#WAPH -Web Application Programming And Hacking ##Instructor: Dr.Phu Phung ##Student **Name** : Sai Krishna Barupati **Email** : barupasa@mail.uc.edu

Repository Information Lab 1

** Repository URL ** : https://github.com/barupasa/waph-barupasa/tree/main/lab1

Lab1 : Foundations on web

Learnings: Through these tasks, I gained hands-on experience in web development and networking. I successfully created CGI web applications in C, starting with a basic "Hello World" program and later integrating another C CGI program with an HTML template. Moving on to PHP, I developed a simple "helloworld.php" page and an "echo.php" application, where I discussed potential security risks associated with the latter.

Furthermore, I delved into understanding HTTP GET and POST requests using Wireshark. I examined the HTTP GET Request and Response for the "echo.php" page, showcasing the practical application of Wireshark in analyzing web traffic. Additionally, I used curl to create an HTTP POST request, capturing the outcome through a screenshot and comparing the differences between HTTP POST and GET requests and their respective responses.

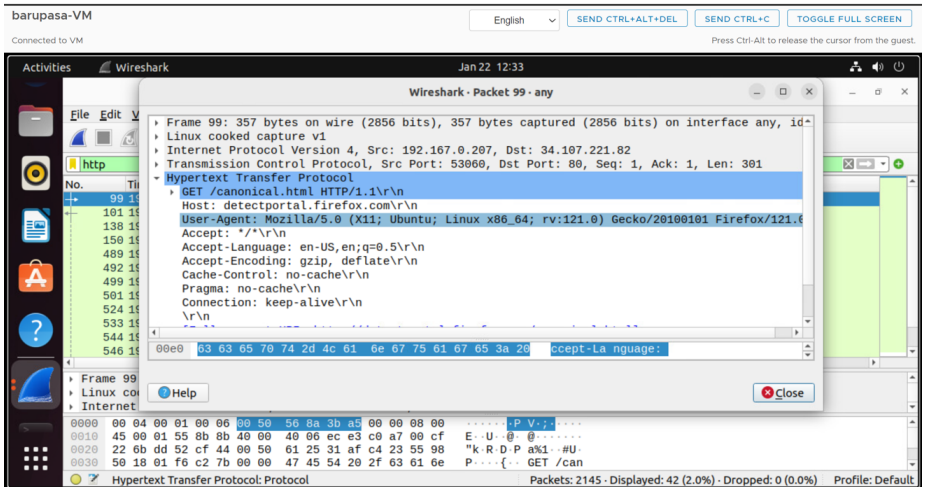
PART1: Web & HTTP Protocol

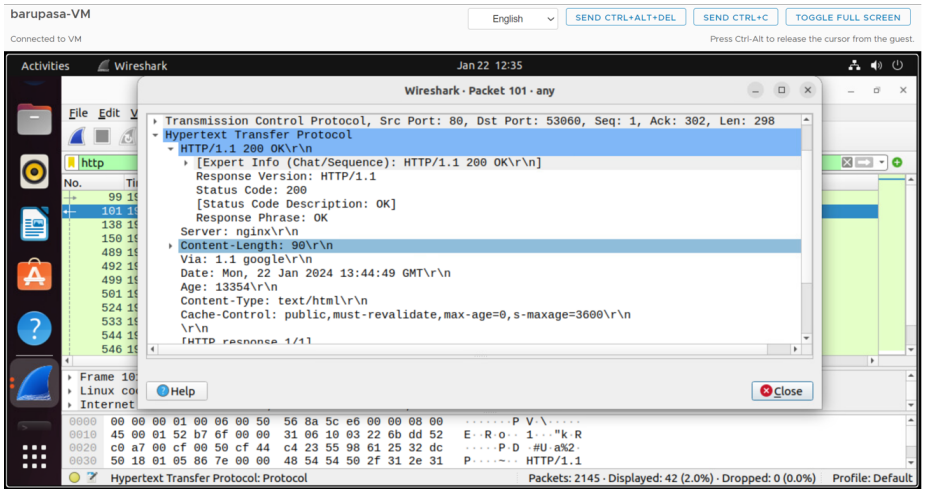
####Task 1: Installing and configuring wireshark ##Installation and configuration: I installed Wireshark by the following cammand in terminal , sudo apt install wireshark-qt and then I have opened wireshark application and started running it using : sudo wireshark and then i configured it to capture the data.

