WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Akshat Chaturvedi

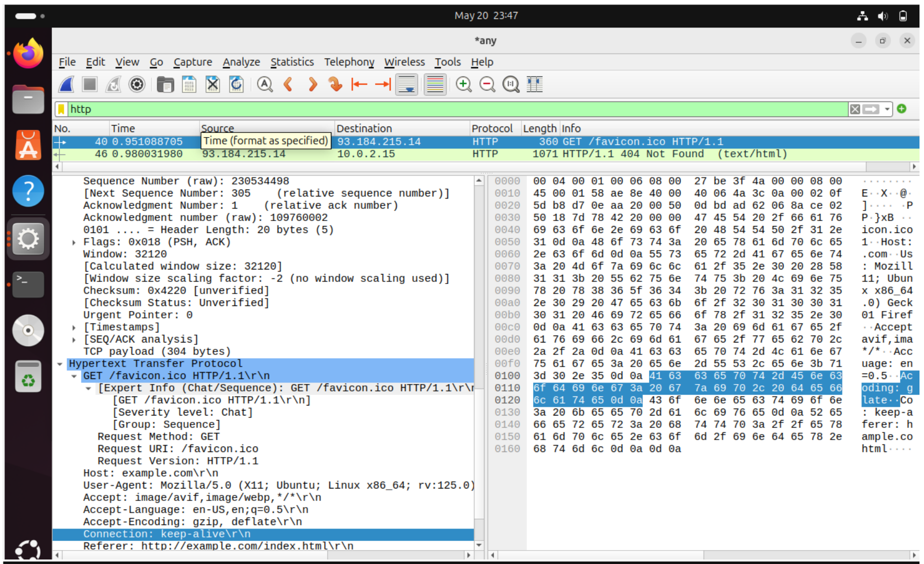
Email: chaturat@mail.uc.edu

Short-bio: I have a keen interest in web development and data science.

Repository Information **Repository-URL**: https://github.com/Akshat3960/waph-chaturat ## Lab1 **Lab Overview**: In this lab I had the chance to use technologies like wireshark and telnet firsthand in this lab. I was able to collect packets and examine requests and responses with the use of wireshark and telenet. I now have a better understanding of http thanks to this exercise. I also gained knowledge in web application programming and CGI. I used C to create a simple online application, then I translated the code to CGI. In addition, we used PHP to create a web application that included user input. Finally, I was able to comprehend get and post requests more clearly.

Part 1

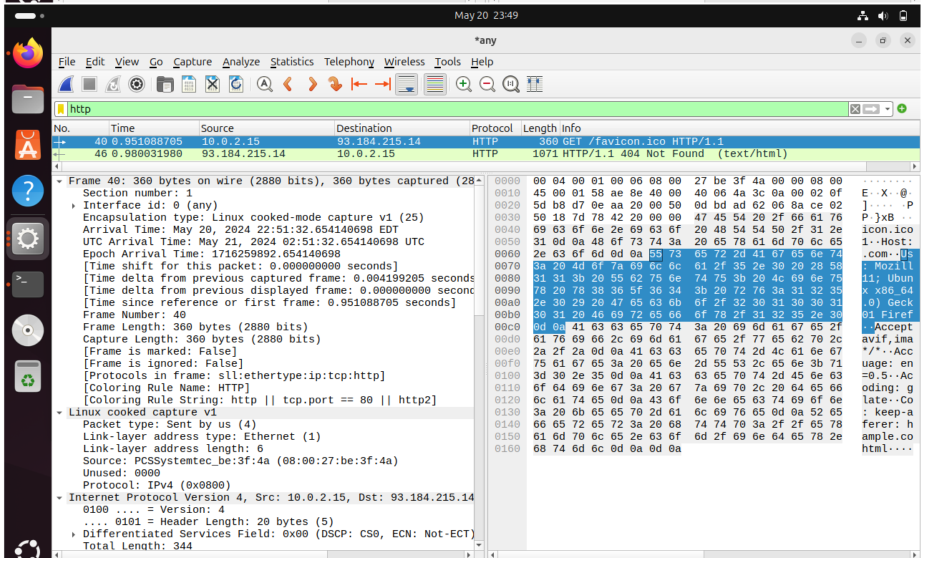
Task1: Familiar with the wireshark tool and HTTP Protocol: In this lab, I examined the HTTP protocol using Wireshark.For instance, I launched a live capture at http://example.com/index.html then ended it. I examined the response to the GET request for /index.html. After that, I followed the HTTP stream to witness the complete client-server conversation. The necessary screenshots are as follows: #### Image 1: Get Request


May 20 23:47
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Time (format as specified)

```
No.   Time           Source          Destination     Protocol Length Info
   40 0.951088705                    93.184.215.14   HTTP      360 GET /favicon.ico HTTP/1.1
   46 0.980031980    93.184.215.14   10.0.2.15       HTTP     1071 HTTP/1.1 404 Not Found  (text/html)
```

```
    Sequence Number (raw): 230534498
    [Next Sequence Number: 305    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 109760002
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x018 (PSH, ACK)
    Window: 32120
    [Calculated window size: 32120]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x4220 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  › [Timestamps]
  › [SEQ/ACK analysis]
    TCP payload (304 bytes)
⌄ Hypertext Transfer Protocol
  ⌄ GET /favicon.ico HTTP/1.1\r\n
    ⌄ [Expert Info (Chat/Sequence): GET /favicon.ico HTTP/1.1\r\r
        [GET /favicon.ico HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /favicon.ico
      Request Version: HTTP/1.1
    Host: example.com\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0)
    Accept: image/avif,image/webp,*/*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Referer: http://example.com/index.html\r\n
```

```
0000  00 04 00 01 00 06 08 00  27 be 3f 4a 00 00 08 00   ········
0010  45 00 01 58 ae 8e 40 00  40 06 4a 3c 0a 00 02 0f   E··X··@·
0020  5d b8 d7 0e aa 20 00 50  0d bd ad 62 06 8a ce 02   ]···· ·P
0030  50 18 7d 78 42 20 00 00  47 45 54 20 2f 66 61 76   P·}xB ··
0040  69 63 6f 6e 2e 69 63 6f  20 48 54 54 50 2f 31 2e   icon.ico
0050  31 0d 0a 48 6f 73 74 3a  20 65 78 61 6d 70 6c 65   1··Host:
0060  2e 63 6f 6d 0d 0a 55 73  65 72 2d 41 67 65 6e 74   .com··Us
0070  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 58   : Mozill
0080  31 31 3b 20 55 62 75 6e  74 75 3b 20 4c 69 6e 75   11; Ubun
0090  78 20 78 38 36 5f 36 34  3b 20 72 76 3a 31 32 35   x x86_64
00a0  2e 30 29 20 47 65 63 6b  6f 2f 32 30 31 30 30 31   .0) Geck
00b0  30 31 20 46 69 72 65 66  6f 78 2f 31 32 35 2e 30   01 Firef
00c0  0d 0a 41 63 63 65 70 74  3a 20 69 6d 61 67 65 2f   ··Accept
00d0  61 76 69 66 2c 69 6d 61  67 65 2f 77 65 62 70 2c   avif,ima
00e0  2a 2f 2a 0d 0a 41 63 63  65 70 74 2d 4c 61 6e 67   */*··Acc
00f0  75 61 67 65 3a 20 65 6e  2d 55 53 2c 65 6e 3b 71   uage: en
0100  3d 30 2e 35 0d 0a 41 63  63 65 70 74 2d 45 6e 63   =0.5··Ac
0110  6f 64 69 6e 67 3a 20 67  7a 69 70 2c 20 64 65 66   oding: g
0120  6c 61 74 65 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   late··Co
0130  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 52 65   : keep-a
0140  66 65 72 65 72 3a 20 68  74 74 70 3a 2f 2f 65 78   ferer: h
0150  61 6d 70 6c 65 2e 63 6f  6d 2f 69 6e 64 65 78 2e   ample.co
0160  68 74 6d 6c 0d 0a 0d 0a                            html····
```


