WAPH-Web Application Programming and Hacking

Instructor : Dr. Phu Phung

Student

Name: Grahika Rampudi

Email: rampudga@mail.uc.edu

Figure 1: Grahika Rampudi

Lab 1 - Foundations of the WEB

Overview: Web technologies, the HTTP protocol, and basic web application programming are the main topics of this lab. The focus is on analysing HTTP requests and responses using Wireshark and TELNET and comparing them to requests made by a web browser. The hands-on components comprise learning how to create CGI programmes, incorporating HTML templates, and exploring PHP web application development. Using Wireshark and curl, examine HTTP GET and POST requests as the final task. The Pandoc tool was used to create the PDF report for submission, and the Labs 1 report was written in Markdown format.

https://github.com/rampudga/waph-rampudga/edit/main/labs/lab1/README.md.

Part 1 : WEB and HTTP Protocol

Task 1. Familiar with Wireshark tool and HTTP protocol

Wireshark is open-source network protocol analyzer that allows users to capture and inspect the data traveling back and forth on a computer network in real-time.

I installed wireshark in VM, then clicked 4th icon and selected any on input interface, and started capturing the packets and http protocol.Themn again captured the screenshots of HTTP requests, HTTP response and stream. HTTP request is a client-initiated message that consists of a method (such as GET, POST), headers, and an optional body that is sent to a server, usually in order to request a resource or carry out an action.HTTP response is which includes headers and an optional response body in addition to a status code indicating the request's success or failure.

