WAPH-Web Application Programming and Hack-ing

Instructor: Dr. Phu Phung Student

Name: Bheemreddy Vikhyath Reddy

Email: bheemrvy@mail.uc.edu

Figure 1: Vikhyath’s headshot

Repository Information

Respository's URL: https://github.com/Vikhyath-Reddy/waph-bheemrvy

This is a private repository for Divakar to store all code from the course. The organization of this repository is as follows.s

The lab's overview

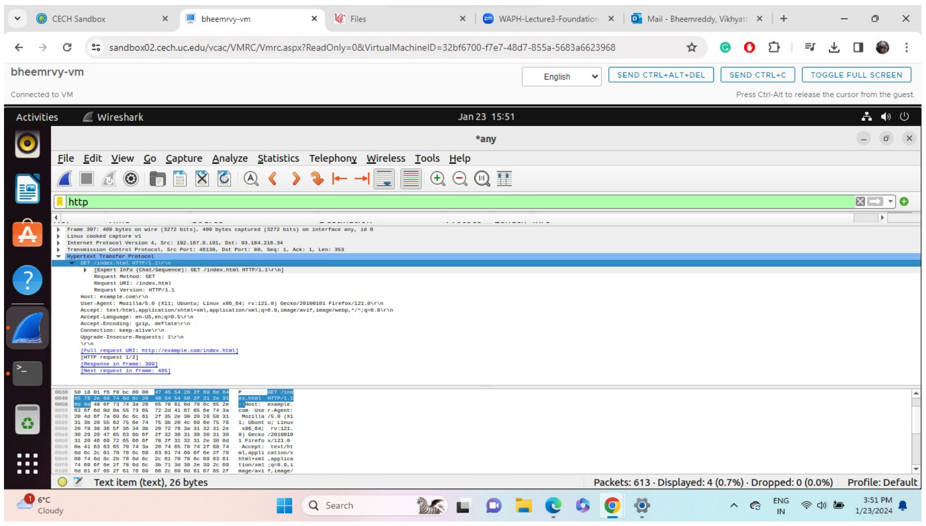
In Lab 1, we will go over the HTTP protocol and how to program web applications. The first section, Part I, walks us through the complexities of HTTP traffic analysis with Wireshark and hands-on contact with HTTP servers via Telnet. The primary goal is to understand the HTTP protocol's core components, methods, and replies. Moving on to Part II, we'll look at web application programming and the Common Gateway Interface. Here, we construct a simple web application in C that handles user input. The lab also covers web application programming using PHP, giving light on the differences between HTTP GET and POST requests, as well as the security implications. This comprehensive lab experience is intended to provide a practical grasp of HTTP, CGI, and the principles of web application development.

Here is the link to the Github repository https://github.com/Vikhyath-Reddy/waph-bheemrvy/tree/main/labs/Lab1.

Part I - The WEB and HTTP Protocol

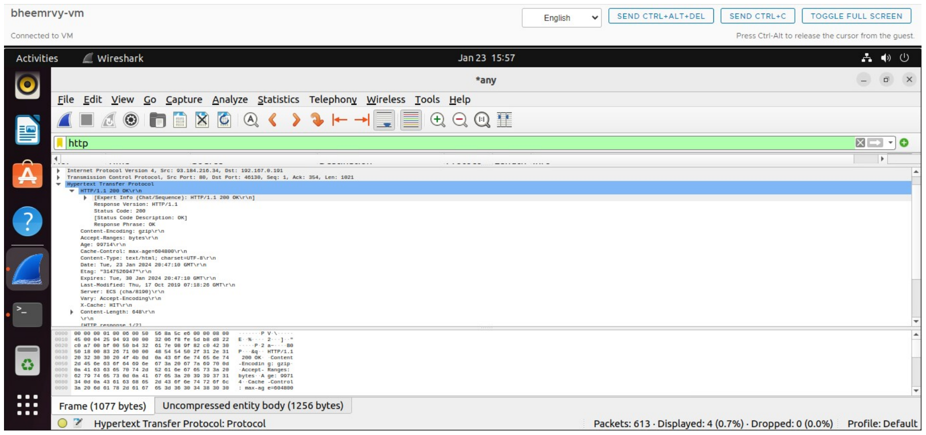
Task 1 - Familiar with the Wireshark tool and HTTP protocol

Wireshark is a powerful tool for in-depth HTTP protocol analysis, providing a thorough knowledge of client-server interactions. To fully utilize Wireshark, begin by selecting the appropriate network interface that will handle the anticipated HTTP traffic. After starting the capture, go to the display filter and type "http" to narrow the emphasis to HTTP-related packets.Examine the collected packets to untangle the complexities of the request-response cycle, paying close attention to headers, methods, and content to gain a complete understanding of the communication dynamics.



