WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Tejeswar Reddy Nalijeni

Email: nalijety@ucmail.uc.edu

Short-bio: Currently I am pursuing Master's in Information Technology and I am interested in Cybersecurity

Lab 1 - Foundations of the Web

The lab's overview

From this lecture 3 of Lab 1, I understand that Wireshark is a great network protocol analyzer widely used for capturing and examining data within a network. One of its primary applications is in the analysis of the HTTP protocol, which serves as the fundamental communication protocol for information exchange on the World Wide Web. HTTP using telnet and Wireshark gave me a good understanding hands-on experience. Telnet is mainly used to manually transmit HTTP requests and receive responses. Wireshark captures and displays the network traffic.

I started by using telnet in the Ubuntu terminal to connect to a web server at port 80. After issuing a GET HTML request with the host (example.com), I received a 200 OK response containing the HTML code of the index page. Notably, the telnet-based request lacked certain fields compared to browser-sent requests, such as cookies and user-agent information.

Subsequently, I undertook various tasks, including creating CGI and PHP scripts,

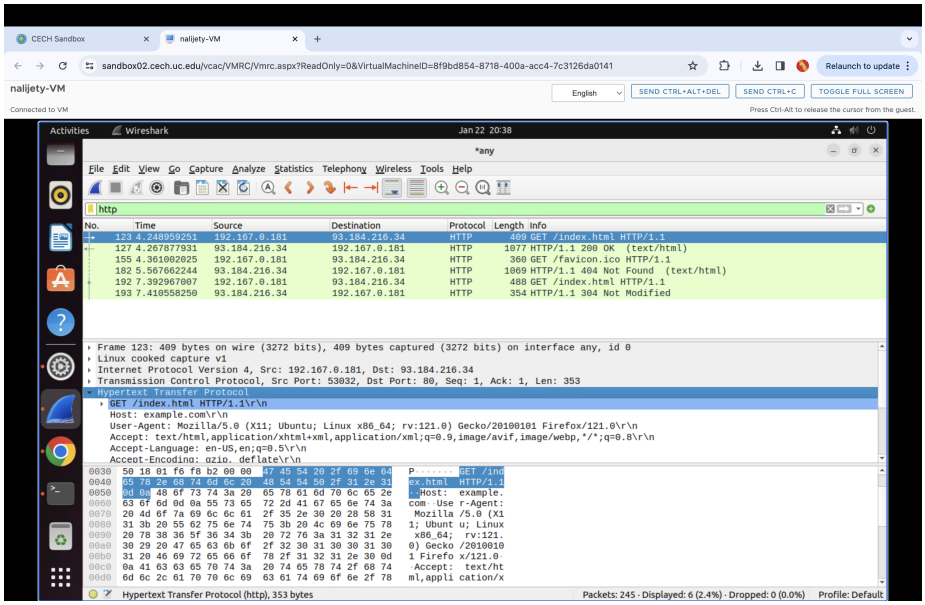
compiling C programs, and capturing traffic using Wireshark. These activities involved executing scripts, moving files to specific directories, and analyzing network traffic for HTTP messages.