May 20 23:49
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

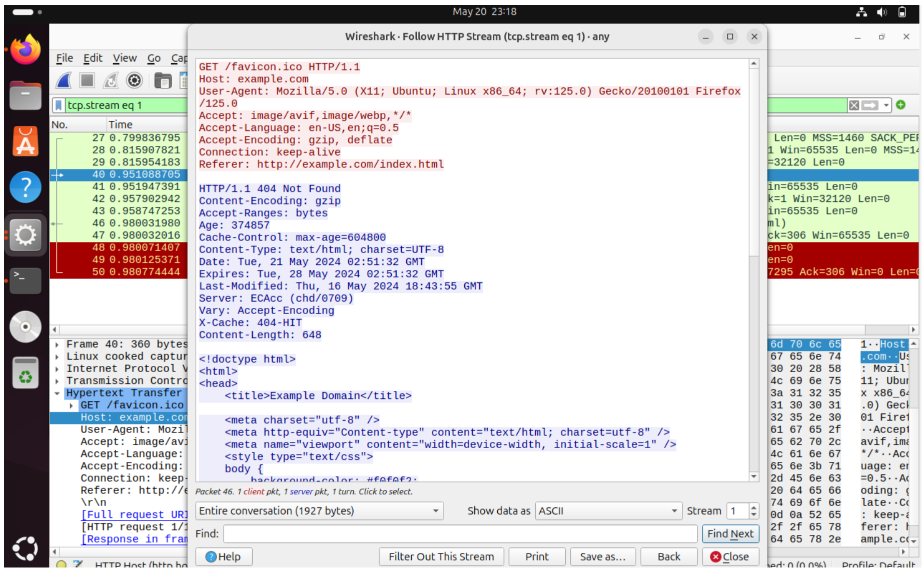
```
No.   Time           Source          Destination     Protocol Length Info
   40 0.951088705    10.0.2.15       93.184.215.14   HTTP      360 GET /favicon.ico HTTP/1.1
   46 0.980031980    93.184.215.14   10.0.2.15       HTTP     1071 HTTP/1.1 404 Not Found  (text/html)
```

```
⌄ Frame 40: 360 bytes on wire (2880 bits), 360 bytes captured (28
    Section number: 1
  › Interface id: 0 (any)
    Encapsulation type: Linux cooked-mode capture v1 (25)
    Arrival Time: May 20, 2024 22:51:32.654140698 EDT
    UTC Arrival Time: May 21, 2024 02:51:32.654140698 UTC
    Epoch Arrival Time: 1716259892.654140698
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.004199205 seconds
    [Time delta from previous displayed frame: 0.000000000 seconc
    [Time since reference or first frame: 0.951088705 seconds]
    Frame Number: 40
    Frame Length: 360 bytes (2880 bits)
    Capture Length: 360 bytes (2880 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: sll:ethertype:ip:tcp:http]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
⌄ Linux cooked capture v1
    Packet type: Sent by us (4)
    Link-layer address type: Ethernet (1)
    Link-layer address length: 6
    Source: PCSSystemtec_be:3f:4a (08:00:27:be:3f:4a)
    Unused: 0000
    Protocol: IPv4 (0x0800)
⌄ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 93.184.215.14
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 344
```

```
0000  00 04 00 01 00 06 08 00  27 be 3f 4a 00 00 08 00   ········
0010  45 00 01 58 ae 8e 40 00  40 06 4a 3c 0a 00 02 0f   E··X··@·
0020  5d b8 d7 0e aa 20 00 50  0d bd ad 62 06 8a ce 02   ]···· ·P
0030  50 18 7d 78 42 20 00 00  47 45 54 20 2f 66 61 76   P·}xB ··
0040  69 63 6f 6e 2e 69 63 6f  20 48 54 54 50 2f 31 2e   icon.ico
0050  31 0d 0a 48 6f 73 74 3a  20 65 78 61 6d 70 6c 65   1··Host:
0060  2e 63 6f 6d 0d 0a 55 73  65 72 2d 41 67 65 6e 74   .com··Us
0070  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 58   : Mozill
0080  31 31 3b 20 55 62 75 6e  74 75 3b 20 4c 69 6e 75   11; Ubun
0090  78 20 78 38 36 5f 36 34  3b 20 72 76 3a 31 32 35   x x86_64
00a0  2e 30 29 20 47 65 63 6b  6f 2f 32 30 31 30 30 31   .0) Geck
00b0  30 31 20 46 69 72 65 66  6f 78 2f 31 32 35 2e 30   01 Firef
00c0  0d 0a 41 63 63 65 70 74  3a 20 69 6d 61 67 65 2f   ··Accept
00d0  61 76 69 66 2c 69 6d 61  67 65 2f 77 65 62 70 2c   avif,ima
00e0  2a 2f 2a 0d 0a 41 63 63  65 70 74 2d 4c 61 6e 67   */*··Acc
00f0  75 61 67 65 3a 20 65 6e  2d 55 53 2c 65 6e 3b 71   uage: en
0100  3d 30 2e 35 0d 0a 41 63  63 65 70 74 2d 45 6e 63   =0.5··Ac
0110  6f 64 69 6e 67 3a 20 67  7a 69 70 2c 20 64 65 66   oding: g
0120  6c 61 74 65 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   late··Co
0130  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 52 65   : keep-a
0140  66 65 72 65 72 3a 20 68  74 74 70 3a 2f 2f 65 78   ferer: h
0150  61 6d 70 6c 65 2e 63 6f  6d 2f 69 6e 64 65 78 2e   ample.co
0160  68 74 6d 6c 0d 0a 0d 0a                            html····
```