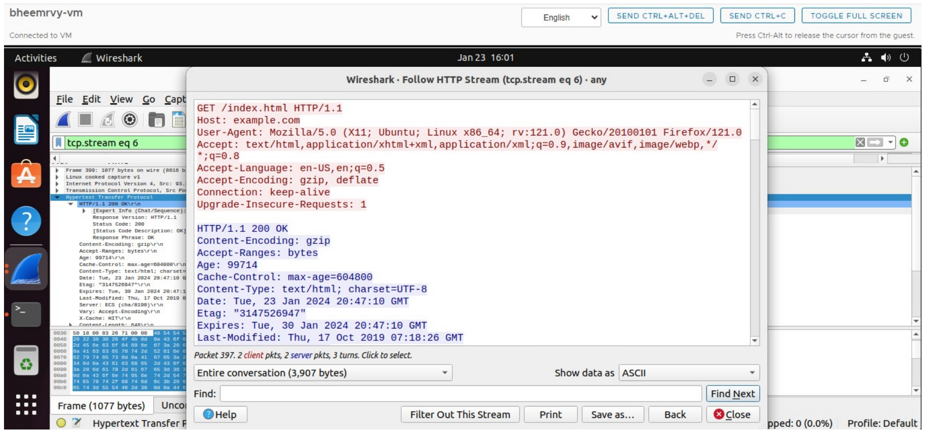
```
Frame 397: 409 bytes on wire (3272 bits), 409 bytes captured (3272 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.167.0.191, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 46130, Dst Port: 80, Seq: 1, Ack: 1, Len: 353
Hypertext Transfer Protocol
  GET /index.html HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /index.html HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /index.html
    Request Version: HTTP/1.1
  Host: example.com\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
  [Full request URI: http://example.com/index.html]
  [HTTP request 1/2]
  [Response in frame: 399]
  [Next request in frame: 465]

0030  50 18 01 f6 f8 bc 00 00  47 45 54 20 2f 69 6e 64   P·······GET /ind
0040  65 78 2e 68 74 6d 6c 20  48 54 54 50 2f 31 2e 31   ex.html HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 20  65 78 61 6d 70 6c 65 2e   ··Host: example.
0060  63 6f 6d 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   com··Use r-Agent:
0070  20 4d 6f 7a 69 6c 6c 61  2f 35 2e 30 20 28 58 31    Mozilla /5.0 (X1
0080  31 3b 20 55 62 75 6e 74  75 3b 20 4c 69 6e 75 78   1; Ubunt u; Linux
0090  20 78 38 36 5f 36 34 3b  20 72 76 3a 31 32 31 2e    x86_64;  rv:121.
00a0  30 29 20 47 65 63 6b 6f  2f 32 30 31 30 30 31 30   0) Gecko /2010010
00b0  31 20 46 69 72 65 66 6f  78 2f 31 32 31 2e 30 0d   1 Firefo x/121.0·
00c0  0a 41 63 63 65 70 74 3a  20 74 65 78 74 2f 68 74   ·Accept:  text/ht
00d0  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli cation/x
00e0  68 74 6d 6c 2b 78 6d 6c  2c 61 70 70 6c 69 63 61   html+xml ,applica
00f0  74 69 6f 6e 2f 78 6d 6c  3b 71 3d 30 2e 39 2c 69   tion/xml ;q=0.9,i
0100  6d 61 67 65 2f 61 76 69  66 2c 69 6d 61 67 65 2f   mage/avi f,image/

Text item (text), 26 bytes          Packets: 613 · Displayed: 4 (0.7%) · Dropped: 0 (0.0%)    Profile: Default
```

Figure 2: example.com_Request



```
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.167.0.191
Transmission Control Protocol, Src Port: 80, Dst Port: 46130, Seq: 1, Ack: 354, Len: 1021
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
  Content-Encoding: gzip\r\n
  Accept-Ranges: bytes\r\n
  Age: 99714\r\n
  Cache-Control: max-age=604800\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  Date: Tue, 23 Jan 2024 20:47:10 GMT\r\n
  Etag: "3147526947"\r\n
  Expires: Tue, 30 Jan 2024 20:47:10 GMT\r\n
  Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT\r\n
  Server: ECS (cha/8190)\r\n
  Vary: Accept-Encoding\r\n
  X-Cache: HIT\r\n
  Content-Length: 648\r\n
  \r\n
  [HTTP response 1/2]

0000  00 00 00 01 00 06 00 50  56 8a 5c e6 00 00 08 00   ·······P V·\·····
0010  45 00 04 25 94 93 00 00  32 06 f8 fe 5d b8 d8 22   E··%···· 2···]··"
0020  c0 a7 00 bf 00 50 b4 32  61 7e 98 9f 82 c0 42 30   ·····P·2 a~····B0
0030  50 18 00 83 26 71 00 00  48 54 54 50 2f 31 2e 31   P···&q·· HTTP/1.1
0040  20 32 30 30 20 4f 4b 0d  0a 43 6f 6e 74 65 6e 74    200 OK· ·Content
0050  2d 45 6e 63 6f 64 69 6e  67 3a 20 67 7a 69 70 0d   -Encodin g: gzip·
0060  0a 41 63 63 65 70 74 2d  52 61 6e 67 65 73 3a 20   ·Accept- Ranges: 
0070  62 79 74 65 73 0d 0a 41  67 65 3a 20 39 39 37 31   bytes··A ge: 9971
0080  34 0d 0a 43 61 63 68 65  2d 43 6f 6e 74 72 6f 6c   4··Cache -Control
0090  3a 20 6d 61 78 2d 61 67  65 3d 36 30 34 38 30 30   : max-ag e=604800

Frame (1077 bytes)    Uncompressed entity body (1256 bytes)
Hypertext Transfer Protocol: Protocol          Packets: 613 · Displayed: 4 (0.7%) · Dropped: 0 (0.0%)    Profile: Default
```

Figure 3: example.com_Response



```
GET /index.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Content-Encoding: gzip
Accept-Ranges: bytes
Age: 99714
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 23 Jan 2024 20:47:10 GMT
Etag: "3147526947"
Expires: Tue, 30 Jan 2024 20:47:10 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
```

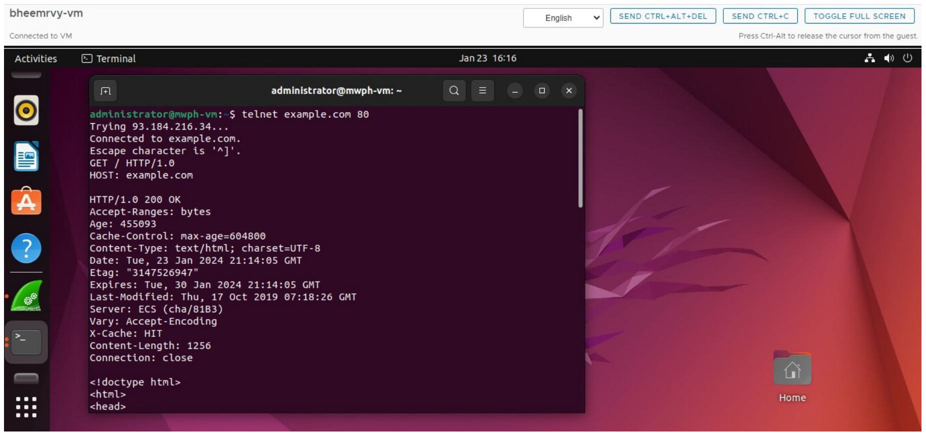
Packet 397. 2 client pkts, 2 server pkts, 3 turns. Click to select.

