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

Figure list ii

Table list iii

Acronyms list iv

Glossário v

1 Introduction 1

2 Phase 1 2

2.1 Milestones . . . 2
2.2 WebClient requirements . . . 2
2.3 Software . . . 2
2.4 Software install . . . 3
2.5 WebClient - Python Code . . . 6
2.6 Wireshark captures . . . 7
2.7 List of headers and replies . . . 9
2.8 Issues and fixes . . . 9

References 9

A Um Apêndice 10

List of Figures

2.1 Changing from false to true the *security.tls.version.enable-deprecated* option 7
2.2 Webclient get 8
2.3 Browser capture 8
2.4 Webclient reply 8

List of Tables

2.1 XAMPP install 5
2.2 Wireshark install 6

Acronyms list

API	Application Programming Interface
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
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

Glossário

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

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.

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.

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. Due to it's nature, it's not recommended for server production environment.

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.

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

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.

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

Apache2
Text

Socket
Text

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

Firewall
Text

VPN
: Text

SSL
: Text

Glossário
: Glossário é uma espécie de pequeno dicionário específico para palavras e expressões pouco conhecidas presentes num texto, seja por serem de natureza técnica, regional ou de outro idioma.

Chapter 1

Introduction

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 Disclaimer: the versions listed below might not be the latest, by present date, given the nature of rolling release distribuitions software updates.

- Client side

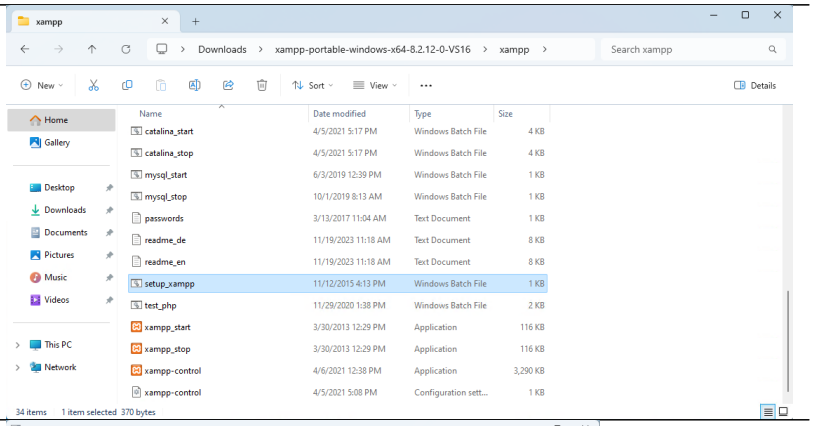
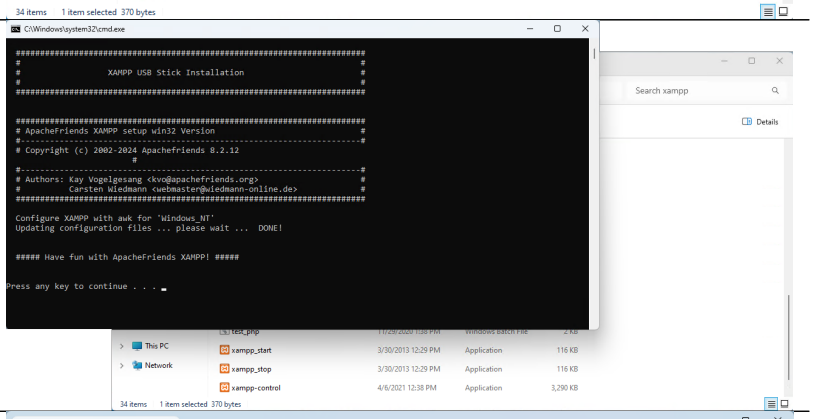
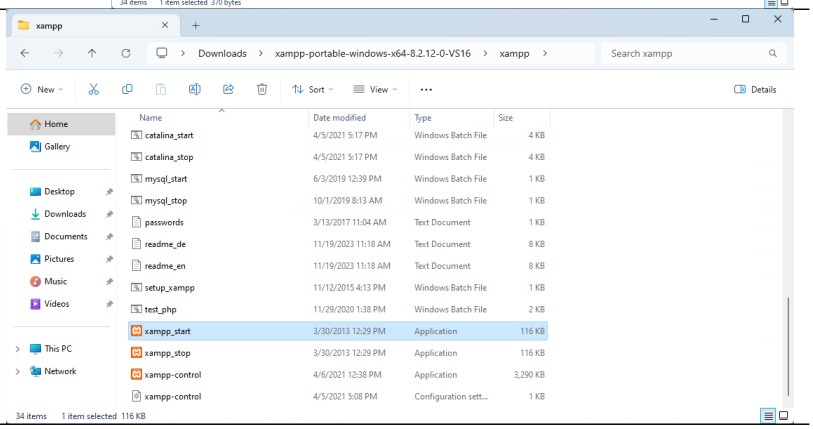
  Operating system: openSUSE Tumbleweed