Image 2: Get Response

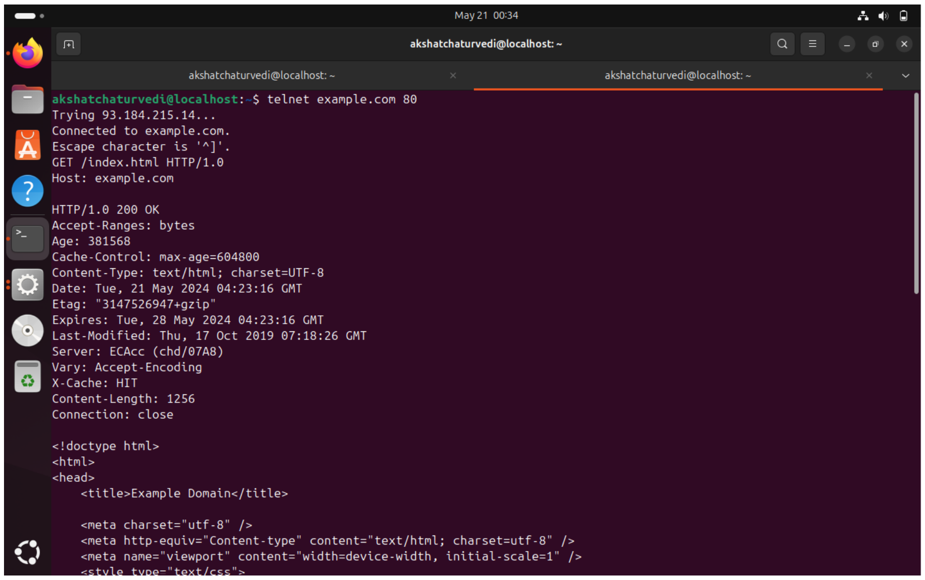
Task2: Understanding HTTP using Telnet and Wireshark: Using Wireshark and telnet, I learned about HTTP in Task 2. I manually established a telnet connection and sent example.com a straightforward HTTP request. I recorded the HTTP traffic using Wireshark while I was making the request. I was able to understand HTTP fundamentals and browser complexities better by researching this procedure. These three screenshots are necessary:



```
GET /favicon.ico HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:125.0) Gecko/20100101 Firefox
/125.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://example.com/index.html

HTTP/1.1 404 Not Found
Content-Encoding: gzip
Accept-Ranges: bytes
Age: 374857
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 21 May 2024 02:51:32 GMT
Expires: Tue, 28 May 2024 02:51:32 GMT
Last-Modified: Thu, 16 May 2024 18:43:55 GMT
Server: ECAcc (chd/0709)
Vary: Accept-Encoding
X-Cache: 404-HIT
Content-Length: 648

<!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;
```

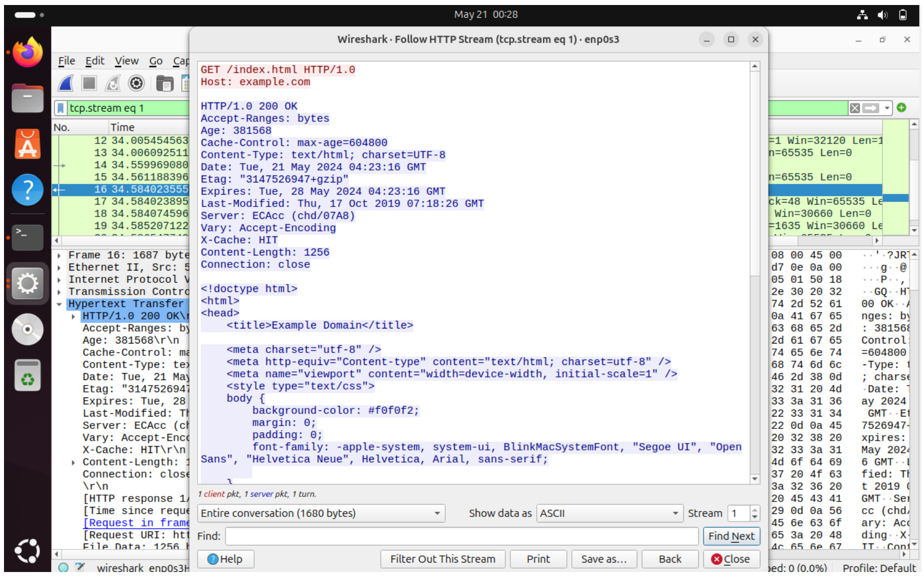
Wireshark · Follow HTTP Stream (tcp.stream eq 1) · any
Packet 46. 1 client pkt, 1 server pkt, 1 turn. Click to select.
Entire conversation (1927 bytes)
Show data as ASCII
Stream 1
Find:
Find Next
Help
Filter Out This Stream
Print
Save as…
Back
Close

Figure 1: Get Response



```
akshatchaturvedi@localhost:~$ telnet example.com 80
Trying 93.184.215.14...
Connected to example.com.
Escape character is '^]'.
GET /index.html HTTP/1.0
Host: example.com

HTTP/1.0 200 OK
Accept-Ranges: bytes
Age: 381568
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 21 May 2024 04:23:16 GMT
Etag: "3147526947+gzip"
Expires: Tue, 28 May 2024 04:23:16 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (chd/07A8)
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">
```

