WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Ruthvik Suvarnakanti

Email: suvarnrk@mail.uc.edu

Figure 1: Ruthvik Suvarnakanti

Lab 1 - Foundations of the WEB

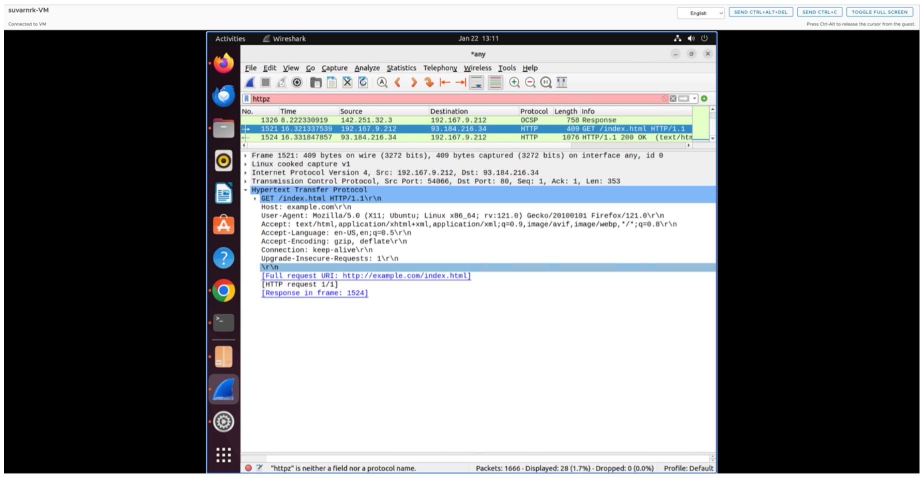
Overview: This lab deals with web technologies , HTTP protocol and basic web application programming. Focusing on Wireshark and TELNET for examining the HTTP requests, responses and comparing them with broswer sent requests. Moving on to the web application programming I got familiarized with develepment 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 genearate the PDF report for submission.

https://github.com/suvarnrk/waph-suvarnrk/blob/main/labs/lab1/README.md

Part 1 : The WEB and the HTTP Protocol

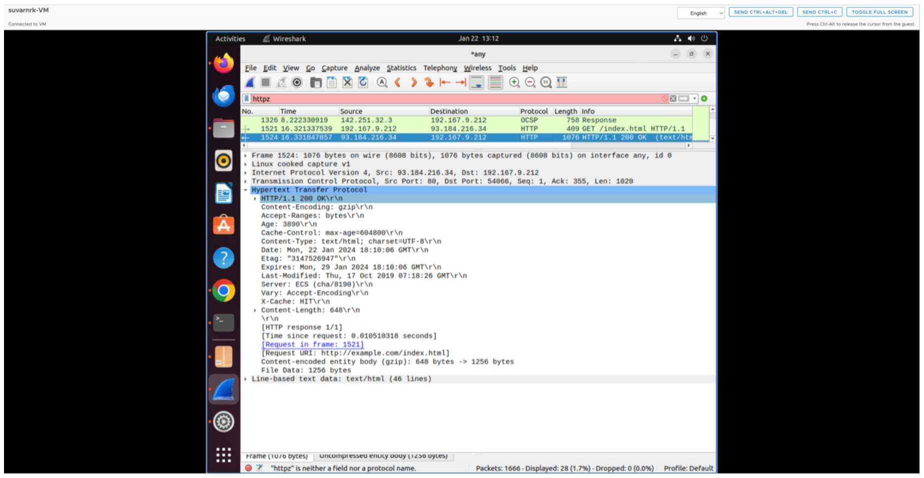
Task 1. Familiar with the Wireshark tool and HTTP protocol

A popular tool for analyzing network packets and protocols, Wireshark records and shows intricate packet data to aid with network problems. Installing it on Ubuntu virtual machines, checking the version, and using it to examine and solve network problems are all possible. Firstly, Install Wireshark in ubuntu using commands from terminal. Open the Wireshark and filter the captures with “any” using which results in all traffics happening. Filter the displaying packets using “http”. Give a request in the browser , here I have given “http://example.com/index.html” as a request and observed the packets in the Wireshark by filtering http in the displayed packets. There will be a HTTP GET request and HTTP response for the request given in browser, we need to capture those packets as part of one this module.


suvarnrk-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 13:11
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
httpz

