WAPH - Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Venkata Ramana Rao Devara
Email: devaravo@mail.uc.edu

Repository

Repository URL: https://www.github.com/devaravo/waph-devaravo/tree/main/labs/lab1.

Lab 1 - Foundations of the Web

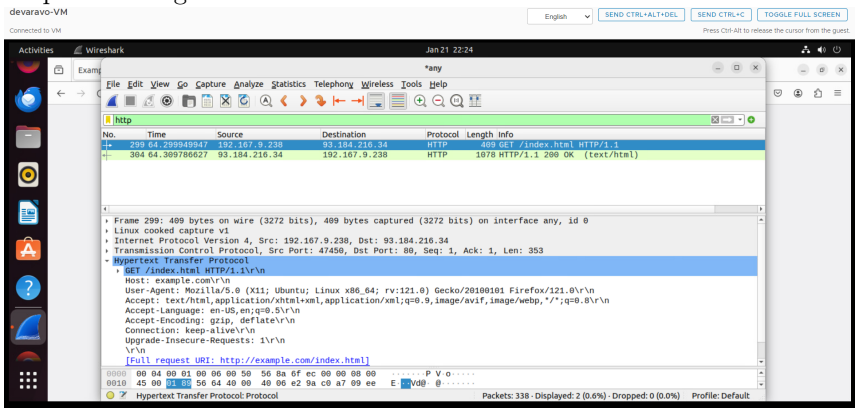
The lab focuses on http request and response messages using Wireshark and telnet, writing CGI programs and deploying them in Apache server, writing simple PHP echo program and testing it with GET and POST requests. It also covers the security risks associated with using REQUEST in PHP instead of dedicated GET and POST.

Part I: The Web and HTTP Protocol

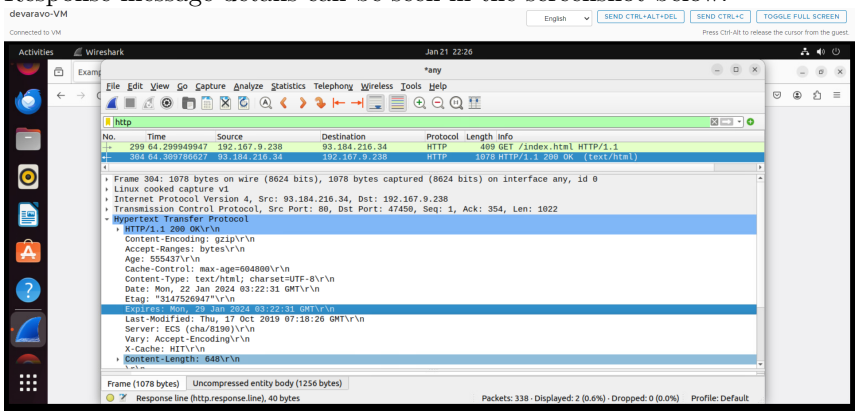
Task 1: Wireshark Tool and HTTP Protocol

- I have installed wireshark using “sudo apt install wireshark-qt”
- I ran wireshark using " sudo wireshark & " from the terminal
- After opening wireshark, I have configured settings to capture all network interfaces

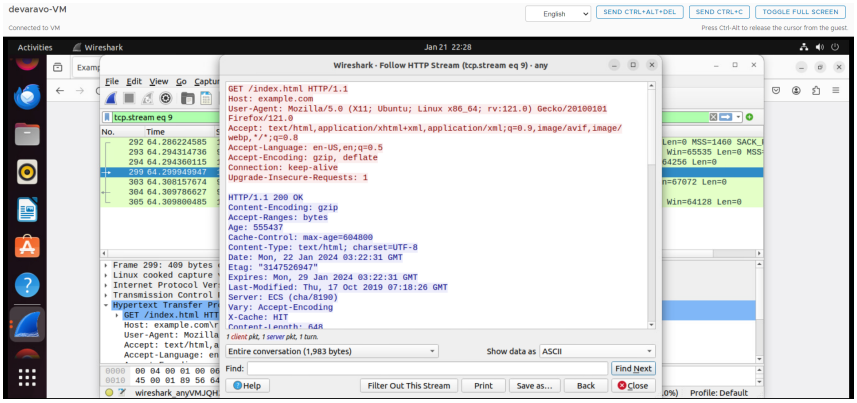
- I clicked on Start icon in wireshark
- I went into browser and browsed example.com
- I clicked on stop icon in wireshark and filtered http results
- Then I clicked on request and response messages to see details of respective packets.
- I also clicked on Follow and then clicked on http stream to view request and response messages.
- Request message details can be seen in the screenshot below:



- Response message details can be seen in the screenshot below:



- Stream of request and response messages can be seen in screenshot below:

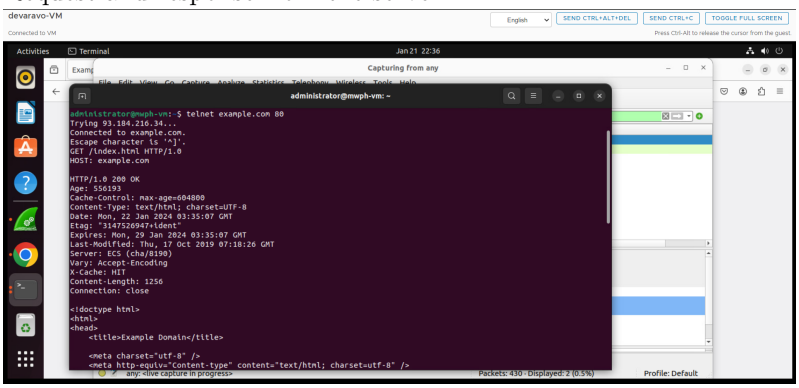


Task 2: HTTP using telnet and Wireshark

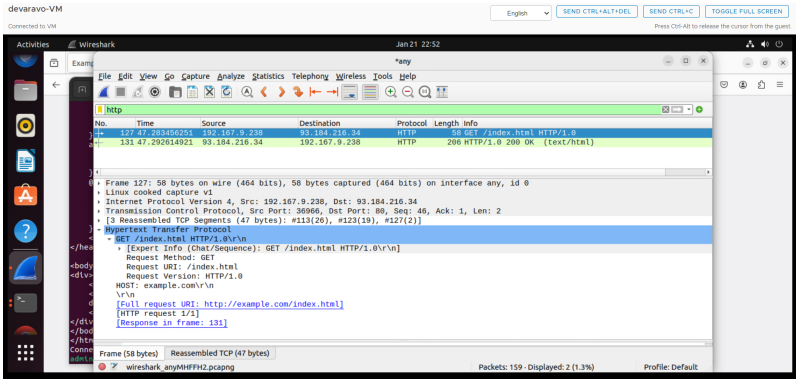
- I clicked on start icon in Wireshark
- I connected to server at port 80 by typing "telnet example.com 80" in terminal
- After the connection is established, I sent request by typing:

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

- I stopped Wireshark and observed http request, response messages in wireshark and terminal
- 1) Request and response from the server:

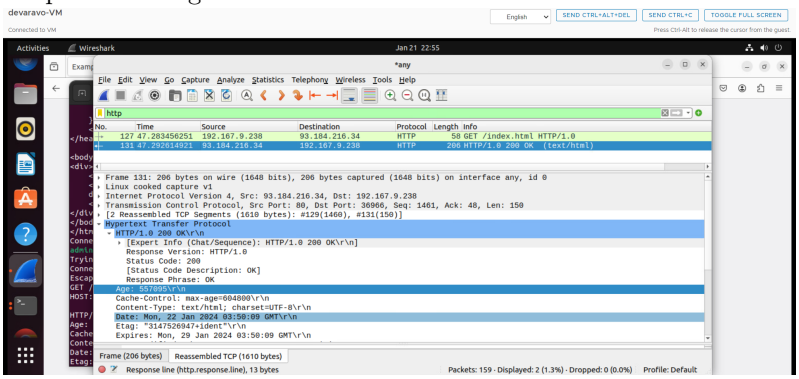


- 2) Request message in wireshark:



I have observed that the request message observed in Task 1 had additional fields when compared to request message in telnet. The following are additional fields: - User-Agent - Accept - Accept-Language - Accept-Encoding - Connection - Upgrade-Insecure-Requests