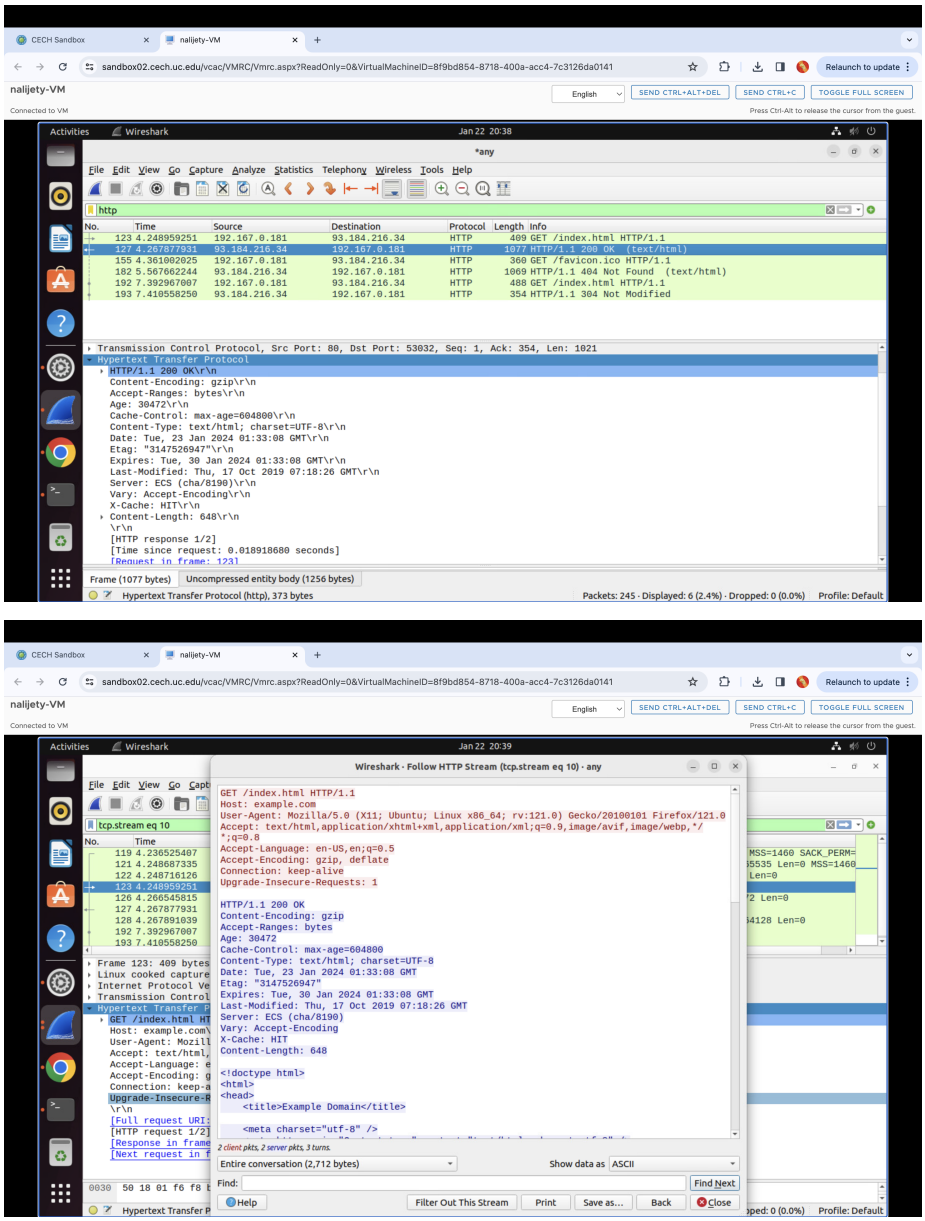
https://github.com/nalijety/waph-nalijety/tree/main/labs/lab1.

Part I - The Web and HTTP Protocol

Task 1. Familiar with the Wireshark tool and HTTP protocol

Primarily, I initiated a wireshark to examine the HTTP protocol for capturing network traffic and analyzing data. After initiating packet capture and I applied a filter for HTTP traffic, the analysis focuses on identifying HTTP request and response messages. When a request is initiated, the browser sends a message to the server which is HTTP request. Simultaneously, HTTP response messages from the server are examined, paying attention to status codes and response headers containing server information. Then I clicked on the follow and choosen the HTTP stream for the view of the data exchange. The HTTP stream is then inspected, revealing the sequence of requests and responses.

