

WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Bhargavi Murari



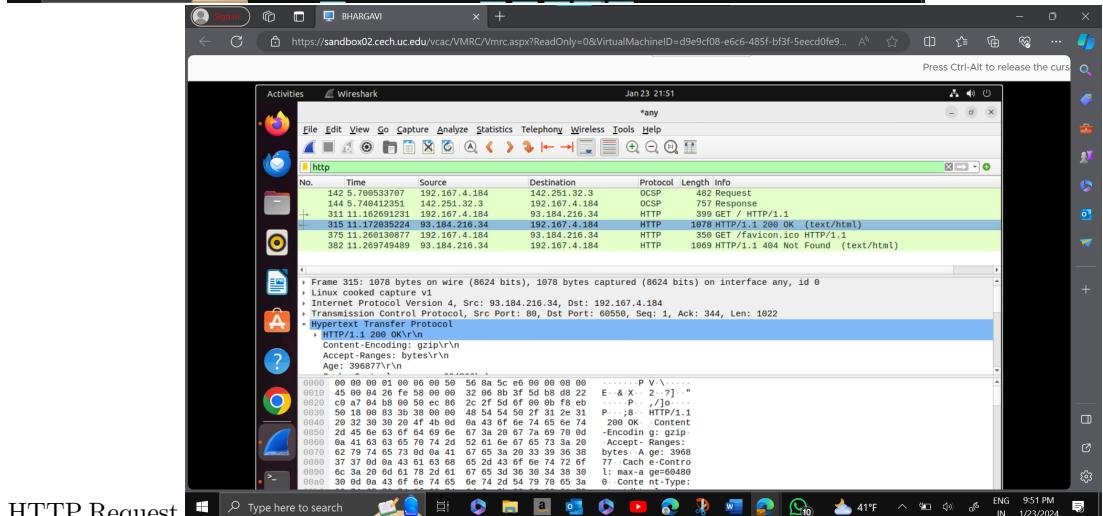
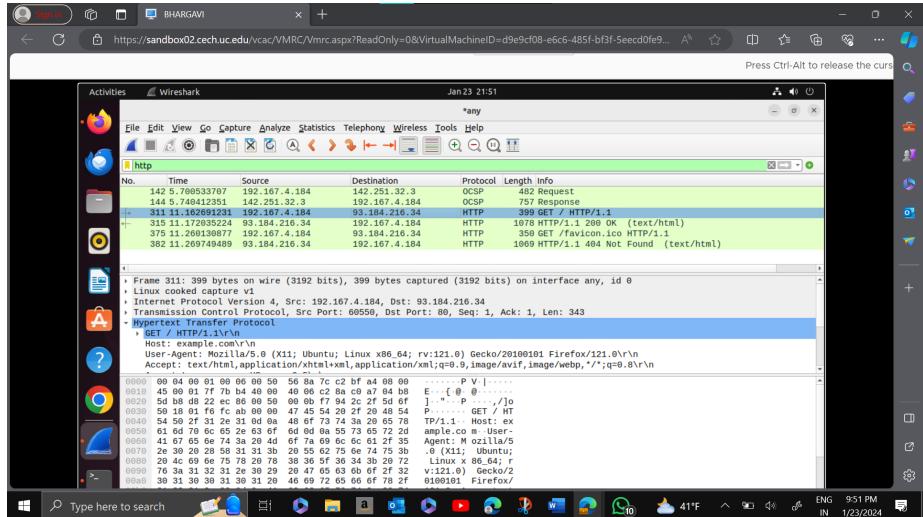
Email: muraribi@mail.uc.edu

Repository Information
Repository's URL: <https://github.com/BhargaviMurari22/waph-muraribi.git>
Lab's Report:
Lab 1 - Foundations of the Web
The lab's overview and outcomes:

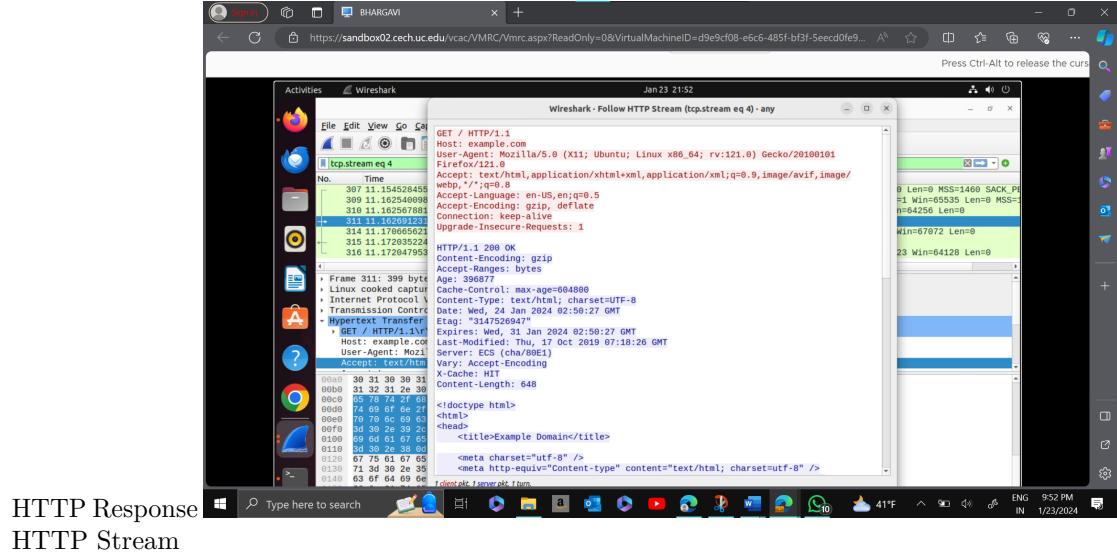
Part I - The Web and HTTP Protocol

Task 1. Familiar with the Wireshark tool and HTTP protocol

I used Mozilla Firefox to search for “http://example.com” on a website and started Wireshark using the “any” option. Once the HTTP stream was accessible, Wireshark showed the HTTP requests and responses. Through the capture and presentation of the network traffic related to the web browsing activity, this made it possible to examine in detail the communication that took place between my machine and the web server.



HTTP Request



HTTP Response
HTTP Stream

Task 2. Understanding HTTP using telnet and Wireshark

To use Telnet to perform HTTP tests, I entered “\$telnet example.com 80” on my terminal. Following that, I typed the instructions’ “GET /index.html HTTP/1.0” and “Host: example.com” commands. The HTTP request I had made and the response I received from the server were then shown by the command prompt, giving me a glimpse of the exchange that was taking place between the computer and the example.com web server.

1. A screenshot of your terminal showing the HTTP Request and HTTP response from the server.

```

administrator@molph-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

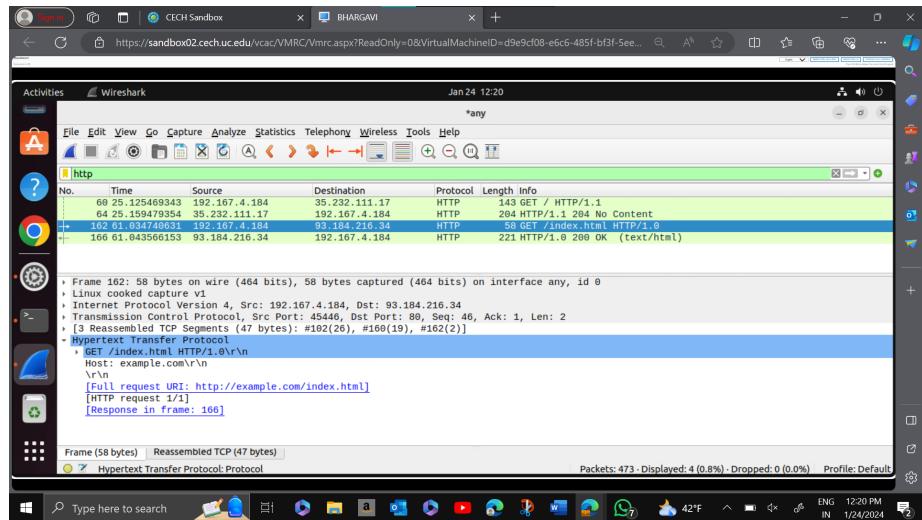
HTTP/1.0 200 OK
Accept-Ranges: bytes
Age: 1789
Content-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Wed, 24 Jan 2024 17:17:46 GMT
Etag: "3147526947"
Expires: Wed, 31 Jan 2024 17:17:46 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (cfa/80E01)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close

<!doctype html>
<html>
<head>
<title>Example Domain</title>
</head>
<body>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />

```

Terminal with Telnet

2. REQUEST: The HTTP request sent in Wireshark Task 1 is similar to the snapshot that was taken using Telnet. Although Wireshark offers more detailed information, Telnet queries still include necessary components such as the host site, HTTP version, and “GET” method. Beyond the fundamental elements observed in Telnet, Wireshark’s expert details include information regarding the “GET” request, severity levels, and arranged groupings.



Telnet Request

3. RESPONSE: The host site’s HTTP version and a successful “200 OK” status are displayed in the Telnet snapshot, which is a reflection of the HTTP response from Wireshark Task 1. Nevertheless, Wireshark goes above and beyond by offering a wealth of information, such as Response version, Status code, Response Phrase, content type, date of update, content length, and time and date of expiration. All in all, Wireshark provides a more thorough analysis of every answer line than Telnet does for the crucial information.

Part II - Basic Web Application Programming

Task 1: CGI Web applications in C

- a. I edited my C program in Sublime Text, compiled it with “gcc,” and ran the executable on the web server using specific commands. The iterative process ensured code modifications were tested and deployed effectively.

```
#include <stdio.h>
int main(void) {
printf("Content-Type: text/plain; charset=utf-8\n\n");
```

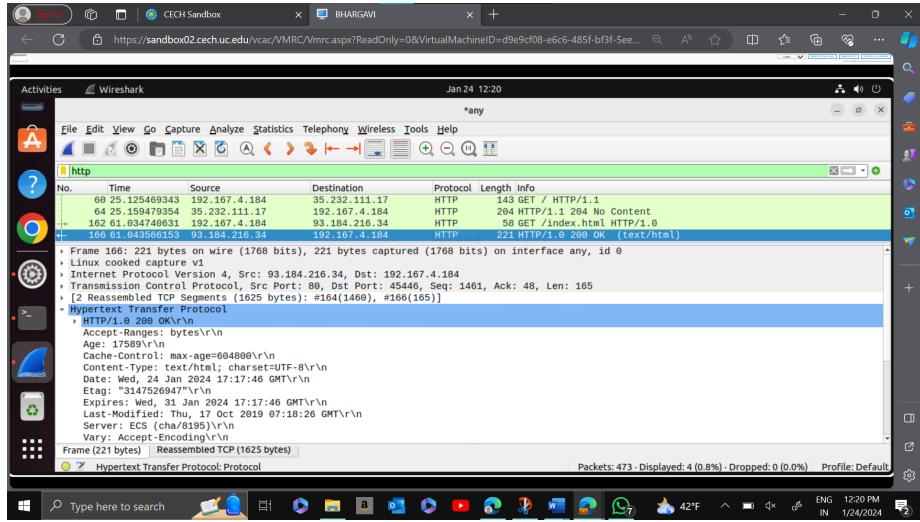


Figure 1: IMAGE-6

```
printf("Hello World CGI! From BHARGAVI, WAPH\n\n");
return 0;
}
```

