meta name="keywords" content="xampp, apache, php, perl, mariadb, open source
    distribution" />

    <link href="/dashboard/stylesheets/normalize.css" rel="stylesheet" type="text/
    css" /><link href="/dashboard/stylesheets/all.css" rel="stylesheet" type="t
------------------------------------------------------------
From Server: HTTP/1.1 301 Moved Permanently
```

```
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Location: http://172.24.1.12/dashboard/
Content-Length: 338
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://172.24.1.12/dashboard/">here</a>.</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at 172.24.1.12 Port
    80</address>
</body></html>
-----------------------------------------------------------
From Server: HTTP/1.1 302 Found
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
X-Powered-By: PHP/8.2.12
Location: http://172.24.1.12/dashboard/
Content-Length: 0
Content-Type: text/html; charset=UTF-8
-----------------------------------------------------------
From Server: HTTP/1.1 400 Bad Request
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Content-Length: 325
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at localhost Port
    80</address>
</body></html>
-----------------------------------------------------------
From Server: HTTP/1.1 404 Not Found
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Content-Length: 297
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at 172.24.1.12 Port
    80</address>
</body></html>
-----------------------------------------------------------
From Server: HTTP/1.1 405 Method Not Allowed
Date: Fri, 29 Mar 2024 18:42:08 GMT
```

```
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Allow: HEAD,GET,POST,OPTIONS,TRACE
Content-Length: 321
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method PUT is not allowed for this URL.</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at 172.24.1.12 Port
    80</address>
</body></html>
-------------------------------------------------------------
From Server: HTTP/1.1 501 Not Implemented
Date: Fri, 29 Mar 2024 18:42:08 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Allow: HEAD,GET,POST,OPTIONS,TRACE
Content-Length: 304
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Not Implemented</title>
</head><body>
<h1>Not Implemented</h1>
<p>BREW not supported for current URL.<br />
</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at 172.24.1.12 Port
    80</address>
</body></html>
```

Listing 2.2: WebClient output

2.6 Wireshark captures

First we must ensure the browser being used can connect to the XAMPP server with HTTP. We can do that by enabling the usage of deprecated TLS.

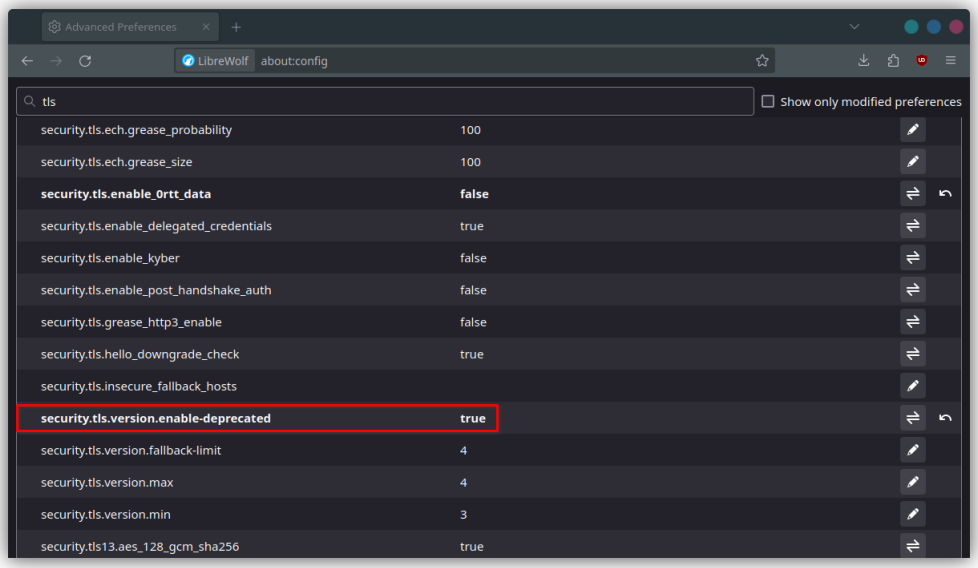


Figure 2.1: Changing from false to true the *security.tls.version.enable-deprecated* option

Then we can start our capture process, next follows some printscreen examples filtered by HTTP.