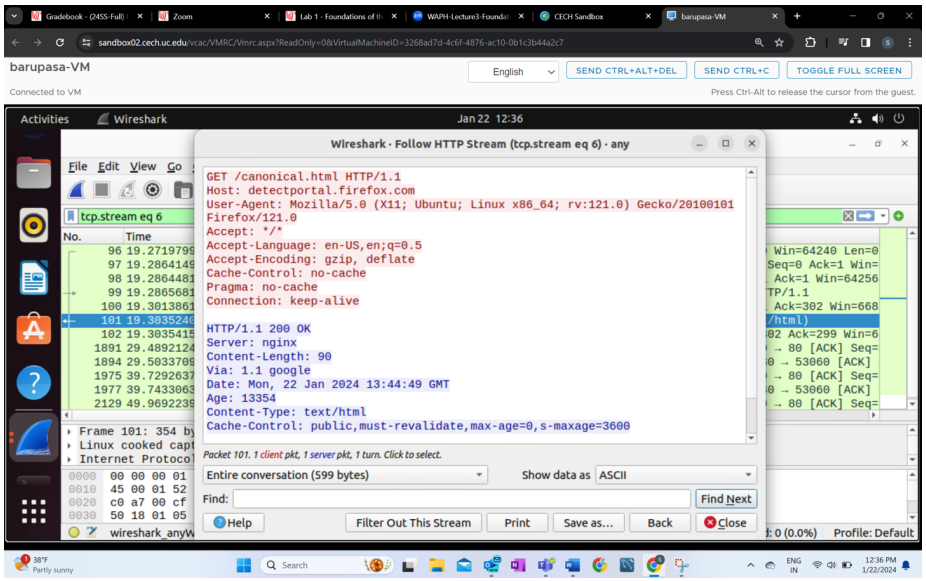
#Browsing: ,then using local browser (Mozilla) I gave a sample hit to a website. Then I have filtered the http request and responses and observed http stream as well .I have observed http stream as well from one of the http request.I have attached those screenshots below

HTTP Request, Response messages, and HTTP Stream

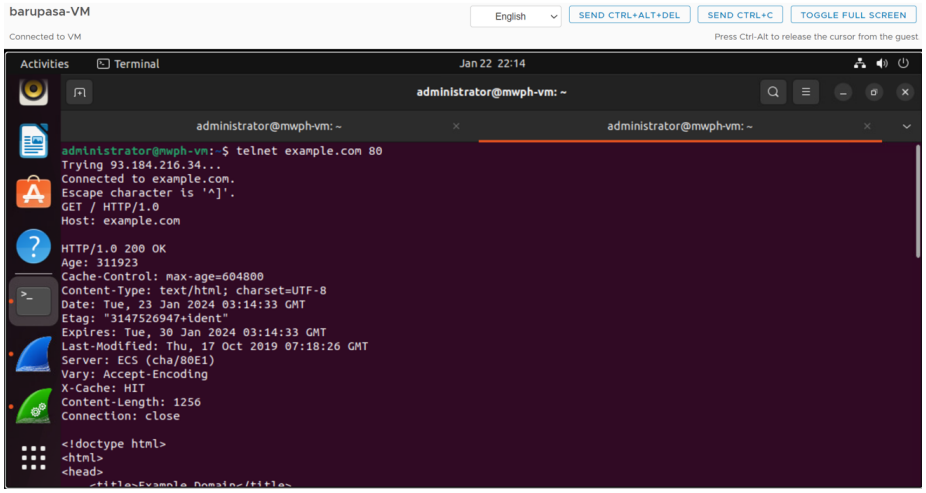
Requesst ,response and Stream:

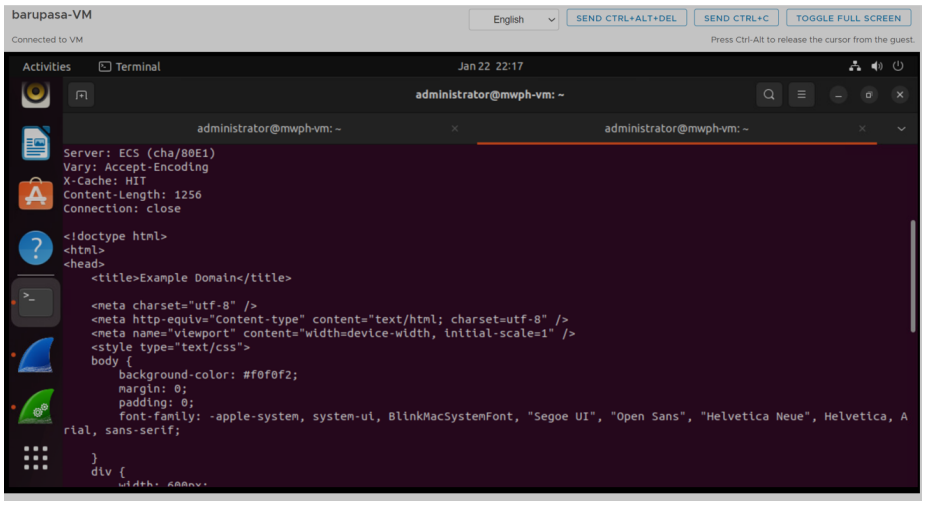

barupasa-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Wireshark
Jan 22 12:33
Wireshark · Packet 99 · any
Frame 99: 357 bytes on wire (2856 bits), 357 bytes captured (2856 bits) on interface any, id
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.167.0.207, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 53060, Dst Port: 80, Seq: 1, Ack: 1, Len: 301
Hypertext Transfer Protocol
GET /canonical.html HTTP/1.1\r\n
Host: detectportal.firefox.com\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.
Accept: */*\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Cache-Control: no-cache\r\n
Pragma: no-cache\r\n
Connection: keep-alive\r\n
\r\n
00e0 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 ccept-La nguage:
Help
Close
Hypertext Transfer Protocol: Protocol
Packets: 2145 · Displayed: 42 (2.0%) · Dropped: 0 (0.0%)
Profile: Default

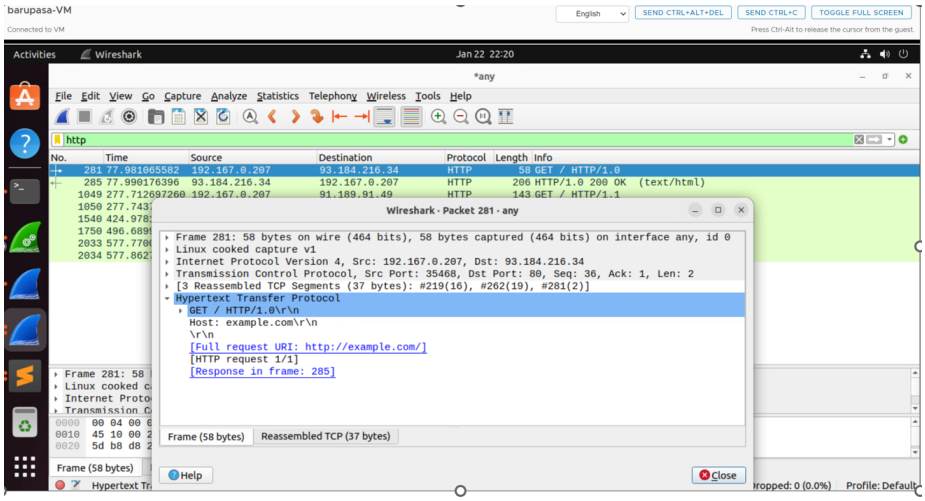

barupasa-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Wireshark
Jan 22 12:35
Wireshark · Packet 101 · any
Transmission Control Protocol, Src Port: 80, Dst Port: 53060, Seq: 1, Ack: 302, Len: 298
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Server: nginx\r\n
Content-Length: 90\r\n
Via: 1.1 google\r\n
Date: Mon, 22 Jan 2024 13:44:49 GMT\r\n
Age: 13354\r\n
Content-Type: text/html\r\n
Cache-Control: public,must-revalidate,max-age=0,s-maxage=3600\r\n
\r\n
Help
Close
Hypertext Transfer Protocol: Protocol
Packets: 2145 · Displayed: 42 (2.0%) · Dropped: 0 (0.0%)
Profile: Default