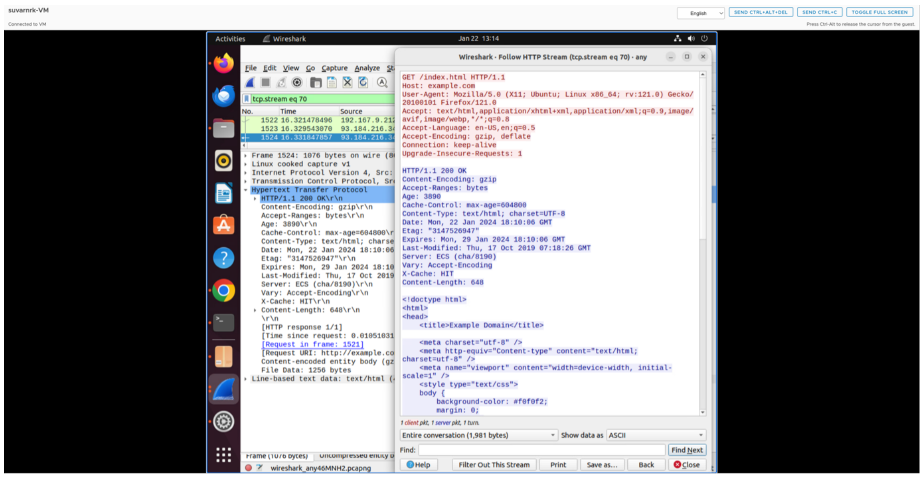
```
No.   Time          Source          Destination     Protocol Length Info
 1326 8.222330919   142.251.32.3    192.167.9.212   OCSP       758 Response
 1521 16.321337539  192.167.9.212   93.184.216.34   HTTP       409 GET /index.html HTTP/1.1
 1524 16.331847857  93.184.216.34   192.167.9.212   HTTP      1076 HTTP/1.1 200 OK  (text/htm

Frame 1521: 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: 192.167.9.212, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 54066, Dst Port: 80, Seq: 1, Ack: 1, Len: 353
Hypertext Transfer Protocol
  GET /index.html HTTP/1.1\r\n
  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 1/1]
  [Response in frame: 1524]

"httpz" is neither a field nor a protocol name.   Packets: 1666 · Displayed: 28 (1.7%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure 2: Wireshark HTTP Request


suvarnrk-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 13:12
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
httpz

```
No.   Time          Source          Destination     Protocol Length Info
 1326 8.222330919   142.251.32.3    192.167.9.212   OCSP       758 Response
 1521 16.321337539  192.167.9.212   93.184.216.34   HTTP       409 GET /index.html HTTP/1.1
 1524 16.331847857  93.184.216.34   192.167.9.212   HTTP      1076 HTTP/1.1 200 OK  (text/htm

Frame 1524: 1076 bytes on wire (8608 bits), 1076 bytes captured (8608 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.167.9.212
Transmission Control Protocol, Src Port: 80, Dst Port: 54066, Seq: 1, Ack: 355, Len: 1020
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Encoding: gzip\r\n
  Accept-Ranges: bytes\r\n
  Age: 3890\r\n
  Cache-Control: max-age=604800\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  Date: Mon, 22 Jan 2024 18:10:06 GMT\r\n
  Etag: "3147526947"\r\n
  Expires: Mon, 29 Jan 2024 18:10:06 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 1/1]
  [Time since request: 0.010510318 seconds]
  [Request in frame: 1521]
  [Request URI: http://example.com/index.html]
  Content-encoded entity body (gzip): 648 bytes -> 1256 bytes
  File Data: 1256 bytes
Line-based text data: text/html (46 lines)

Frame (1076 bytes)   Uncompressed entity body (1256 bytes)
"httpz" is neither a field nor a protocol name.   Packets: 1666 · Displayed: 28 (1.7%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure 3: Wireshark HTTP Response


Activities
Wireshark
Jan 22 13:14
Wireshark · Follow HTTP Stream (tcp.stream eq 70) · any
File Edit View Go Capture Analyze
tcp.stream eq 70

```
No.   Time            Source
1522 16.321478496  192.167.9.212
1523 16.329543070  93.184.216.34
1524 16.331847857  93.184.216.34

▸ Frame 1524: 1076 bytes on wire (8
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src:
▸ Transmission Control Protocol, Src
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Content-Encoding: gzip\r\n
    Accept-Ranges: bytes\r\n
    Age: 3890\r\n
    Cache-Control: max-age=604800\r
    Content-Type: text/html; charse
    Date: Mon, 22 Jan 2024 18:10:06
    Etag: "3147526947"\r\n
    Expires: Mon, 29 Jan 2024 18:10
    Last-Modified: Thu, 17 Oct 2019
    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 1/1]
    [Time since request: 0.01051031
    [Request in frame: 1521]
    [Request URI: http://example.co
    Content-encoded entity body (gz
    File Data: 1256 bytes
▸ Line-based text data: text/html (
```

```
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: 3890
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Mon, 22 Jan 2024 18:10:06 GMT
Etag: "3147526947"
Expires: Mon, 29 Jan 2024 18:10:06 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (cha/8190)
Vary: Accept-Encoding
X-Cache: 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;
        margin: 0;
```