Image1: Request and Response from the server

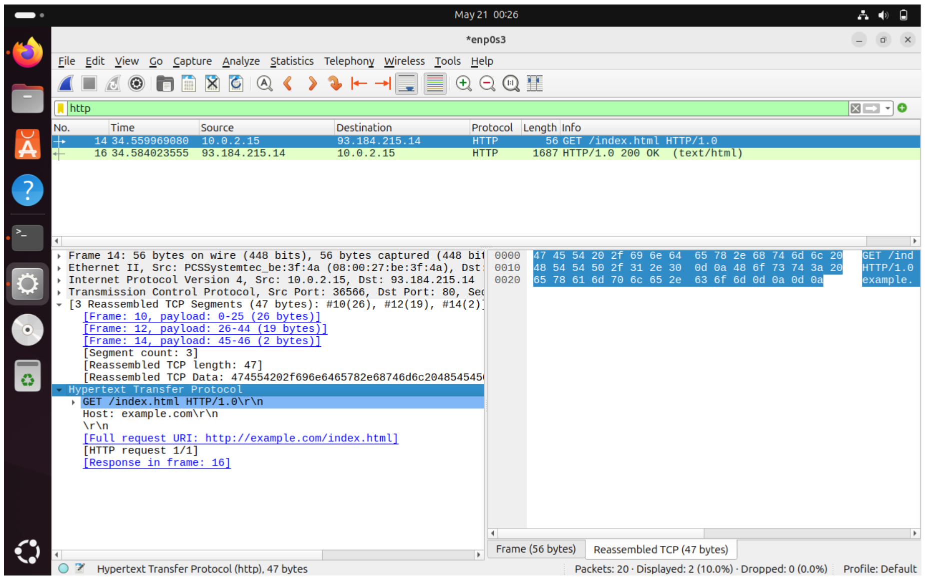


```
GET /index.html HTTP/1.0
Host: example.com

HTTP/1.0 200 OK
Accept-Ranges: bytes
Age: 381568
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 21 May 2024 04:23:16 GMT
Etag: "3147526947+gzip"
Expires: Tue, 28 May 2024 04:23:16 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (chd/07A8)
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, Arial, sans-serif;
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (1680 bytes) · Show data as: ASCII · Stream: 1

Image2: Request message in wireshark



```
No.  Time          Source        Destination   Protocol Length Info
  14 34.559969080  10.0.2.15     93.184.215.14 HTTP       56 GET /index.html HTTP/1.0
  16 34.584023555  93.184.215.14 10.0.2.15     HTTP     1687 HTTP/1.0 200 OK  (text/html)
```

```
Frame 14: 56 bytes on wire (448 bits), 56 bytes captured (448 bit
Ethernet II, Src: PCSSystemtec_be:3f:4a (08:00:27:be:3f:4a), Dst
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 93.184.215.14
Transmission Control Protocol, Src Port: 36566, Dst Port: 80, Sec
[3 Reassembled TCP Segments (47 bytes): #10(26), #12(19), #14(2)]
    [Frame: 10, payload: 0-25 (26 bytes)]
    [Frame: 12, payload: 26-44 (19 bytes)]
    [Frame: 14, payload: 45-46 (2 bytes)]
    [Segment count: 3]
    [Reassembled TCP length: 47]
    [Reassembled TCP Data: 474554202f696e6465782e68746d6c2048545450
Hypertext Transfer Protocol
    GET /index.html HTTP/1.0\r\n
    Host: example.com\r\n
    \r\n
    [Full request URI: http://example.com/index.html]
    [HTTP request 1/1]
    [Response in frame: 16]
```

```
0000  47 45 54 20 2f 69 6e 64  65 78 2e 68 74 6d 6c 20   GET /ind
0010  48 54 54 50 2f 31 2e 30  0d 0a 48 6f 73 74 3a 20   HTTP/1.0
0020  65 78 61 6d 70 6c 65 2e  63 6f 6d 0d 0a 0d 0a       example.
```

Hypertext Transfer Protocol (http), 47 bytes · Packets: 20 · Displayed: 2 (10.0%) · Dropped: 0 (0.0%) · Profile: Default

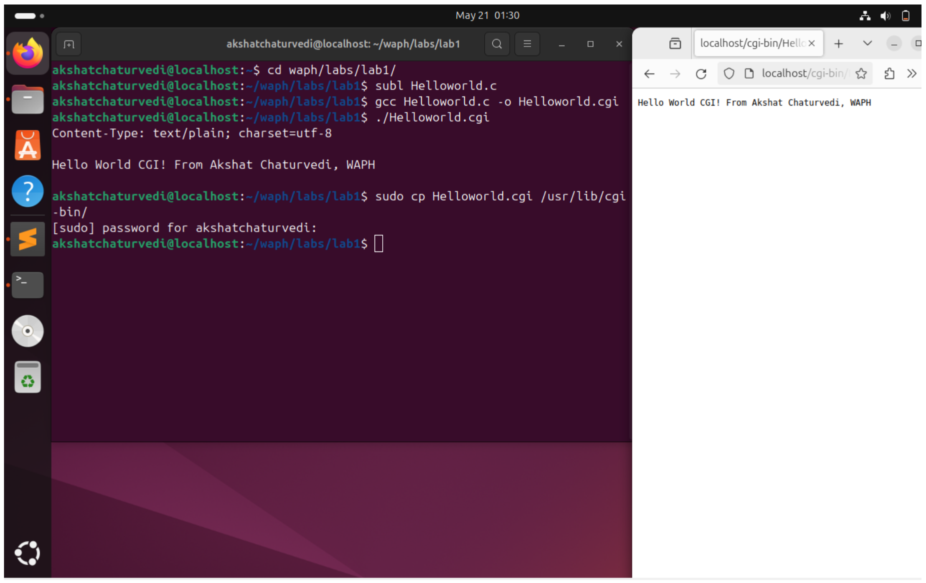
May 21 00:26
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Frame (56 bytes)
Reassembled TCP (47 bytes)

Image3: Response

Part2

Task1: a): I built the C program Helloworld.c to generate a “Hello World” CGI program. It printed the message “Hello World CGI! From Akshat Chaturvedi, WAPH”. I used gcc to compile the code, and to accomplish this, I turned on Apache’s CGI module, dropped the built program to

/usr/lib/cgi-bin, and checked that it worked by opening a web browser and going to http://localhost/cgi-bin/Helloworld.cgi. To illustrate, here is the screenshot:

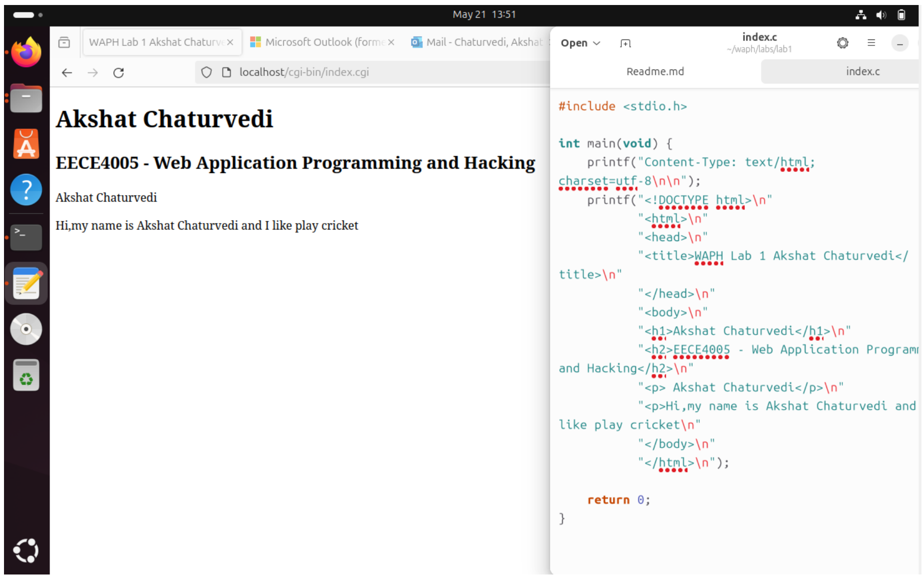


