

WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Tulasiram Nakkantaboina

Email: nakkantm@mail.uc.edu



Figure 1: Tulasiram Nakkantaboina

Lab 1 - Foundations of the WEB

Overview: This lab deals with web technologies , HTTP protocol and basic web application programming. Focusing on Wireshark and TELNET for examining the HTTP requests, responses and comparing them with browser sent requests. Moving on to the web application programming I got familiarized with development of CGI programs in C and incorporating HTML templates. Additionally this lab covers PHP web application development. The final task explores HTTP GET and POST request utilizing wireshark and curl. The Labs1 report was written in Markdown format and Pandoc tool was used to generate the PDF report for submission.

https://github.com/nakkantm-uc/waph-nakkantm/blob/main/labs/lab1/R_EADME.md.

Part 1 : The WEB and the HTTP Protocol

Task 1. Familiar with the Wireshark tool and HTTP protocol

Wireshark is a protocol Analyzer tool which is used for network analysis and troubleshooting. After installing the wireshark in Ubuntu VM , I have clicked on the capture options (4th icon) selected any on the input interface and enabled the promiscous mode on filters checkbox. Now started to capture the packets and filtered HTTP protocol among all the other protocols.Then , respective HTTP requests and responses are selected to analyze and observe the HTTP stream. The HTTP request gives the information about the type of request, target URL , HTTP version, content-type , authorization and the data that is sent to the web servers. The HTTP response gives the information about status code, status text , content-type and the data that is sent back to the web browser etc.