- b. The “index.c” file’s structure is shown in a screenshot, which also features a URL the professor gave.

```
#include<stdio.h>
int main(void)
{
printf("Content-Type: text/html\n\n");
printf("<html><head><h1>Name: BHARGAVI MURARI</h1><h2>DEpartment: INFORMATION TECHNOLOGY</h2>");
return 0;
}
```

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

- a. I created a “helloworld.php” file in Sublime Text, used the command “\$ sudo cp helloworld.php /var/www/html” to move it to the web server, then used a browser to view it locally.

```
<?php
echo "Hello World, this is the first PHP by Bhargavi, WAPH";
phpinfo();
?>
```

- b. I created a “echo.php” file in Sublime Text, used the command “\$ sudo cp echo.php /var/www/html” to move it to the web server, then used a

A screenshot of a Linux desktop environment. At the top, there is a horizontal bar with icons for user sign-in, network, and system status. Below this is a dock with various application icons. In the center, there is a terminal window titled "Activities" with the title "Firefox Web Browser". The terminal shows the following command-line session:

```
administrator@mwph-vm: ~/waph-muraribl/labs/lab1
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ gcc helloworld.c -o helloworld.cgi
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ ./helloworld.cgi
Content-Type: text/plain; charset=utf-8
Hello World CGI! From BHARGAVI, WAPH

[sudo] password for administrator:
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ sudo cp helloworld.cgi /usr/lib/cgi-bin
[sudo] password for administrator:
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ 
```

Below the terminal is a web browser window titled "localhost/cgi-bin/helloworld.cgi" which displays the output of the CGI script: "Hello World CGI! From BHARGAVI, WAPH".

Figure 2: IMAGE-7

A screenshot of a Linux desktop environment, similar to Figure 2. It features a terminal window and a web browser window. The terminal window shows the following command-line session:

```
administrator@mwph-vm: ~/waph-muraribl/labs/lab1
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ gcc index.c -o index.cgi
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ ./index.cgi
Content-Type: text/html
<html><head><title>BHARGAVI MURARI</title></head><body><h1>Name: BHARGAVI MURARI</h1><h2>DEPARTMENT: INFORMATION TECHNOLOGY</h2><p>This is web programming Hands-on lab</p></body></html>
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ sudo cp index.cgi /usr/lib/cgi-bin
administrator@mwph-vm: ~/waph-muraribl/labs/lab1$ 
```

The web browser window titled "localhost/cgi-bin/index.cgi" displays the generated HTML content, showing the name "BHARGAVI MURARI" and the department "INFORMATION TECHNOLOGY", along with the message "This is web programming Hands-on lab".

Figure 3: IMAGE-8

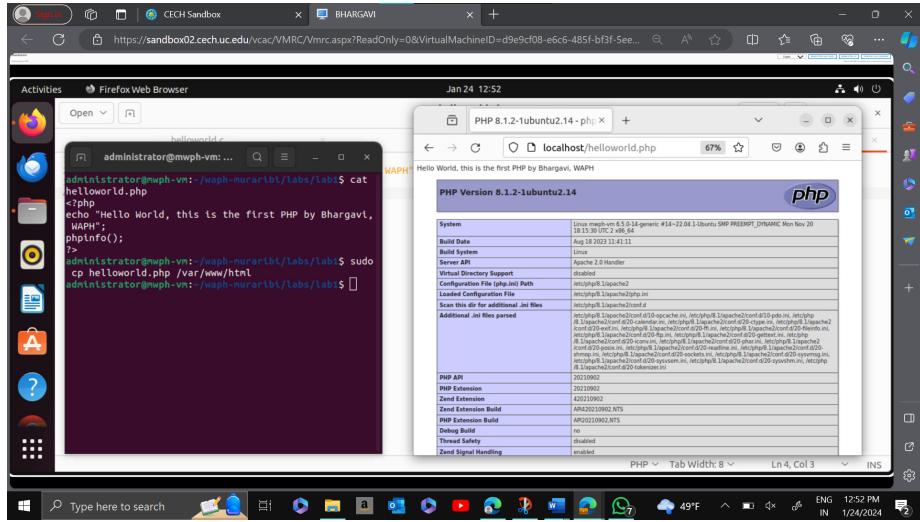


Figure 4: IMAGE-9

browser to view it locally.