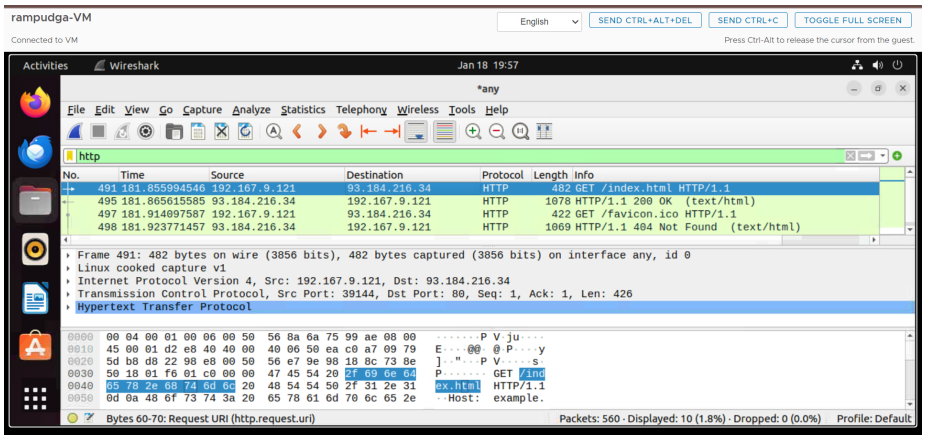

Figure 2: Wireshark HTTP Request

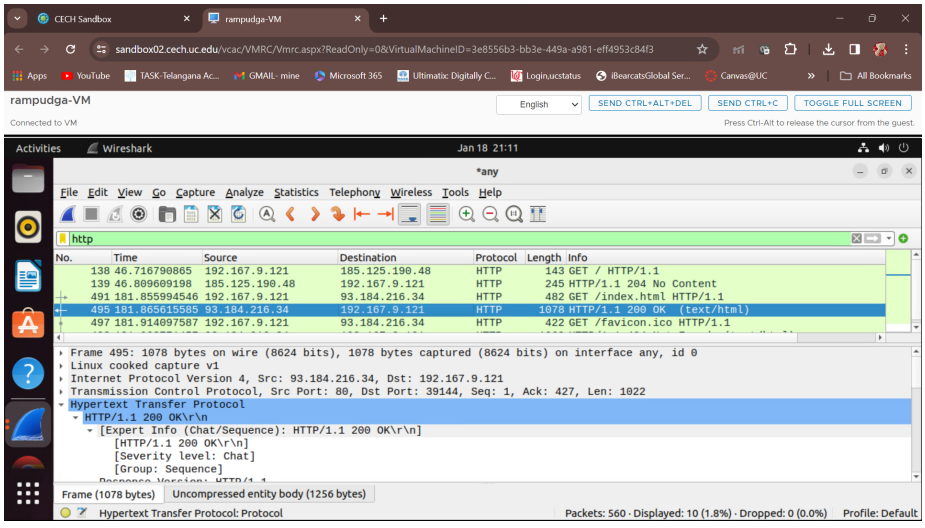

Figure 3: Wireshark HTTP Response

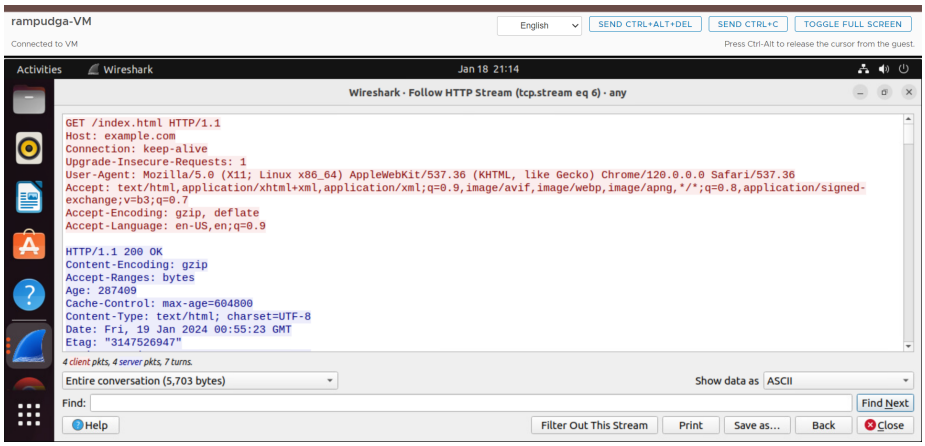

Figure 4: Wireshark HTTP Stream

Task 2. Understanding HTTP using telnet and Wireshark

Using TELNET in the terminal, Wireshark was launched to capture network packets that preceded the HTTP request to example.com/index.html. The example.com web server was first connected to using TELNET, with the syntax "telnet example.com portNumber." The HTTP was excecuted using details like host name , path, HTTP version,and request. So next hit enter twice to recieve response.

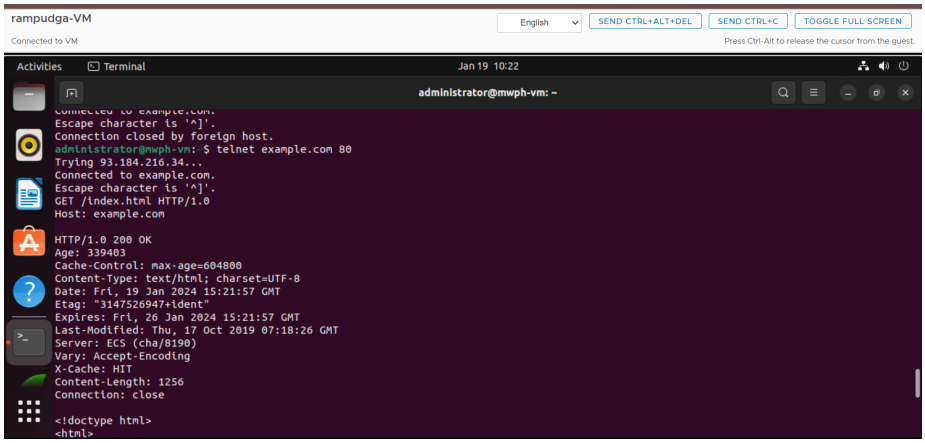

Figure 5: Telnet request

after comparing the http requests through browsers an d telnet in wireshark, it is noted that server fields were missing in the telnet made request. The telnet HTTP request is created by hand, but the browser automatically fills in request

headers like user-agent, accept, accept-language, authorization, encoding, and content when sending a request.

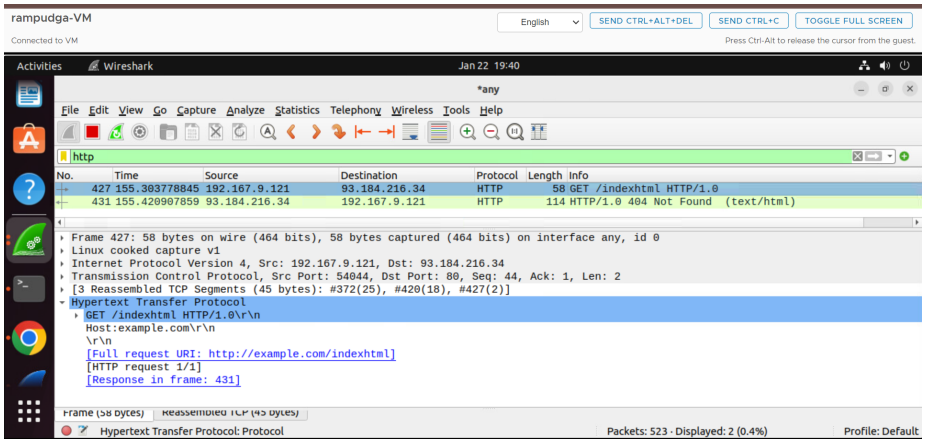

Figure 6: Telnet request in wireshark

The HTTP responses in wireshark in browser and telnet are same.