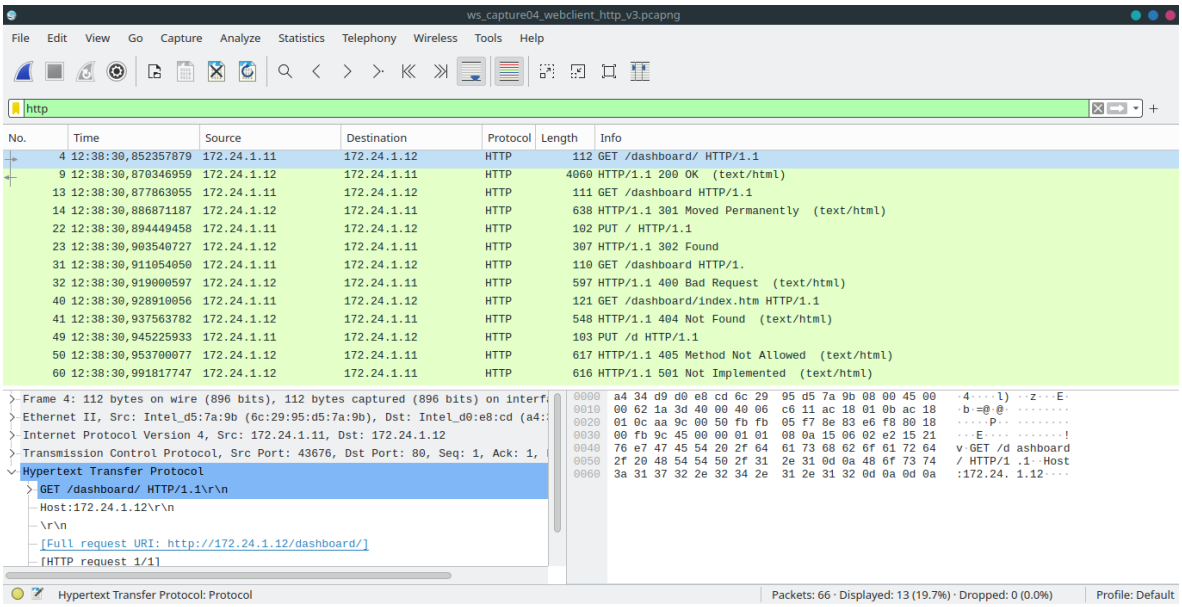
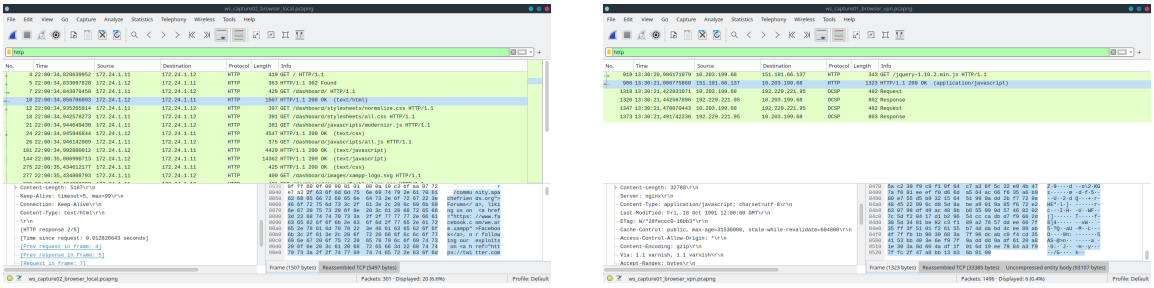


Figure 2.2: Webclient get capture



(a) Browser (b) Browser (VPN)

Figure 2.3: Browser capture

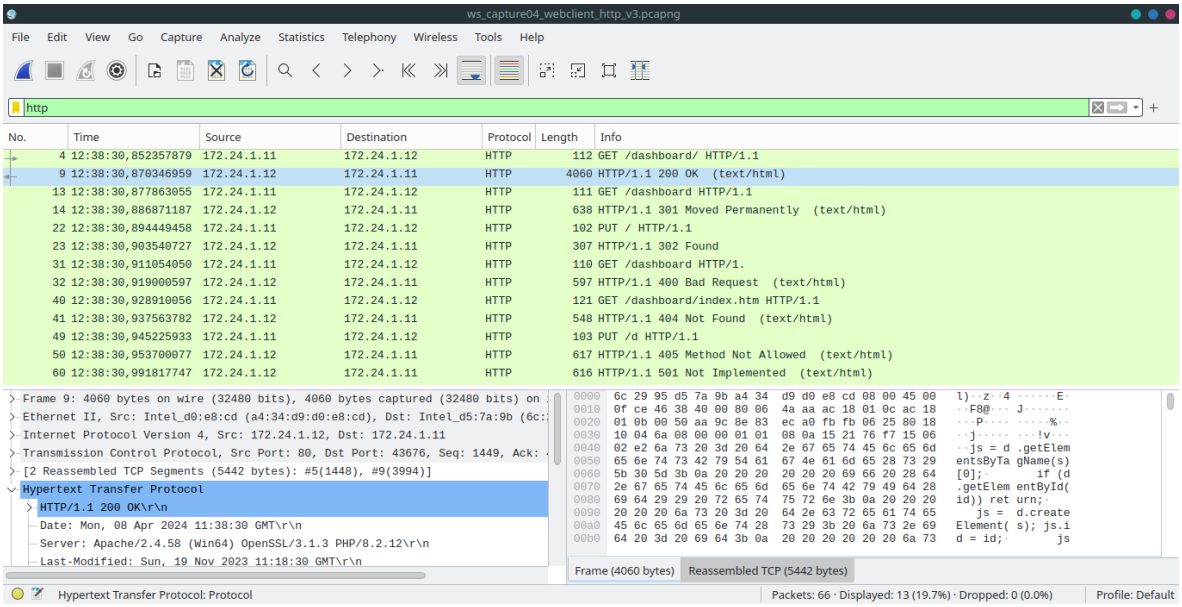


Figure 2.4: Webclient reply capture

To compliment the images, below are segmented outputs (the important parts):

```
No.     Time               Source                Destination           Protocol     Length Info
    919 13:30:20,986171079 10.203.199.68         151.101.66.137        HTTP      343    GET /jquery-1.10.2.min.js HTTP/1.1
Hypertext Transfer Protocol
    GET /jquery-1.10.2.min.js HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /jquery-1.10.2.min.js HTTP/1.1\r\n]
            [GET /jquery-1.10.2.min.js HTTP/1.1\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Request Method: GET
        Request URI: /jquery-1.10.2.min.js
        Request Version: HTTP/1.1
    Host: code.jquery.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Referer: http://172.24.1.12/\r\n
    \r\n
    [Full request URI: http://code.jquery.com/jquery-1.10.2.min.js]
    [HTTP request 1/1]
    [Response in frame: 986]
No.     Time               Source                Destination           Protocol     Length Info
    986 13:30:21,006775860 151.101.66.137        10.203.199.68         HTTP      1323   HTTP/1.1 200 OK  (application/javascript)
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
            [HTTP/1.1 200 OK\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Response Version: HTTP/1.1
        Status Code: 200
        [Status Code Description: OK]
        Response Phrase: OK
    Connection: keep-alive\r\n
    Content-Length: 32788\r\n
        [Content length: 32788]
    Server: nginx\r\n
    Content-Type: application/javascript; charset=utf-8\r\n
    Last-Modified: Fri, 18 Oct 1991 12:00:00 GMT\r\n
    ETag: W/"28feccc0-16bb3"\r\n
    Cache-Control: public, max-age=31536000, stale-while-revalidate=604800\r\n
    Access-Control-Allow-Origin: *\r\n
    Content-Encoding: gzip\r\n
    Via: 1.1 varnish, 1.1 varnish\r\n
    Accept-Ranges: bytes\r\n
    Date: Mon, 11 Mar 2024 13:30:20 GMT\r\n
    Age: 15363215\r\n
    X-Served-By: cache-lga13622-LGA, cache-lis1490024-LIS\r\n