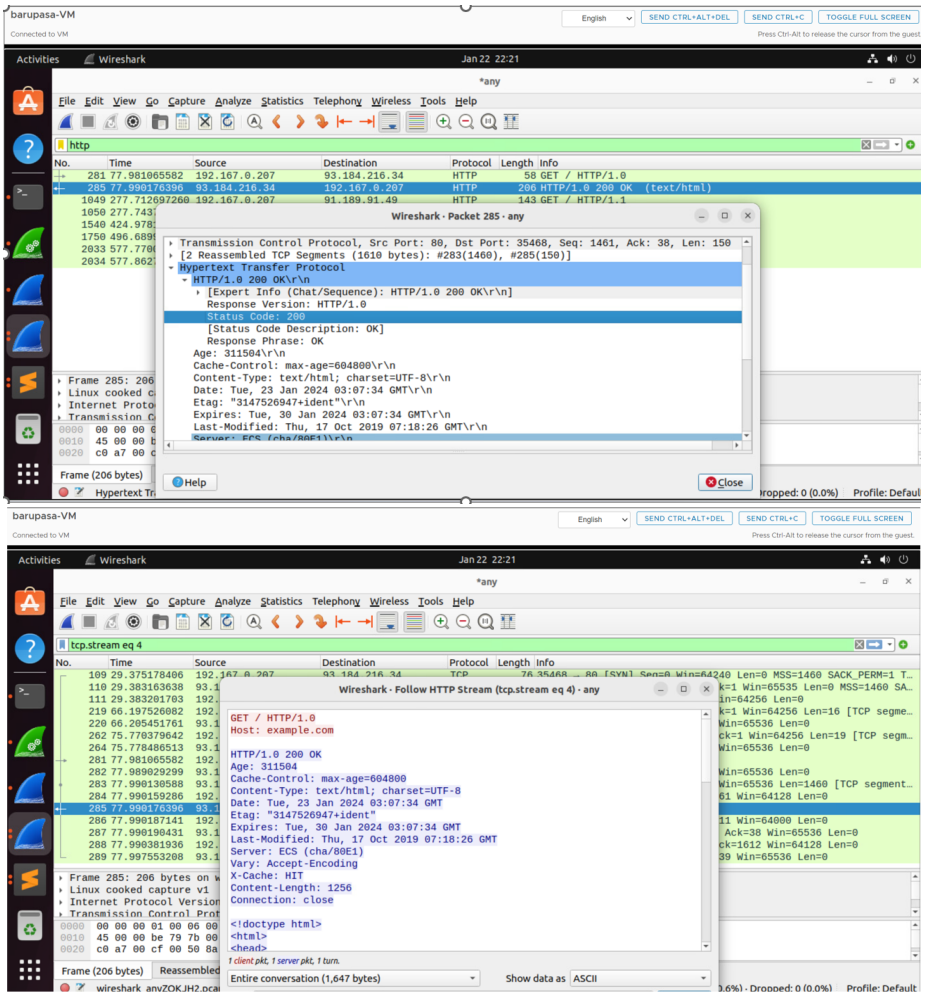


##TASK 2: Telnet connection and Analysis Here we used Telnet for HTTP examinations, open the terminal and enter the command "$telnet example.com 80." Then, use the command "GET /index.html HTTP/1.0" ,then "Host: example.com" and press double we get the HTTP request and response in the command prompt. I performed the same procedure while running the wireshark ,and ihave observed Telnet provides basic information like the request with GET, HTTP version, and host, while Wireshark offers more detailed insights like expert info, severity level, group, request method, URI, version, response version, status code, response phrase, content type. In summary, Wireshark provides a comprehensive breakdown of each line in the HTTP request and response.I have attached the screenshots for each step below.

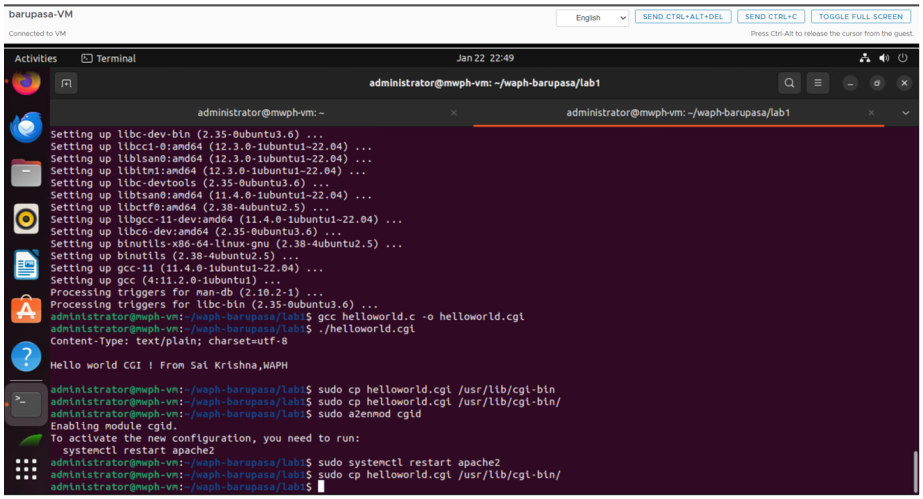



barupasa-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Terminal
Jan 22 22:17
administrator@mwph-vm: ~
administrator@mwph-vm: ~
administrator@mwph-vm: ~
Server: ECS (cha/80E1)
Vary: Accept-Encoding
X-Cache: HIT
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;
margin: 0;
padding: 0;
font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, A
rial, sans-serif;
}
div {


barupasa-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Wireshark
Jan 22 22:20
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
281 77.981065582 192.167.0.207 93.184.216.34 HTTP 58 GET / HTTP/1.0
285 77.990176396 93.184.216.34 192.167.0.207 HTTP 206 HTTP/1.0 200 OK (text/html)
1049 277.712697260 192.167.0.207 91.189.91.49 HTTP 143 GET / HTTP/1.1
Wireshark · Packet 281 · any
Frame 281: 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: 192.167.0.207, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 35468, Dst Port: 80, Seq: 36, Ack: 1, Len: 2
[3 Reassembled TCP Segments (37 bytes): #219(16), #262(19), #281(2)]
Hypertext Transfer Protocol
GET / HTTP/1.0\r\n
Host: example.com\r\n
\r\n
[Full request URI: http://example.com/]
[HTTP request 1/1]
[Response in frame: 285]
Frame (58 bytes)
Reassembled TCP (37 bytes)
Help
Close
Profile: Default



