Contents

waph-najarzs

WAPH-Web Application Programming and Hacking

Lab 1 - Foundations of the Web

Name: Zaid Najar

Instructor: Dr. Phu Phung

Email: najarzs@mail.uc.edu

Overview

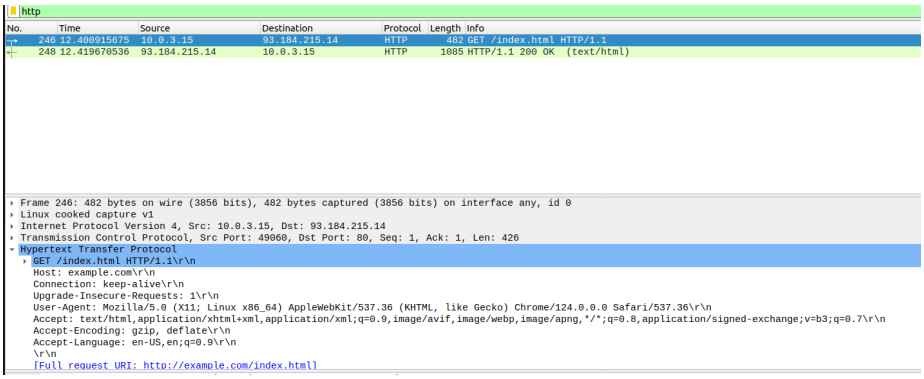
this lab shows core concepts of the web and HHTP protocol through Wireshark and web application programming. There are two parts: HTTP protocol and basic web application programming. I should be bale to use Wireshark to analyze HTTP traffic, use telnet, and develop applications using CGI and PHP.

Part 1 - The Web and HTTP Protocol

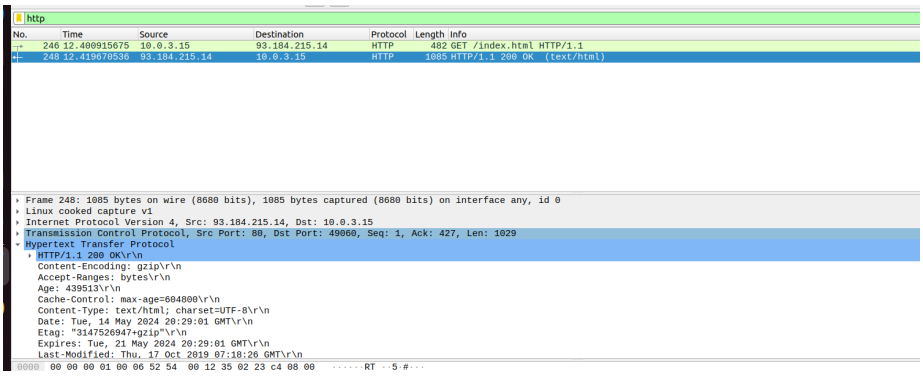
Task 1. Familiar with the Wireshark tool and HTTP protocol

I used Wireshark by running it as the lecture instructed me to do and opened an example html page in chrome. I then used the filter to search through all HTTP protocols and used that to take my screenshots.