  Browser: LibreWolf version 123.0-1

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

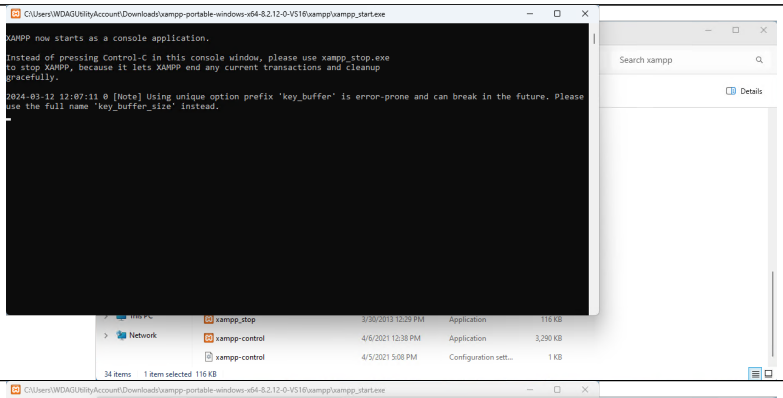
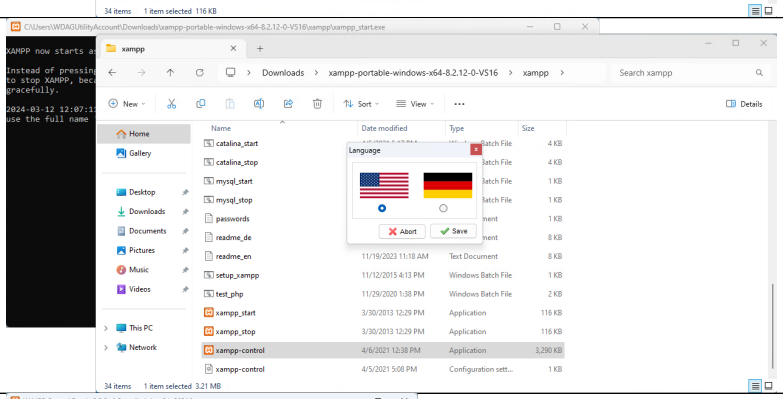
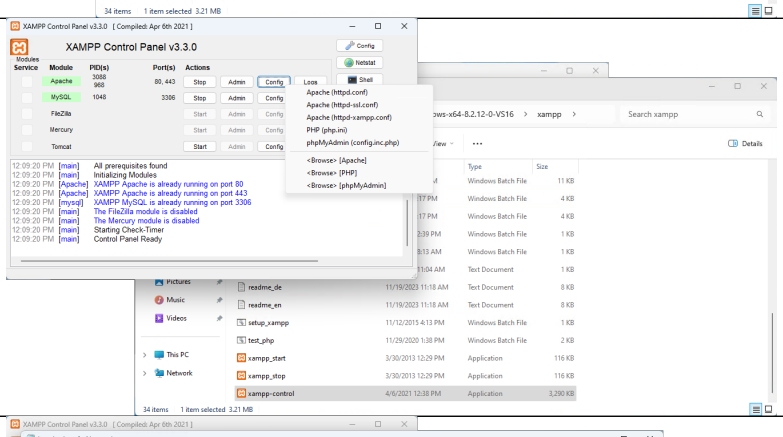
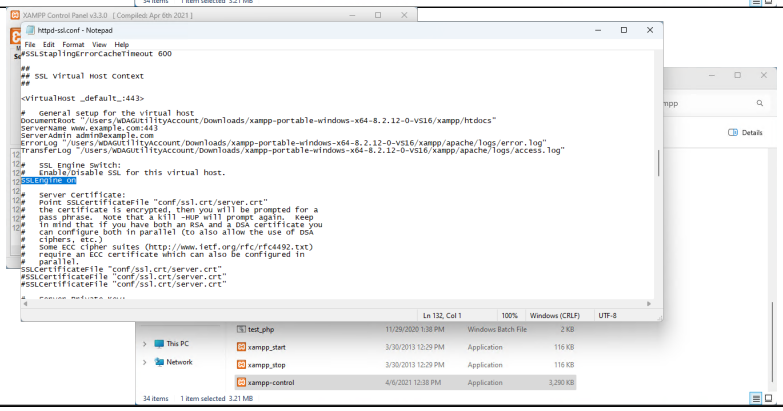
2.4 Software install