Entire conversation (3,907 bytes) · Show data as: ASCII

Figure 4: example.com_HTTP_Stream

Task 2 - Understanding HTTP using telnet and Wireshark

Using the telnet software to send a simple HTTP request entails starting a terminal connection and connecting to the specified server on port 80. Once the connection has been established, manually create a basic HTTP request by giving necessary components such as the HTTP method (e.g.,GET), the resource path, and the HTTP version. Finally, hit the “Enter” key twice to send the request. To examine the complexities of HTTP communications using Wireshark, start a capture on the appropriate network interface. Use a display filter for “http” to narrow the focus on HTTP traffic. The Wireshark output will then clearly display the details of the sent HTTP request, including headers and data for further inspection.



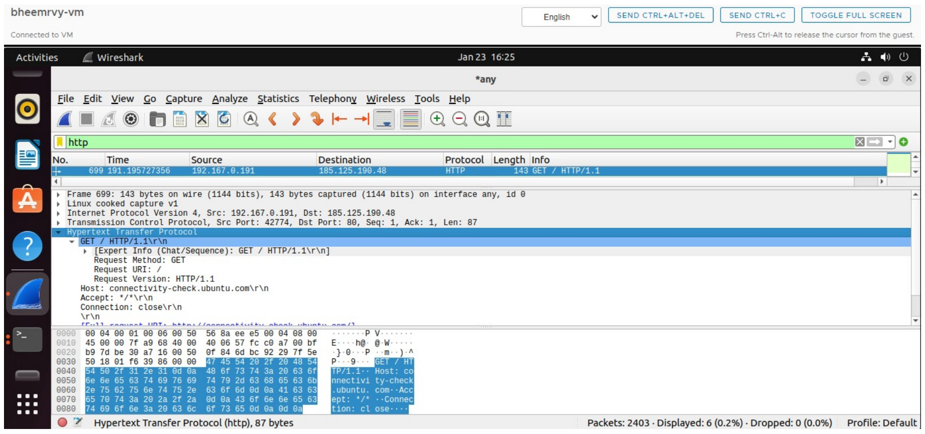
```
administrator@mwph-vm:~$ telnet example.com 80
Trying 93.184.216.34...
Connected to example.com.
Escape character is '^]'.
GET / HTTP/1.0
HOST: example.com

HTTP/1.0 200 OK
Accept-Ranges: bytes
Age: 455093
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 23 Jan 2024 21:14:05 GMT
Etag: "3147526947"
Expires: Tue, 30 Jan 2024 21:14:05 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (cha/81B3)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close

<!doctype html>
<html>
<head>
```

Figure 5: Sending_Request_by_using_Telnet_in_terminal

Part II - Basic Application Programming

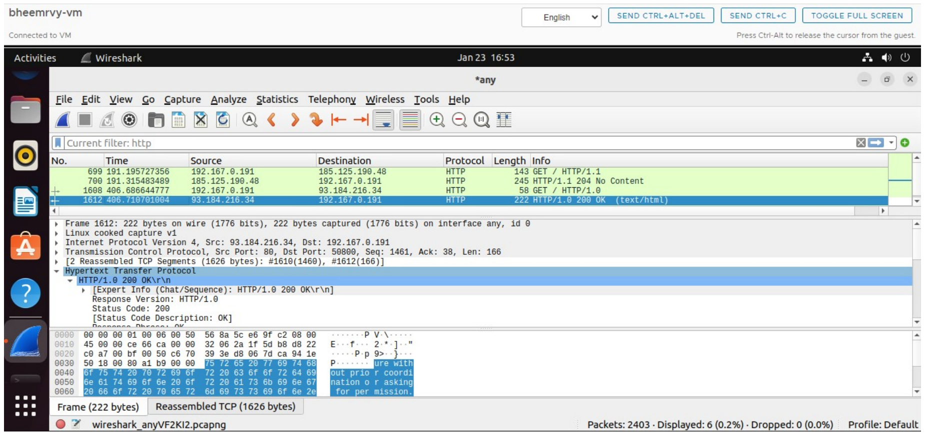
Task 1 - CGI Web applications in C

To develop a Hello World CGI application in C, first create a C source code file named "helloworld.c" that includes the necessary CGI headers. Include the HTTP Content-Type header in this file first, then the HTML content. Next, use a C compiler, such as GCC, to build the C program and create an executable file called "helloworld.cgi." Once compiled, move the executable to the web server's CGI directory ("/usr/lib/cgi-bin/"). To allow CGI execution, set the CGI directory in the web server's settings. Use a web browser to navigate to the relevant URL (for example, http://localhost/cgi-bin/helloworld.cgi). The web server will execute the CGI application, and the browser will show the produced output. This simplified approach demonstrates how to compile and install a basic CGI application in C on a web server, allowing for dynamic content production.


bheemrvy-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 23 16:25
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

