

# **WAPH-Web Application Programming and Hacking**

**Instructor: Dr. Phu Phung**

**Student**

**Name:** Ruthvik Suvarnakanti

**Email:** suvarnrk@mail.uc.edu

## **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 browser sent requests. Moving on to the web application programming I got familiarized with development 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 generate 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**

Wireshark is a protocol Analyzer tool which is used for network analysis and troubleshooting. After installing the wirshark in Ubuntu VM , I have clicked on the capture options (4th icon) selected any on the input interface and enabled the promiscous mode on filters checkbox. Now started to capture the packets and filtered HTTP protocol among all the other protocols.Then , respective HTTP requests and responses are selected to analyze and observe the HTTP stream. The HTTP request gives the information about the type of request, target URL , HTTP version, content-type , authorization and the data that is sent to the web servers. The HTTP response gives the information about status code, status text , content-type and the data that is sent back to the web browser etc.

### **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.



Figure 1: Ruthvik Suvarnakanti



Figure 2: Wireshark HTTP Request

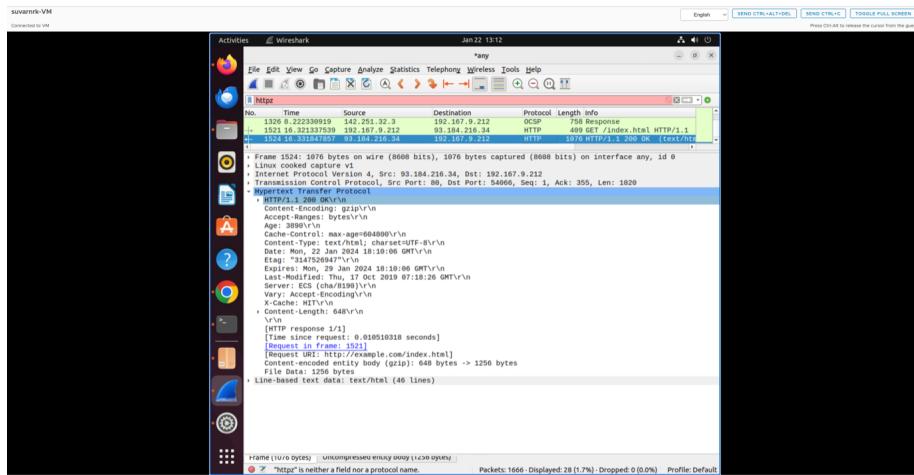


Figure 3: Wireshark HTTP Response

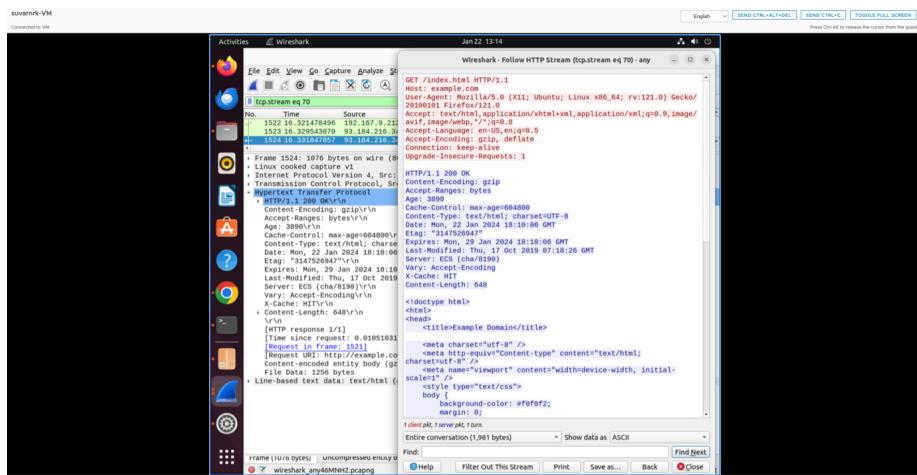