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*.	

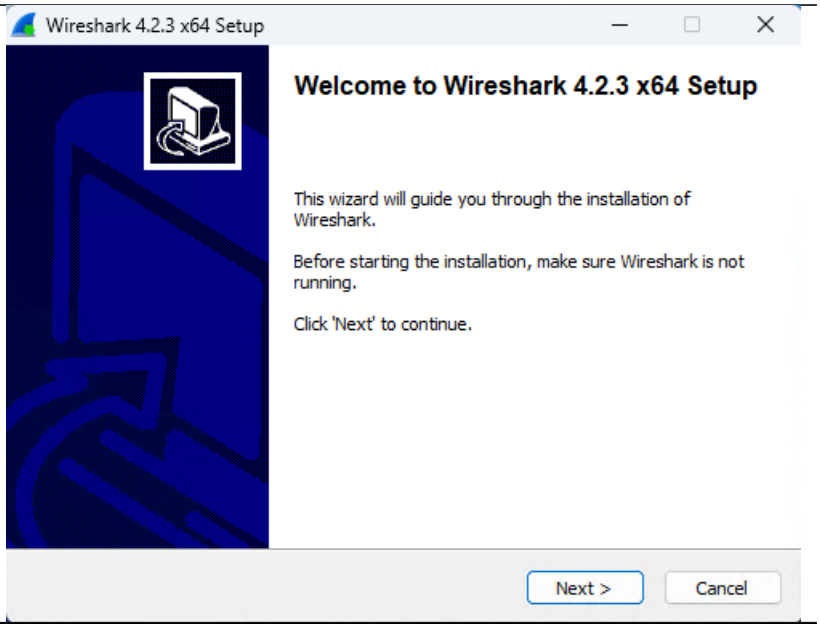
Continued on next page

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

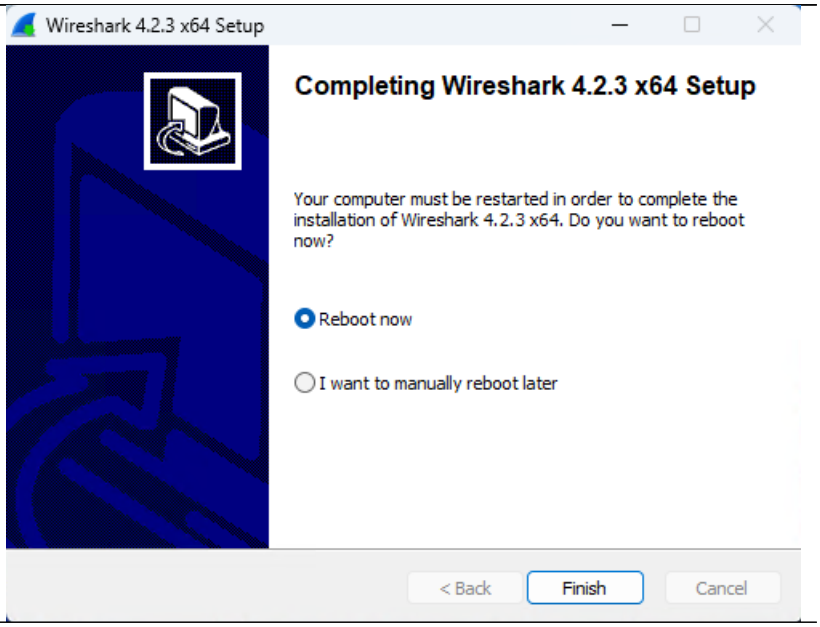
Steps	Example
After completing the installation, reboot the computer.	