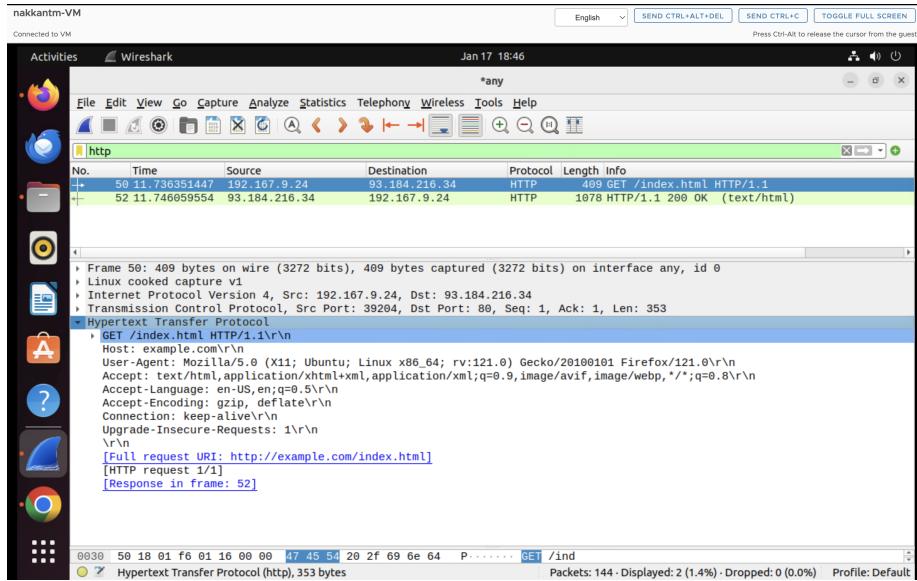


Figure 2: Wireshark HTTP Request

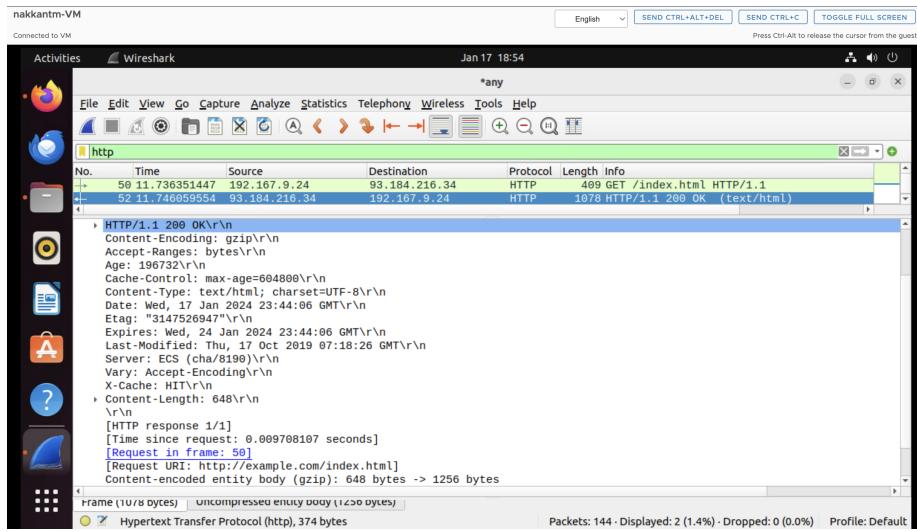


Figure 3: Wireshark HTTP Response

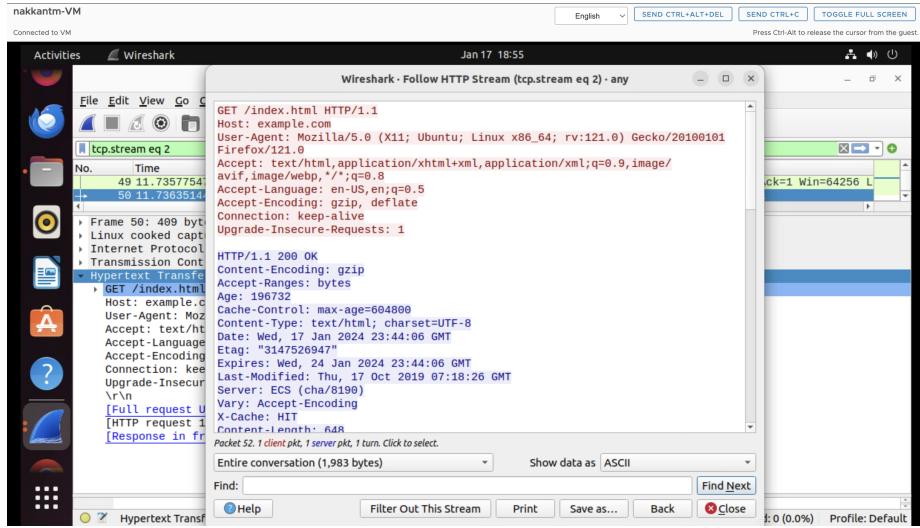


Figure 4: Wireshark HTTP Stream

Task 2. Understanding HTTP using telnet and Wireshark

Wireshark was started to capture the network packets before making the HTTP request to exmaple.com/index.html via TELNET through the terminal. For using the TELNET first the connection was established to the exmaple.com webserver through the syntax telnet example.com portNumber. After the connection is established the type of request , path file , http version and host name were given for making the HTTP Request. And the response was received after clicking on the enter twice.

Comparing the HTTP requests through browser and TELNET in wireshark, it is noted that server fields were missing in the telnet made request . The telnet HTTP request is manually constructed where as in the browser sent request the browser automatically populates request headers such as user-agent, accept, accept-language, authorization , encoding and content etc.

both the HTTP responses in wireshark through browser and TELNET were same.

Part II - Basic Web Application Programming

Task 1: CGI Web applications in C

A. I have developed a basic CGI program in C which just prints Hello world! , I have compiled this using gcc and deployed the generated cgi file by copying it to usr/lib/cgi-bin before accessing it on localhost/cgi-bin/helloWorld.cgi in the browser.

```

nakkantm-VM
Connected to VM
Activities Terminal Jan 17 19:03
administrator@molph-vm:~$ telnet example.com 80
Trying 93.184.216.34...
Connected to example.com.
Escape character is '^J'.
GET /index.html HTTP/1.0
Host: example.com

HTTP/1.0 200 OK
Age: 197835
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Thu, 18 Jan 2024 00:02:29 GMT
Etag: "3147526947+ident"
Expires: Thu, 25 Jan 2024 00:02:29 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close

<!DOCTYPE html>
<html>
<head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
background-color: #f0f0f2;
}

```

Figure 5: Telnet request

No.	Time	Source	Destination	Protocol	Length	Info
6	0.035685388	192.167.9.24	35.232.111.17	HTTP	143	GET / HTTP/1.1
8	0.069506107	35.232.111.17	192.167.9.24	HTTP	204	HTTP/1.1 204 No Content
107	88.687428458	192.167.9.24	93.184.216.34	HTTP	58	GET /index.html HTTP/1.0
111	88.696440617	93.184.216.34	192.167.9.24	HTTP	206	HTTP/1.0 200 OK (text/html)