Figure 4: Wireshark HTTP Stream

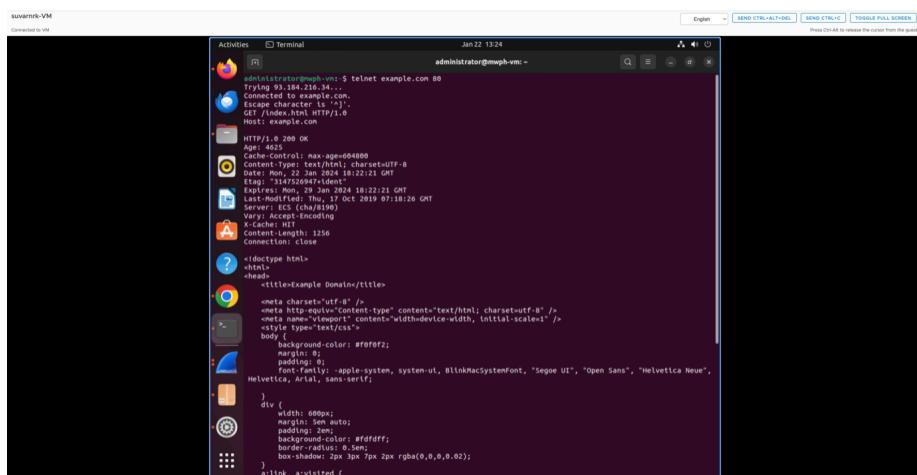


Figure 5: Telnet request

Comparing the HTTP requests through browser and TELNET in wireshark, it is noted that server fields were missing in the telnet made request . The telnet HTTP request is manually constructed where as in the browser sent request the browser automatically populates request headers such as user-agent, accept, accept-language, authorization , encoding and content etc.

#### Telnet request in wireshark

both the HTTP responses in wireshark through browser and TELNET were same.

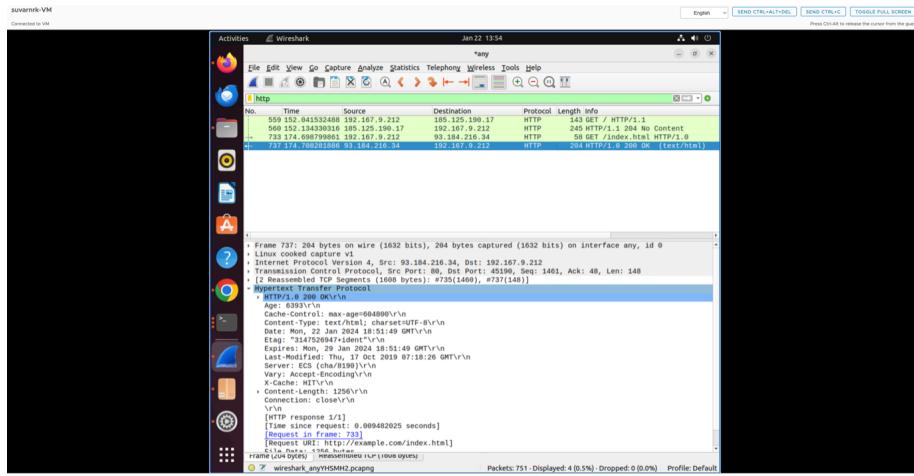


Figure 6: Telent response in wireshark

## Part II - Basic Web Application Programming

### Task 1: CGI Web applications in C

A. I have developed a basic CGI program in C which just prints Hello world! , I have compiled this using gcc and deployed the generated cgi file by copying it to `/usr/lib/cgi-bin` before accessing it on `localhost/cgi-bin/helloWorld.cgi` in the browser.

B. Now , I have developed another CGI programming in C, this time incorporating a basic HTML template in the C code . This template has the course name as title, student name as Heading and other details as paragraph. This file was also compiled using gcc and copied to `/usr/lib/cgi-bin` before accessing on the browser.

Included file `helloworld.c`:

```
#include<stdio.h>
int main() {
    const char *htmlContent = "<!DOCTYPE html> <html> <head> <title>Web Application Program">
```