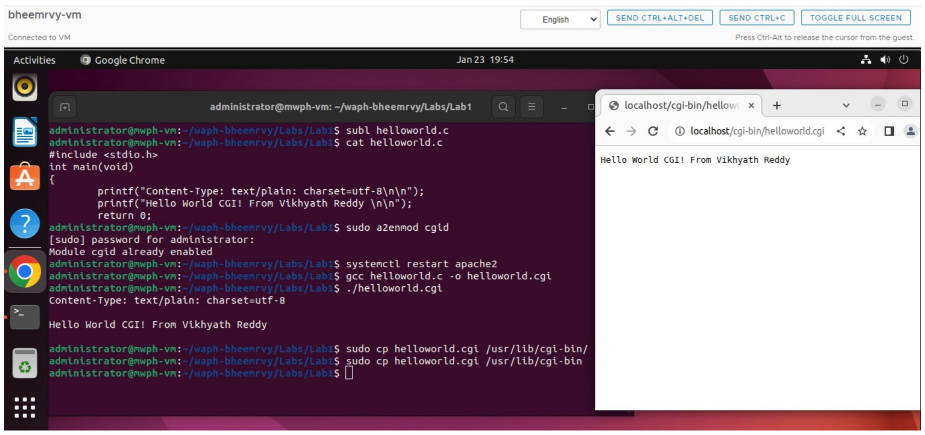
```
No.   Time            Source          Destination     Protocol Length Info
  699 191.195727356   192.167.0.191   185.125.190.48  HTTP        143 GET / HTTP/1.1

Frame 699: 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: 192.167.0.191, Dst: 185.125.190.48
Transmission Control Protocol, Src Port: 42774, Dst Port: 80, Seq: 1, Ack: 1, Len: 87
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: connectivity-check.ubuntu.com\r\n
  Accept: */*\r\n
  Connection: close\r\n
  \r\n

0000  00 04 00 01 00 06 00 50  56 8a ee e5 00 04 08 00   ·······P V·······
0010  45 00 00 7f a9 68 40 00  40 06 57 fc c0 a7 00 bf   E····h@· @·W·····
0020  b9 7d be 30 a7 16 00 50  0f 84 6d bc 92 29 7f 5e   ·}·0···P ··m··)·^
0030  50 18 01 f6 39 86 00 00  47 45 54 20 2f 20 48 54   P···9··· GET / HT
0040  54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a 20 63 6f   TP/1.1·· Host: co
0050  6e 6e 65 63 74 69 76 69  74 79 2d 63 68 65 63 6b   nnectivi ty-check
0060  2e 75 62 75 6e 74 75 2e  63 6f 6d 0d 0a 41 63 63   .ubuntu. com··Acc
0070  65 70 74 3a 20 2a 2f 2a  0d 0a 43 6f 6e 6e 65 63   ept: */* ··Connec
0080  74 69 6f 6e 3a 20 63 6c  6f 73 65 0d 0a 0d 0a      tion: cl ose····

Hypertext Transfer Protocol (http), 87 bytes     Packets: 2403 · Displayed: 6 (0.2%) · Dropped: 0 (0.0%)     Profile: Default
```

Figure 6: Telnet_Request_in_Wireshark



```
No.   Time            Source          Destination     Protocol Length Info
  699 191.195727356   192.167.0.191   185.125.190.48  HTTP        143 GET / HTTP/1.1
  700 191.315483489   185.125.190.48  192.167.0.191   HTTP        245 HTTP/1.1 204 No Content
 1608 406.686644777   192.167.0.191   93.184.216.34   HTTP         58 GET / HTTP/1.0
 1612 406.710701004   93.184.216.34   192.167.0.191   HTTP        222 HTTP/1.0 200 OK  (text/html)

Frame 1612: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.167.0.191
Transmission Control Protocol, Src Port: 80, Dst Port: 50800, Seq: 1461, Ack: 38, Len: 166
[2 Reassembled TCP Segments (1626 bytes): #1610(1460), #1612(166)]
Hypertext Transfer Protocol
  HTTP/1.0 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
    Response Version: HTTP/1.0
    Status Code: 200
    [Status Code Description: OK]

0000  00 00 00 01 00 06 00 50  56 8a 5c e6 9f c2 08 00   ·······P V·\·····
0010  45 00 00 ce 66 ca 00 00  32 06 2a 1f 5d b8 d8 22   E···f··· 2·*·]··"
0020  c0 a7 00 bf 00 50 c6 70  39 3e d8 06 7d ca 94 1e   ·····P·p 9>··}···
0030  50 18 00 80 a1 b9 00 00  75 72 65 20 77 69 74 68   P······· ure with
0040  6f 75 74 20 70 72 69 6f  72 20 63 6f 6f 72 64 69   out prio r coordi
0050  6e 61 74 69 6f 6e 20 6f  72 20 61 73 6b 69 6e 67   nation o r asking
0060  20 66 6f 72 20 70 65 72  6d 69 73 73 69 6f 6e 2e    for per mission.

Frame (222 bytes)   Reassembled TCP (1626 bytes)
wireshark_anyVF2KI2.pcapng     Packets: 2403 · Displayed: 6 (0.2%) · Dropped: 0 (0.0%)     Profile: Default
```