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

Task 3 (10 pts). Understanding HTTP GET and POST requests.

- Using Wireshark, I examined the POST and GET HTTP requests made by the “echo.php” application. I used Wireshark to record from any network before starting the server, and I took screenshots while examining and recording the requests and responses using the http filter.

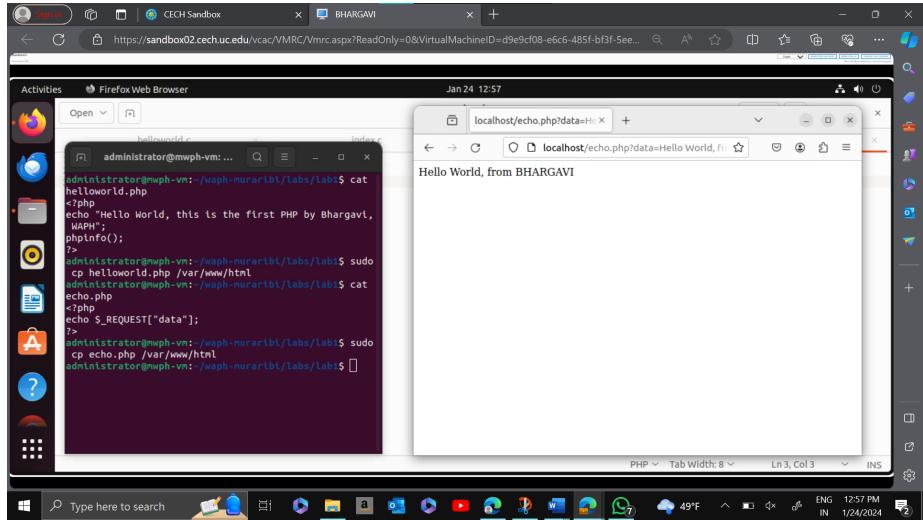
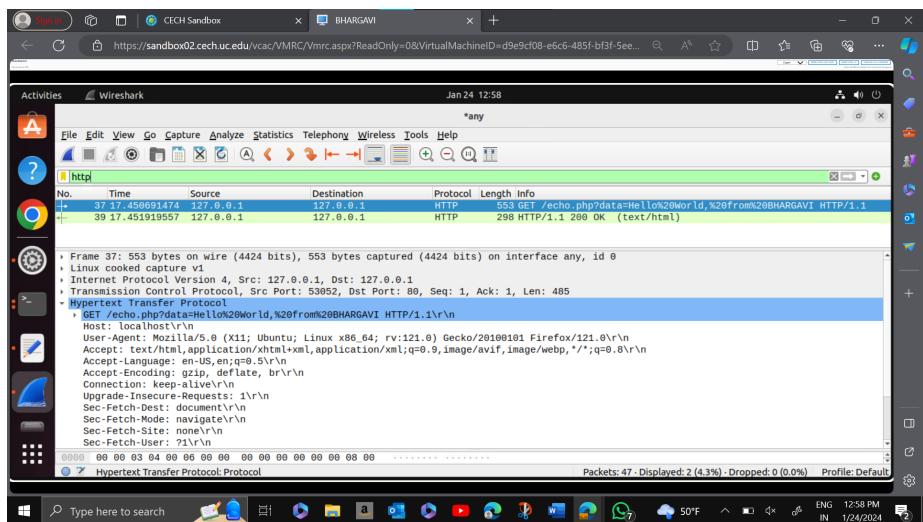
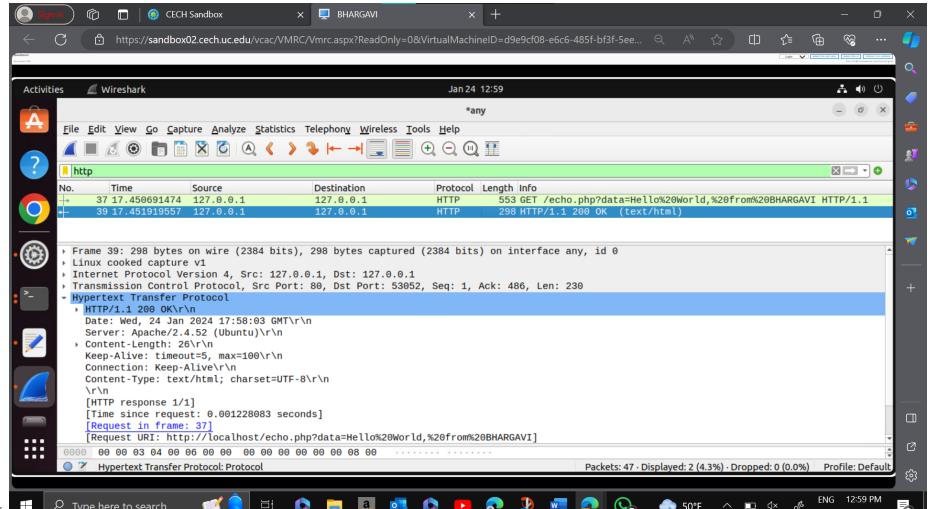


