WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Bridget May

Name: Bridget May

Email: mailto:mayb05@udayton.edu

Short-bio: Hi! My name is Bridget May! I am a senior at the University of Dayton studying computer science and graphic design.

Figure 1: My headshot

Repository Information

Respository's URL: https://github.com/mayb05/waph-mayb05.git

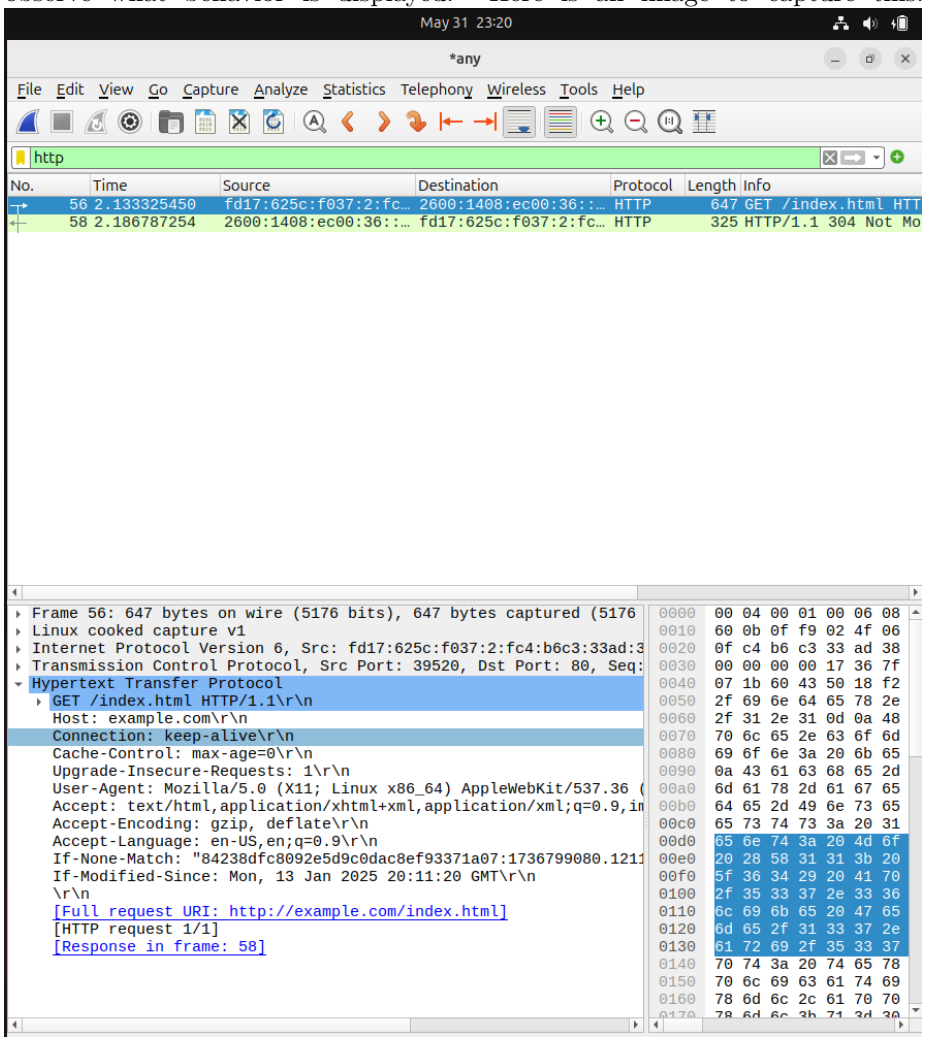
This is a private repository for Bridget May to store all code from the course. The organization of this repository is as follows.