Figure 7: Telnet_Response_in_Wireshark

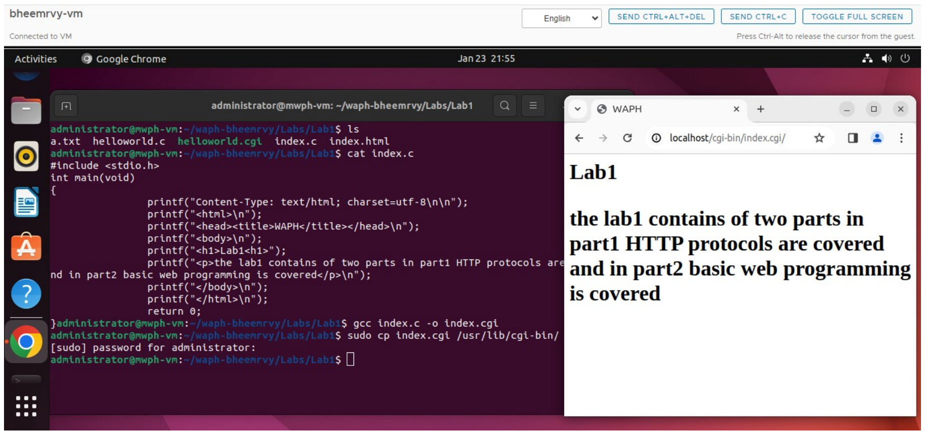

bheemrvy-vm
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Google Chrome
Jan 23 19:54
administrator@mwph-vm: ~/waph-bheemrvy/Labs/Lab1
localhost/cgi-bin/helloworld.cgi

```
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ subl helloworld.c
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ cat helloworld.c
#include <stdio.h>
int main(void)
{
        printf("Content-Type: text/plain: charset=utf-8\n\n");
        printf("Hello World CGI! From Vikhyath Reddy \n\n");
        return 0;
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo a2enmod cgid
[sudo] password for administrator:
Module cgid already enabled
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ systemctl restart apache2
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ gcc helloworld.c -o helloworld.cgi
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ ./helloworld.cgi
Content-Type: text/plain: charset=utf-8

Hello World CGI! From Vikhyath Reddy

administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo cp helloworld.cgi /usr/lib/cgi-bin/
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo cp helloworld.cgi /usr/lib/cgi-bin
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ 
```

```
Hello World CGI! From Vikhyath Reddy
```

Figure 8: Helloworld.c_CGI_Program

1. Created a “index.c” file to demonstrate competency in building C CGI applications,following a simple HTML template with proper title, header, and paragraph components.


bheemrvy-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
Google Chrome
Jan 23 21:55
administrator@mwph-vm: ~/waph-bheemrvy/Labs/Lab1
WAPH
localhost/cgi-bin/index.cgi/

```
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ ls
a.txt  helloworld.c  helloworld.cgi  index.c  index.html
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ cat index.c
#include <stdio.h>
int main(void)
{
                printf("Content-Type: text/html; charset=utf-8\n\n");
                printf("<html>\n");
                printf("<head><title>WAPH</title></head>\n");
                printf("<body>\n");
                printf("<h1>Lab1<h1>");
                printf("<p>the lab1 contains of two parts in part1 HTTP protocols are
nd in part2 basic web programming is covered</p>\n");
                printf("</body>\n");
                printf("</html>\n");
                return 0;
}administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ gcc index.c -o index.cgi
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo cp index.cgi /usr/lib/cgi-bin/
[sudo] password for administrator:
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ 
```

Lab1

the lab1 contains of two parts in part1 HTTP protocols are covered and in part2 basic web programming is covered

Figure 9: Index.c_CGI_Program

Included file index.c: c

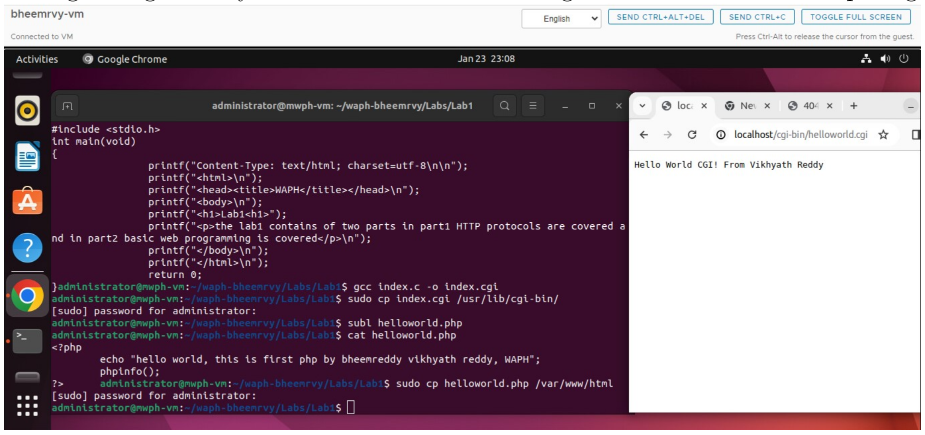
```
#include <stdio.h> int main(void){
printf("Content-Type: text/html; charset=utf-8\n\n");
printf("<html>\n");
printf("<head><title>WAPH</title></head>\n");
printf("<body>\n");
printf("<h1>Lab1</h1>\n");
printf("<p>The lab1 consists of two parts in part 1 HTTP protocols are covered and in part 2
printf("</body>\n");
printf("</html>\n"); return 0; }
```

Task 2 - A simple PHP Web Application with user input. 1. Create a PHP script with the content “helloworld” and save it as a “.php” extension. Transfer the PHP script to the web server’s page root directory, which is normally located at /var/www/html.Launch a web browser and navigate to the PHP script using the server’s URL, such as http://localhost/helloworld.php. When you visit this URL, the server will execute the PHP script,and the browser will render and show the results.

2) For the “echo.php” application:

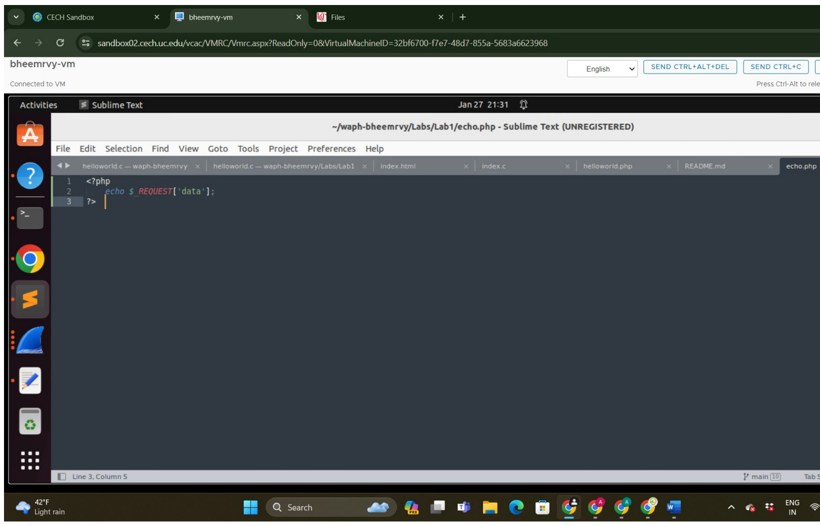
Create a “echo.php” script to function as an application. Navigate to the relevant URL on the web server to execute the PHP script. The script will process the request, and the browser will display the output from the “echo.php” application.

