WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Shivani Eli

Email: elisi@mail.uc.edu

Figure 1: Shivani Eli

Lab 1 - Foundations of the WEB

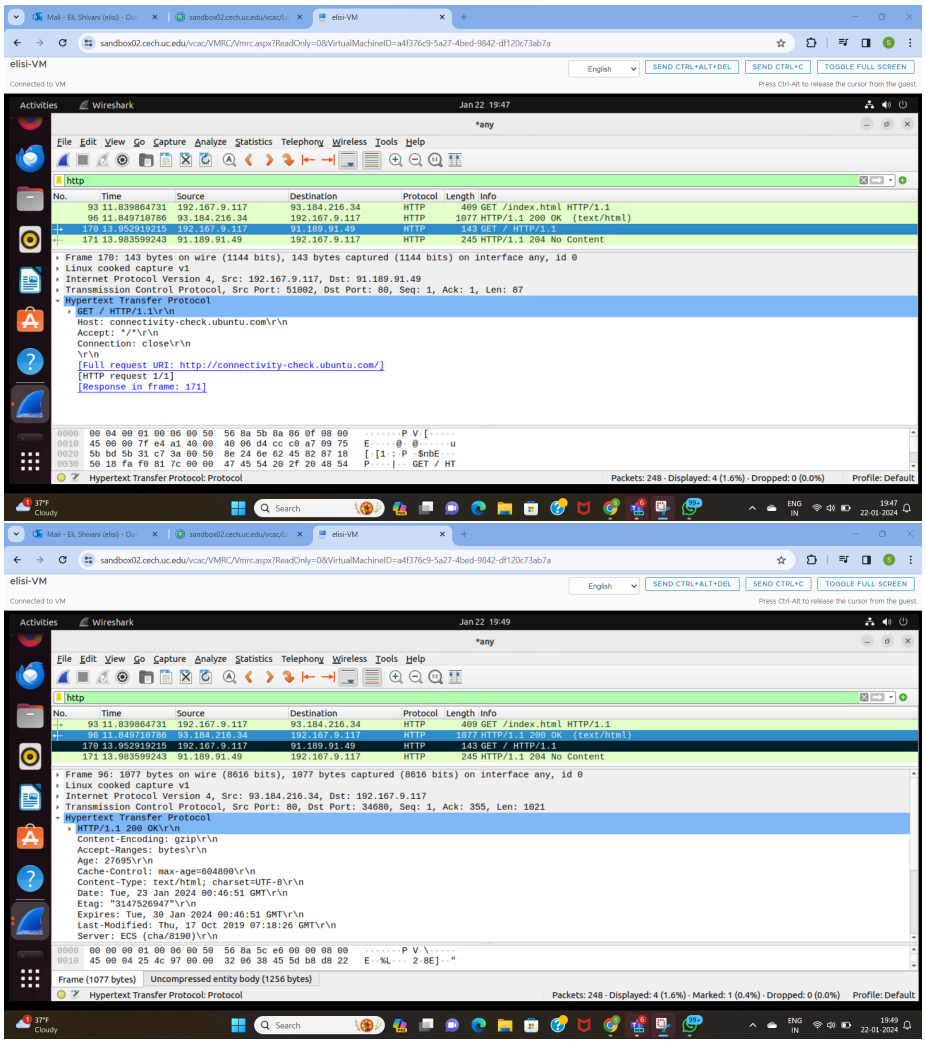
Overview: The lab1 of the WAPHT course, has two parts, part1 consists of HTTP protocol, wireshark, telnet and task2 deals with basic cgi programming in C and HTML, and examining the GET, POST, REQUEST for the browser requests. This also includes PHP basic programming and 'POST curl' commands. For this report pdf genaration pandoc was used.