Frame 107: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface any, id 0
 > Linux cooked capture v1
 > Internet Protocol Version 4, Src: 192.167.9.24, Dst: 93.184.216.34
 > Transmission Control Protocol, Src Port: 54654, Dst Port: 80, Seq: 46, Ack: 1, Len: 2
 > [3 Reassembled TCP Segments (47 bytes): #94(26), #105(19), #107(2)]
 Hypertext Transfer Protocol
 > GET /index.html HTTP/1.0\r\n
 Host: example.com\r\n\r\n
 [Full request URL: http://example.com/index.html]
 [HTTP request 1/1]
 [Response in frame: 111]

Figure 6: Telnet request in wireshark

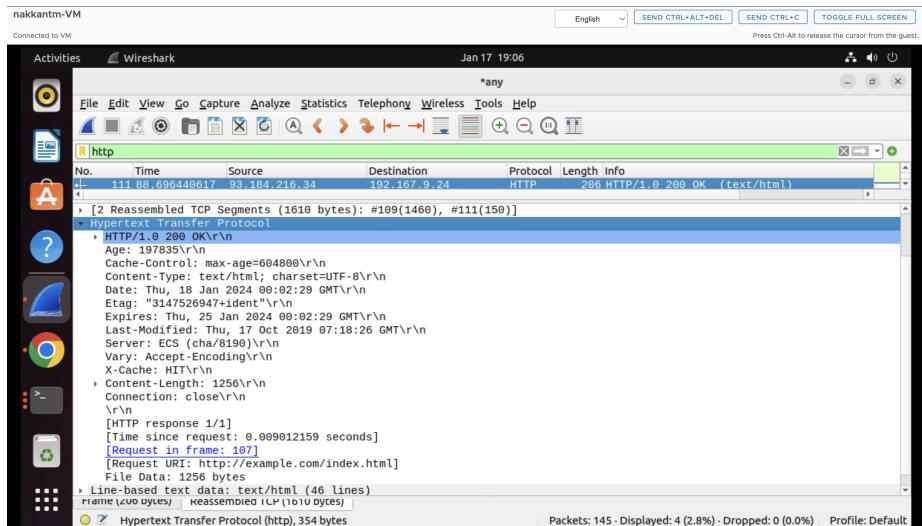


Figure 7: Telent response in wireshark

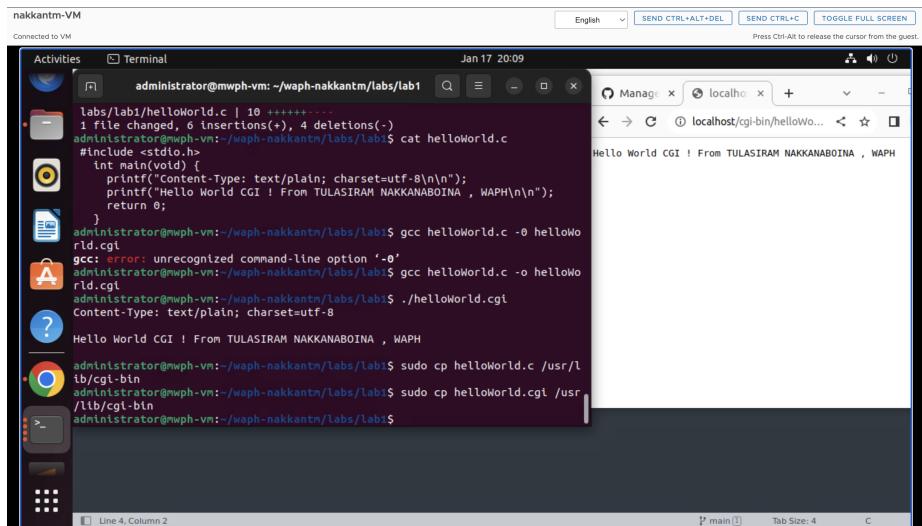


Figure 8: CGI program in C

B. Now , I have developed another CGI programming in C, this time incorporating a basic HTML template in the C code . This template has the course name as title, student name as Heading and other details as paragraph. This file was also compiled using gcc and copied to usr/lib/cgi-bin before accessing on the browser.