    X-Cache: HIT, HIT\r\n
    X-Cache-Hits: 12, 7599\r\n
    X-Timer: S1710163821.991495,VS0,VE0\r\n
    Vary: Accept-Encoding\r\n
    \r\n
    [HTTP response 1/1]
```

```
    [Time since request: 0.020604781 seconds]
    [Request in frame: 919]
    [Request URI: http://code.jquery.com/jquery-1.10.2.min.js]
    Content-encoded entity body (gzip): 32788 bytes -> 93107 bytes
    File Data: 93107 bytes
Media Type
    Media type: application/javascript; charset=utf-8 (93107 bytes)
```

Listing 2.3: Wireshark capture output sample - VPN

```
No.     Time               Source                Destination           Protocol
    Length Info
      4 22:00:34,820639952 172.24.1.11           172.24.1.12           HTTP     419
       GET / HTTP/1.1
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
            [GET / HTTP/1.1\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Request Method: GET
        Request URI: /
        Request Version: HTTP/1.1
    Host: 172.24.1.12\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:123.0) Gecko/20100101 Firefox
    /123.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
    webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    DNT: 1\r\n
    Sec-GPC: 1\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://172.24.1.12/]
    [HTTP request 1/5]
    [Response in frame: 5]
    [Next request in frame: 7]
No.     Time               Source                Destination           Protocol
    Length Info
      5 22:00:34,833097828 172.24.1.12           172.24.1.11           HTTP     363
       HTTP/1.1 302 Found
Hypertext Transfer Protocol
    HTTP/1.1 302 Found\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 302 Found\r\n]
            [HTTP/1.1 302 Found\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Response Version: HTTP/1.1
        Status Code: 302
        [Status Code Description: Found]
        Response Phrase: Found
    Date: Tue, 12 Mar 2024 22:00:35 GMT\r\n
    Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12\r\n
    X-Powered-By: PHP/8.2.12\r\n
    Location: http://172.24.1.12/dashboard/\r\n
    Content-Length: 0\r\n
        [Content length: 0]
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
```

```