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
```

```
29 for sentence in httpTestMessages:
30
31     # Socket open and connect
32     clientSocket = socket(AF_INET, SOCK_STREAM)
33     clientSocket.connect((serverName,serverPort))
34
35     # Join serverName to the current sentence
36     sentence += "Host:" + serverName + "\r\n\r\n"
37
38     # Socket encode message and send
39     clientSocket.send(sentence.encode())
40
41     # Receive and out the response message
42     modifiedSentence = clientSocket.recv(1024)
43
44     # Close socket connection
45     clientSocket.close()
46
47     # Print the requested message
48     print ("-"*60)
49     print ("From Server:", modifiedSentence.decode())
```

Listing 2.1: Simple HTTP WebClient using sockets in python

The code 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.

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.

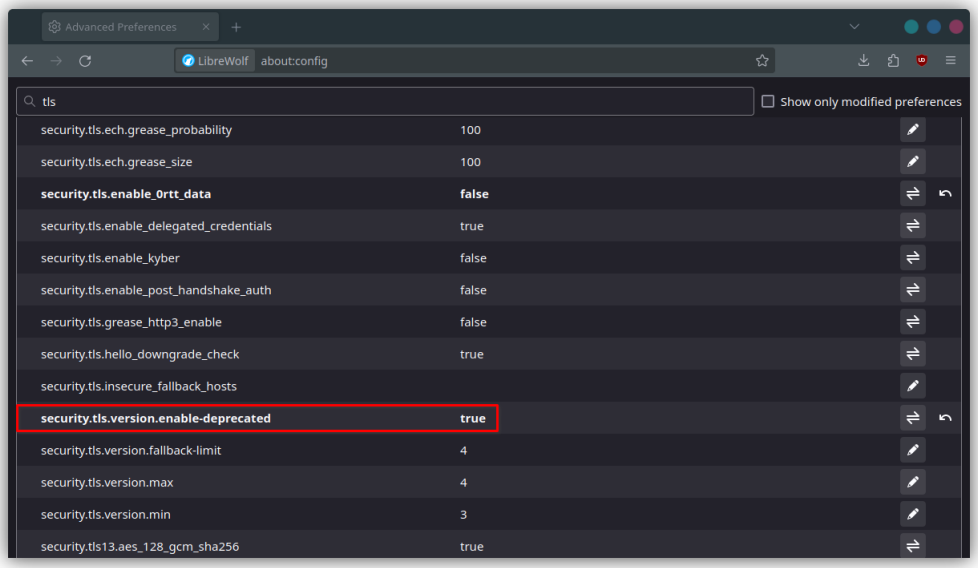


Figure 2.1: Changing from false to true the *security.tls.version.enable-deprecated* option

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

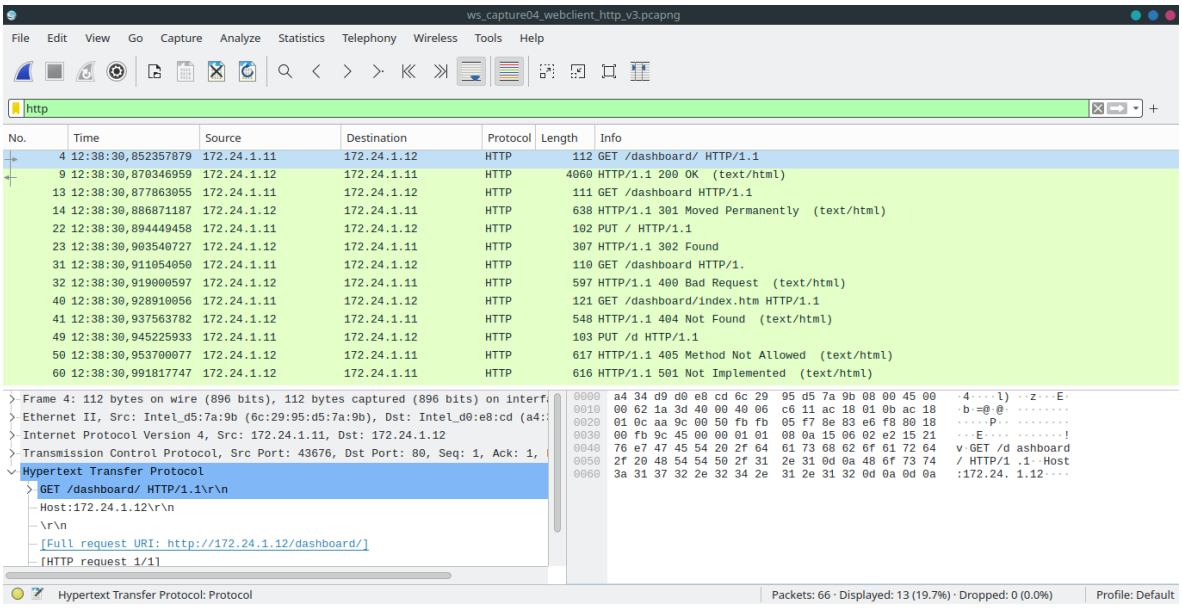
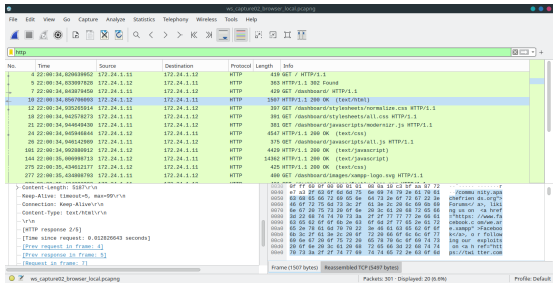
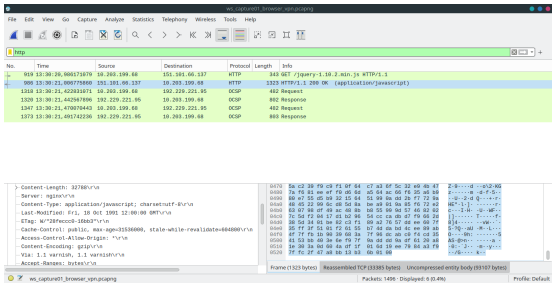


Figure 2.2: Webclient get



(a) Browser



(b) Browser (VPN)

Figure 2.3: Browser capture

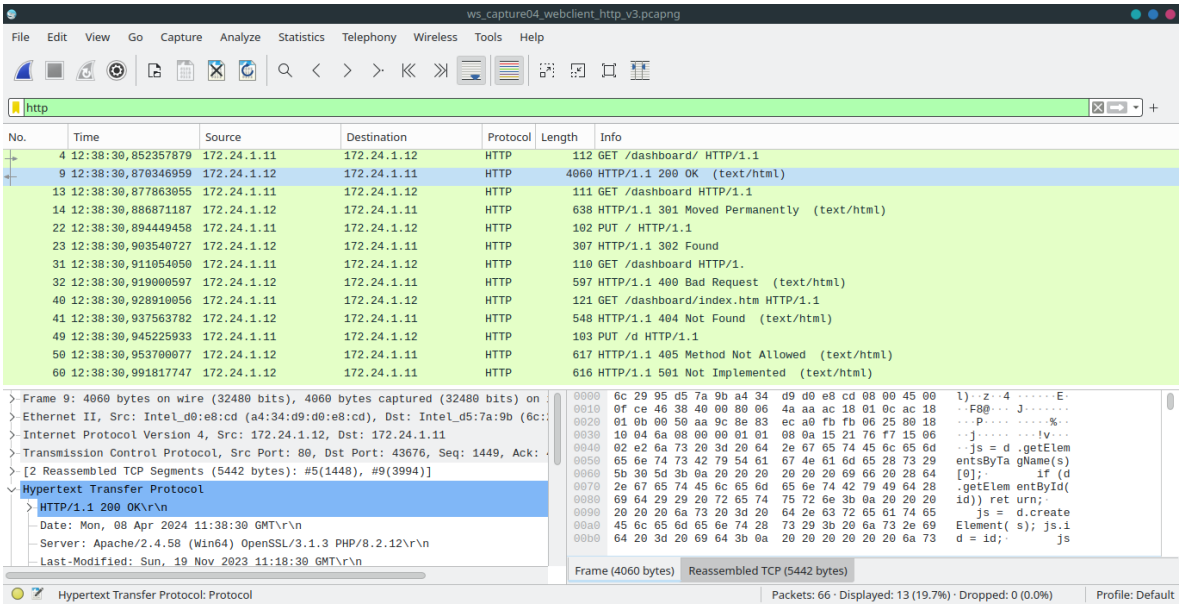


Figure 2.4: Webclient reply

Capture output txt

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.

2.8 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.

Appendix A

Um Apêndice