- 3) Response message in wireshark:



In response message sent in Task 1, I have observed additional field: Content-Encoding:gzip

Part II - Basic Web Application Programming

Task 1: CGI Applications in C

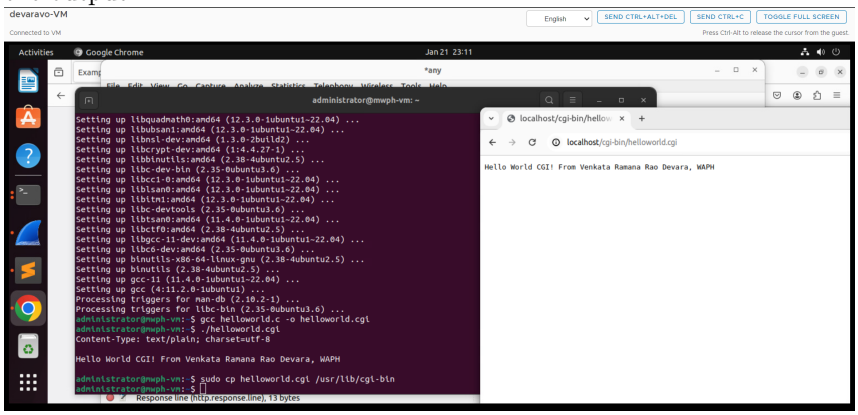
a)

- I created a new helloworld.c program
- I have installed gcc compiler
- I have compiled the program to cgi output file using " gcc helloworld.c -o helloworld.cgi"
- As cgi daemon is not enabled by default, I have enabled it and restarted apache server using the following commands:

```
sudo a2enmod cgid
```

```
sudo systemctl restart apache2
```

- I copied the cgi output file to /usr/lib/cgi-bin
- I opened http://localhost/cgi-bin/helloworld.cgi in browser and observed the output:



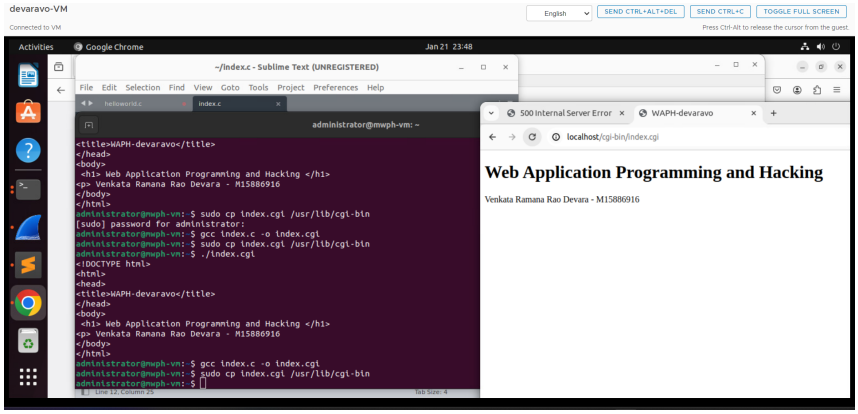
b)

- Similarly, I created c program index.c
- Program source code:

```
#include <stdio.h>
int main(void){
    printf("Content-Type: text/html; charset=utf-8\n\n");
    printf("<!DOCTYPE html>\n");
    printf("<html>\n");
    printf("<head>\n");
    printf("<title>WAPH-devaravo</title>\n");
    printf("</head>\n");
    printf("<body> \n ");
    printf("<h1> Web Application Programming and Hacking </h1>\n");
    printf("<p> Venkata Ramana Rao Devara - M15886916\n");
    printf("</body>\n");
    printf("</html>\n");


    return 0;
}
```