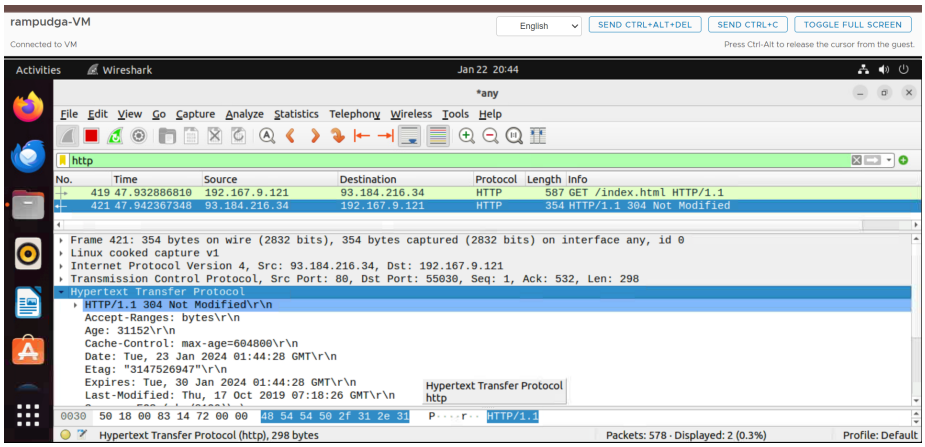

Figure 7: Telent response in wireshark

Part II - Basic Web Application Programming

Task 1: CGI Web applications in C

A. In this instance, I developed a CGI program that outputs “Hello world!” Using GCC, I compiled the program and subsequently moved the generated CGI file to usr/lib/cgi-bin for deployment. The result could be viewed in the browser by navigating to localhost/cgi-bin/helloWorld.cgi.

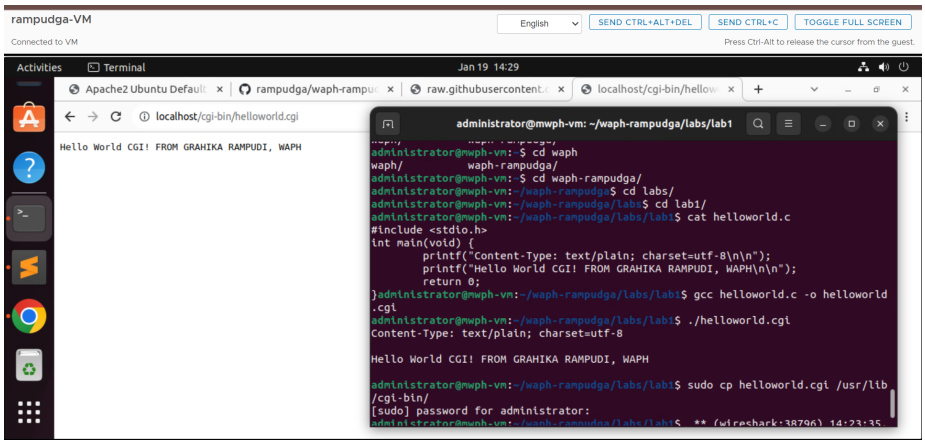

Figure 8: CGI program in C

B. I've now created another CGI programme, but this time I included a simple HTML template into the C code. The heading in this code is the student's name, the course name is the title, and other details are In this template, the heading is the student's name, the course name is the title, and other details are paragraphs. Before being viewed in a browser, this file was copied to usr/lib/cgi-bin and compiled with GCC.