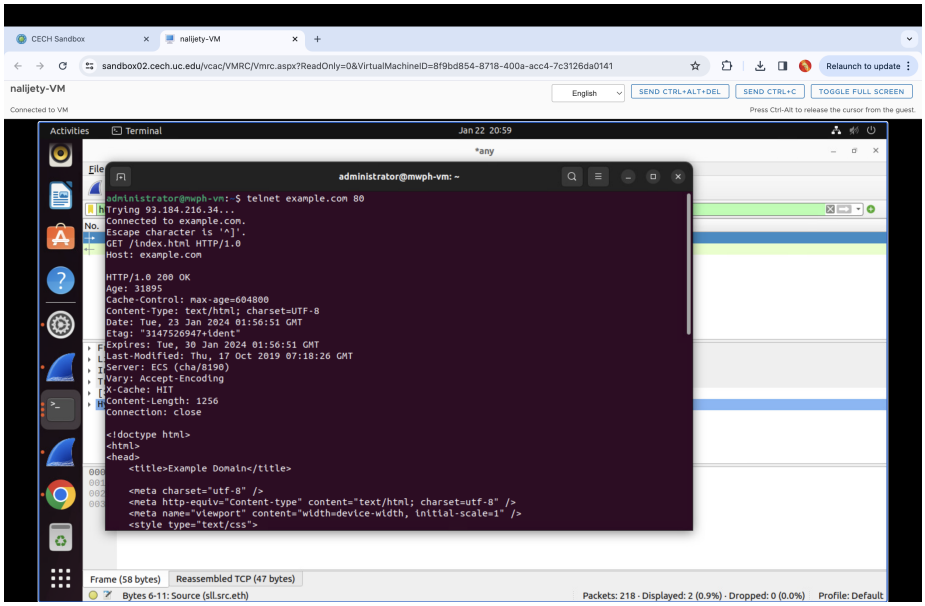




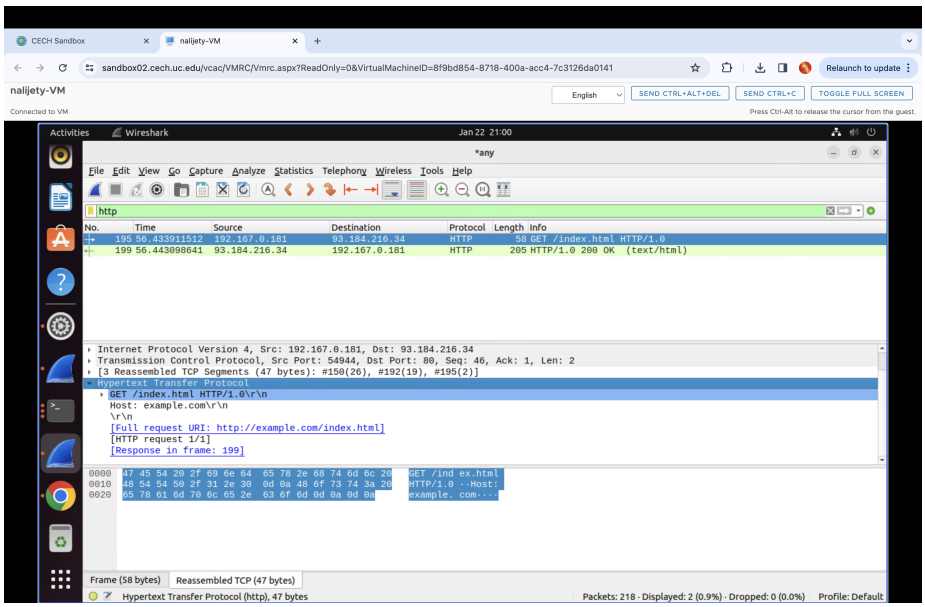
Task 2. Understanding HTTP using telnet and Wireshark

Initially, using the Ubuntu terminal, I used the telnet command for connecting the web server at port 80. After that I gave the command GET HTML request with the host (example.com). With this request I got the response with 200 OK including the html code of the index page.

1.