Screenshot 1: Request



Screenshot 2: Response



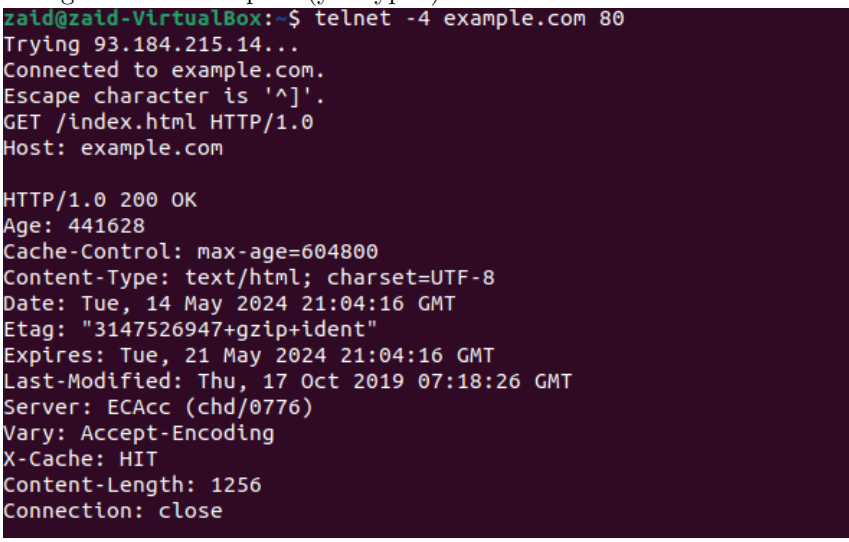
Screenshot 3: HTTP Stream

```
GET /index.html HTTP/1.1
Host: example.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

HTTP/1.1 200 OK
Content-Encoding: gzip
Accept-Ranges: bytes
Age: 439513
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 14 May 2024 20:29:01 GMT
Etag: "3147526947+gzip"
Expires: Tue, 21 May 2024 20:29:01 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (chd/0776)
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;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial,
sans-serif;

    }
    div {
        width: 600px;
```

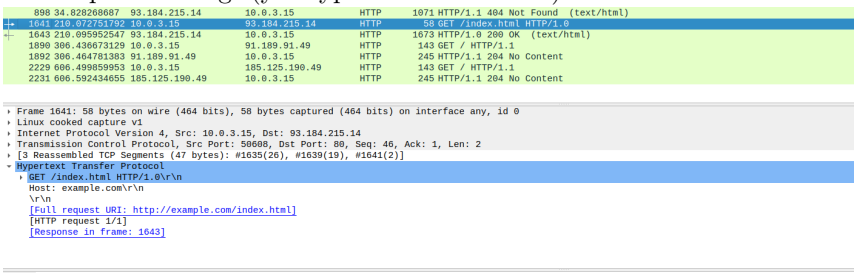
Task 2. Understanding HTTP using telnet and Wireshark

Telnet was used in this task to send a minimal HTTP request to Wireshark. After opening the example.com link and filtering by HTTP, I opened the terminal and used the command telnet example.com 80. This connected and then I used a GET method and the Host header to analyze the capture.

1. A screenshot of your terminal showing the HTTP Request (you typed) and HTTP response from the server

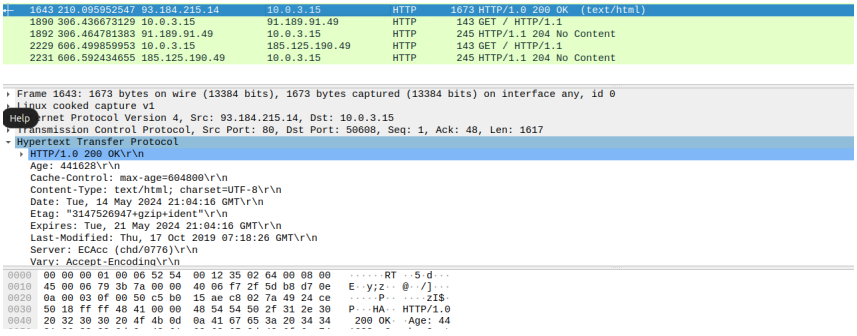


2. A screenshot of the HTTP Request message (you typed in telnet above) in Wireshark as in Task 1



The differences are there are two different HTTP versions used with Telnet using 1.0 and the browser using 1.1. The telnet is also missing the User-Agent field, the Accept, the Accept-Language, Accept-Encoding, Connection, and Upgrade-Insecure-Reuqests fields.