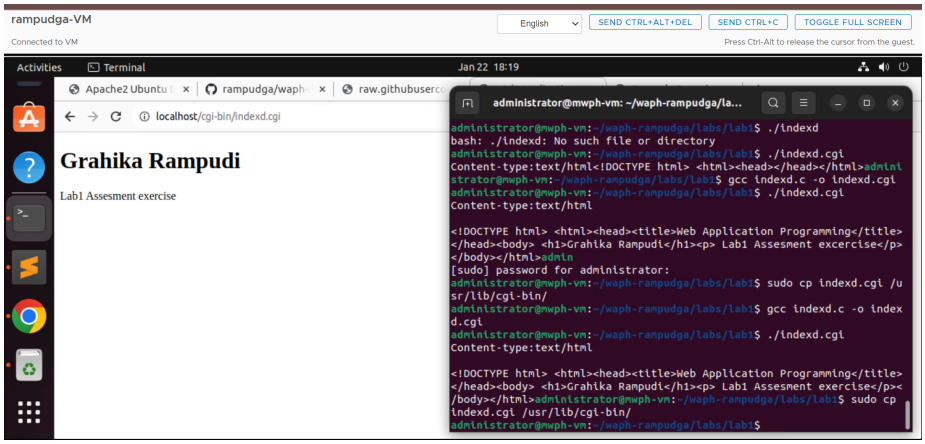

Figure 9: CGI in C and HTML

Included file `helloworld.c`:

```
#include<stdio.h>
int main() {
const char *htCont = "<!DOCTYPE html> <html> <head> <title>Web Application Programming</
```

```
                                    </head> <body> <h1>Grahika Rampudi</h1>\
                                    <p>Lab1 Assesment Excercise</p></body></html>";

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

Task 2: PHP Web Application with user input.

A. This task involves developing a PHP web application and deploying it to the root apache2 var/www/html directory. The basic syntax includes my name and the PHP version. Next, it was viewed in the browser at IP address/helloworld.php.

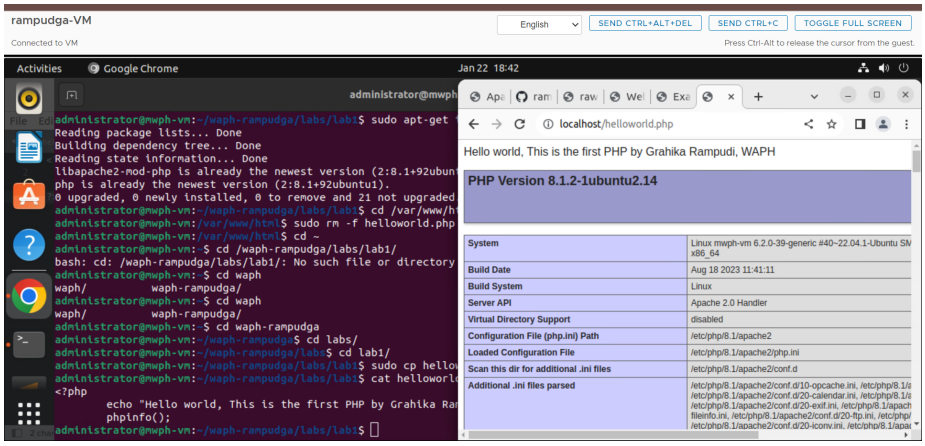

Figure 10: helloWorld.php

Included file `helloworld.php`:

```
<?php
    echo "Hello World! This is my first PHP by Grahika Rampudi , WAPH";
?>
```

B. PHP has been used to create a basic echo web application that outputs the path variable passed through the http request. There are several security flaws when using PHP's $_REQUEST('data') function to capture path variables in GET and POST requests, including the possibility of data tampering, SQL injections, and remote code execution. These risks can be reduced by sanitising user inputs, preparing statements for SQL inputs, and implementing input validation.

Included file `echo.php`:

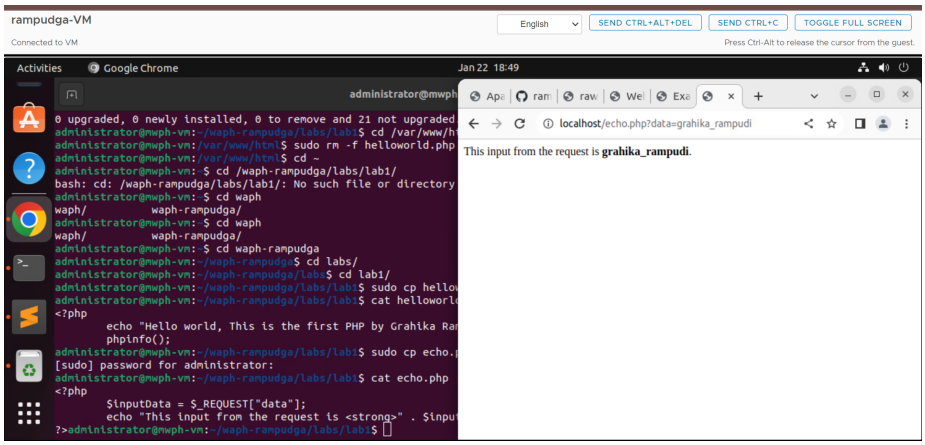

Figure 11: echo.php