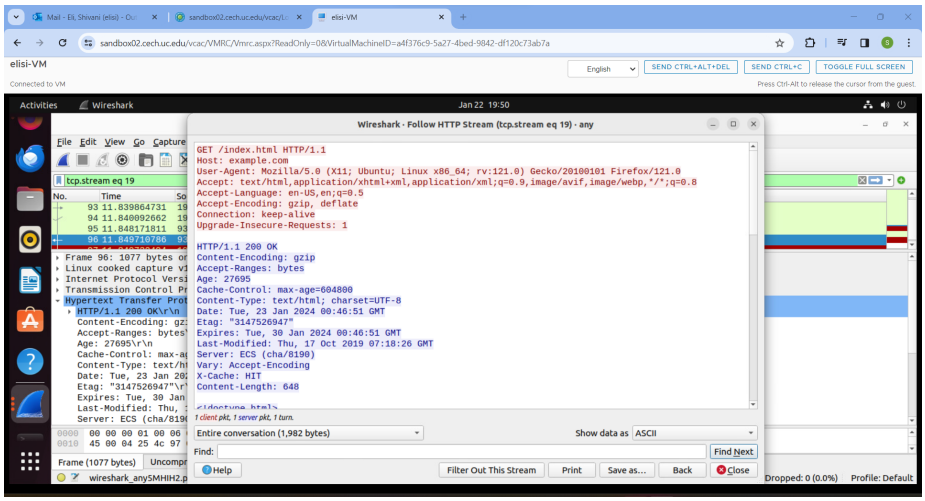
https://github.com/ShivaniEli/waph-elisi/blob/main/labs/lab1/README.md

Part 1 : The WEB and the HTTP Protocol

Task 1. Familiar with the Wireshark tool and HTTP protocol

The wireshark tool is used to analyse the packets for HTTP protocol. For this I installed the wireshark in the VM. After installation, opened the wireshark and clicked on the '4th icon' and clicked 'any' , ok start. Then in the browser opened a 'example.com' website then came back to wire shark and clicked 2nd icon then filtered the 'http' protocols, and got the GET and Response for the http request.


elisi-VM
Activities
Wireshark
Jan 22 19:47
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
93 11.839864731 192.167.9.117 93.184.216.34 HTTP 409 GET /index.html HTTP/1.1
96 11.849710786 93.184.216.34 192.167.9.117 HTTP 1077 HTTP/1.1 200 OK (text/html)
170 13.952919215 192.167.9.117 91.189.91.49 HTTP 143 GET / HTTP/1.1
171 13.983599243 91.189.91.49 192.167.9.117 HTTP 245 HTTP/1.1 204 No Content
Frame 170: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.167.9.117, Dst: 91.189.91.49
Transmission Control Protocol, Src Port: 51002, Dst Port: 80, Seq: 1, Ack: 1, Len: 87
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: connectivity-check.ubuntu.com\r\n
Accept: */*\r\n
Connection: close\r\n
\r\n
[Full request URI: http://connectivity-check.ubuntu.com/]
[HTTP request 1/1]
[Response in frame: 171]
Hypertext Transfer Protocol: Protocol
Packets: 248 · Displayed: 4 (1.6%) · Dropped: 0 (0.0%)
Profile: Default
elisi-VM
Activities
Wireshark
Jan 22 19:49
*any
http
No. Time Source Destination Protocol Length Info
93 11.839864731 192.167.9.117 93.184.216.34 HTTP 409 GET /index.html HTTP/1.1
96 11.849710786 93.184.216.34 192.167.9.117 HTTP 1077 HTTP/1.1 200 OK (text/html)
170 13.952919215 192.167.9.117 91.189.91.49 HTTP 143 GET / HTTP/1.1
171 13.983599243 91.189.91.49 192.167.9.117 HTTP 245 HTTP/1.1 204 No Content
Frame 96: 1077 bytes on wire (8616 bits), 1077 bytes captured (8616 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.167.9.117
Transmission Control Protocol, Src Port: 80, Dst Port: 34680, Seq: 1, Ack: 355, Len: 1021
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Content-Encoding: gzip\r\n
Accept-Ranges: bytes\r\n
Age: 27695\r\n
Cache-Control: max-age=604800\r\n
Content-Type: text/html; charset=UTF-8\r\n
Date: Tue, 23 Jan 2024 00:46:51 GMT\r\n
Etag: "3147526947"\r\n
Expires: Tue, 30 Jan 2024 00:46:51 GMT\r\n
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT\r\n
Server: ECS (cha/8190)\r\n
Frame (1077 bytes)
Uncompressed entity body (1256 bytes)
Hypertext Transfer Protocol: Protocol
Packets: 248 · Displayed: 4 (1.6%) · Marked: 1 (0.4%) · Dropped: 0 (0.0%)
Profile: Default