Included file echo.php: php Security Risks: • When the “data” argument is filled in with user input without sufficient validation or sanitization,it becomes vulnerable to XSS (Cross-Site Scripting) attacks. In such circumstances, attackers may insert malicious scripts that run when other people access the website, providing a huge security risk. • The usage of $_REQUEST enables data to come from a variety of sources, including user input via URL parameters. However, if the input does not include adequate validation or sanitization mechanisms, this strategy opens the door to data manipulation. This emphasizes the significance of having strong security measures to reduce the danger of unwanted data tampering.


bheemrvy-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
Google Chrome
Jan 23 23:08
administrator@mwph-vm: ~/waph-bheemrvy/Labs/Lab1
localhost/cgi-bin/helloworld.cgi

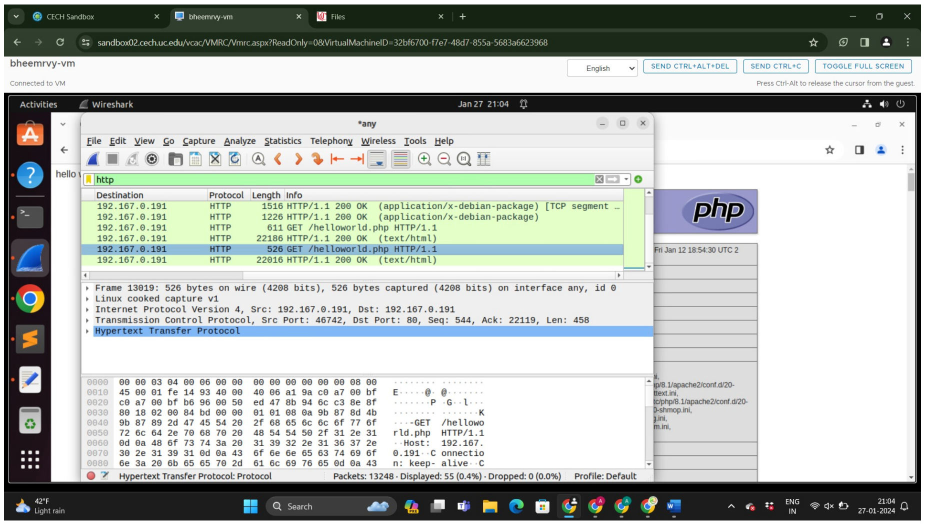
```
#include <stdio.h>
int main(void)
{
                printf("Content-Type: text/html; charset=utf-8\n\n");
                printf("<html>\n");
                printf("<head><title>WAPH</title></head>\n");
                printf("<body>\n");
                printf("<h1>Lab1<h1>");
                printf("<p>the lab1 contains of two parts in part1 HTTP protocols are covered a
nd in part2 basic web programming is covered</p>\n");
                printf("</body>\n");
                printf("</html>\n");
                return 0;
}administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ gcc index.c -o index.cgi
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo cp index.cgi /usr/lib/cgi-bin/
[sudo] password for administrator:
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ subl helloworld.php
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ cat helloworld.php
<?php
        echo "hello world, this is first php by bheemreddy vikhyath reddy, WAPH";
        phpinfo();
?>      administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo cp helloworld.php /var/www/html
[sudo] password for administrator:
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$
```

```
Hello World CGI! From Vikhyath Reddy
```



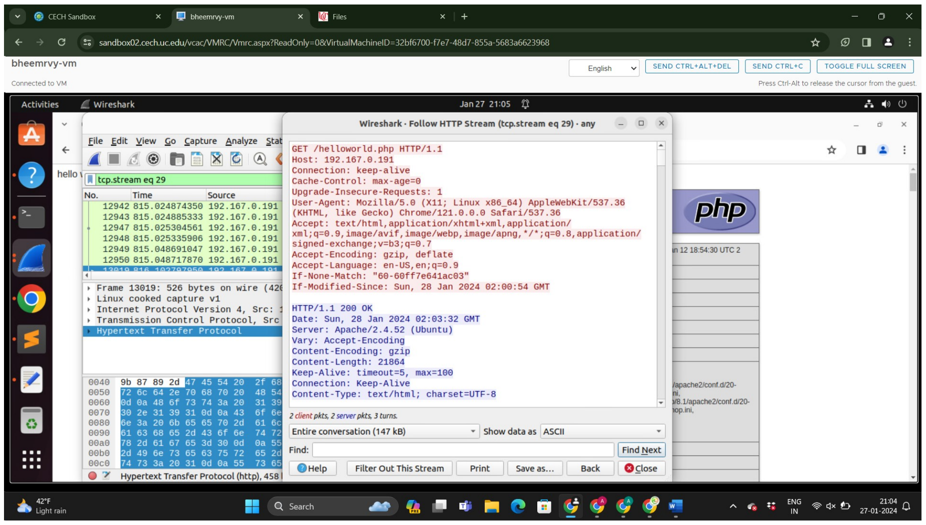
```
<?php
    echo $_REQUEST['data'];
?>
```