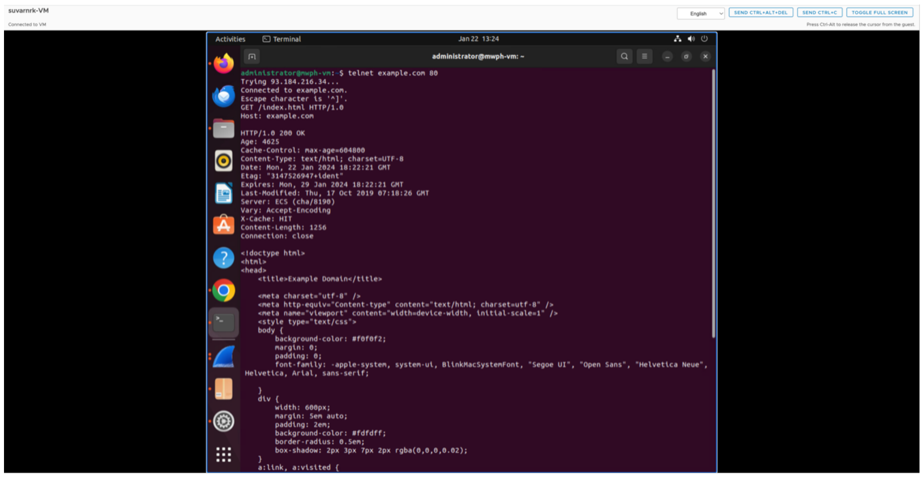
1 client pkt, 1 server pkt, 1 turn.

Entire conversation (1,981 bytes) Show data as ASCII

Figure 4: Wireshark HTTP Stream

Task 2. Understanding HTTP using telnet and Wireshark

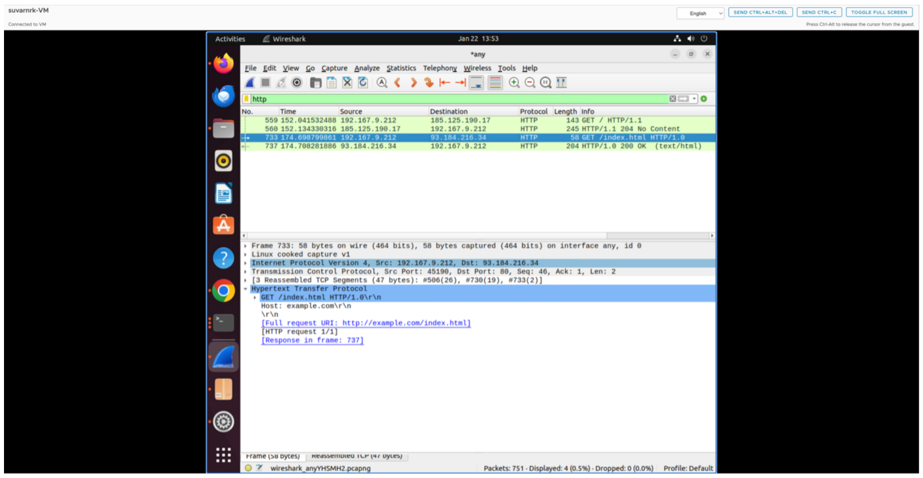
Wireshark was started to capture the network packets before making the HTTP request to exmaple.com/index.html via TELNET through the terminal. For using the TELNET first the connection was established to the exmaple.com webserver through the syntax telnet example.com portNumber. After the connection 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.



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

HTTP/1.0 200 OK
Age: 4625
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Mon, 22 Jan 2024 18:22:21 GMT
Etag: "3147526947+ident"
Expires: Mon, 29 Jan 2024 18:22:21 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (cha/8190)
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;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
```

Figure 5: Telnet request

Telnet, a network application utilizing the telnet protocol, facilitates TCP connections to servers for data exchange. In this task we will be requesting the example.com/index.html through telnet instead of using browser request. Connection should be established firstly between the telnet and example.com webserver using command “telnet example.com 80” here 80 is the port number. After a successful connection is established, GET command and Host commands are entered manually to get the required response.



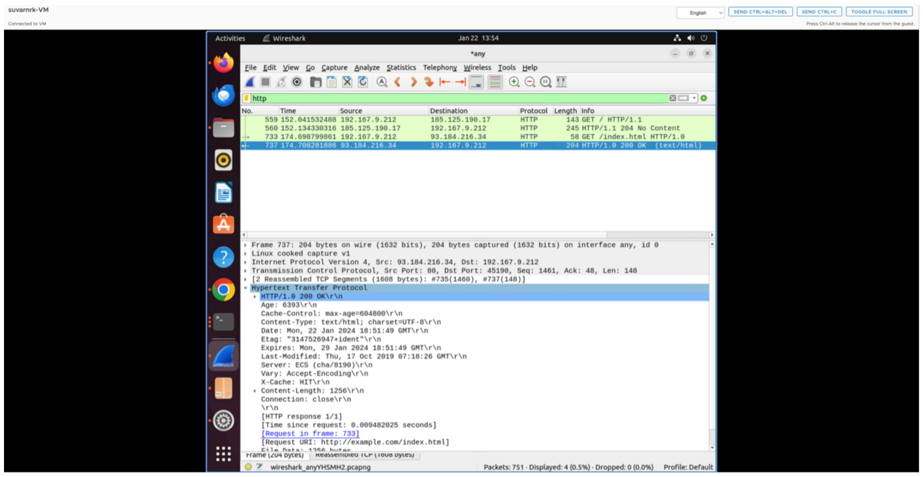
```
No.  Time             Source          Destination     Protocol Length Info
 559 152.041532488 192.167.9.212   185.125.190.17  HTTP      143 GET / HTTP/1.1
 560 152.134330316 185.125.190.17  192.167.9.212   HTTP      245 HTTP/1.1 204 No Content
 733 174.698799861 192.167.9.212   93.184.216.34   HTTP       58 GET /index.html HTTP/1.0
 737 174.708281886 93.184.216.34   192.167.9.212   HTTP      204 HTTP/1.0 200 OK  (text/html)
```