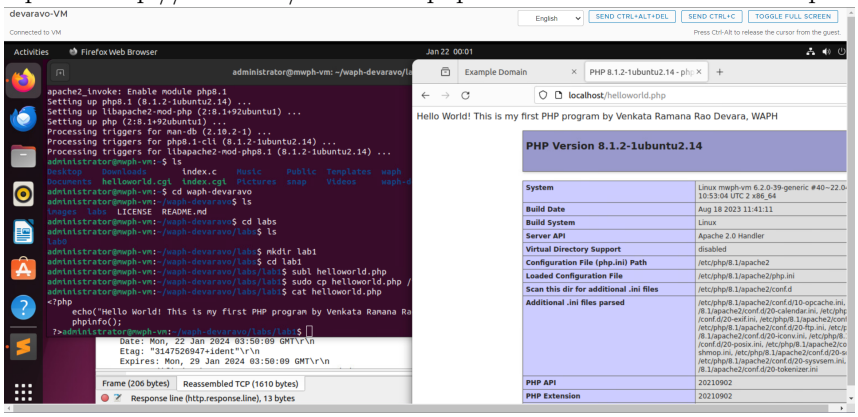
- Compiled the program (gcc index.c -o index.cgi)
- Copied the output file to /usr/lib/cgi-bin
- Opened index.cgi in browser



Task 2: Simple PHP application with user input

a)

- Installed php using "sudo apt-get install php libapache2-mod-php -y
- Created file “helloworld.php” in lab1 folder
- Copied helloworld.php to /var/www/html
- Opened http://localhost/helloworld.php in browser and observed output



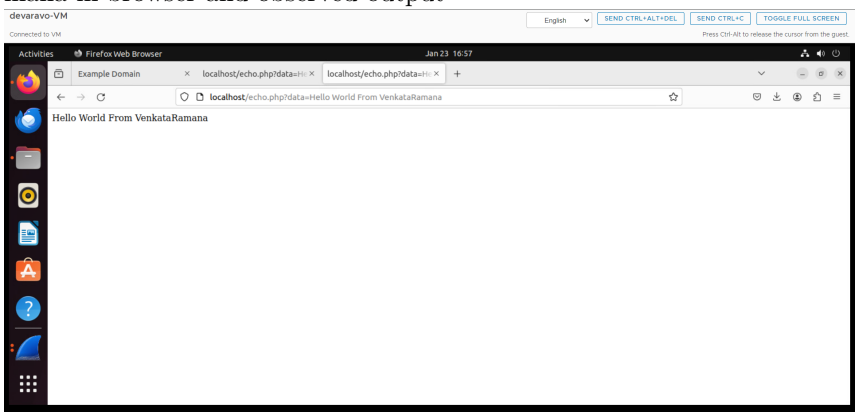
b)

- Created new php file echo.php
- Source code:

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

- Copied the file to /var/www/html

- Opened http://localhost/echo.php?data=Hello World from VenkataRamana in browser and observed output



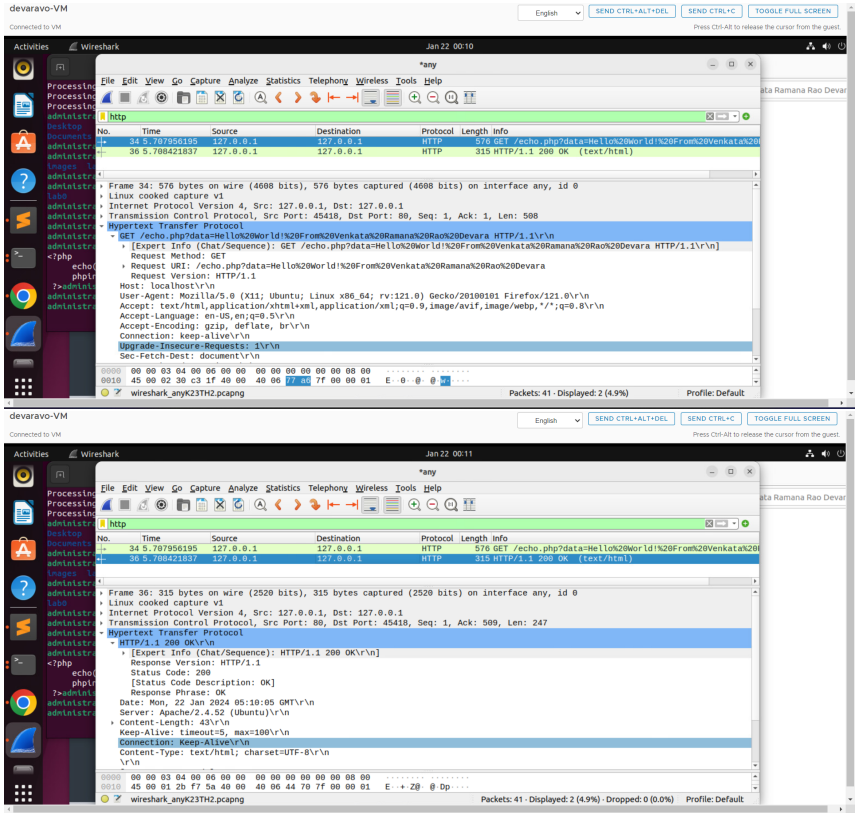
- There are security risks associated with this web application. The application uses REQUEST method instead of dedicated GET or POST methods. So, the application will capture data field regardless the data comes from a GET method or POST method. This gives access to the application for the attacker as he can just include the data in the URL and send it to application (as the application catches data from GET method). Although the attacker can still modify the POST request, it is still difficult than just sending data in URL.