```
akshatchaturvedi@localhost:~$ cd waph/labs/lab1/
akshatchaturvedi@localhost:~/waph/labs/lab1$ subl Helloworld.c
akshatchaturvedi@localhost:~/waph/labs/lab1$ gcc Helloworld.c -o Helloworld.cgi
akshatchaturvedi@localhost:~/waph/labs/lab1$ ./Helloworld.cgi
Content-Type: text/plain; charset=utf-8

Hello World CGI! From Akshat Chaturvedi, WAPH

akshatchaturvedi@localhost:~/waph/labs/lab1$ sudo cp Helloworld.cgi /usr/lib/cgi
-bin/
[sudo] password for akshatchaturvedi:
akshatchaturvedi@localhost:~/waph/labs/lab1$
```

```
Hello World CGI! From Akshat Chaturvedi, WAPH
```

Figure 2: P2T1

b): To deploy the code, I first copied the file in the same folder and named it as index.c. Then, I modified the program and inside the printf command, I had written the HTML code. After it, I converted it into the cgi file and opened it in the browser after which I had written the command,"localhost/cgi-bin/index.cgi" and then got the output as shown in the image below.



Akshat Chaturvedi

EECE4005 - Web Application Programming and Hacking

Akshat Chaturvedi

Hi,my name is Akshat Chaturvedi and I like play cricket

```
#include <stdio.h>

int main(void) {
    printf("Content-Type: text/html;
charset=utf-8\n\n");
    printf("<!DOCTYPE html>\n"
           "<html>\n"
           "<head>\n"
           "<title>WAPH Lab 1 Akshat Chaturvedi</
title>\n"
           "</head>\n"
           "<body>\n"
           "<h1>Akshat Chaturvedi</h1>\n"
           "<h2>EECE4005 - Web Application Program
and Hacking</h2>\n"
           "<p> Akshat Chaturvedi</p>\n"
           "<p>Hi,my name is Akshat Chaturvedi and
like play cricket\n"
           "</body>\n"
           "</html>\n");

    return 0;
}
```

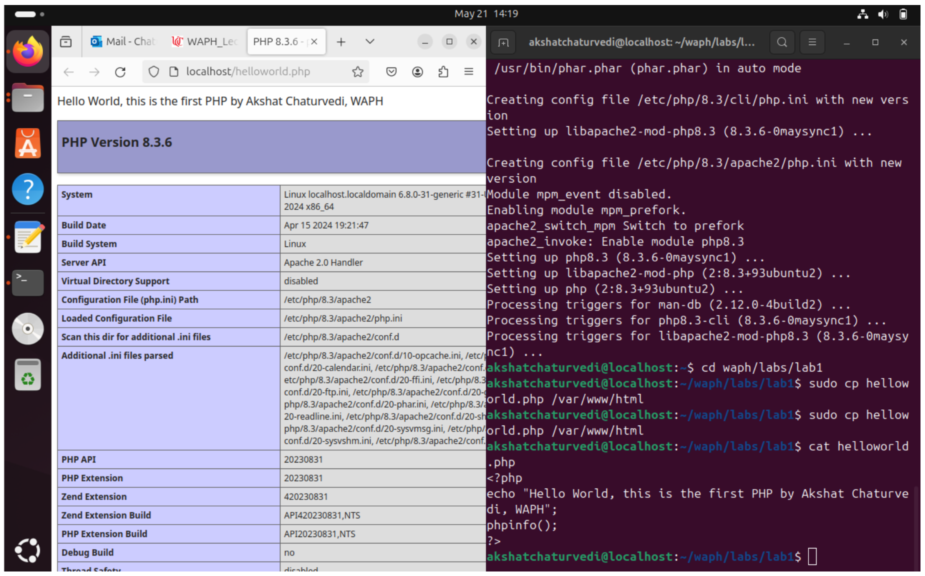
Chaturvedi, Akshat (chaturat) ##### #include <stdio.h>

int main(void) {

```
printf("Content-Type: text/html; charset=utf-8\n\n");
printf("<!DOCTYPE html>\n"
       "<html>\n"
       "<head>\n"
       "<title>WAPH Lab 1 Akshat Chaturvedi</title>\n"
       "</head>\n"
       "<body>\n"
       "<h1>Akshat Chaturvedi</h1>\n"
       "<h2>EECE4005 - Web Application Programming and Hacking</h2>\n"
       "<p> Akshat Chaturvedi</p>\n"
       "<p>Hi,my name is Akshat Chaturvedi and I like play cricket\n"
       "</body>\n"
       "</html>\n");

return 0;
```

}

Task2: **a)**: Installing PHP and making a helloworld.php file in my lab1 folder was my initial step. After that, I copied and pasted the given code and deployed file to the six root directory of the webserver. I checked the file's functionality in the browser once it was deployed to make sure it was up to level.


Hello World, this is the first PHP by Akshat Chaturvedi, WAPH

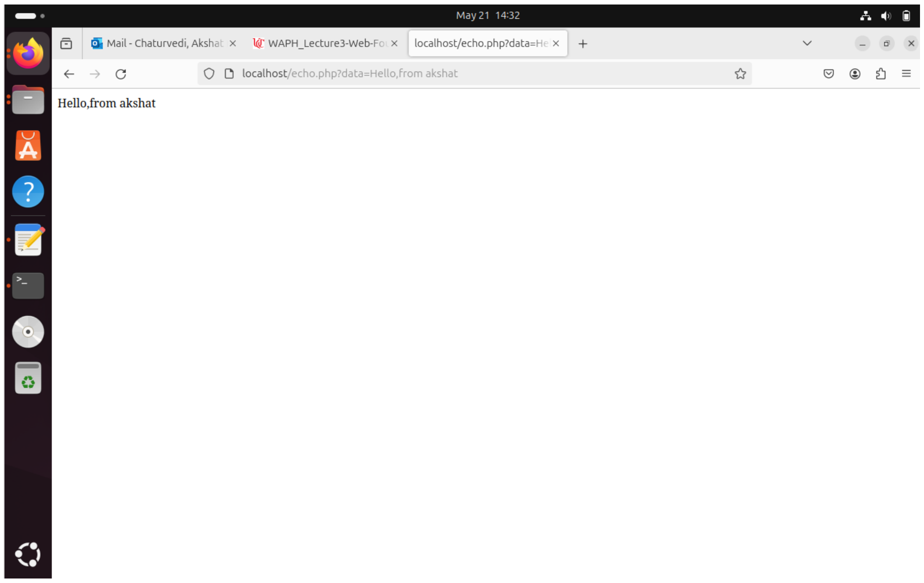
PHP Version 8.3.6

System	Linux localhost.localdomain 6.8.0-31-generic #31- 2024 x86_64
Build Date	Apr 15 2024 19:21:47
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/ conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf. etc/php/8.3/apache2/conf.d/20-ffi.ini, /etc/php/8.3 conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20- php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/ 20-readline.ini, /etc/php/8.3/apache2/conf.d/20-sh php/8.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/ conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no
Thread Safety	disabled