```php
<?php
    $iData = $_REQUEST["data"];
    echo "The input from the request is <strong>" . $iData . "</strong>.<br>";
?>
```

Task 3: Understanding HTTP GET and POST requests.

A.By default the call that was made through the browser was a HTTP GET call and the path variable was passed using ? in the URL IPaddress/echo.php?data="value". The input varaible was then disaplyed as part of the response. This request, response and HTTP stream were analyzed through wireshark.

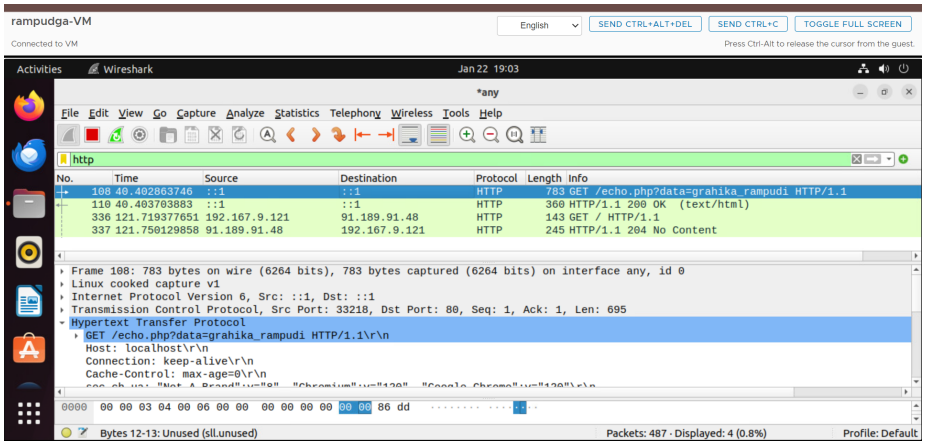

Figure 12: HTTP GET request in WireShark

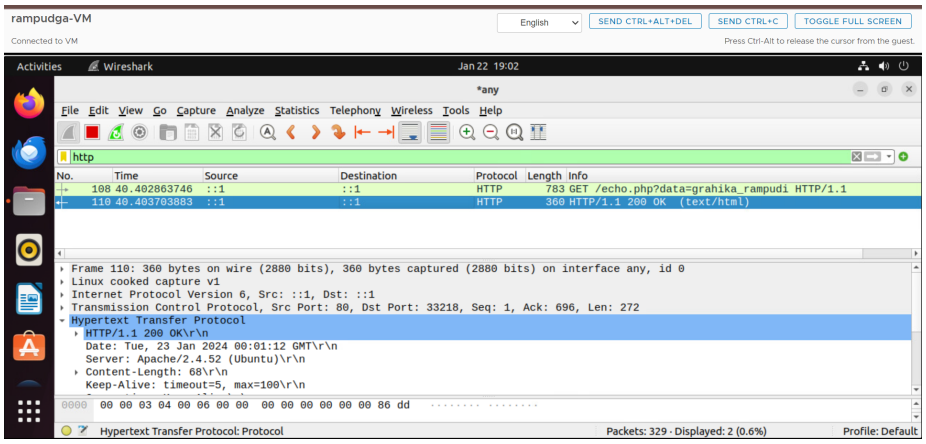

Figure 13: HTTP response in WireShark

B.A command-line tool for processing data using different n/w protocols is called Client URL of CUrl. My post request to echo.php was made via the terminal using CURL.

curl -X POST localhost/echo.php -d “data=Grahika Rampudi N”

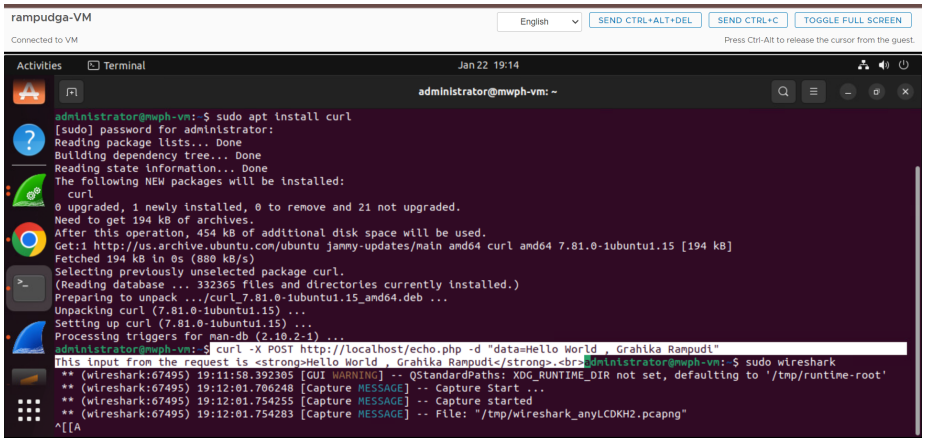

Figure 14: HTTP POST request using CURL