Figure 10: Hello_world_PHP_program
Figure 11: Echo_PHP_Program Task 3 - Understanding HTTP GET and POST requests. 1. Start by selecting the network interface through which the HTTP traffic is expected in my case I selected any to capture the HTTP traffic. Open the web browser and navigate to the PHP script using the server's URL http://localhost/echo.php?data=Hi%20this%20is%20vikhyath%20reddy%20Bheemreddy%20PHP%20echo.php%
Once the capture is initiated, navigate to the display filter and enter "http" to filter out HTTP-related packets. The below images demonstrates the Request and Response of the echo.php script



```
Destination       Protocol  Length Info
192.167.0.191     HTTP        1516 HTTP/1.1 200 OK  (application/x-debian-package) [TCP segment …
192.167.0.191     HTTP        1226 HTTP/1.1 200 OK  (application/x-debian-package)
192.167.0.191     HTTP         611 GET /helloworld.php HTTP/1.1
192.167.0.191     HTTP       22186 HTTP/1.1 200 OK  (text/html)
192.167.0.191     HTTP         526 GET /helloworld.php HTTP/1.1
192.167.0.191     HTTP       22016 HTTP/1.1 200 OK  (text/html)

› Frame 13019: 526 bytes on wire (4208 bits), 526 bytes captured (4208 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 192.167.0.191, Dst: 192.167.0.191
› Transmission Control Protocol, Src Port: 46742, Dst Port: 80, Seq: 544, Ack: 22119, Len: 458
› Hypertext Transfer Protocol

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 01 fe 14 93 40 00  40 06 a1 9a c0 a7 00 bf   E·····@· @·······
0020  c0 a7 00 bf b6 96 00 50  ed 47 8b 94 6c c3 8e 8f   ·······P ·G··l···
0030  80 18 02 00 84 bd 00 00  01 01 08 0a 9b 87 8d 4b   ········ ·······K
0040  9b 87 89 2d 47 45 54 20  2f 68 65 6c 6c 6f 77 6f   ···-GET  /hellowo
0050  72 6c 64 2e 70 68 70 20  48 54 54 50 2f 31 2e 31   rld.php  HTTP/1.1
0060  0d 0a 48 6f 73 74 3a 20  31 39 32 2e 31 36 37 2e   ··Host:  192.167.
0070  30 2e 31 39 31 0d 0a 43  6f 6e 6e 65 63 74 69 6f   0.191··C onnectio
0080  6e 3a 20 6b 65 65 70 2d  61 6c 69 76 65 0d 0a 43   n: keep- alive··C

Hypertext Transfer Protocol: Protocol    Packets: 13248 · Displayed: 55 (0.4%) · Dropped: 0 (0.0%)    Profile: Default
```

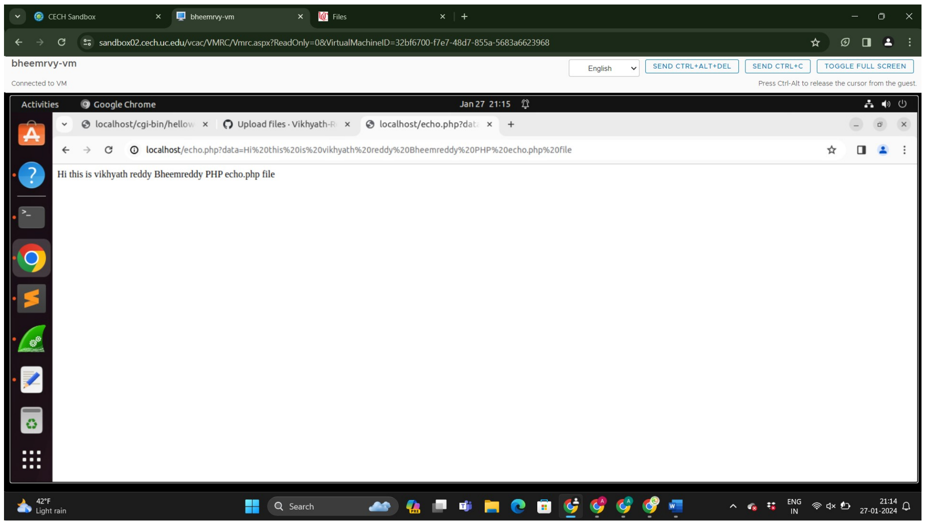
Figure 12: Echo_PHP_Wireshark_Request



```
GET /helloworld.php HTTP/1.1
Host: 192.167.0.191
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
If-None-Match: "60-60ff7e641ac03"
If-Modified-Since: Sun, 28 Jan 2024 02:00:54 GMT

HTTP/1.1 200 OK
Date: Sun, 28 Jan 2024 02:03:32 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 21864
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Wireshark · Follow HTTP Stream (tcp.stream eq 29) · any
2 client pkts, 2 server pkts, 3 turns.
Entire conversation (147 kB)
Show data as ASCII
Find:
Find Next
Help
Filter Out This Stream
Print
Save as…
Back
Close

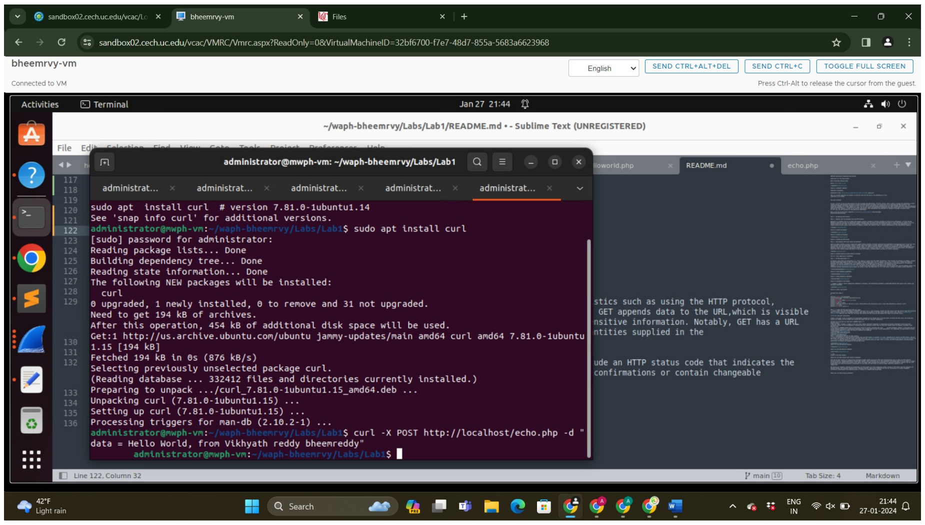
Figure 13: Echo_PHP_Wireshark_Response