###PART II : Basic Web app programming ##Task 1: a)To deploy a CGI C program on a web server, start by using the Sublime text editor to write your code using the command "$ subl helloworld.c." After writing the code(I have attached the code snippet below), compile it with the command "gcc helloworld.c -o helloworld.cgi." Once the code is compiled, enable the CGI module in Apache by using the commands "$ sudo a2enmod cgid" and restart Apache using "$ sudo systemctl restart apache2."we need to create a dedicated folder to store your CGI program. To see the output, open a web browser and access "localhost/cgi-bin/helloworld.cgi." This indicateds that CGI C program is successfully deployed and running on the web server.



b) HTML template: Sample:

```
#include<stdio.h>
Int main(void)
{
Printf("Content-type: text/plain; charset=utf-8\n\n");
Printf("Hello world CGI ,From Sai Krishna,WAPH\n\n");
Return 0;
}
```

Below is the code I have used and i have attched screenshots Helloworld.c program :

```
#include<stdio.h>

int main(void) {

printf("content-Type : text/html ; charset= utf-8\n\n");

printf("<!DOCTYPE html>\n");

printf( "<html>\n");

printf(" <head>\n");

printf("<title> WAPH-SAI KRISHNA </title>\n");

printf("<head> \n");

printf("<body>\n");
```

```
printf(" <h1> WAPH -web application programming and hacking </h1>\n");

printf(" <p> Sai Krishna\n");

printf(" <p> I have basic knowledge in web development \n");

printf(" <body> \n");

printf(" <html> \n");

return 0;
```

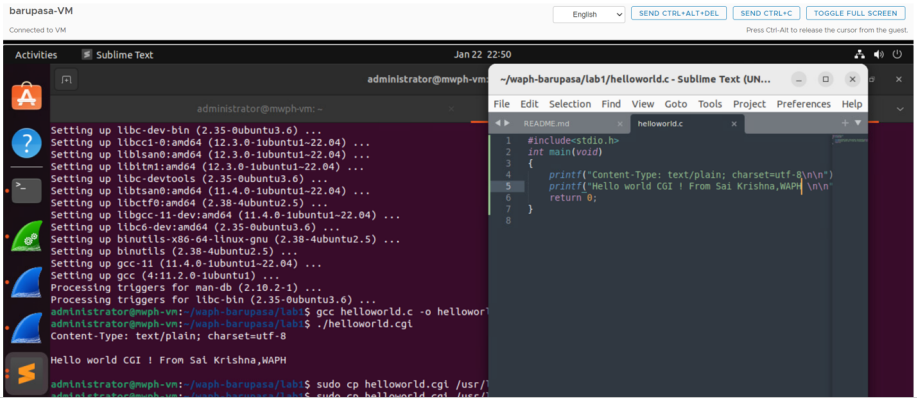


Figure 1: cs12

##Task2: a) Following the instructions in Lecture 3 on Sublime, I have successfully created a basic helloworld.php PHP page with my name and PHP configuration.I wrote the code after using the command subl helloworld.php to accomplish that.

b) I developed a simple "helloworld.php" file in Sublime and deployed it on the web server b
`![cs16](images/CS16.png)`

###Task 3: a)I used Wireshark to inspect the requests and responses of the "echo.php" program. To do this, I selected the "any network" option in Wireshark before starting the server. Then, I applied an HTTP filter to capture and examine the requests and responses successfully. I've included screenshots below for reference.