```
 /usr/bin/phar.phar (phar.phar) in auto mode

Creating config file /etc/php/8.3/cli/php.ini with new vers
ion
Setting up libapache2-mod-php8.3 (8.3.6-0maysync1) ...

Creating config file /etc/php/8.3/apache2/php.ini with new
version
Module mpm_event disabled.
Enabling module mpm_prefork.
apache2_switch_mpm Switch to prefork
apache2_invoke: Enable module php8.3
Setting up php8.3 (8.3.6-0maysync1) ...
Setting up libapache2-mod-php (2:8.3+93ubuntu2) ...
Setting up php (2:8.3+93ubuntu2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for php8.3-cli (8.3.6-0maysync1) ...
Processing triggers for libapache2-mod-php8.3 (8.3.6-0maysy
nc1) ...
akshatchaturvedi@localhost:~$ cd waph/labs/lab1
akshatchaturvedi@localhost:~/waph/labs/lab1$ sudo cp hellow
orld.php /var/www/html
akshatchaturvedi@localhost:~/waph/labs/lab1$ sudo cp hellow
orld.php /var/www/html
akshatchaturvedi@localhost:~/waph/labs/lab1$ cat helloworld
.php
<?php
echo "Hello World, this is the first PHP by Akshat Chaturve
di, WAPH";
phpinfo();
?>
akshatchaturvedi@localhost:~/waph/labs/lab1$ 
```

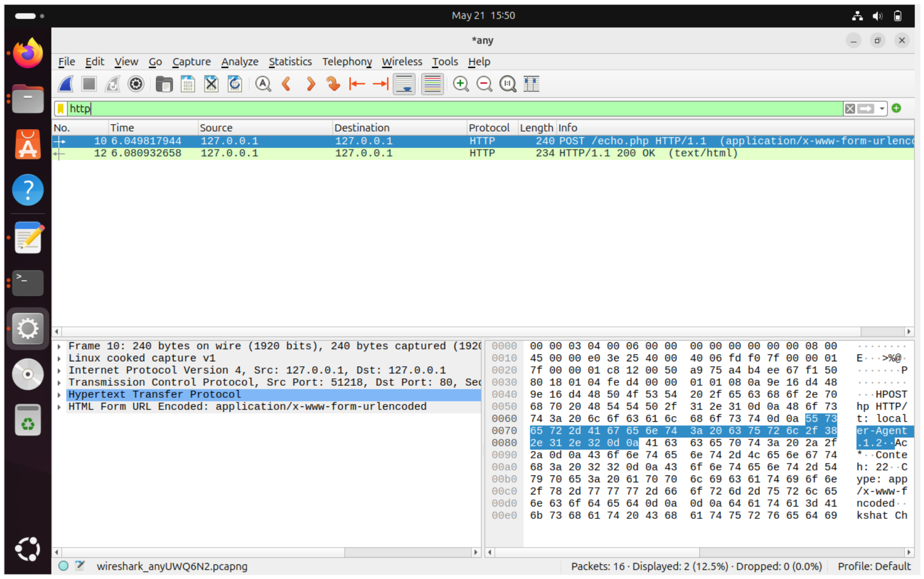
b): I typed the command HTTP-POST http://localhost/echo.php-d "data=Hello, from akshat" on the terminal to test echo.php using an HTTP POST request with curl. The echoed data was shown in the response. I simultaneously ran Wireshark to capture packets while this was going on. Wireshark then recorded the HTTP requests and displayed the data that was delivered and received. I believe that one of the application's security threats is the possibility of data capture during transmission and reception, which could result in the disclosure of private or sensitive information. This is how echo.php is coded:

Task3: **a)**: I used the browser to navigate to http://localhost/echo.php?data=Akshat Chaturvedi in order to study the HTTP GET Request and Response from the echo.php page. At the same time, I turned on Wireshark and simultaneously captured the request and response packets. I was able to observe the request and answer with the aid of wireshark.


May 21 14:32
Mail - Chaturvedi, Akshat
WAPH_Lecture3-Web-Fo
localhost/echo.php?data=He
localhost/echo.php?data=Hello,from akshat

Hello,from akshat

Figure 3: PHP


May 21 15:50
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.   Time          Source       Destination  Protocol Length Info
   10 6.049817944   127.0.0.1    127.0.0.1    HTTP        240 POST /echo.php HTTP/1.1  (application/x-www-form-urlenc
   12 6.080932658   127.0.0.1    127.0.0.1    HTTP        234 HTTP/1.1 200 OK  (text/html)
```