```
Frame 733: 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: 192.167.9.212, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 45190, Dst Port: 80, Seq: 46, Ack: 1, Len: 2
[3 Reassembled TCP Segments (47 bytes): #506(26), #730(19), #733(2)]
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: 737]
```

suvarnrk-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Activities
Wireshark
Jan 22 13:53
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Frame (58 bytes)
Reassembled TCP (47 bytes)
wireshark_anyYHSMH2.pcapng
Packets: 751 · Displayed: 4 (0.5%) · Dropped: 0 (0.0%)
Profile: Default

Figure 6: Telnet request in wireshark

The HTTP replies in Wireshark were identical whether viewed using a browser and TELNET.



```
No.  Time             Source          Destination     Protocol Length Info
 559 152.041532488 192.167.9.212   185.125.190.17  HTTP      143 GET / HTTP/1.1
 560 152.134330316 185.125.190.17  192.167.9.212   HTTP      245 HTTP/1.1 204 No Content
 733 174.698799861 192.167.9.212   93.184.216.34   HTTP       58 GET /index.html HTTP/1.0
 737 174.708281886 93.184.216.34   192.167.9.212   HTTP      204 HTTP/1.0 200 OK  (text/html)
```

```
Frame 737: 204 bytes on wire (1632 bits), 204 bytes captured (1632 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.167.9.212
Transmission Control Protocol, Src Port: 80, Dst Port: 45190, Seq: 1461, Ack: 48, Len: 148
[2 Reassembled TCP Segments (1608 bytes): #735(1460), #737(148)]
Hypertext Transfer Protocol
  HTTP/1.0 200 OK\r\n
  Age: 6393\r\n
  Cache-Control: max-age=604800\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  Date: Mon, 22 Jan 2024 18:51:49 GMT\r\n
  Etag: "3147526947+ident"\r\n
  Expires: Mon, 29 Jan 2024 18:51:49 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: 1256\r\n
  Connection: close\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.009482025 seconds]
  [Request in frame: 733]
  [Request URI: http://example.com/index.html]
  File Data: 1256 bytes
```

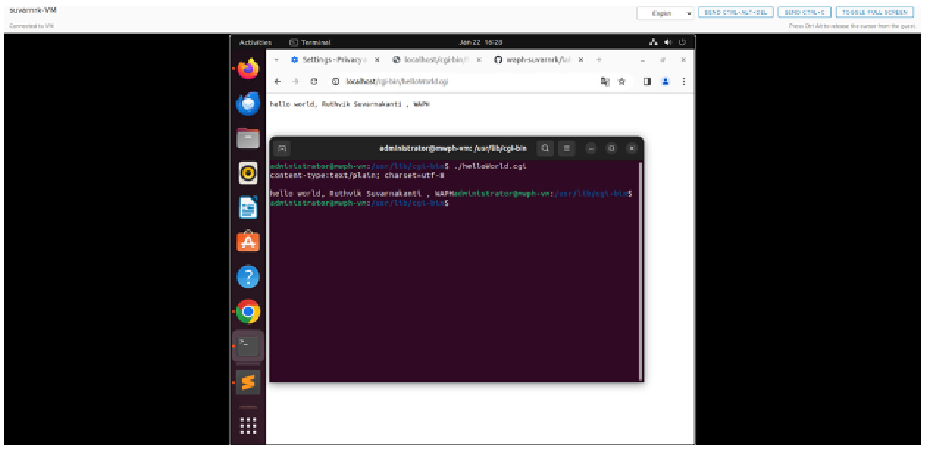
suvarnrk-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Activities
Wireshark
Jan 22 13:54
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Frame (204 bytes)
Reassembled TCP (1608 bytes)
wireshark_anyYHSMH2.pcapng
Packets: 751 · Displayed: 4 (0.5%) · Dropped: 0 (0.0%)
Profile: Default

Figure 7: Telent response in wireshark

Part 2 - Basic Web Application Programming

Task 1: CGI Web applications in C

A. CGI is a commonly used protocol allowing web servers to communicate with external applications by executing them as command-line programs. To write a C "Hello World!" CGI program, write a script that outputs "Hello World CGI! From Ruthvik Suvarnakanti, WAPH." The code may be run as a standard binary application and built using gcc installable with "$ sudo apt install gcc". After creating a helloworld.c program copy it to the helloworld.cgi in cgi-bin folder. After entering the commands, enter " http://localhost/cgi-bin/helloworld.cgi " you will get the response in the browser for which the code you entered.