Lab 1 Overview

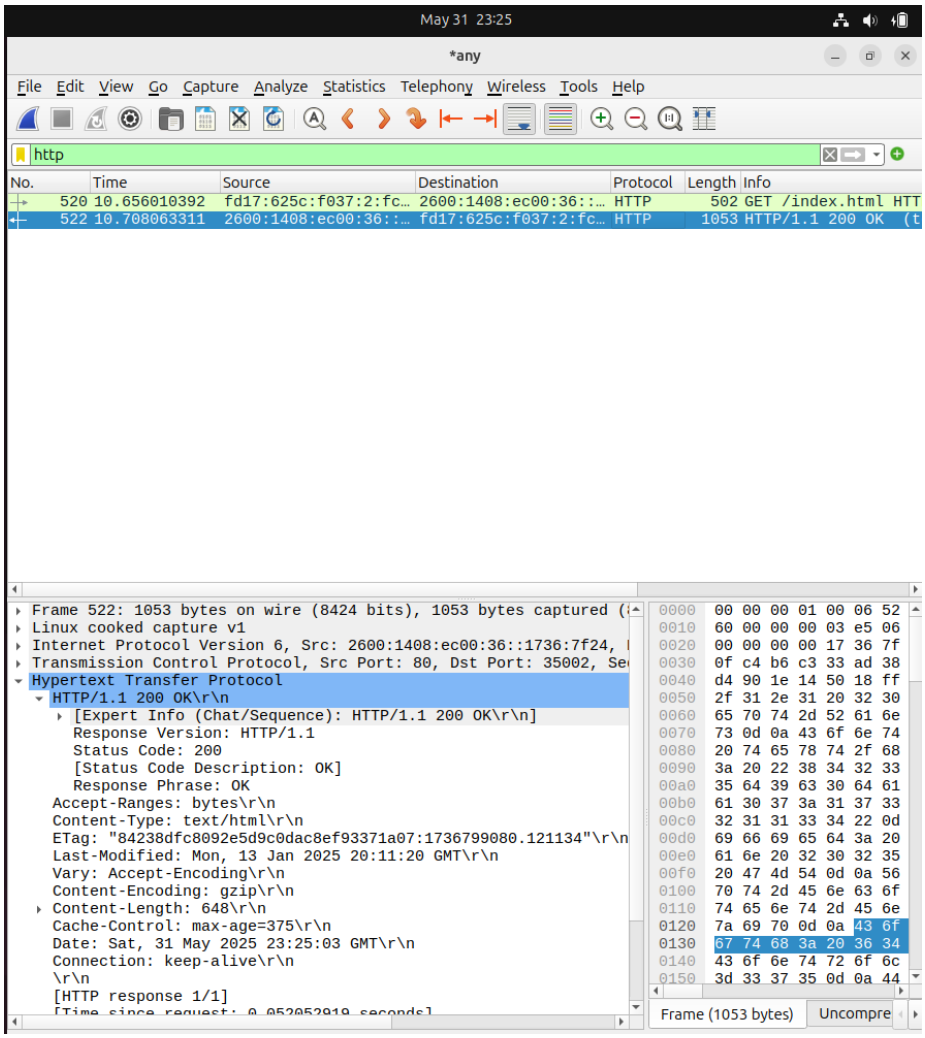
Lab link: https://github.com/mayb05/waph-mayb05/tree/main/labs/lab1 In this lab, I use wireshark to understand HTTP protocol of GET/REQUEST/RESPONSE. I also create programs in c, and php to gain more experience. We also use CGI to deploy my webpages.