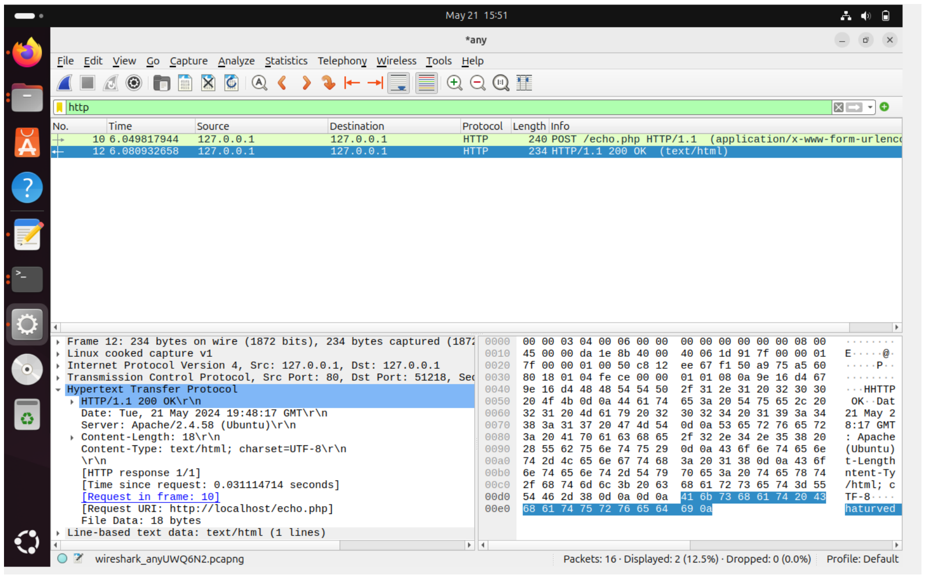
```
Frame 10: 240 bytes on wire (1920 bits), 240 bytes captured (1920
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 51218, Dst Port: 80, Sec
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········
0010  45 00 00 e0 3e 25 40 00  40 06 fd f0 7f 00 00 01   E···>%@·
0020  7f 00 00 01 c8 12 00 50  a9 75 a4 b4 ee 67 f1 50   ·······P
0030  80 18 01 04 fe d4 00 00  01 01 08 0a 9e 16 d4 48   ········
0040  9e 16 d4 48 50 4f 53 54  20 2f 65 63 68 6f 2e 70   ···HPOST
0050  68 70 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   hp HTTP/
0060  74 3a 20 6c 6f 63 61 6c  68 6f 73 74 0d 0a 55 73   t: local
0070  65 72 2d 41 67 65 6e 74  3a 20 63 75 72 6c 2f 38   er-Agent
0080  2e 31 2e 32 0d 0a 41 63  63 65 70 74 3a 20 2a 2f   .1.2··Ac
0090  2a 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   *··Conte
00a0  68 3a 20 32 32 0d 0a 43  6f 6e 74 65 6e 74 2d 54   h: 22··C
00b0  79 70 65 3a 20 61 70 70  6c 69 63 61 74 69 6f 6e   ype: app
00c0  2f 78 2d 77 77 77 2d 66  6f 72 6d 2d 75 72 6c 65   /x-www-f
00d0  6e 63 6f 64 65 64 0d 0a  0d 0a 64 61 74 61 3d 41   ncoded··
00e0  6b 73 68 61 74 20 43 68  61 74 75 72 76 65 64 69   kshat Ch
```

wireshark_anyUWQ6N2.pcapng   Packets: 16 · Displayed: 2 (12.5%) · Dropped: 0 (0.0%)   Profile: Default

Image1: Get Request

Image2: Get Response b): Using curl, I typed curl-X POST http://localhost/echo.php-d "data=[DATA]" in the console to make an HTTP POST request with my name. I also used Wireshark to record the HTTP stream during this procedure, which allowed me to see the requests and answers.



```
No.  Time          Source      Destination  Protocol Length Info
  10 6.049817944   127.0.0.1   127.0.0.1    HTTP     240 POST /echo.php HTTP/1.1  (application/x-www-form-urlenc
  12 6.080932658   127.0.0.1   127.0.0.1    HTTP     234 HTTP/1.1 200 OK  (text/html)
```

```
Frame 12: 234 bytes on wire (1872 bits), 234 bytes captured (1872
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: 51218, Sec
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Tue, 21 May 2024 19:48:17 GMT\r\n
  Server: Apache/2.4.58 (Ubuntu)\r\n
  Content-Length: 18\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.031114714 seconds]
  [Request in frame: 10]
  [Request URI: http://localhost/echo.php]
  File Data: 18 bytes
Line-based text data: text/html (1 lines)
```

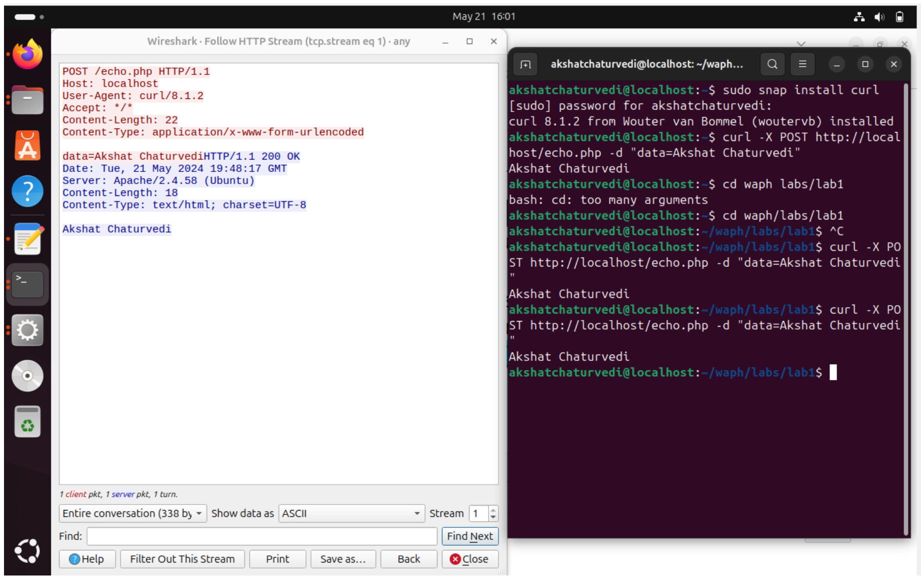
```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········
0010  45 00 00 da 1e 8b 40 00  40 06 1d 91 7f 00 00 01   E·····@·
0020  7f 00 00 01 00 50 c8 12  ee 67 f1 50 a9 75 a5 60   ·····P··
0030  80 18 01 04 fe ce 00 00  01 01 08 0a 9e 16 d4 67   ········
0040  9e 16 d4 48 48 54 54 50  2f 31 2e 31 20 32 30 30   ···HHTTP
0050  20 4f 4b 0d 0a 44 61 74  65 3a 20 54 75 65 2c 20    OK··Dat
0060  32 31 20 4d 61 79 20 32  30 32 34 20 31 39 3a 34   21 May 2
0070  38 3a 31 37 20 47 4d 54  0d 0a 53 65 72 76 65 72   8:17 GMT
0080  3a 20 41 70 61 63 68 65  2f 32 2e 34 2e 35 38 20   : Apache
0090  28 55 62 75 6e 74 75 29  0d 0a 43 6f 6e 74 65 6e   (Ubuntu)
00a0  74 2d 4c 65 6e 67 74 68  3a 20 31 38 0d 0a 43 6f   t-Length
00b0  6e 74 65 6e 74 2d 54 79  70 65 3a 20 74 65 78 74   ntent-Ty
00c0  2f 68 74 6d 6c 3b 20 63  68 61 72 73 65 74 3d 55   /html; c
00d0  54 46 2d 38 0d 0a 0d 0a  41 6b 73 68 61 74 20 43   TF-8····
00e0  68 61 74 75 72 76 65 64  69 0a                     haturved
```

```
wireshark_anyUWQ6N2.pcapng    Packets: 16 · Displayed: 2 (12.5%) · Dropped: 0 (0.0%)    Profile: Default
```