Task 2. Understanding HTTP using telnet and Wireshark

For TELNET, I ran the telnet example.com 80 command, then captured the packets in the wireshark. For this after telnet command establishing the connection to the example.com server, the request type file path http and host name were given as shown in the below screenshots, for the http request. then clicked two times 'enter' for http response

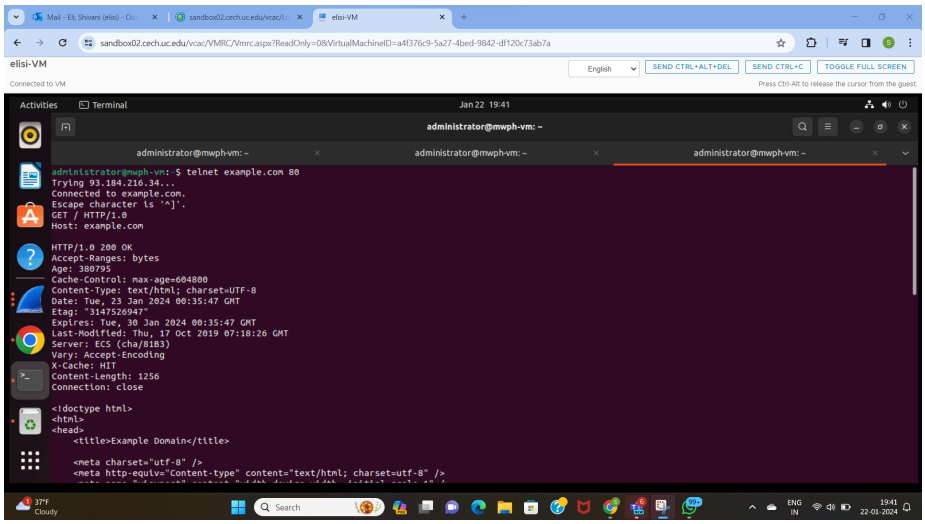
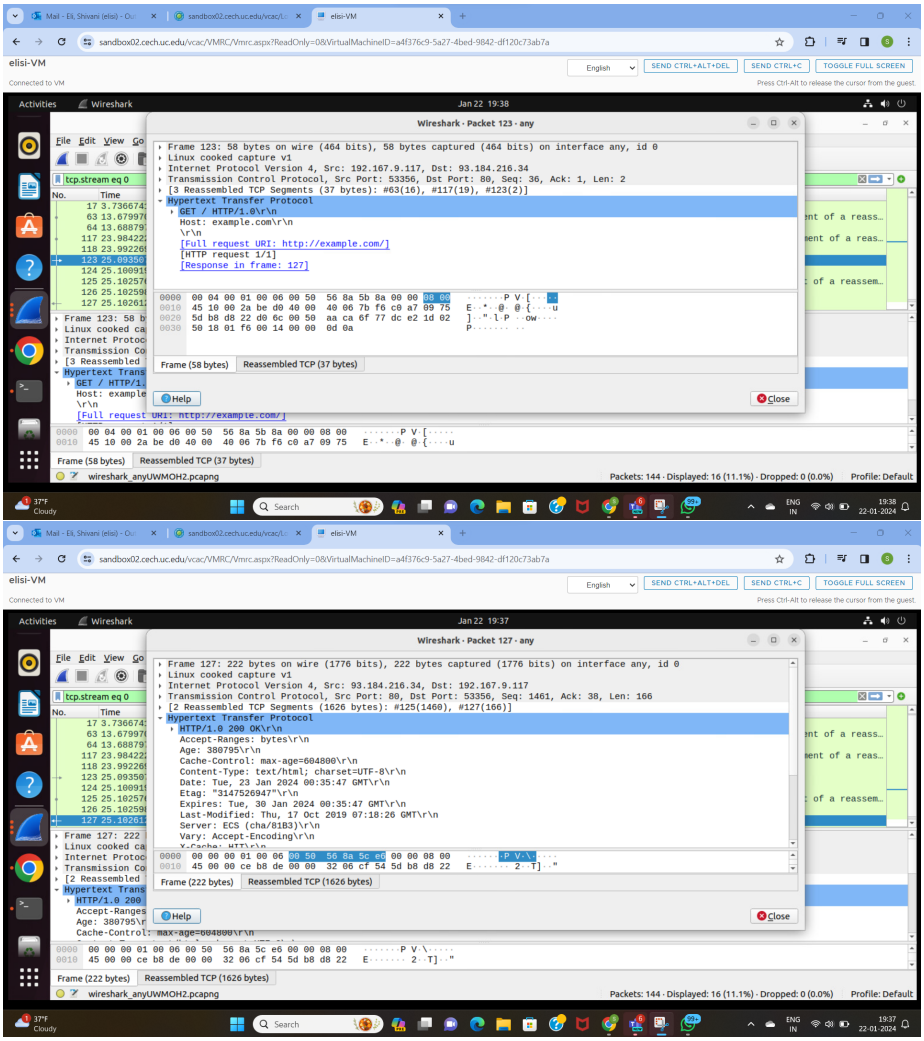