```
hello world, Ruthvik Suvarnakanti , WAPH
```

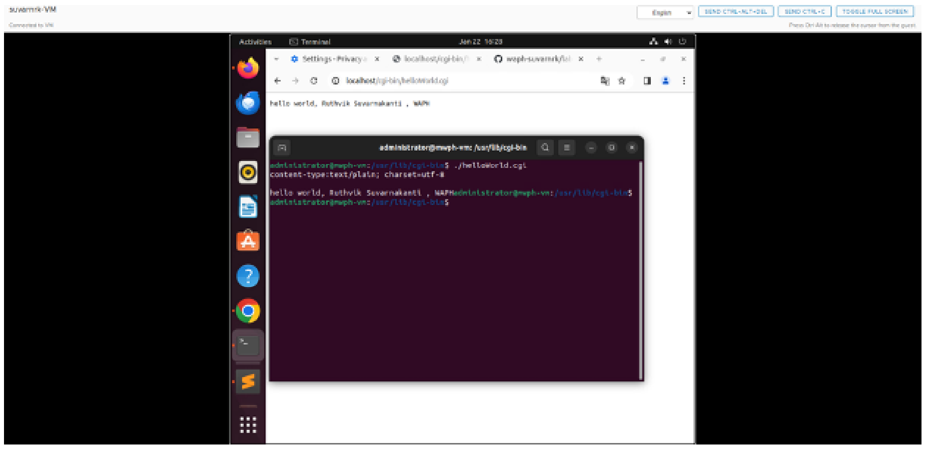
```
administrator@waph-vm:/usr/lib/cgi-bin$ ./helloWorld.cgi
content-type:text/plain; charset=utf-8

hello world, Ruthvik Suvarnakanti , WAPHadministrator@waph-vm:/usr/lib/cgi-bin$
administrator@waph-vm:/usr/lib/cgi-bin$
```

Figure 8: CGI program in C

B. I enhanced my CGI programming in C by integrating a basic HTML template featuring the course name as the title and student details. After compiling the code with gcc, I placed the resulting file in usr/lib/cgi-bin for browser access.

Included file `helloworld.c`:

```
    #include<stdio.h>
    int main() {
    const char *htmlContent = "<!DOCTYPE html> <html> <head> <title>Web Application Programm
                              "</head> <body> <h1>Student: Ruthvik Suvarnakanti</h1>"
                              "<p>This exercise is done as part of Lab1 assessment i.e CGI W

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

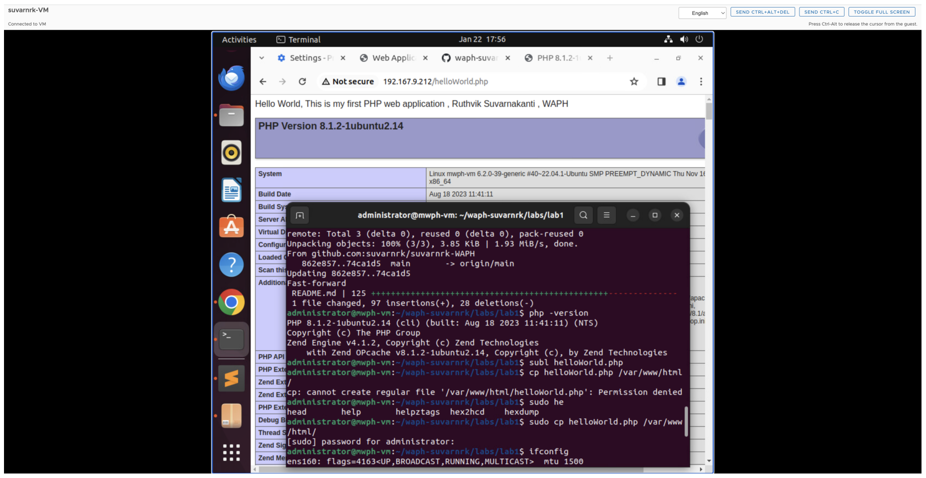

```
hello world, Ruthvik Suvarnakanti , WAPH
```

```
administrator@mwph-vm:/usr/lib/cgi-bin$ ./helloWorld.cgi
content-type:text/plain; charset=utf-8

hello world, Ruthvik Suvarnakanti , WAPHadministrator@mwph-vm:/usr/lib/cgi-bin$
administrator@mwph-vm:/usr/lib/cgi-bin$
```

Figure 9: CGI in C and HTML

Task 2: A simple PHP Web Application with user input.