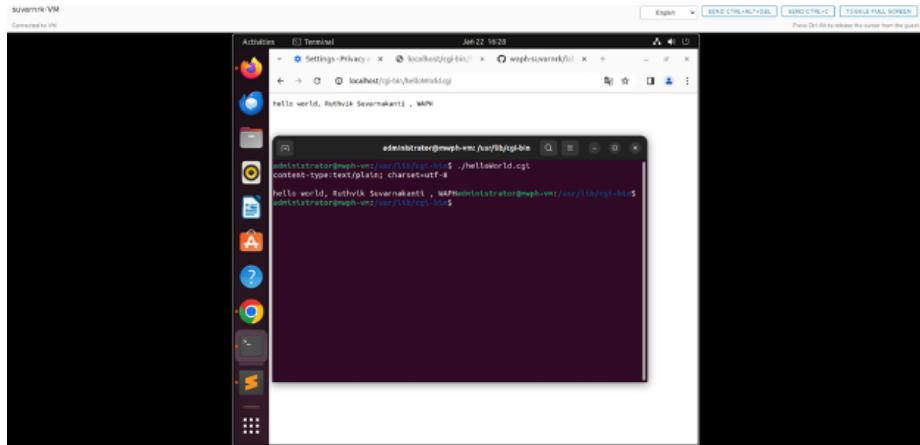


Figure 7: CGI program in C

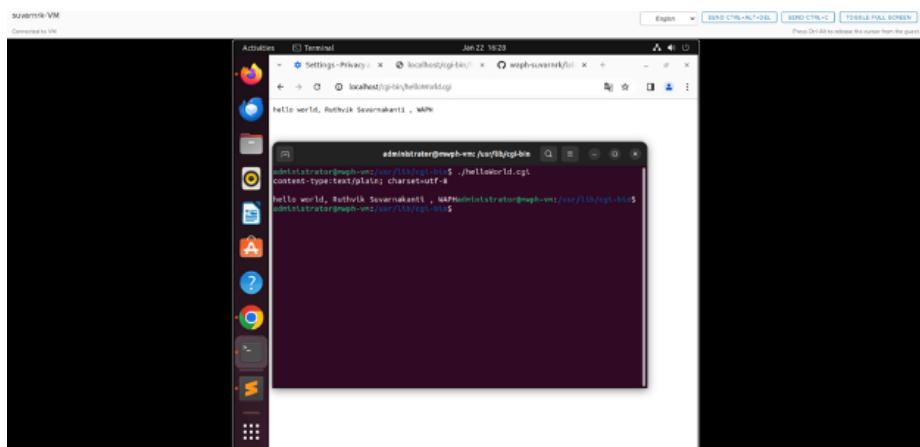


Figure 8: CGI in C and HTML

```

        "</head> <body> <h1>Student: Tulasiram Nakkanaboina</h1>
        "<p>This exercise is done as part of Lab1 assessment i.e CGI WAPPH

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

```

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

A. As part of this task , a PHP web application has been developed with basic syntax which includes my name and PHP version , deployed it to the root apache2 var/www/html directory . It was then accessed on the browser with the url IP address/helloworld.php

helloWorld.php

Included file helloworld.php:

```

<?php
    echo "Hello World! This is my first PHP program, Tulasiram Nakkanaboina , WAPH";
?>

```

B. A simple echo web application in PHP has been developed which prints the path variable passed through the http request. using `$_REQUEST['data']` in PHP for capturing the path variables in GET and POST requests possess various security vulnerabilities such as data tampering, SQL injections and Remote Code Execution. By implementing input validation , prepared statements for SQL inputs and sanitizing the usser inputs can mitigate these risks.

Included file echo.php:

```

<?php
    $inputData = $_REQUEST["data"];
    echo "The input from the request is <strong>" . $inputData . "</strong>.<br>";
?>

```

**Task 3: Understanding HTTP GET and POST requests.**

A.By default the call that was made through the browser was a HTTP GET call and the path variable was passed using ? in the URL IPaddress/echo.php?data=“value”. The input varaiable was then disaplyed as part of the response. This request, response and HTTP stream were analyzed through wireshark.

HTTP GET request in WireShark

HTTP response in WireShark

B.Client URL of CUrl is a command line tool for processing data using various n/w protocols. I have used CURL in terminal to make a post request to echo.php.