C.The similarities and differences between HTTP GET/ POST requests and respones Similarities: - They are both essential for sending and retrieving data between a client and a server. - To provide more details about the request, both may contain headers. - A status code indicating the request’s success or failure is included in both responses. Differences are: - POST sends data in the request body, keeping it hidden, whereas GET adds data to the URL, visible in the query string. - Because POST requests send data in the request body, they

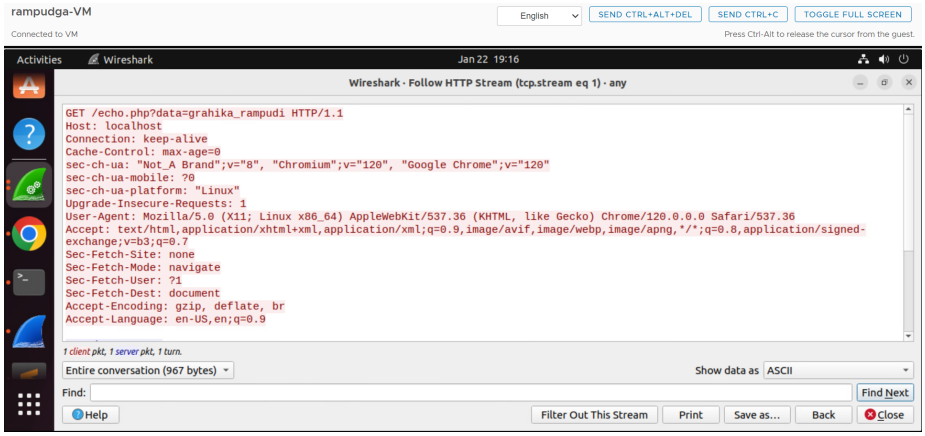

Figure 15: HTTP Stream in Wireshark

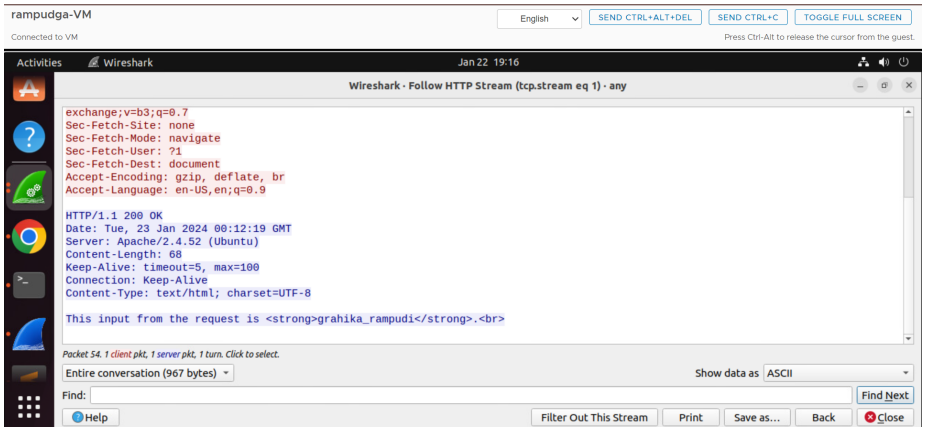

Figure 16: HTTP Stream in Wireshark

have a higher limit for data transmission than GET requests, which are usually limited by the maximum length of the URL. - Depending on the type of request (POST or GET) and the particular action being carried out, the body of the response may have a different structure and content.

Post this Labs/Lab1 folder was created to accomodate the project report and the changes were pushed. Pandoc tool was used to generate the project report from the README.md file