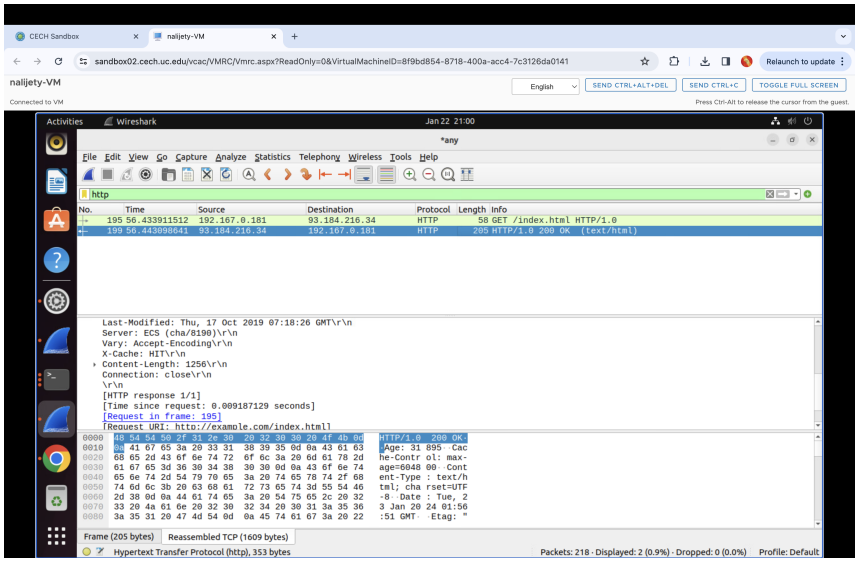


2.



From the above screenshot, the HTTP request made using telnet in the Ubuntu terminal have lacked certain fields compared to the one sent by a browser in Task 1. The Common omissions might include headers related to cookies, user-agent information, or specific content negotiation preferences. When using telnet,

the request will be more minimalistic, containing only essential details such as the requested resource and host, as it's a manual command line interaction without the automatic inclusion of various headers that browsers typically send for enhanced functionality and compatibility.

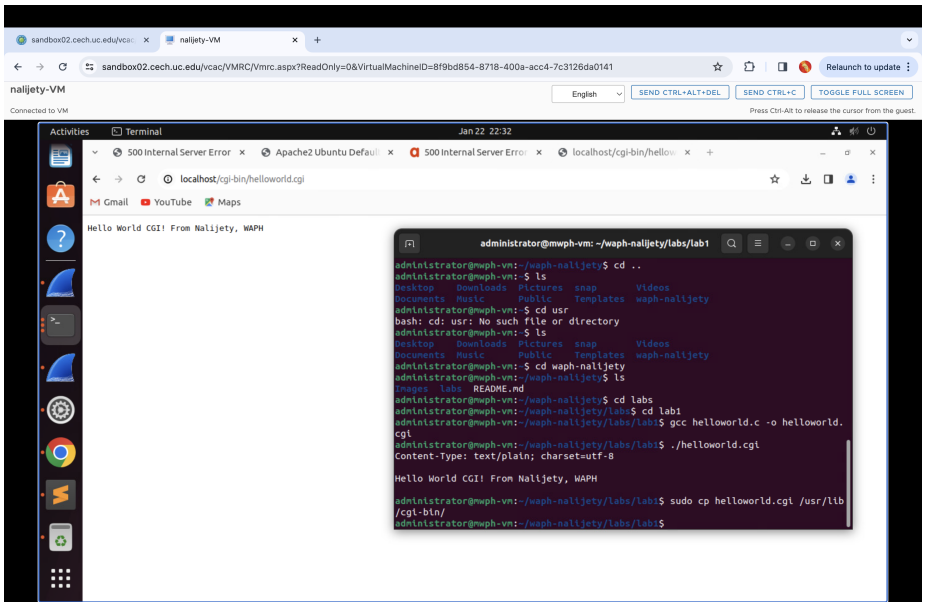
3. 

From the above screenshot, I found that there is no major differences between the HTTP response by the telnet and the web browser.