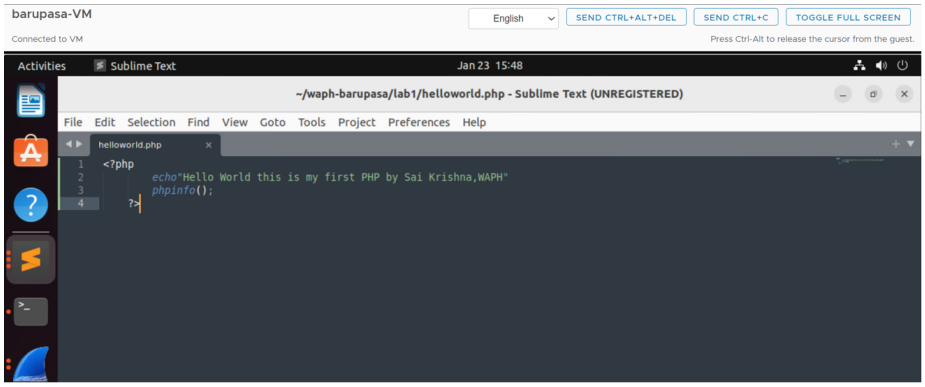


Figure 2: cs11

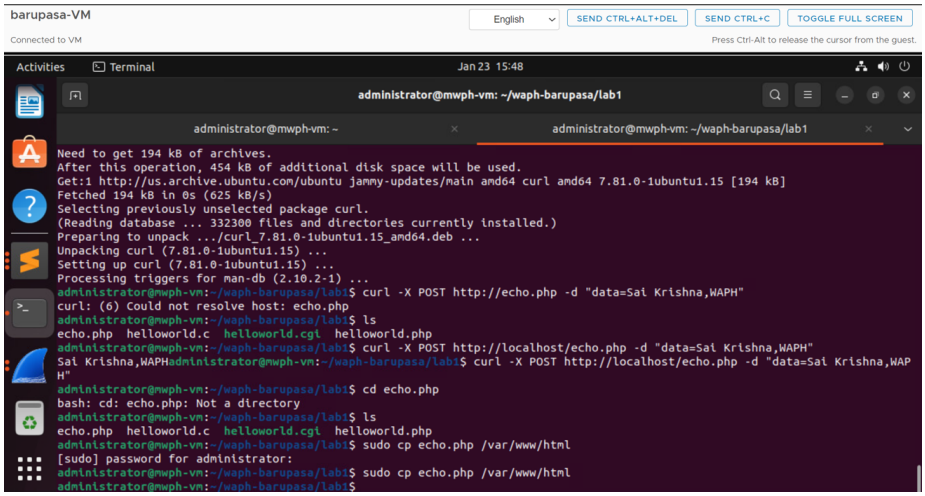