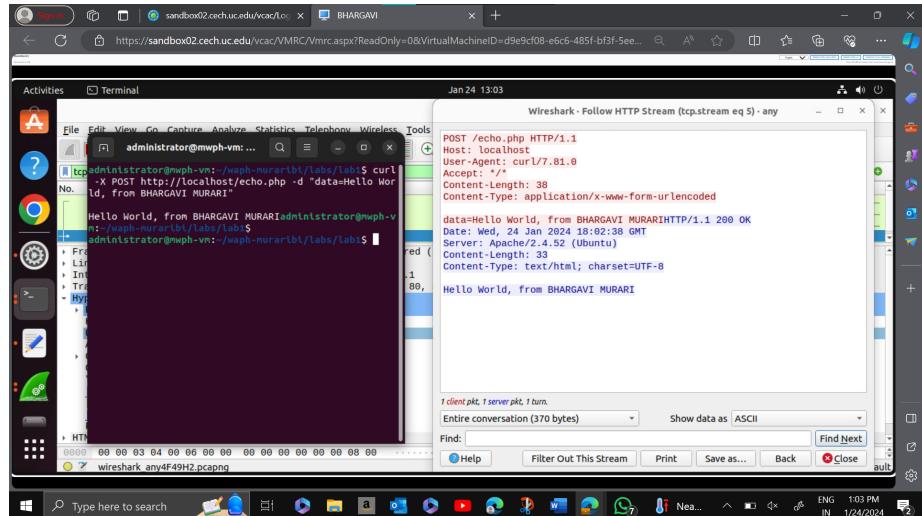
Figure 5: IMAGE-10





Echo REQUEST Screenshot
Echo RESPONSE Screenshot

- b. After using “sudo apt install curl” to install Curl on Ubuntu, I ran a command, as seen in the screenshot of the command prompt. I was able to view the HTTP request and response using Wireshark by right-clicking on the request and choosing the HTTP stream. Refer to the screenshot that is included.



Echo STREAM Screenshot

- c. Similarities and Differences: Headers add extra information during the HTTP request and response cycle, including POST. POST is not visible in the address bar; GET is preferred for smaller data; and POST is used for data submission; GET is used for retrieval with parameters in the URL.

These are the main distinctions.

Submission

Use the `pandoc` tool to generate the PDF report for submission from the `README.md` file, make sure that the report and contents are rendered properly.

The PDF file should be named `your-username-waph-lab1.pdf`, e.g., `phungph-waph-lab1.pdf`