Part 1 - The Web and HTTP Protocol

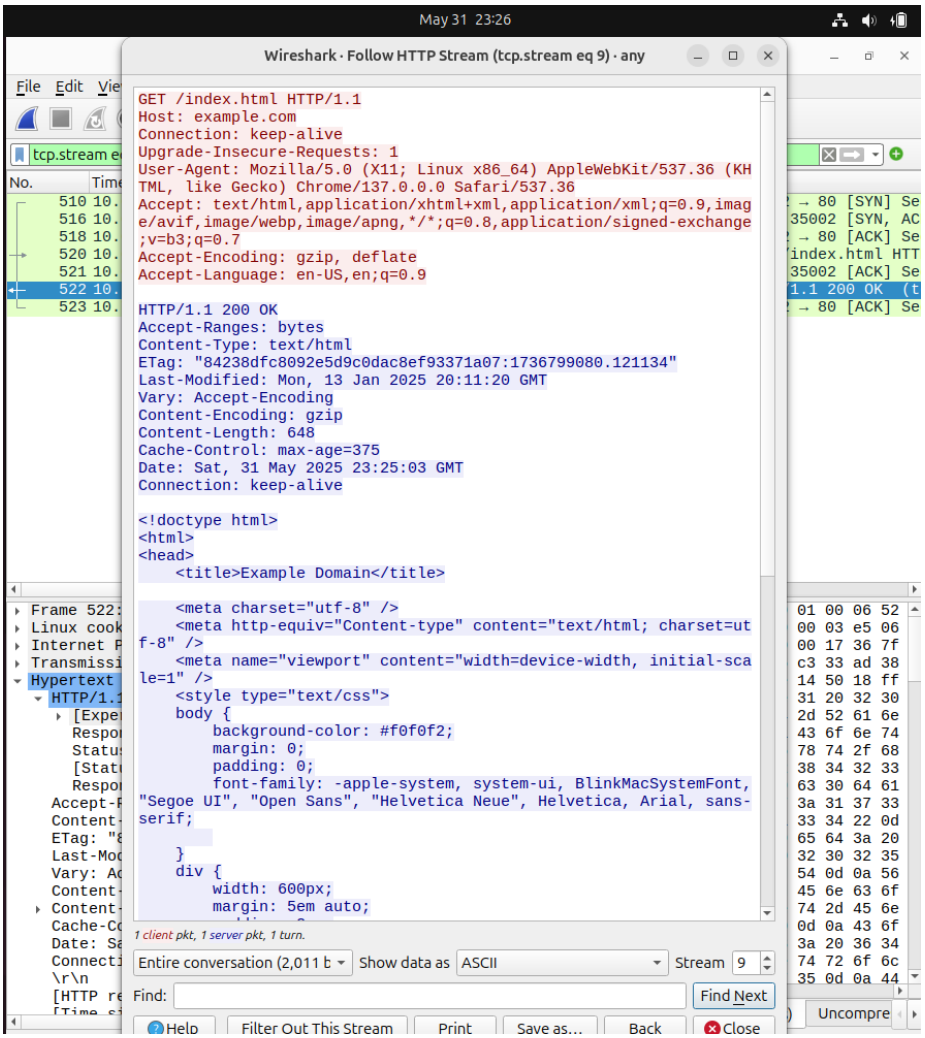
Task 1 - Wireshark and HTTP Protocol To start this lab, I installed wireshark in order to view the exchange of packets when visiting example.com. The first thing I did was set the tool to look at any activity then filtered down to HTTP. Only my requests for example.com showed up. After doing this, I clicked on the GET Request to observe what behavior is displayed. Here is an image to capture this:



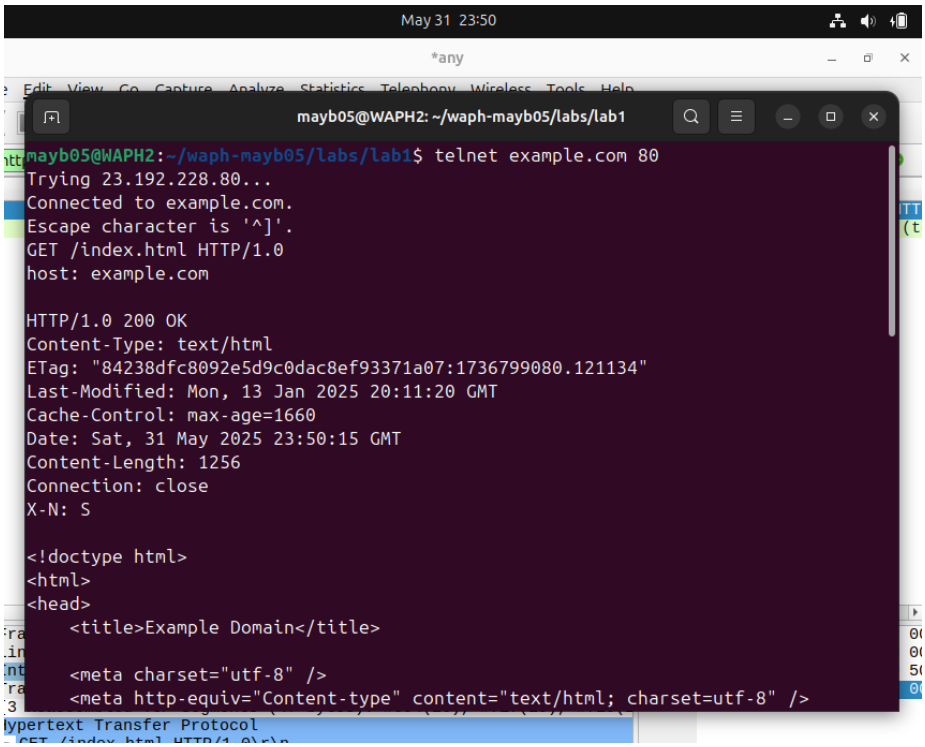
I then clicked on the Response message to note the differences in Response. At first I got the 304 error and had to clear my history and do the exercise a second time. We want the code 200.