May 21 15:51
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

Figure 4: PHP

The screenshots are attached here:



```
POST /echo.php HTTP/1.1
Host: localhost
User-Agent: curl/8.1.2
Accept: */*
Content-Length: 22
Content-Type: application/x-www-form-urlencoded

data=Akshat ChaturvediHTTP/1.1 200 OK
Date: Tue, 21 May 2024 19:48:17 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Length: 18
Content-Type: text/html; charset=UTF-8

Akshat Chaturvedi
```

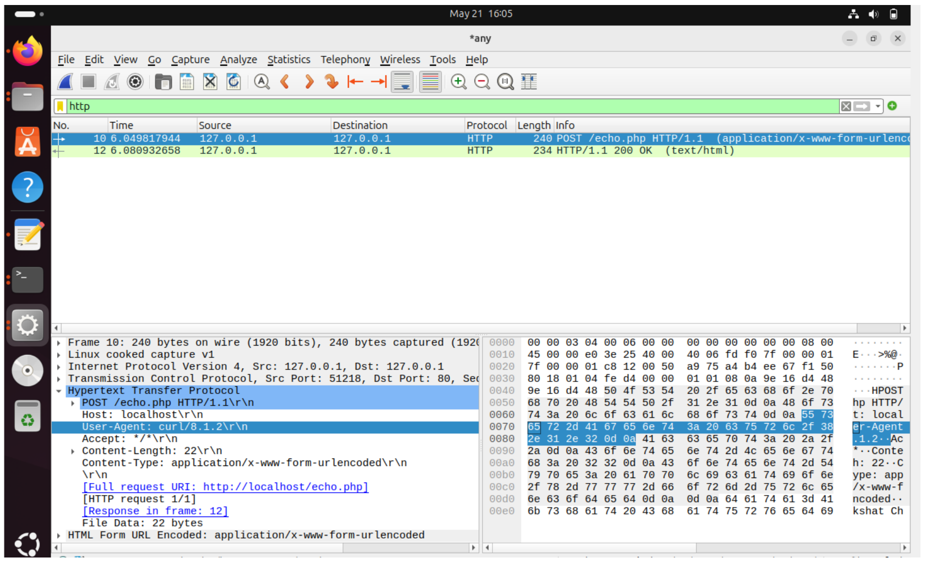
```
akshatchaturvedi@localhost:~$ sudo snap install curl
[sudo] password for akshatchaturvedi:
curl 8.1.2 from Wouter van Bommel (woutervb) installed
akshatchaturvedi@localhost:~$ curl -X POST http://local
host/echo.php -d "data=Akshat Chaturvedi"
Akshat Chaturvedi
akshatchaturvedi@localhost:~$ cd waph labs/lab1
bash: cd: too many arguments
akshatchaturvedi@localhost:~$ cd waph/labs/lab1
akshatchaturvedi@localhost:~/waph/labs/lab1$ ^C
akshatchaturvedi@localhost:~/waph/labs/lab1$ curl -X PO
ST http://localhost/echo.php -d "data=Akshat Chaturvedi
"
Akshat Chaturvedi
akshatchaturvedi@localhost:~/waph/labs/lab1$ curl -X PO
ST http://localhost/echo.php -d "data=Akshat Chaturvedi
"
Akshat Chaturvedi
akshatchaturvedi@localhost:~/waph/labs/lab1$
```

May 21 16:01
Wireshark · Follow HTTP Stream (tcp.stream eq 1) · any
akshatchaturvedi@localhost: ~/waph...
1 client pkt, 1 server pkt, 1 turn.
Entire conversation (338 by
Show data as
ASCII
Stream
1
Find:
Find Next
Help
Filter Out This Stream
Print
Save as…
Back
Close

Image1: Curl(HTTP Request)

Image2: Wireshark Capturing Packets

c): While GET queries allow visitors to view the material in the URL, POST


May 21 16:05
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.    Time          Source       Destination  Protocol Length Info
    10 6.049817944   127.0.0.1    127.0.0.1    HTTP       240 POST /echo.php HTTP/1.1  (application/x-www-form-urlenco
    12 6.080932658   127.0.0.1    127.0.0.1    HTTP       234 HTTP/1.1 200 OK  (text/html)
```

```
Frame 10: 240 bytes on wire (1920 bits), 240 bytes captured (1920
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 51218, Dst Port: 80, Se
Hypertext Transfer Protocol
  POST /echo.php HTTP/1.1\r\n
  Host: localhost\r\n
  User-Agent: curl/8.1.2\r\n
  Accept: */*\r\n
  Content-Length: 22\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  \r\n
  [Full request URI: http://localhost/echo.php]
  [HTTP request 1/1]
  [Response in frame: 12]
  File Data: 22 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········
0010  45 00 00 e0 3e 25 40 00  40 06 fd f0 7f 00 00 01   E···>%@·
0020  7f 00 00 01 c8 12 00 50  a9 75 a4 b4 ee 67 f1 50   ·······P
0030  80 18 01 04 fe d4 00 00  01 01 08 0a 9e 16 d4 48   ········
0040  9e 16 d4 48 50 4f 53 54  20 2f 65 63 68 6f 2e 70   ···HPOST
0050  68 70 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   hp HTTP/
0060  74 3a 20 6c 6f 63 61 6c  68 6f 73 74 0d 0a 55 73   t: local
0070  65 72 2d 41 67 65 6e 74  3a 20 63 75 72 6c 2f 38   er-Agent
0080  2e 31 2e 32 0d 0a 41 63  63 65 70 74 3a 20 2a 2f   .1.2··Ac
0090  2a 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   *··Conte
00a0  68 3a 20 32 32 0d 0a 43  6f 6e 74 65 6e 74 2d 54   h: 22··C
00b0  79 70 65 3a 20 61 70 70  6c 69 63 61 74 69 6f 6e   ype: app
00c0  2f 78 2d 77 77 77 2d 66  6f 72 6d 2d 75 72 6c 65   /x-www-f
00d0  6e 63 6f 64 65 64 0d 0a  0d 0a 64 61 74 61 3d 41   ncoded··
00e0  6b 73 68 61 74 20 43 68  61 74 75 72 76 65 64 69   kshat Ch
```

Figure 5: PHP

requests are typically used for sensitive or larger data because they conceal the contents from users. Nevertheless, we may continue to use programs like Wireshark to examine and alter both kinds of requests. Regarding responses, both techniques provide server data; however, POST responses are not limited by URL length restrictions, in contrast to GET responses.