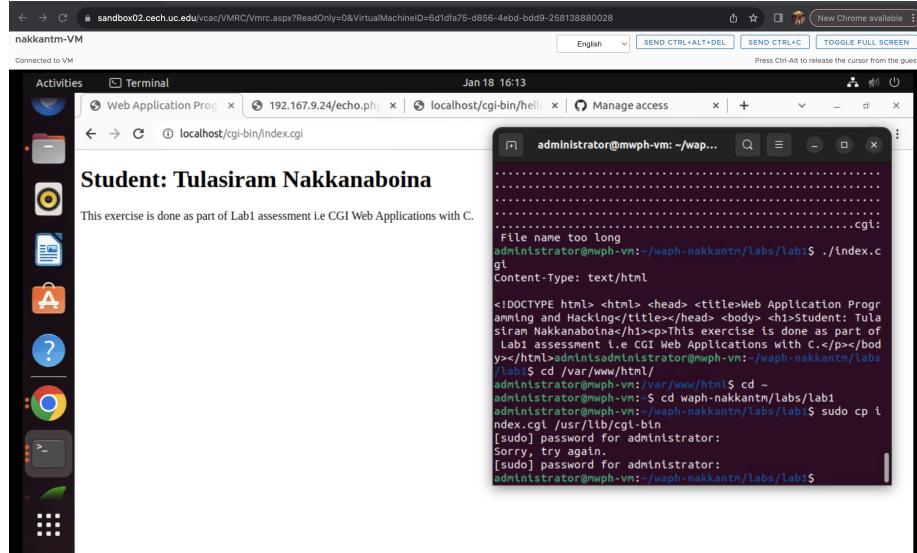


Figure 9: CGI in C and HTML

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>Student: Tulasiram Nakkanaboina</h1><p>This exercise is done as part of Lab1 assessment i.e CGI Web Applications with C.</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. As part of this task , a PHP web application has been developed with basic syntax which includes my name and PHP version , deployed it to the root apache2 var/www/html directory . It was then accessed on the browser with the url IP address/helloworld.php

Included file `helloworld.php`:

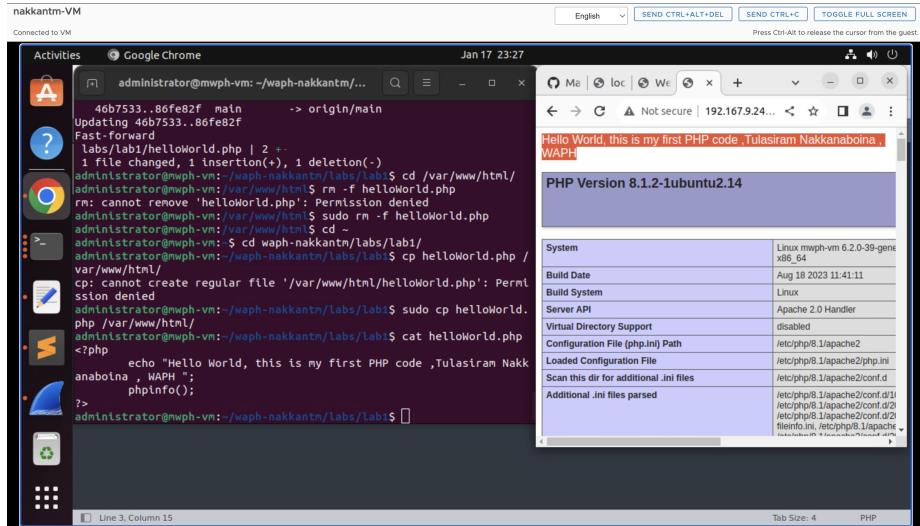


Figure 10: helloworld.php

```
<?php
    echo "Hello World! This is my first PHP program, Tulasiram Nakkanaboina , WAPH";
?>
```

B. A simple echo web application in PHP has been developed which prints the path variable passed through the http request. using `$_REQUEST['data']` in PHP for capturing the path variables in GET and POST requests possess various security vulnerabilities such as data tampering, SQL injections and Remote Code Execution. By implementing input validation , prepared statements for SQL inputs and sanitizing the usser inputs can mitigate these risks.