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/5]
    [Time since request: 0.012457876 seconds]
    [Request in frame: 4]
    [Next request in frame: 7]
    [Next response in frame: 10]
    [Request URI: http://172.24.1.12/]
```

Listing 2.4: Wireshark capture output sample - Browser

```
No.     Time               Source                Destination           Protocol
    Length Info
      4 12:38:30,852357879 172.24.1.11           172.24.1.12           HTTP     112
       GET /dashboard/ HTTP/1.1
Hypertext Transfer Protocol
No.     Time               Source                Destination           Protocol
    Length Info
      9 12:38:30,870346959 172.24.1.12           172.24.1.11           HTTP     4060
       HTTP/1.1 200 OK  (text/html)
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Mon, 08 Apr 2024 11:38:30 GMT\r\n
    Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12\r\n
    Last-Modified: Sun, 19 Nov 2023 11:18:30 GMT\r\n
    ETag: "1443-60a7f87754d80"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 5187\r\n
    Content-Type: text/html\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.017989080 seconds]
    [Request in frame: 4]
    [Request URI: http://172.24.1.12/dashboard/]
    File Data: 5187 bytes
Line-based text data: text/html (130 lines)
```

Listing 2.5: Wireshark capture output sample - WebClient

2.7 List of headers and replies

Request: GET /dashboard/ HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 200 OK
Meaning: this header complies with what the server expects from a webclient request.

Request: GET /dashboard HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 301 Moved Permanently
Meaning: this header request a relative directory without a forward slash at the end, prompting the server to reply with a "moved" answer.

Request: PUT / HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 302 Found
Meaning: this header request is an upload request to an unexistent directory.

Request: GET /dashboard HTTP/1.\r\nHost:127.24.1.12 \r\n\r\n

Reply: HTTP/1.1 400 Bad Request
Meaning: this header request, although it has an invalid directory, has the HTTP protocol version badly writen (HTTP/1.1 vs. actual HTTP/1.) which causes a "bad request" reply from the server.