Figure 2: Telnet request for the example.com

Now, coming to the stream packets in wireshark for the http request via browser and telnet, the response for both ways remain same. Yet the server fields in the

packets stream were missing for http request via telnet.



Part II - Basic Web Application Programming

Task 1: CGI Web applications in C

A.A basic 'helloworld' printf C program was saved in the 'helloworld.c' file. This file was compiled using the gcc and the compiled file was deployed generated cgi, then copied this to the path usr/lib/cgi-bin, for it to be accessed on browser i.e. 'localhost/cgi-bin/helloworld.cgi'.

B.After a simple C program, another 'index.c' file was created with HTML template. In this program it has the head with tittle as Name, with body contaning the course details. Once saved, the file is compiled with gcc and copied

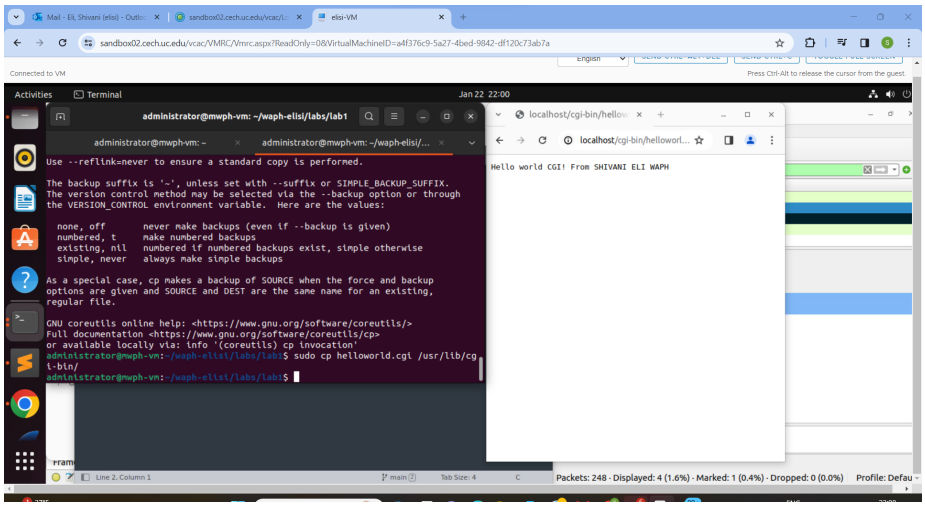


Figure 3: CGI program in C

to usr/lib/cgi-bin before accessing in the browser. This way the index.c file has C code with HTML basic template.

The .c file is included below.

Included file **helloworld.c**:

```
#include<stdio.h>
int main() {
const char *htmlcOntent = "<!DOCTYPE html> <html> <head> "
                          "<title>Web application programming and Hacking</title>"
                          "</head> <body> <h1>Name: Shivani Eli</h1>"
                          "<p>lab1 assessment assignment exercises. "
                          "This is a CGI application.</p></body></html>";

printf("Content-Type: text/html\n\n");
printf("%s", htmlcOntent);
return 0;
}
```

Task 2: A simple PHP Web Application with user input.

A. For task 2 developed a simple PHP web application '.php' , and deployed it to apache2 'var/www/html'. Firstly, Installed the required apache2, then created a helloworld.php file with some basic echo php code in it. Then ran the command 'sudo cp helloworld.php /var/www/html', in the browser it was accessed by 'localhost/helloworld.php'.

Included file **helloworld.php**:

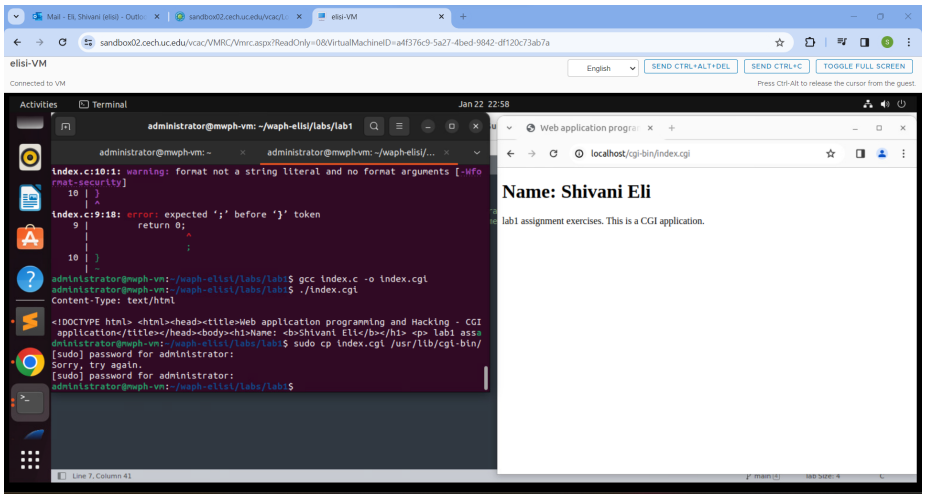


Figure 4: The CGI C file embedded with HTML

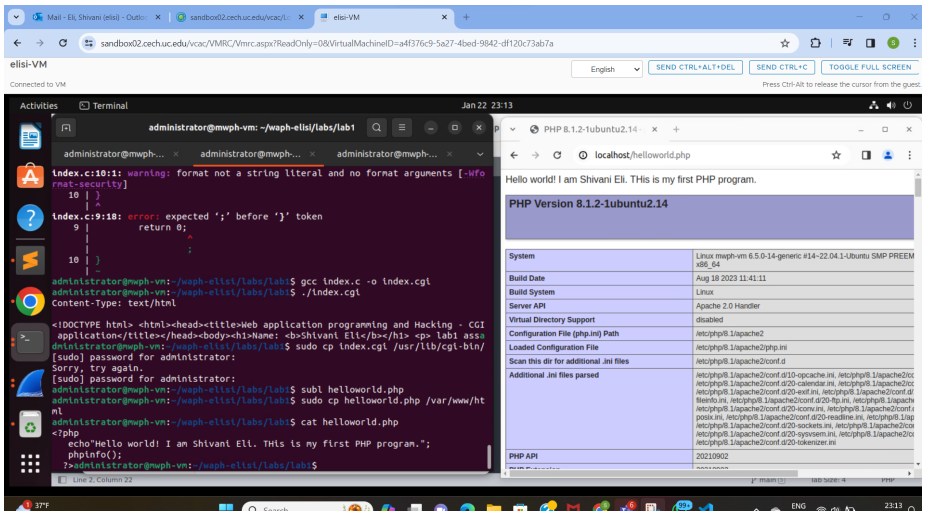
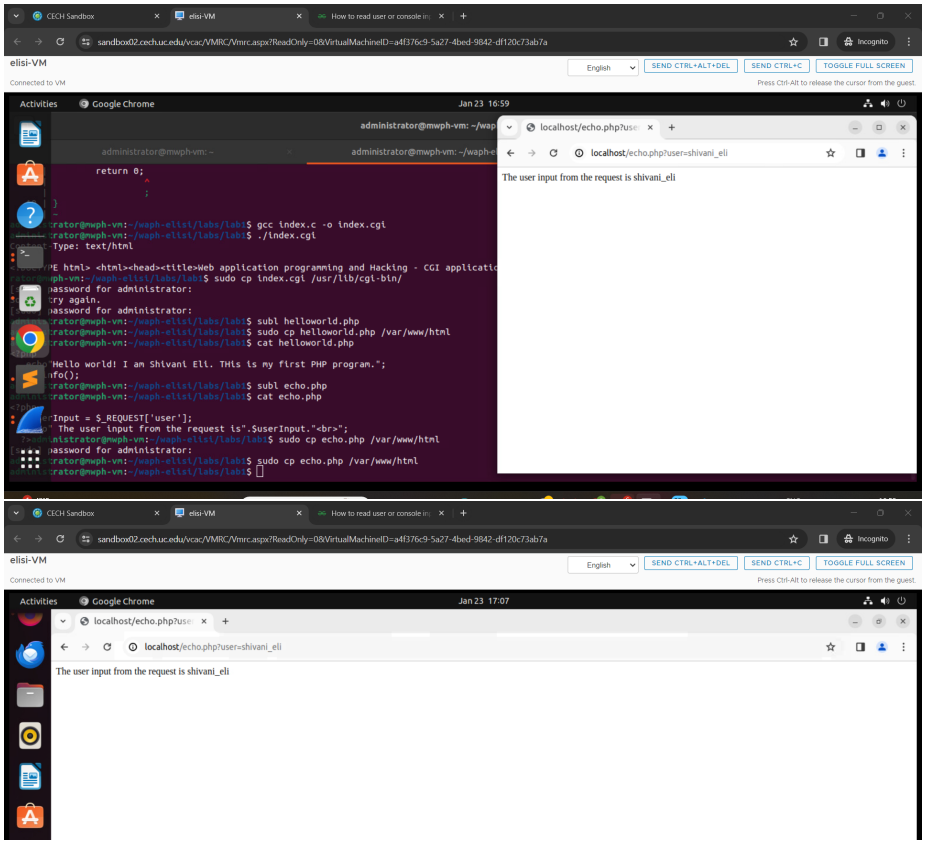


Figure 5: helloworld.php

```php
<?php
    echo "Hello World! I am Shivani Eli. THis is my first PHP program.";
?>
```

B. Then another file echo.php was created with simple echo statement, and '$_REQUEST['variable']' for getting the user input. Then after the /var/www/html, in the browser url the url is the 'localhost/echo.php?user="shivani_eli" '. This $_REQUEST enables the input of user, in PHP.