Included file `echo.php`:

```
<?php
    $inputData = $_REQUEST["data"];
    echo "The input from the request is <strong>" . $inputData . "</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 varaiable was then dispalyed as part of the response. This request, response and HTTP stream were analyzed through wireshark.

B.Client URL of CUrl is a command line tool for processing data using various n/w protocols. I have used CURL in terminal to make a post request to echo.php.

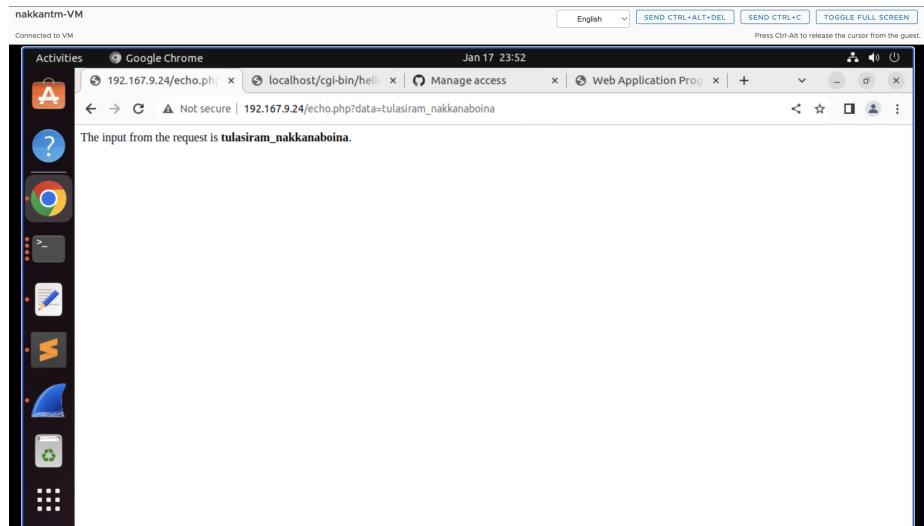


Figure 11: echo.php

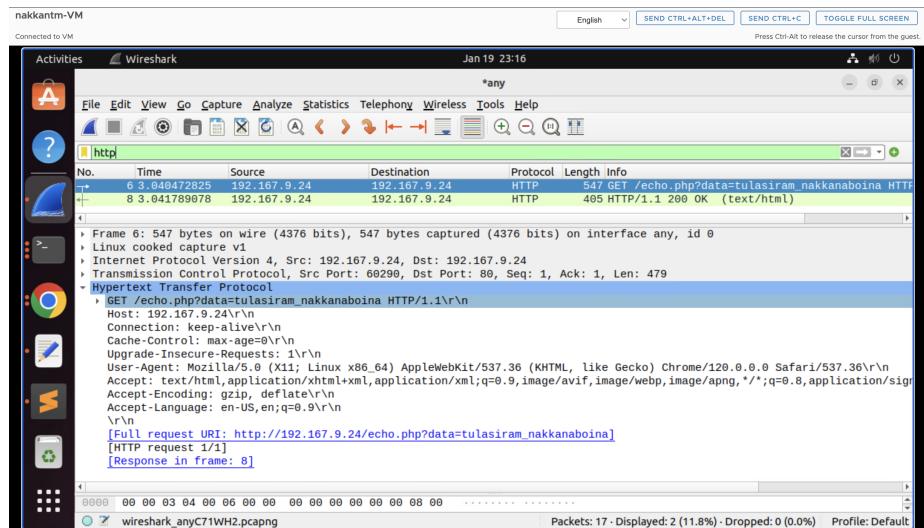


Figure 12: HTTP GET request in Wireshark

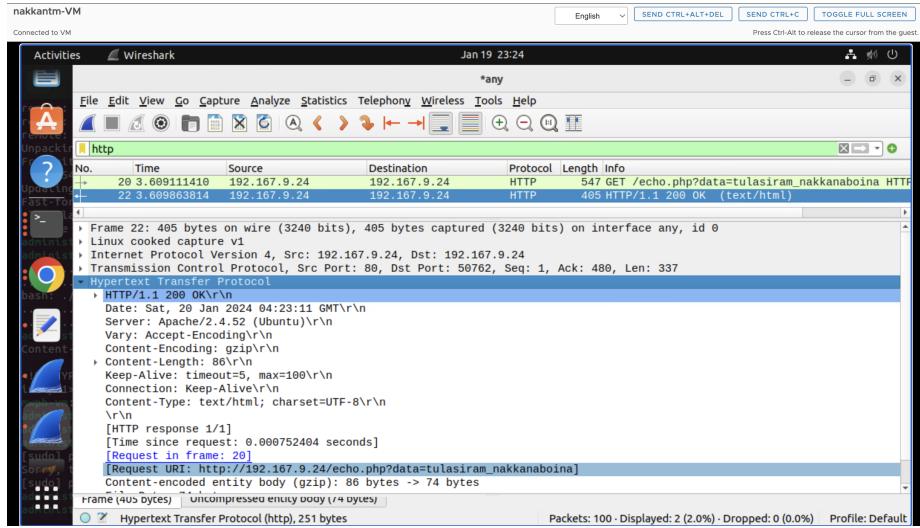


Figure 13: HTTP response in Wireshark