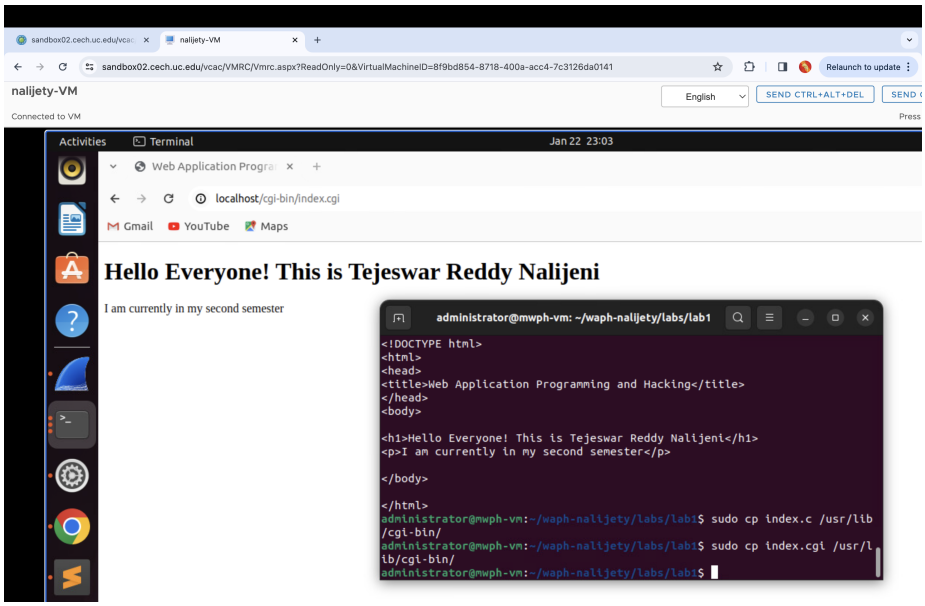
Part II - Basic Web Application Programming

Task 1. (10 pts) CGI Web applications in C

a. Initially, I navigated to the lab 1 file which is located in my local repository. Then I created a helloworld.c file using sublime in the terminal. In the file, I provided a c code which specify content type plain text and printing hello world. After that, I used GNU Compiler collection to compile the C program in "helloworld.c" into an executable named "helloworld.cgi". Then, the compiled program is executed in the current directory, as part of a CGI script in environment of web server. Since the CGI programs are stored in bin/usr/1ib/cgi folder, the helloworld.cgi file is moved to that path. Then I opened the browser and visited the URL- "http://localhost/cgi-bin/helloworld.cgi". The execution of the cgi file is successful and the output obtained is as shown below.



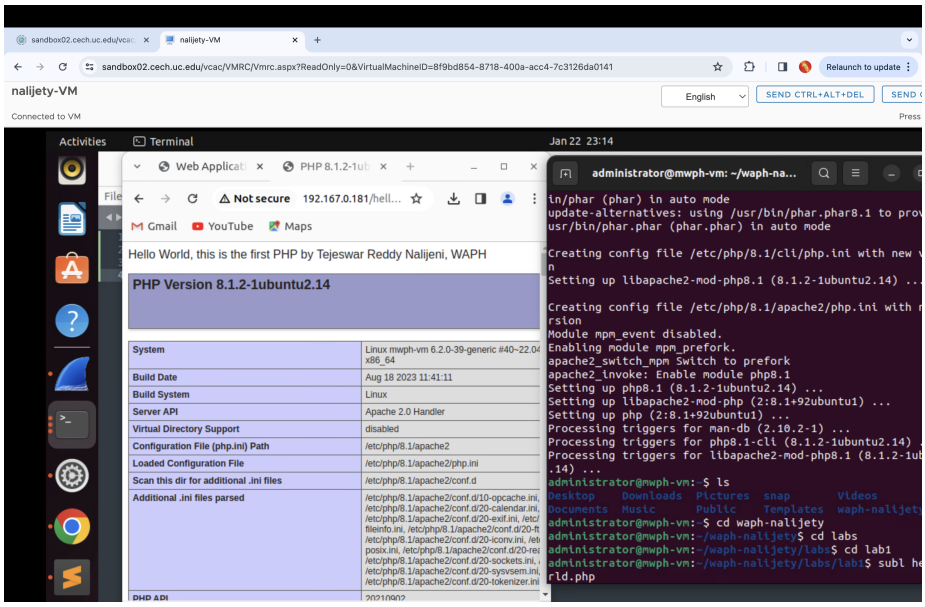
b. Initially, I have created the file named “index.c” in the sublime. Next, the c code printed each line of a sample HTML code with the title, head, and the body. I have given the head includes course name and the body of the page included a information about me. Thereafter, I converted the index file to a .cgi file using gcc command and the I saved it in the “bin/usr/lib/cgi” folder. Then I opened the browser and searched with URL - http://localhost/cgi-bin/index.cgi. Now the file is executed successfully and the below shows a simple HTML code in terminal and page.