Request: GET /dashboard/index.htm HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 404 Not Found
Meaning: this header request tries to get a file that doesn't exist in the local server.

Request: PUT /d HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 405 Method Not Allowed
Meaning: this header request tries to upload something to the relative directory "d".

Request: BREW /coffee/ HTTP/1.1\r\nHost:127.24.1.12 \r\n\r\n
Reply: HTTP/1.1 501 Not Implemented
Meaning: a poorly attempt to get the 1998 April fool's day. It should have replied with 418 I'm a teapot. Even with GET instead of BREW it didn't work. Apache doesn't have the implementation.

Chapter 3

Phase 2

Commands: ping, tracert, ipconfig

Part 1 Group 7 -> N = 7 -> 192.168.7.0/24 Laptop0 -> 192.168.7.1 PC0 -> 192.168.7.2

Question: How can a PC know if it is connected to a switch? Is traceroute useful in this situation? Answer:

Part 2

```
                External
                    ^
                    |
                    |
                ---------
                |       |
      --------->|Router0|<---------
      |         ---------          |
      |LAN                      LAN|
      |Transit              Transit|
      |C                         B|
      |                           |
      v                           v
  ---------   LAN Transit A   ---------
  |Router1|<----------------->|Router2|
  ---------                   ---------
      |                           |
  ---------                       |
  |       |                       |
  v       v                       v
LAN A   LAN B                 LAN Server
```

Listing 3.1: Network plan

The above table will be used for the next phases. Instead of planning for each phase and re-assigning the entire network, the network was fully detailed to accomodate all phases.

However, here we'll focus on router R1 and LAN A. For now let's just focus on the IP addresses, the values will be explained in Phase 3.

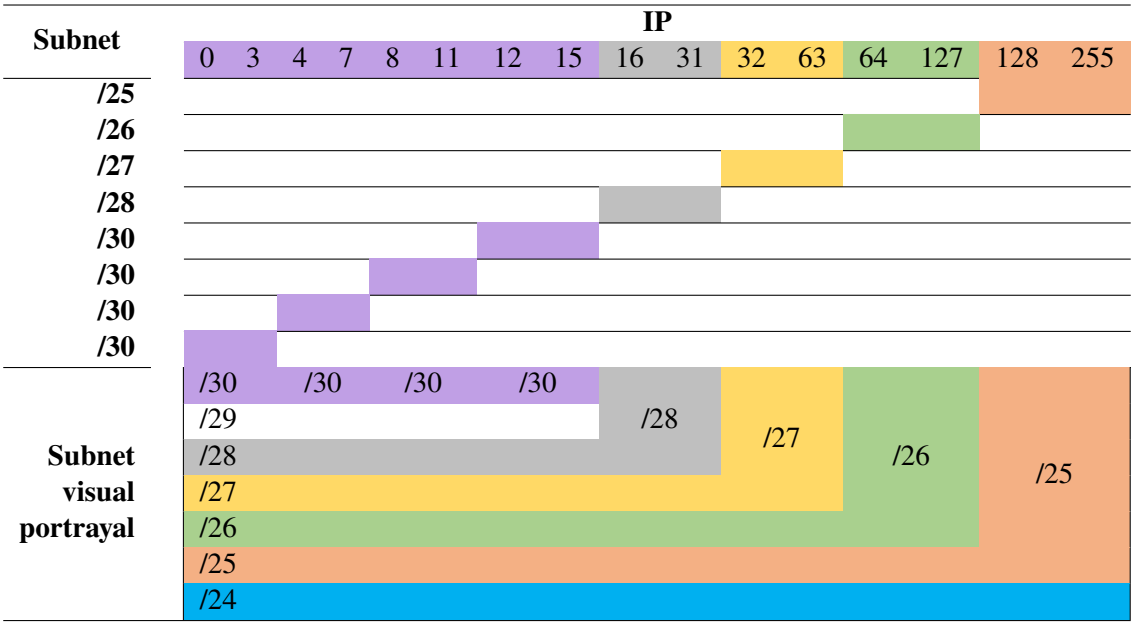


Table 3.1: Visual LAN allocation

Name	Network	Usable IPs	Router	Broadcast	Subnet Mask	Populated
	192.168.7.					