A. In the labs/lab1 folder of your private repository, create a new file named helloworld.php. The file should contain content that utilizes the echo language construct in PHP for printing strings from texts or expressions. The echo construct can be employed with or without parentheses, wrapping texts within either double or single quotes. Additionally, phpinfo() is included to display PHP information on the server for testing purposes only. Deploy the code to the root directory of the webserver using the following commands$ cd $ sudo cp helloworld.php /var/www/html Browsers should be able to view the deployed page at http://<192.167.9.212>/helloworld.php.


```
Hello World, This is my first PHP web application , Ruthvik Suvarnakanti , WAPH
```

PHP Version 8.1.2-1ubuntu2.14

System	Linux mwph-vm 6.2.0-39-generic #40~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 16 x86_64
Build Date	Aug 18 2023 11:41:11

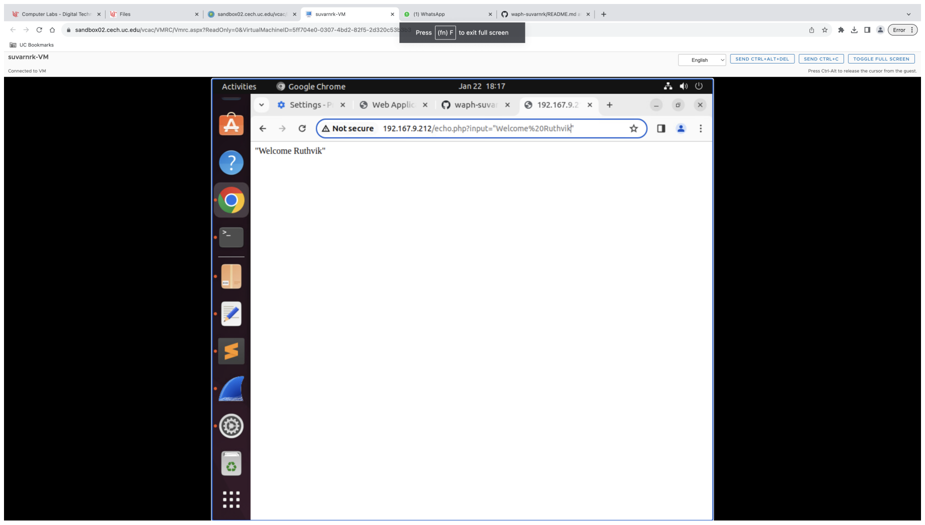
```
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 3.85 KiB | 1.93 MiB/s, done.
From github.com:suvarnrk/suvarnrk-WAPH
   862e857..74ca1d5  main       -> origin/main
Updating 862e857..74ca1d5
Fast-forward
 README.md | 125 +++++++++++++++++++++++++++++++++++++++++++++++++-------------
 1 file changed, 97 insertions(+), 28 deletions(-)
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ php -version
PHP 8.1.2-1ubuntu2.14 (cli) (built: Aug 18 2023 11:41:11) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.14, Copyright (c), by Zend Technologies
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ subl helloWorld.php
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ cp helloWorld.php /var/www/html
/
cp: cannot create regular file '/var/www/html/helloWorld.php': Permission denied
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ sudo he
head       help       helpztags  hex2hcd    hexdump
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ sudo cp helloWorld.php /var/www
/html/
[sudo] password for administrator:
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

Figure 10: helloWorld.php

Included file `helloworld.php`:

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

B. Using the echo function, I've developed a simple PHP web application that outputs the path variable obtained from HTTP queries. It's crucial to remember that using PHP's $_REQUEST('data') to record path variables in GET and POST requests exposes you to security flaws including remote code execution, SQL injections, and data manipulation. It is essential to apply input validation, use prepared statements for SQL inputs, and sanitize user inputs in order to reduce these risks.



```
192.167.9.212/echo.php?input="Welcome%20Ruthvik"
```

```
"Welcome Ruthvik"
```

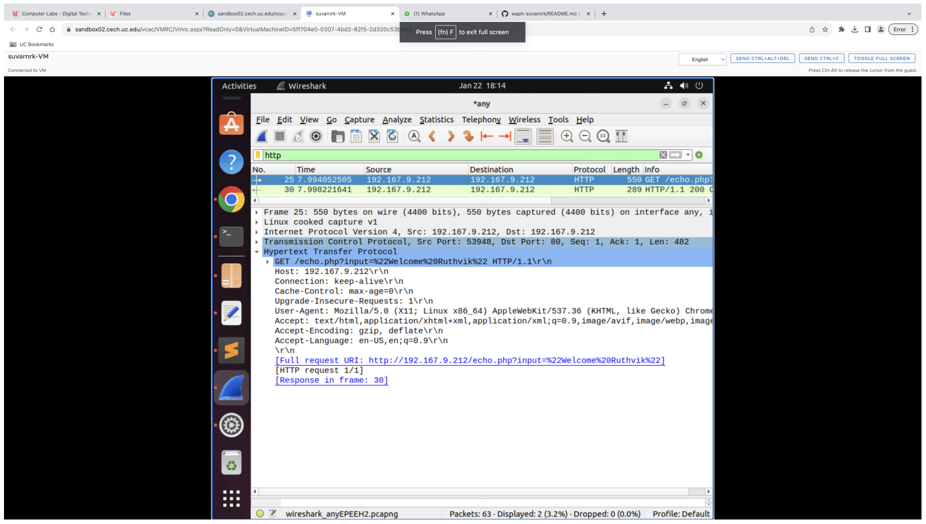
Figure 11: echo.php

Included file `echo.php`:

```php
<?php
    $inputData = $_REQUEST["input"];
    echo $inputData ;
?>
```

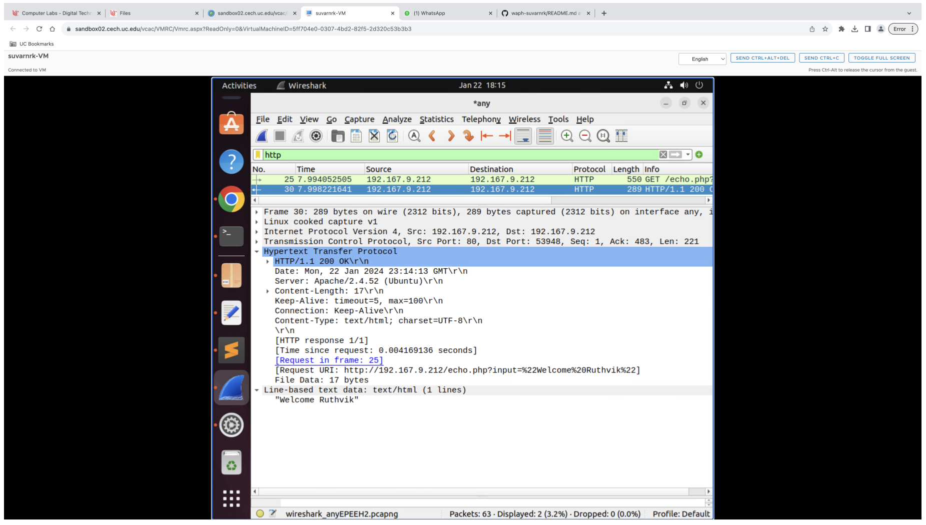
Task 3: Understanding HTTP GET and POST requests.

A. The call made by the browser was a standard HTTP GET request, and it used the "?" character in the URL syntax i.e., IPaddress/echo.php?input="value" to send the path variable. The result included the input variable after that. Wireshark was used to examine the request itself, the response, and HTTP stream in its entirety.



```
No.  Time         Source         Destination    Protocol Length Info
  25 7.994052505  192.167.9.212  192.167.9.212  HTTP        550 GET /echo.php?
  30 7.998221641  192.167.9.212  192.167.9.212  HTTP        289 HTTP/1.1 200 O

Frame 25: 550 bytes on wire (4400 bits), 550 bytes captured (4400 bits) on interface any, i
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.167.9.212, Dst: 192.167.9.212
Transmission Control Protocol, Src Port: 53948, Dst Port: 80, Seq: 1, Ack: 1, Len: 482
Hypertext Transfer Protocol
  GET /echo.php?input=%22Welcome%20Ruthvik%22 HTTP/1.1\r\n
  Host: 192.167.9.212\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  \r\n
  [Full request URI: http://192.167.9.212/echo.php?input=%22Welcome%20Ruthvik%22]
  [HTTP request 1/1]
  [Response in frame: 30]
```

Figure 12: HTTP GET request in WireShark

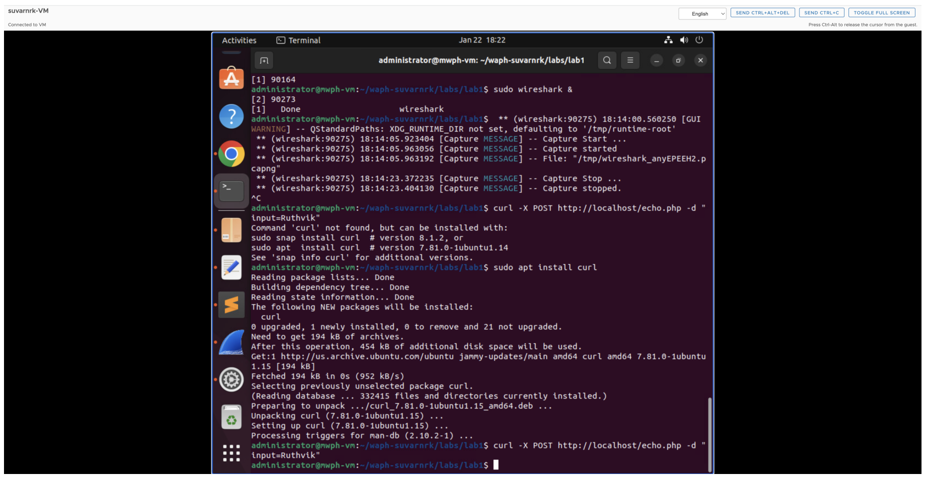


```
No.  Time         Source         Destination    Protocol Length Info
  25 7.994052505  192.167.9.212  192.167.9.212  HTTP        550 GET /echo.php?
  30 7.998221641  192.167.9.212  192.167.9.212  HTTP        289 HTTP/1.1 200 O