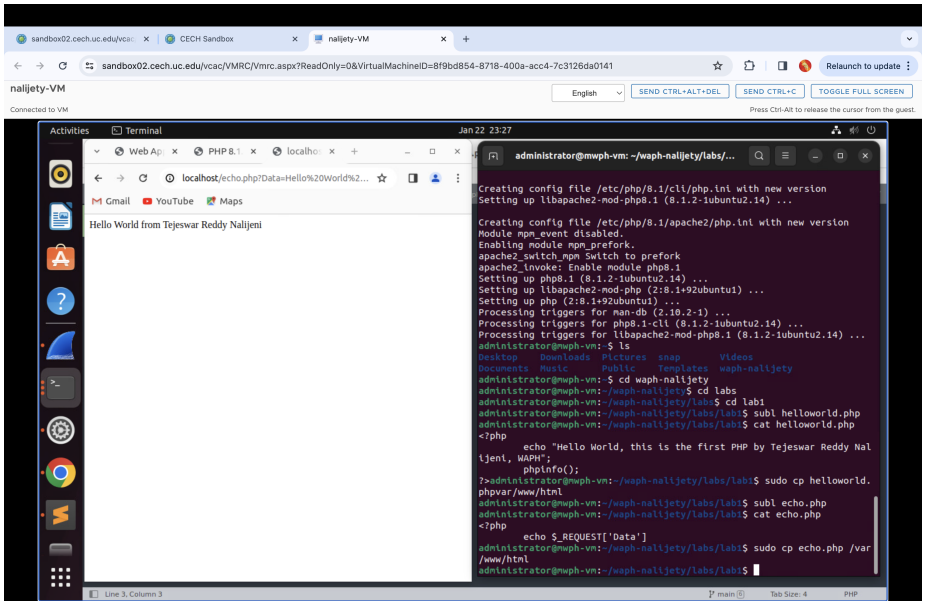
Included file `index.c`: C `#include <stdio.h> int main() { printf("Content-Type: text/html\n\n"); printf("<!DOCTYPE html>\n"); printf("<html>\n"); printf("<head>\n"); printf("<title>Web Application Programming and Hacking</title>\n"); printf("</head>\n"); printf("<body>\n\n"); printf("<h1>Hello Everyone! This is Tejeswar Reddy Nalijeni</h1>\n"); printf("<p>I am currently in my second semester</p>\n\n"); printf("</body>\n\n"); printf("</html>\n"); return 0; }`

Task 2 (10 pts). A simple PHP Web Application with user input.

a. Initially, in the Ubuntu terminal I have installed the PHP. After that I opened my local lab 1 folder and then created a helloworld.php file in sublime. Now in this file, I have given a code to print out a string containing hello world and my name using the echo function which is followed by using a phpinfo() function which is to print PHP information on the server. Then I copied the file to the web server root directory which is "var/www/html". Now I have opened the browser and then I have opened the page using the URL- http://192.167.0.181/helloworld.php.