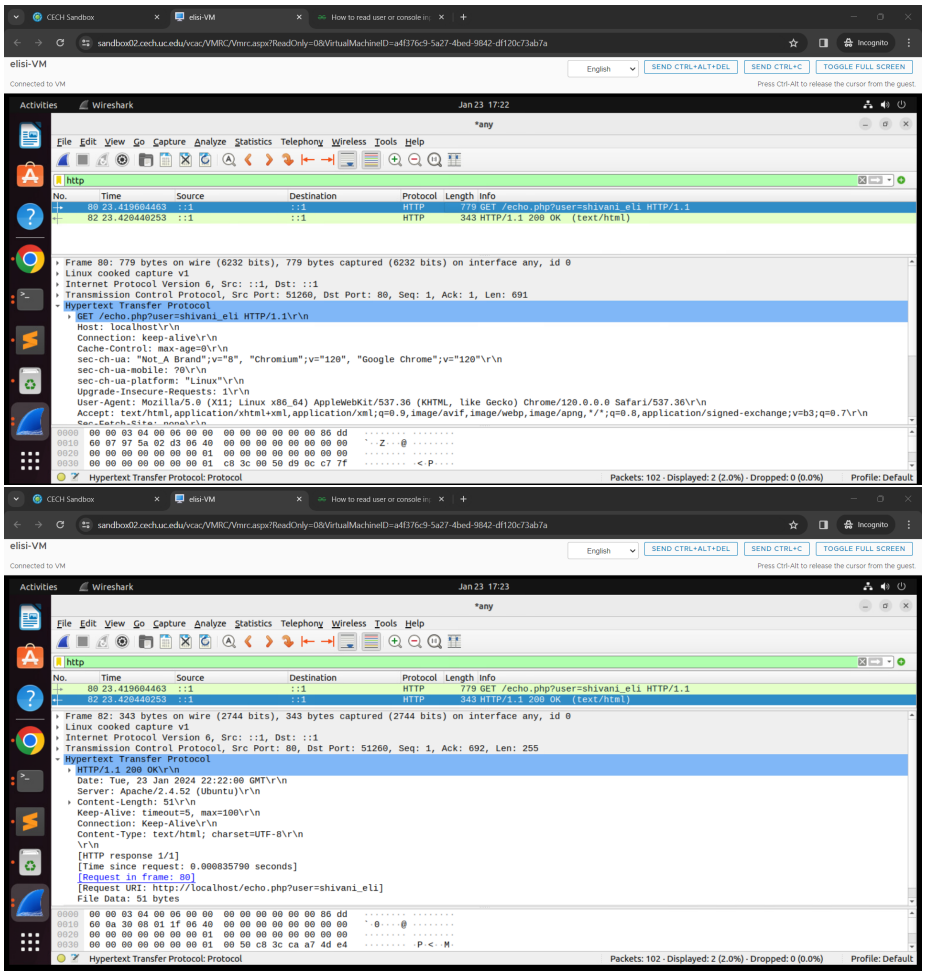
Included file `echo.php`:

```php
<?php
    $userInput = $_REQUEST["user"];
    echo "The user input from the request is " . $userInput . "<br>";
?>
```