Frame 30: 289 bytes on wire (2312 bits), 289 bytes captured (2312 bits) on interface any, i
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.167.9.212, Dst: 192.167.9.212
Transmission Control Protocol, Src Port: 80, Dst Port: 53948, Seq: 1, Ack: 483, Len: 221
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Mon, 22 Jan 2024 23:14:13 GMT\r\n
  Server: Apache/2.4.52 (Ubuntu)\r\n
  Content-Length: 17\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: 0.004169136 seconds]
  [Request in frame: 25]
  [Request URI: http://192.167.9.212/echo.php?input=%22Welcome%20Ruthvik%22]
  File Data: 17 bytes
Line-based text data: text/html (1 lines)
  "Welcome Ruthvik"
```

Figure 13: HTTP response in WireShark

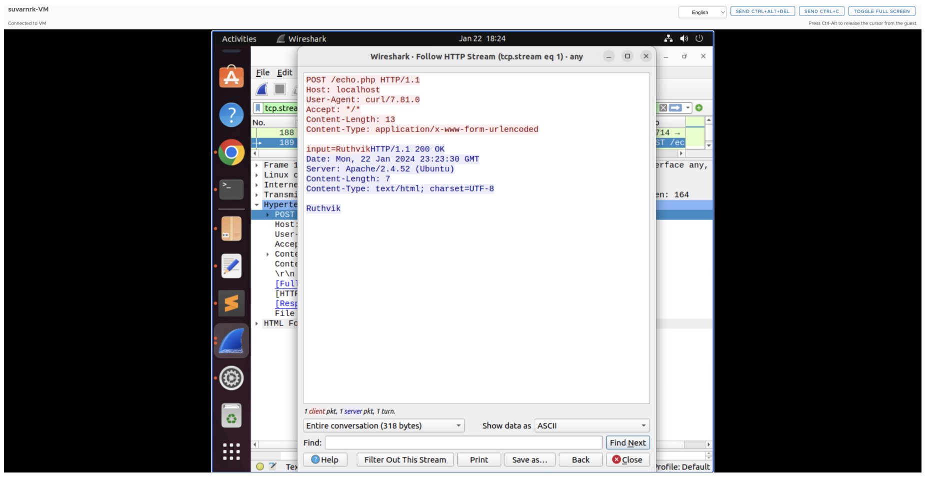
B. A command-line program called Client URL (cURL) is intended for data processing over various network protocols. In this case, I sent a POST call to the echo.php endpoint using cURL in the terminal. This required utilizing the cURL command-line tool to deliver data to the server using the HTTP POST protocol.

curl -X POST localhost/echo.php -d “input= Ruthvik”



```
[1] 90164
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ sudo wireshark &
[2] 90273
[1]   Done                    wireshark
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$  ** (wireshark:90275) 18:14:00.560250 [GUI
WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
 ** (wireshark:90275) 18:14:05.923404 [Capture MESSAGE] -- Capture Start ...
 ** (wireshark:90275) 18:14:05.963056 [Capture MESSAGE] -- Capture started
 ** (wireshark:90275) 18:14:05.963192 [Capture MESSAGE] -- File: "/tmp/wireshark_anyEPEEH2.p
capng"
 ** (wireshark:90275) 18:14:23.372235 [Capture MESSAGE] -- Capture Stop ...
 ** (wireshark:90275) 18:14:23.404130 [Capture MESSAGE] -- Capture stopped.
^C
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ curl -X POST http://localhost/echo.php -d "
input=Ruthvik"
Command 'curl' not found, but can be installed with:
sudo snap install curl  # version 8.1.2, or
sudo apt  install curl  # version 7.81.0-1ubuntu1.14
See 'snap info curl' for additional versions.
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$ sudo apt install curl
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 21 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 (952 kB/s)
Selecting previously unselected package curl.
(Reading database ... 332415 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-suvarnrk/labs/lab1$ curl -X POST http://localhost/echo.php -d "
input=Ruthvik"
administrator@mwph-vm:~/waph-suvarnrk/labs/lab1$
```

Figure 14: HTTP POST request using CURL



```
POST /echo.php HTTP/1.1
Host: localhost
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

input=RuthvikHTTP/1.1 200 OK
Date: Mon, 22 Jan 2024 23:23:30 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 7
Content-Type: text/html; charset=UTF-8

Ruthvik
```

Figure 15: HTTP Stream in Wireshark

C.An analysis tool such as Wireshark may be used to compare the similarities between HTTP GET and POST requests, which are essential for client-server communication and contain request headers and response codes. Interestingly, they communicate data differently: POST uses the HTTP body, whereas GET uses the URL. GET is usually used for retrieving, whereas POST is used for updating, and POST is seen to be more secure. The GET and POST replies in the echo.php web application are the same. The project report also had a Labs/Lab1 folder created for it, and the README.md file was used to construct the report using the Pandoc tool.