LAN Server	128	129 - 253	254	255	128	126
LAN A	64	65 - 125	126	127	192	48
LAN B	32	33 - 61	62	63	224	27
Unused remaining	16	17 - 31		32		0
	12	13 - 14		15		0
LAN Transit C	8	9 - 10		11	252	2
LAN Transit B	4	5 - 6		7	252	2
LAN Transit A	0	1 - 2		3	252	2

Table 3.2: LAN allocation table

Name	Ports Link	Network	IP	Subnet Mask	Gateway
PC0	Fa0 ->Sw0 Fa0/2	LAN A	192.168.7.65	255.255.255.192	192.168.7.126
Laptop0	Fa0 ->Sw0 Fa0/3		192.168.7.66	255.255.255.192	192.168.7.126
PC1	Fa0 ->Sw1 Fa0/2	LAN B	192.168.7.33	255.255.255.224	192.168.7.62
Laptop1	Fa0 ->Sw1 Fa0/3		192.168.7.34	255.255.255.224	192.168.7.62
R0	Fa5/0 ->R1 Fa5/0	LAN Transit B	192.168.7.5	255.255.255.252	
	Fa4/0 ->R2 Fa4/0	LAN Transit C	192.168.7.9	255.255.255.252	
	Fa0/0	External			
R1	Fa4/0 ->R2 Fa5/0	LAN Transit A	192.168.7.1	255.255.255.252	
	Fa5/0 ->R1 Fa4/0	LAN Transit B	192.168.7.6	255.255.255.252	
	Fa0/0 ->Sw0 Fa0/1	LAN A	192.168.7.126	255.255.255.192	
	Fa1/0 ->Sw1 Fa0/1	LAN B	192.168.7.62	255.255.255.224	
R2	Fa5/0 ->R1 Fa4/0	LAN Transit A	192.168.7.2	255.255.255.252	
	Fa4/0 ->R0 Fa4/0	LAN Transit C	192.168.7.10	255.255.255.252	
	Fa0/0 ->Sw2 Fa0/4	LAN Server	192.168.7.254	255.255.255.128	
DHCP Server	Fa0 ->Sw2 Fa0/3	LAN Server	192.168.7.129	255.255.255.128	192.168.7.254
DNS Server	Fa0 ->Sw2 Fa0/2		192.168.7.130	255.255.255.128	192.168.7.254
HTTP Server	Fa0 ->Sw2 Fa0/1		192.168.7.131	255.255.255.128	192.168.7.254
Sw0	Fa0/1 ->R1 Fa0/0	LAN A			
	Fa0/2 ->PC0				
	Fa0/3 ->Laptop0				
Sw1	Fa0/1 ->R1 Fa1/0	LAN B			
	Fa0/2 ->PC1				
	Fa0/3 ->Laptop1				
Sw2	Fa0/1 ->HTTP	LAN Server			
	Fa0/2 ->DNS				
	Fa0/3 ->DHCP				
	Fa0/4 ->R2 Fa0/0				

Table 3.3: IP configuration table

Chapter 4

Issues and fixes

Running python code:

Python3 wasn't installed by default. Then had to run the code with: $ python3 httpsocketv3.py.

Encrypted html body in wireshark:

Initially I had to run wireshark in a remote virtual private network (VPN) connection. Fortunately I could see the VPN doing it's magic but also couldn't see the HTTP body, since it was encrypted.

Default HTTP protocol, HTTPS:

To guarantee the HTTP connection I had to disable SSLEngine in Apache2 WebServer.

Chapter 5

Conclusions

During phase 1 many challenges were met. By creating (or in this case adapting) a webclient without using the http library, it allowed a better understanding of the protocol requests and replies by taking advantage of the provided protocol stack in a operating system. Employing the wireless packet monitor, wireshark, concepts related to http were better understood as all transactions between webclient and webserver were seen in real time, allowing a greater furthering of knowledge.

Appendix A

Appendix