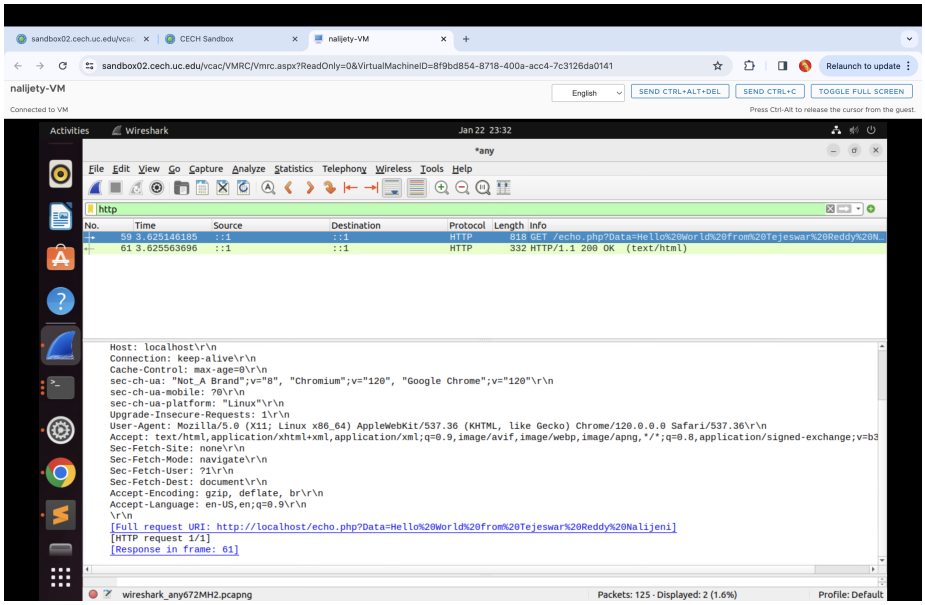


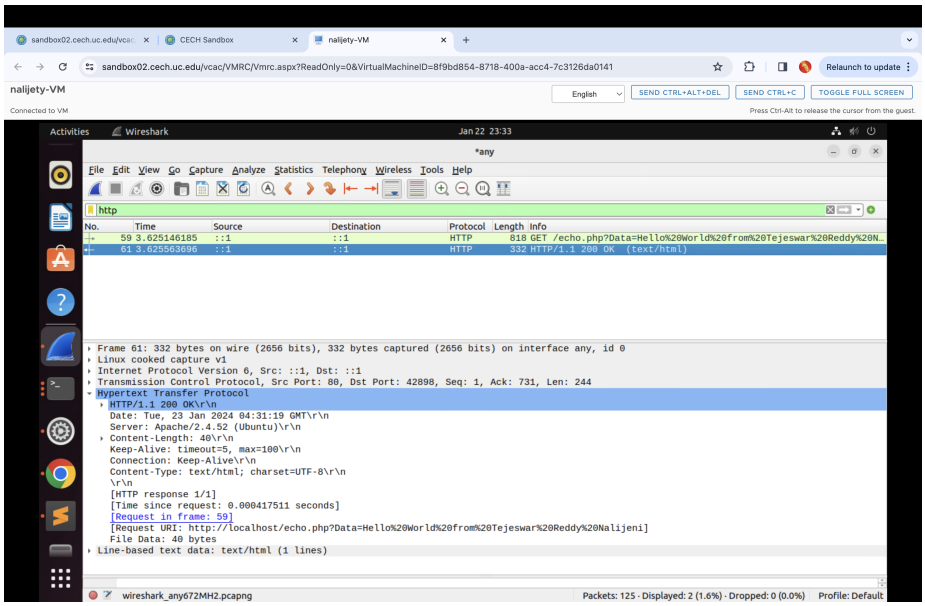
b. Initially, I have created the PHP file in sublime which is named as "echo.php".This PHP code retrieves the value associated with the "Data" parameter from the incoming request. It can be done by checking either the URL for GET requests or the request body for POST requests. Once it is obtained, the code echoes (prints) the value to the output. Now, in this I have given my name as the input for the variable 'Data'. FInally, I have moved this file to the /var/www/html path and then opened the page using the URL - http://192.167.0.181/echo.php. As the code can be accessed by the user input straightly, it might be manipulated by the users in order to inject malicious content to the application.