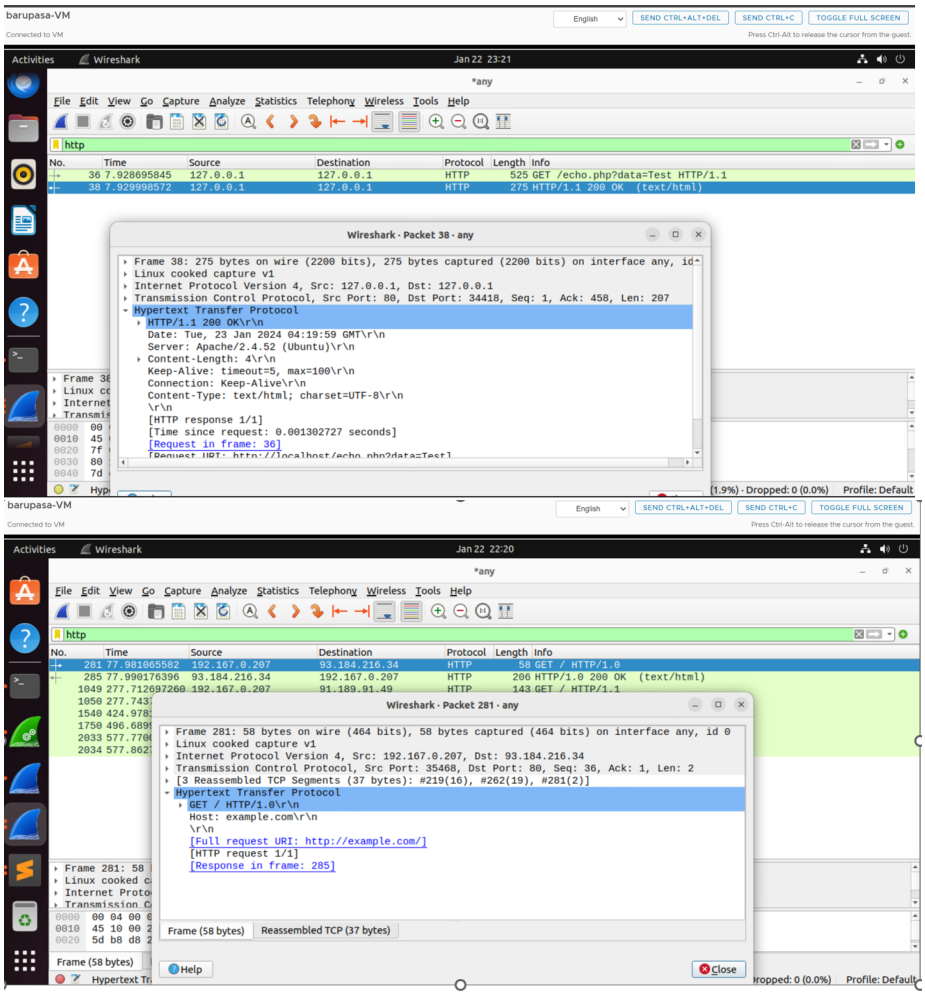
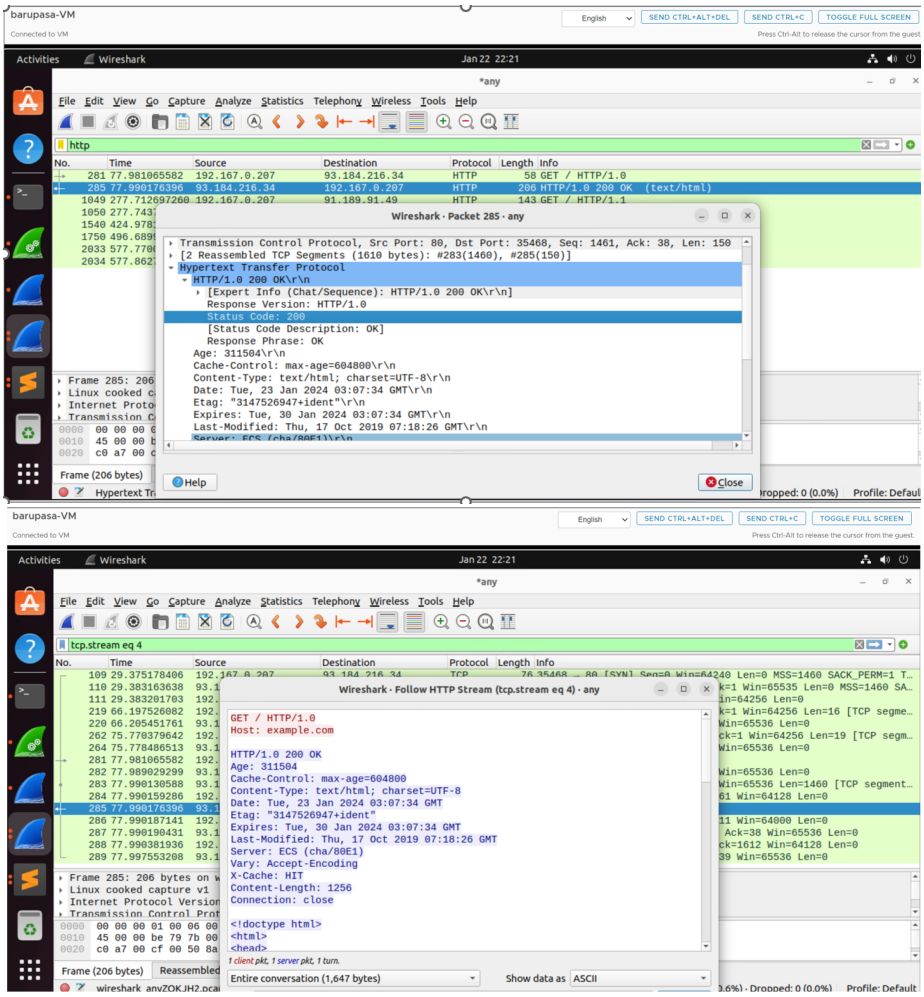
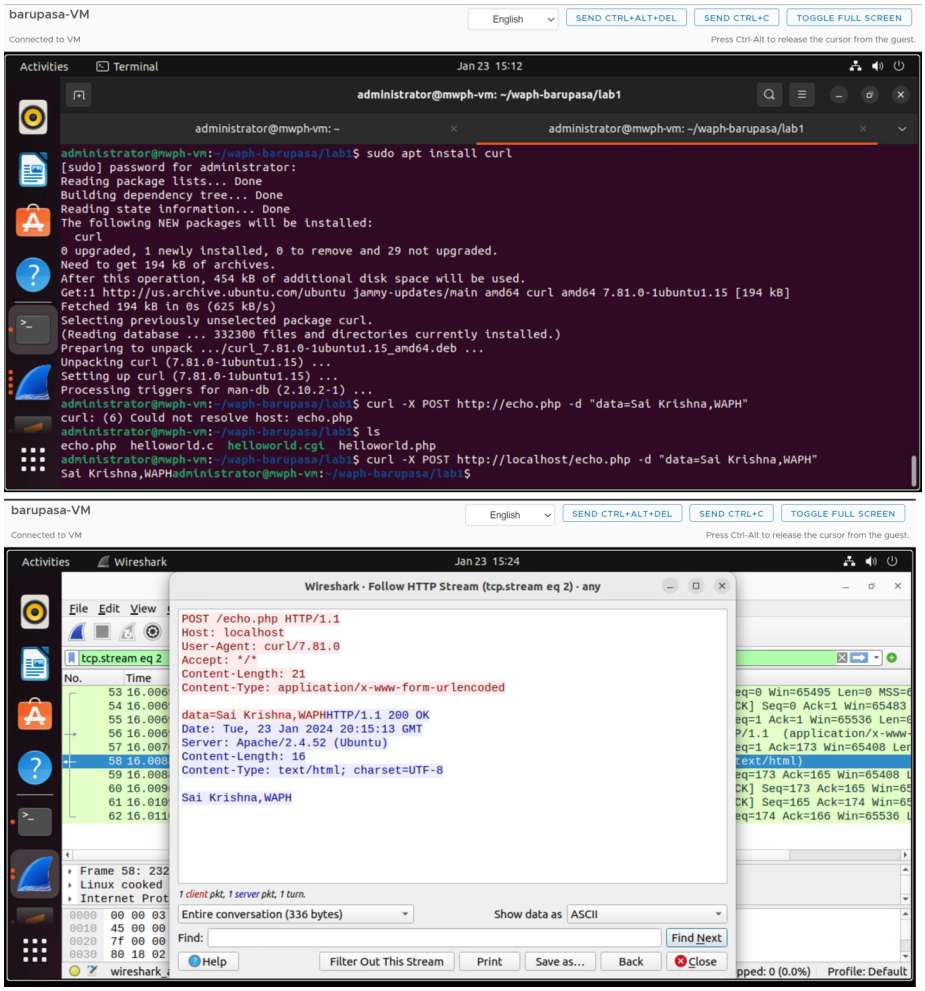