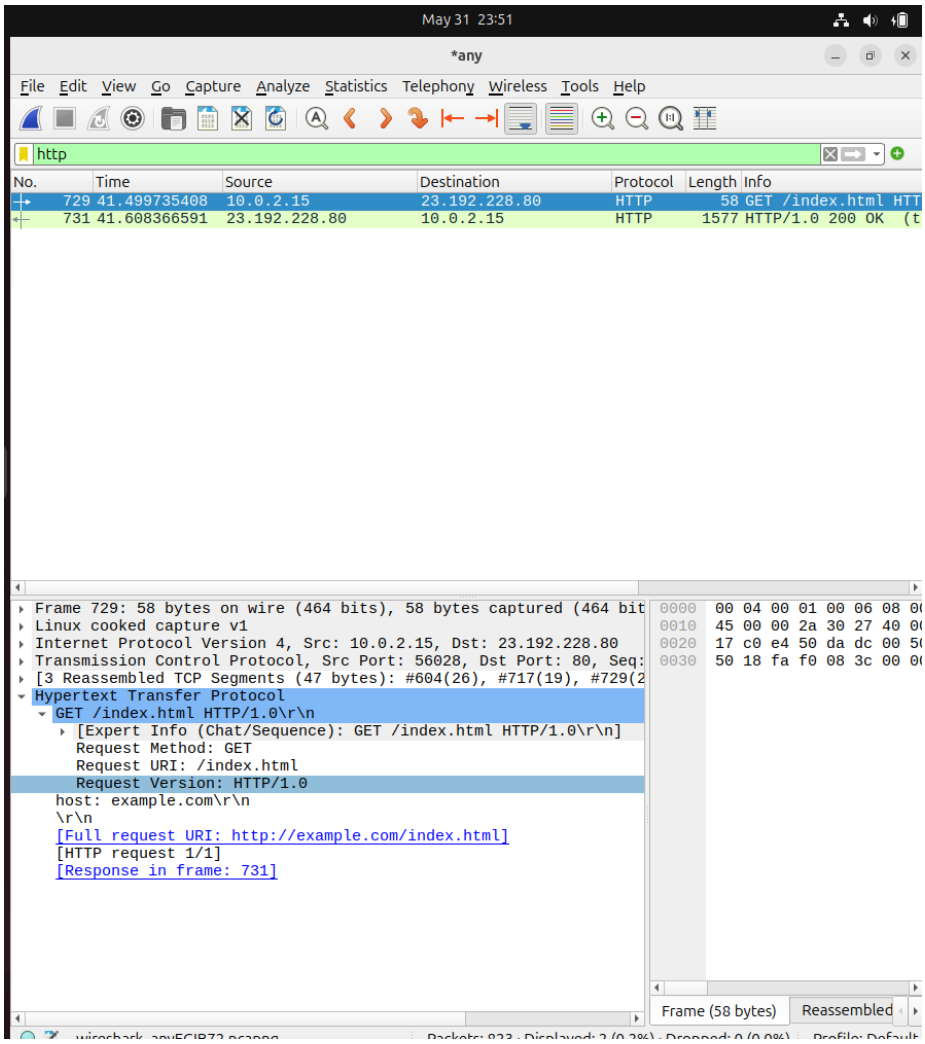
Then I clicked right clicked the response and clicked follow–>HTTP to see the messages. Red text is requests and blue text is response.