```
curl -X POST localhost/echo.php -d "data=Tulasiram N"
```

C.The similarities and differences between HTTP GET/ POST requests and responses Similarities: Both are HTTP methods used to communicate between client and server. Both can be captured and analyzed using tools like Wireshark. Both have request headers (though the headers might differ). Both receive responses with status codes, headers Differences: In GET request data is sent in the URL whereas in the POST request data is sent in the HTTP body. POST is more secure as data is not exposed to normal users. GET is used for retrieving the information, whereas POST is used to update the information .

as the echo.php web application is a mirror application , i.e just printing the received input , the responses in both the HTTP GET and HTTP POST are identical.

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

```

remote: Compressing objects: 100% (10/10), done.
remote: Total 10 (delta 6), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), 2.15 KB / 733.00 KB/s, done.
From github.com:nakkantm-uc/waph-nakkantm
   bd754d1..bf56b9 > origin/main
Up-to-date with bd754d1..bf56b9
Fast-forward
  labs/lab1/index.c | 15 ++++++-----+
   1 file changed, 9 insertions(+), 6 deletions(-)
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ gcc index.c -o index.cgi
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ ./index.cgi
Content-Type: text/html
<!DOCTYPE html> <html> <head> <title>Web Application Programming and Hacking</title></head> <body> <h1>Student: Tulasiram N<br/>Nakkanaboina</h1><p>This exercise is done as part of Lab1 assessment i.e CGI Web Applications with C.</p></body></html>
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ cd ~
administrator@mwpf-vn:~$ cd waph-nakkantm/labs/lab1
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ sudo cp index.cgi /usr/lib/cgi-bin
[sudo] password for administrator:
Sorry, try again.
[sudo] password for administrator:
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ curl -X POST http://localhost/echo.php -d "data=Tulasiram N"
administrator@mwpf-vn:~/waph-nakkantm/labs/lab1$ 
```

Figure 14: HTTP POST request using CURL

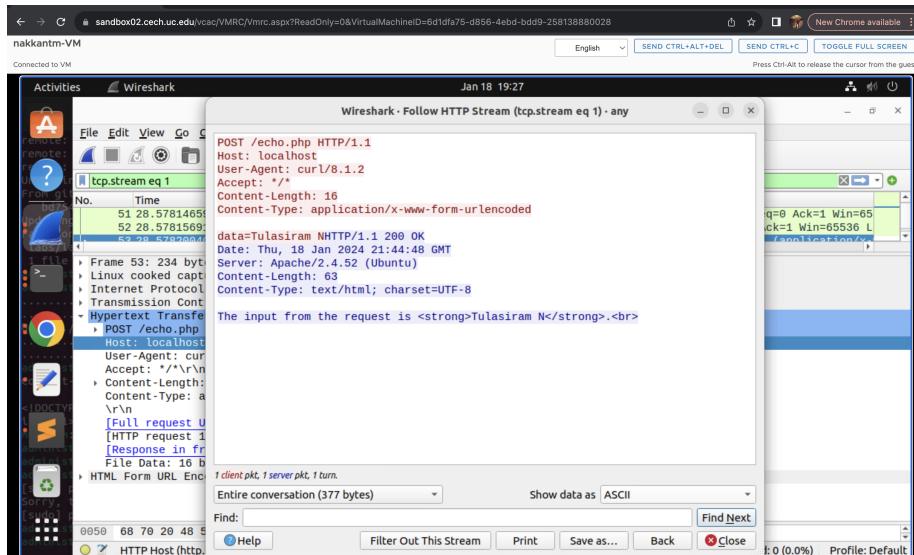


Figure 15: HTTP Stream in Wireshark