3. A screenshot of the HTTP Response message in Wireshark that the server

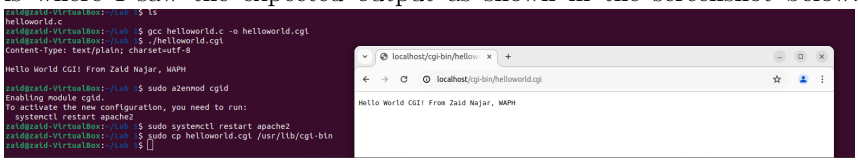


responded to your request. The differences are that the Telnet uses another HTTP version. It is also missing the fields: Connection Handling, Transger Encoding, Content-Encoding, and other headers such as Cache-Control, ETag, Set-Cookie, etc.

Part II - Basic Web Application Programming

Task 1. CGI Web applications in C

a. Summarize how you developed a Hello World CGI program in C and compiled and deployed the program on the web server. I created a new file called hellowrold.c that was a simple print statement. I compiled it using gcc to turn it into a .cgi file. I then configured the Apache web server. I then copied the CGI program to the cgi-bin directory and then navigated to the given link. This is where I saw the expected output as shown in the screenshot below.



b. Summarize and demonstrate with a screenshot that you can write another C CGI program and deploy it with a simple HTML template provided on https://www.w3schools.com/html/ with proper title, heading, and paragraph, i.e., the course and your information should be there. I copied the earlier helloworld file and repeated the exact steps as above. This time, I just changed the code up a bit following a template. Here is the source code from index.c as well as a screenshot of the output.

Included file `index.c`:

```
#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- Zaid Najar</title>\n");
    printf("</head>\n");
    printf("<body>\n");
    printf("<h1>Welcome to WAPH! I am Zaid Najar.</h1>\n");
    printf("<p> I am a Computer Science Major, class of 2025.</p>\n");
    printf("</body>\n");
    printf("</html>\n");
    return 0;
}
```

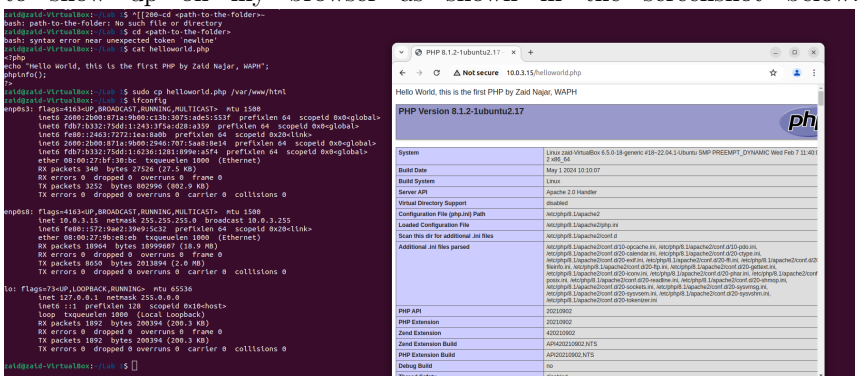
Welcome to WAPH! I am Zaid Najar.

I am a Computer Science Major, class of 2025.

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

a. Summarize and demonstrate with a screenshot that you have successfully developed a simple helloworld.php PHP page with your name and PHP configuration as guided in Lecture 3. I installed php and followed the instructions given in the lecture. I

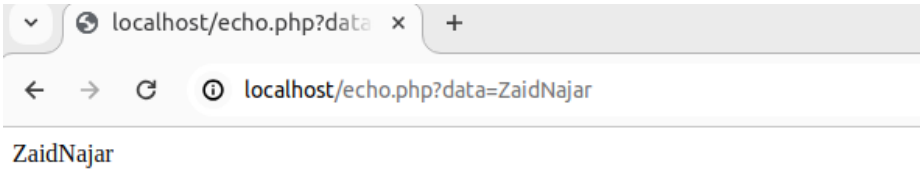
created a new file called helloworld.php and then configured it to show up on my browser as shown in the screenshot below.



b. Demonstrate that you developed and deployed an echo Web application in PHP, e.g., echo.php with a screenshot with your name in the data. I created a new file called echo.php and deployed it the same as the previous file. I then made data equal my name.