```
localhost/echo.php?data=Hi%20this%20is%20vikhyath%20reddy%20Bheemreddy%20PHP%20echo.php%20file
```

Hi this is vikhyath reddy Bheemreddy PHP echo.php file

2. curl create an HTTP Post Request

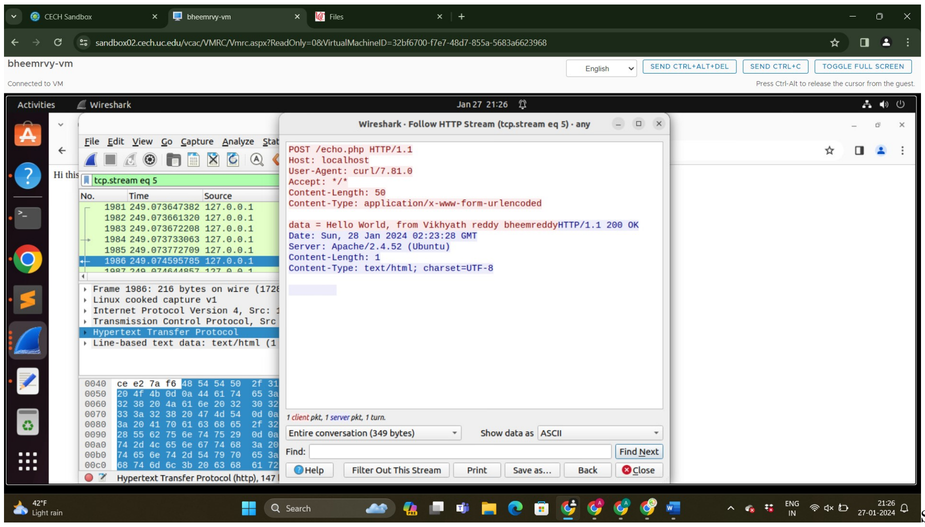

bheemrvy-vm
Connected to VM
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
administrator@mwph-vm: ~/waph-bheemrvy/Labs/Lab1
~/waph-bheemrvy/Labs/Lab1/README.md • - Sublime Text (UNREGISTERED)

```
sudo apt  install curl  # version 7.81.0-1ubuntu1.14
See 'snap info curl' for additional versions.
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ sudo apt install curl
[sudo] password for administrator:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 194 kB of archives.
After this operation, 454 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu
1.15 [194 kB]
Fetched 194 kB in 0s (876 kB/s)
Selecting previously unselected package curl.
(Reading database ... 332412 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0-1ubuntu1.15_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.15) ...
Setting up curl (7.81.0-1ubuntu1.15) ...
Processing triggers for man-db (2.10.2-1) ...
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$ curl -X POST http://localhost/echo.php -d "
data = Hello World, from Vikhyath reddy bheemreddy"
        administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab1$
```

Figure 15: curl_POST_request HTTP stream in Wireshark

HTTP GET and POST requests are communication techniques between clients and servers that share characteristics such as using the HTTP protocol, including headers, including specifying a Request URI. However, their differences lay in data submission: GET appends data to the URL,which is visible in the address bar, whereas POST transmits data within the request body, providing more protection for sensitive information. Notably, GET has a URL length-based size restriction,making it appropriate for less data, but POST excels at handling bigger quantities supplied in the request body.

GET queries are cacheable, enabling URL bookmarking, but POST answers are rarely cached.Both answers include an HTTP status code that indicates the outcome; however, GET responses frequently offer requested material, whereas POST responses may serve as confirmations or contain changeable information. Understanding these characteristics is critical for creating good online apps.


CECH Sandbox
bheemrvy-vm
Files
sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=32bf6700-f7e7-48d7-855a-5683a6623968
bheemrvy-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 27 21:26
Wireshark · Follow HTTP Stream (tcp.stream eq 5) · any
File Edit View Go Capture Analyze Stat
Hi this
tcp.stream eq 5
1 client pkt, 1 server pkt, 1 turn.
Entire conversation (349 bytes)
Show data as ASCII
Find:
Find Next
Help
Filter Out This Stream
Print
Save as…
Back
Close
42°F
Light rain
Search
ENG
IN
21:26
27-01-2024

```
No.  Time             Source
1981 249.073647382 127.0.0.1
1982 249.073661320 127.0.0.1
1983 249.073672208 127.0.0.1
1984 249.073733063 127.0.0.1
1985 249.073772709 127.0.0.1
1986 249.074595785 127.0.0.1
1987 249.074644857 127.0.0.1
```

```
Frame 1986: 216 bytes on wire (1728
Linux cooked capture v1
Internet Protocol Version 4, Src: 1
Transmission Control Protocol, Src
Hypertext Transfer Protocol
Line-based text data: text/html (1
```

```
0040  ce e2 7a f6 48 54 54 50  2f 31
0050  20 4f 4b 0d 0a 44 61 74  65 3a
0060  32 38 20 4a 61 6e 20 32  30 32
0070  33 3a 32 38 20 47 4d 54  0d 0a
0080  3a 20 41 70 61 63 68 65  2f 32
0090  28 55 62 75 6e 74 75 29  0d 0a
00a0  74 2d 4c 65 6e 67 74 68  3a 20
00b0  74 65 6e 74 2d 54 79 70  65 3a
00c0  68 74 6d 6c 3b 20 63 68  61 72
```

Hypertext Transfer Protocol (http), 147

```
POST /echo.php HTTP/1.1
Host: localhost
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 50
Content-Type: application/x-www-form-urlencoded

data = Hello World, from Vikhyath reddy bheemreddyHTTP/1.1 200 OK
Date: Sun, 28 Jan 2024 02:23:28 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 1
Content-Type: text/html; charset=UTF-8
```

s

Figure 16: curl_HTTP_stream_in_Wireshark