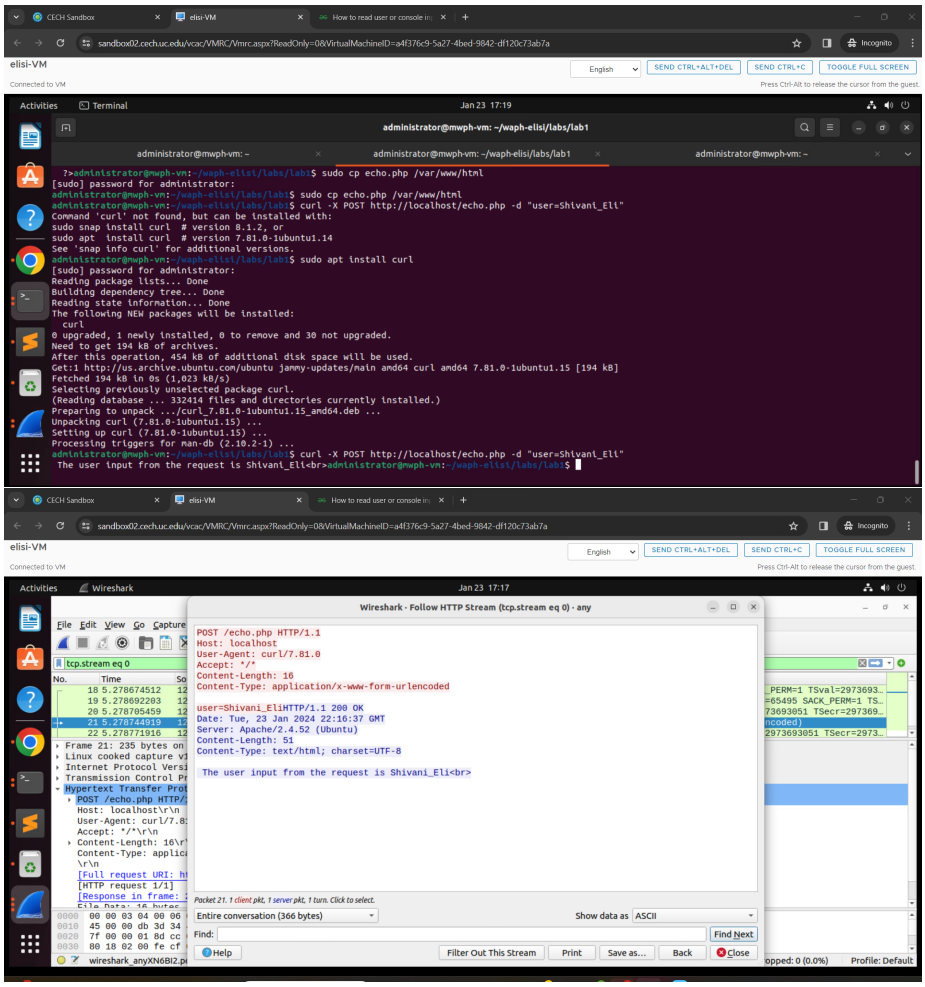
Task 3: Understanding HTTP GET and POST requests.

A. Now, the HTTP GET and POST requests are analysed. The browser call was made with GET request with syntax 'localhost/echo.php?user="shivani_eli" '

where user is the input. Then along with the echo statement, the user value ‘shivani_eli’ was also displayed in the browser. This GET and response for the http protocol was captured in the wireshark.



B. The curl tool was used to get the POST requests in the htttp request for the echo.php file. Installed the curl, then entered the command curl -X POST localhost/echo.php -d “user=Shivani_Eli”. To get the post request, this before excuting started the wireshark, then entered the above curl command then after this, in the wireshark the POST http request was captured.



C.The similarities and differences between HTTP GET/ POST requests and respones Similarities: Both HTTP techniques are employed in client-server communication. Both contain request headers. Responses with headers and status codes are sent to both. Wireshark used to record and analyse the GET/POST requests. Differences: Data is sent in the URL for a GET request, while it is transmitted in the HTTP body for a POST request in the above curl was used. Information may be retrieved using GET, while updates can be made using POST. Since data is not accessible to regular users, POST is more secure.

Lastly, The echo.php is simple echo statement program, the GET/POST are similar responses.

After completing the tasks, the labs/lab1 folders were created and pushed, the README.md file was edited then pushed. Using pandoc the README.md file's pdf was generated for the lab1 submission.