Included file `echo.php`:

```
<?php

echo $_REQUEST["data"];

?>
```



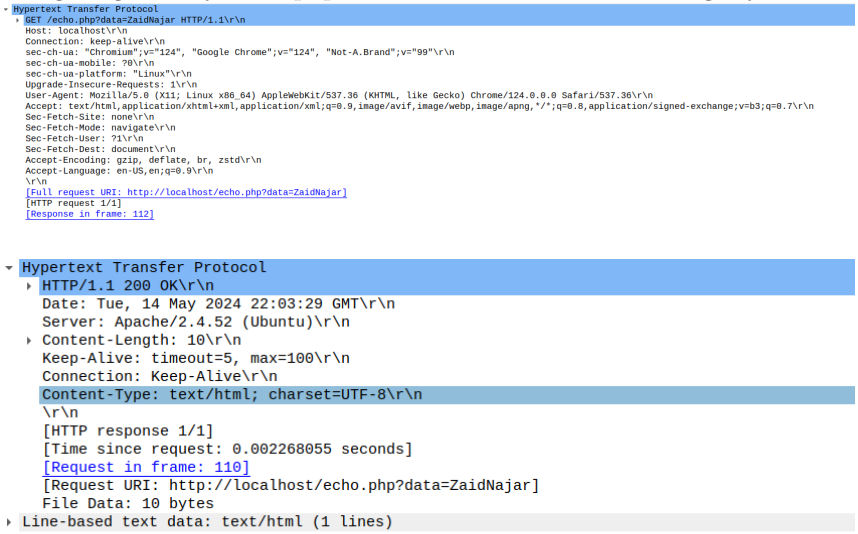
Screenshot of output:

Security Risks: There are a few risks such as URL manipulation which allows users to manipulate the URL to add unintended data. There is also a lack of input validation which could lead to errors or unwanted access from other users.

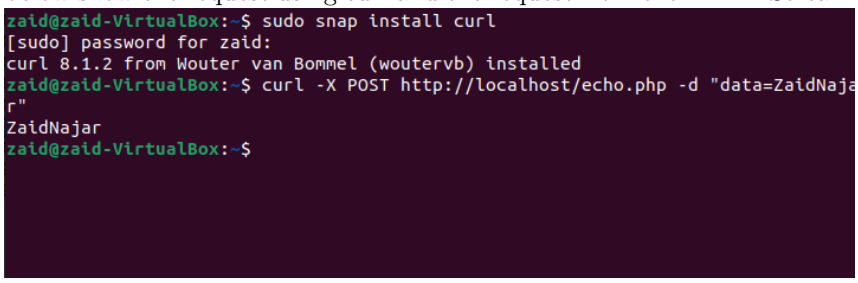
Task 3. Understanding HTTP GET and POST requests.

a. Briefly describe how you used Wireshark to examine the HTTP GET Request and Response for the echo.php page with your name in the data. I used Wireshark by running it as I did in the earlier tasks,

navigating to my echo.php in the browser, and filtering by HTTP.



b. Summarize using curl to create an HTTP POST request with your name in the data. I install curl then used the command as instructed with my data input to receive the POST request. The two screenshots below show the request using curl and the request from the HTTP Stream.



```
GET /echo.php?data=ZaidNajar HTTP/1.1
Host: localhost
Connection: keep-alive
sec-ch-ua: "Chromium";v="124", "Google Chrome";v="124", "Not-A.Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

HTTP/1.1 200 OK
Date: Tue, 14 May 2024 22:03:29 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 10
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

ZaidNajar
```

c. Compare the similarity/difference between HTTP POST Request and HTTP GET Request and between the two HTTP Responses above. In the request method, GET adds data to the URL while the POST sends it in the request body. For data visbility, you can see the data in the URL in GET, but in POST the data is in the request body. GET is less secure while post is more secure. GET is for getting data while POST is for sending data to be processed.