- Also, another security risk is that it is easier for the attacker to make the user click on a link with data in it than making the user enter data in POST request.

Task 3: HTTP GET and POST requests

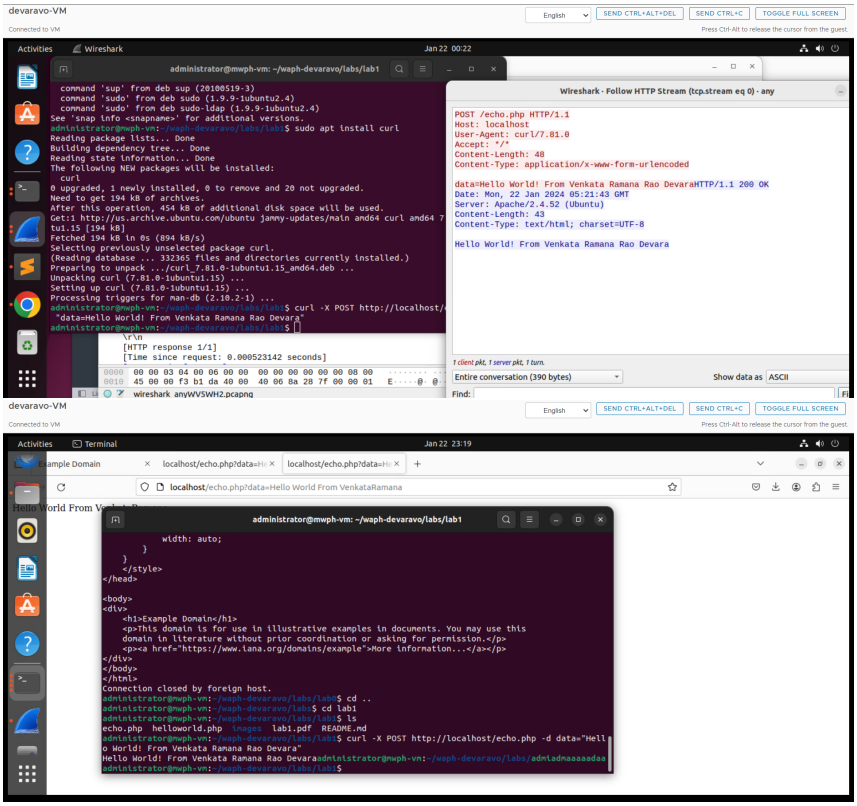
a)

- Started wireshark
- Opened http://localhost/echo.php?data=Hello World from VenkataRamana in browser
- Stopped wireshark and observed request and response messages



b)

- Started wireshark
- Ran curl -X POST http://localhost/echo.php -d “data=Hello World! From Venkata Ramana Rao Devara”
- Observed output “Hello World! From Venkata Ramana Rao Devara” in terminal
- Closed wireshark and observed http stream



c) From the two request messages for GET and POST observed in Wireshark, although the response message for both the requests print same data "Hello World! From Venkata Ramana Rao Devara", the difference lies in request message. - In GET request, the data is embedded in the url http://localhost/echo.php?data=Hello World! From Venkata Ramana - In POST request, the data is sent in the header in data field: data=Hello World! From Venkata Ramana Rao Devara. The data is not visible in URL itself.