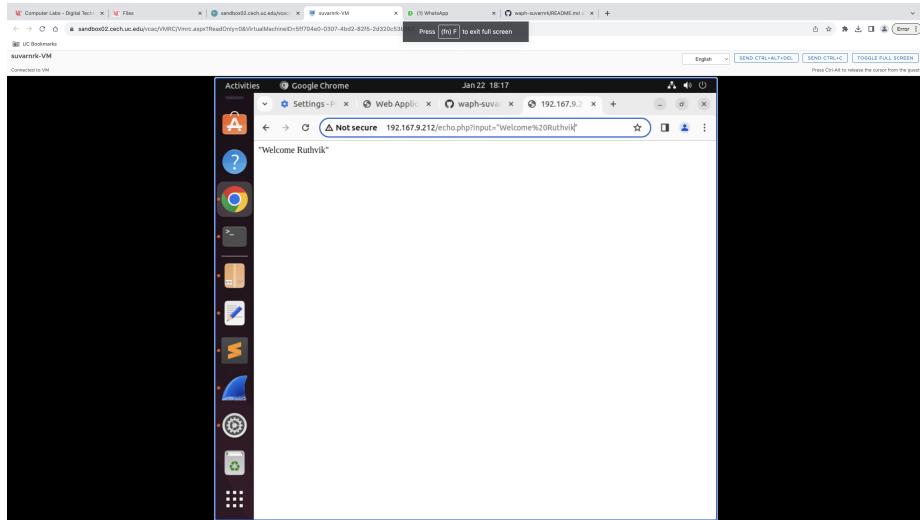


Figure 9: echo.php

```
curl -X POST localhost/echo.php -d "input= Ruthvik"
```

C.The similarities and differences between HTTP GET/ POST requests and responses Similarities: Both are HTTP methods used to communicate between client and server. Both can be captured and analyzed using tools like Wireshark. Both have request headers (though the headers might differ). Both receive responses with status codes, headers Differences: In GET request data is sent in the URL whereas in the POST request data is sent in the HTTP body. POST is more secure as data is not exposed to normal users. GET is used for retrieving the information, whereas POST is used to update the information .

as the echo.php web application is a mirror application , i.e just printing the received input , the responses in both the HTTP GET and HTTP POST are identical.

Post this Labs/Lab1 folder was created to accommodate the project report and the changes were pushed. Pandoc tool was used to generate the project report from the README.md file

```

Administrator@mvph-vm: /waph-suvarnjk/labs/lab1$ sudo wireshark &
Administrator@mvph-vm: /waph-suvarnjk/labs/lab1$ curl -X POST http://localhost/echo.php -d "InputRuthvik"
curl: command not found, but can be installed with:
  sudo snap install curl # version 8.1.2, or
  sudo apt install curl # version 7.8.1.0~ubuntu1.14
See 'snap info curl' for additional versions.
Reading package lists... Done
Building dependency tree... Done
Reading information from status file... 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 jannetty/main amd64 curl amd64 7.81.0~ubuntu1.14 [194 kB]
Fetched 194 kB in 0s (952 kB/s)
Selecting previously unselected package curl.
(Reading database ... 33,390 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0~ubuntu1.14_amd64.deb ...
Unpacking curl (7.81.0~ubuntu1.15) ...
Setting up curl (7.81.0~ubuntu1.15) ...
Processing triggers for man-db (2.8.3-2.1) ...
Administrator@mvph-vm: /waph-suvarnjk/labs/lab1$ curl -X POST http://localhost/echo.php -d "InputRuthvik"
Administrator@mvph-vm: /waph-suvarnjk/labs/lab1$ 

```

Figure 10: HTTP POST request using CURL

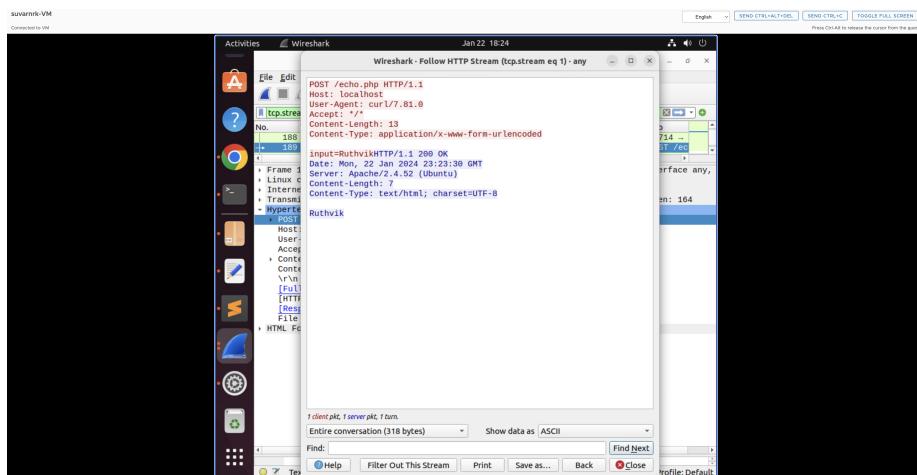


Figure 11: HTTP Stream in Wireshark