Figure 3: cs13


barupasa-VM
Activities
Wireshark
Jan 22 23:21
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
36 7.928695845 127.0.0.1 127.0.0.1 HTTP 525 GET /echo.php?data=Test HTTP/1.1
38 7.929998572 127.0.0.1 127.0.0.1 HTTP 275 HTTP/1.1 200 OK (text/html)
Wireshark · Packet 38 · any
Frame 38: 275 bytes on wire (2200 bits), 275 bytes captured (2200 bits) on interface any, id
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 34418, Seq: 1, Ack: 458, Len: 207
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Tue, 23 Jan 2024 04:19:59 GMT\r\n
Server: Apache/2.4.52 (Ubuntu)\r\n
Content-Length: 4\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=UTF-8\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.001302727 seconds]
[Request in frame: 36]
(1.9%) · Dropped: 0 (0.0%) Profile: Default
barupasa-VM
Activities
Wireshark
Jan 22 22:20
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
281 77.981065582 192.167.0.207 93.184.216.34 HTTP 58 GET / HTTP/1.0
285 77.990176396 93.184.216.34 192.167.0.207 HTTP 206 HTTP/1.0 200 OK (text/html)
1049 277.712697260 192.167.0.207 91.189.91.49 HTTP 143 GET / HTTP/1.1
Wireshark · Packet 281 · any
Frame 281: 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: 192.167.0.207, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 35468, Dst Port: 80, Seq: 36, Ack: 1, Len: 2
[3 Reassembled TCP Segments (37 bytes): #219(16), #262(19), #281(2)]
Hypertext Transfer Protocol
GET / HTTP/1.0\r\n
Host: example.com\r\n
\r\n
[Full request URI: http://example.com/]
[HTTP request 1/1]
[Response in frame: 285]
Frame (58 bytes)
Reassembled TCP (37 bytes)
Help
Close
Frame (58 bytes)
Profile: Default



b) First, I installed the Curl application on Ubuntu using the command "sudo apt install curl." Afterward, as shown in the command prompt screenshot below, I executed a command before starting Wireshark. Then, I right-clicked on the HTTP request followed by the HTTP stream in Wireshark. This allowed me to clearly see the HTTP request and response of the application using Curl. I've included a screenshot for better understanding.



c)

HTTP POST Requests and Responses share similarities as they both follow a cycle of making a request and receiving a response. They also allow the inclusion of headers in the request to provide extra information. However, there are several differences between them. For example, the GET method's data is visible in the browser's address bar, while the POST method's data is not. GET is suitable for smaller amounts of data compared to POST. Additionally, GET is used for retrieval, where parameters like names are provided in the URL, POST method is used to submit data, which is sent in request body of the HTTP request..