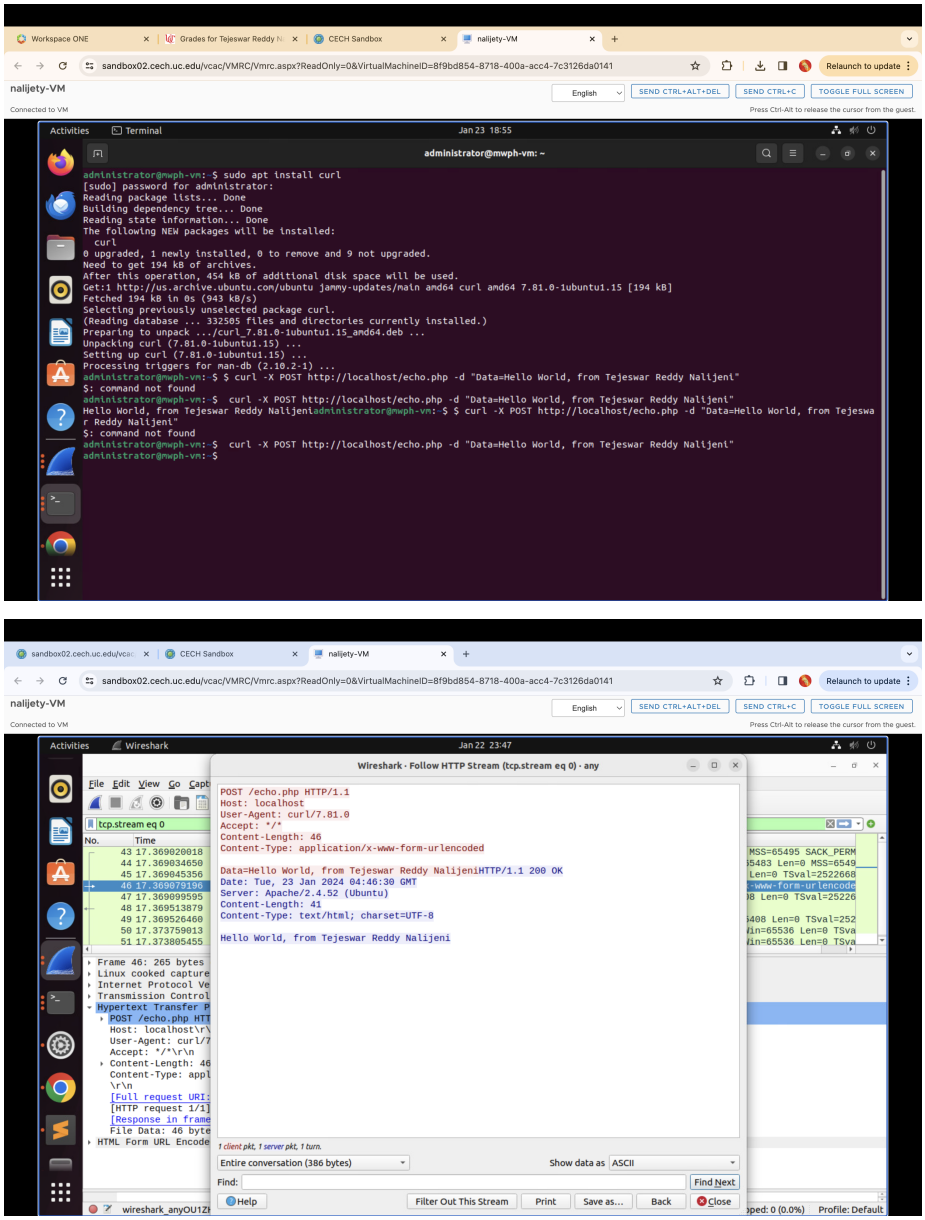
Task 3 (10 pts). Understanding HTTP GET and POST requests.

a. I have opened wireshark and then initiated the browser for capturing the traffic of the echo file. After that, I have stopped the wireshark and then filtered the HTTP messages. The message is a GET request, and it incorporates the data I provided within its information. Finally, the response obtained as 200 OK status code.





b. Initially, I have opened wireshark and then initiated the browser for capturing the traffic. For this, the curl command is used for making the HTTP requests. This command simulates an HTTP POST request to the local server which sends the data “Hello World from Tejeswar Reddy Nalijeni” to the PHP script named echo.php. Then Wireshark traffic capture is stopped and then filtered the http messages. Finally, I have clicked on the request message and then clicked on follow which leads to the HTTP stream.



c. The data transmission is the difference that lies between the GET and POST requests. The GET will include the parameters in URL, whereas POST will send them directly to the request body. In both of the tasks, the response is a 200 OK status code, which signifies the requests successfully.