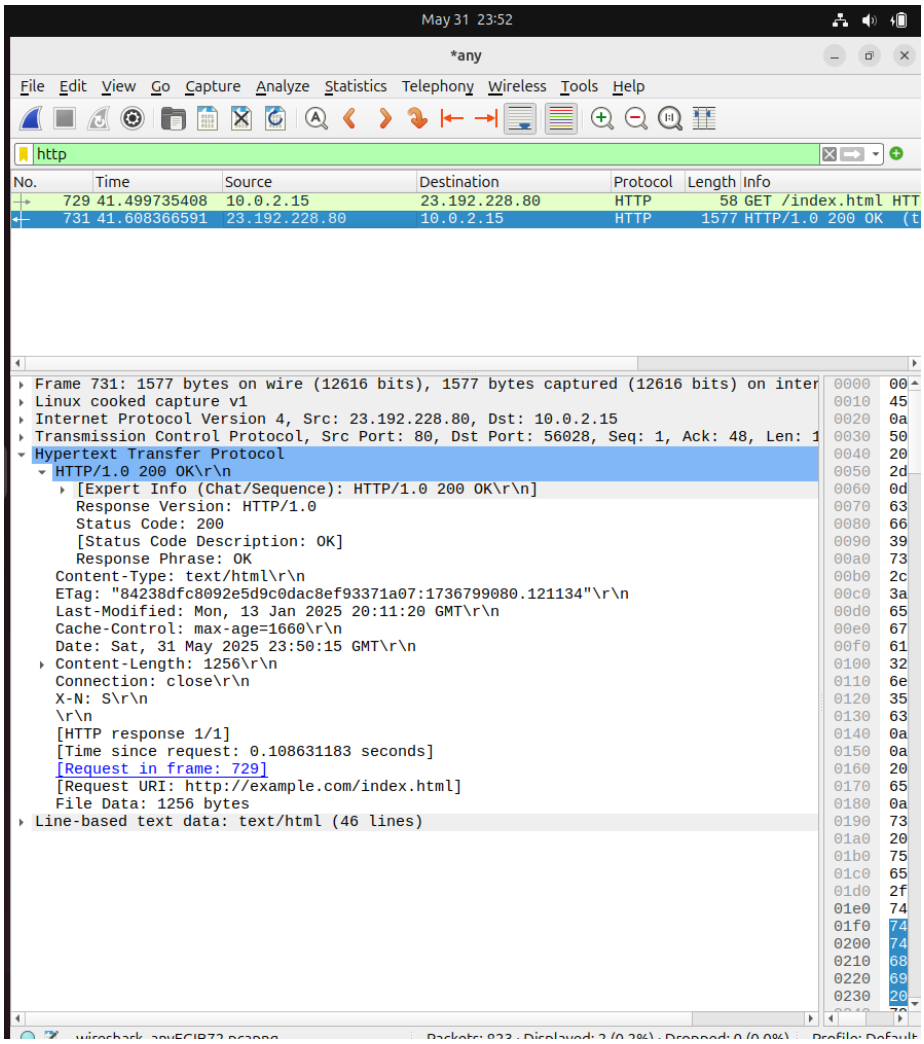
Task 2 - Telnet and Wireshark The first step was to clear the wireshark capture and then type the telnet command. The command is `telnet example.com 80` After doing so, I typed the Request needed and the host website. Here is a screenshot demonstrating this.



The next step was using wireshark to analyze the differences between using my web browser and telnet. The differences were that using the web browser, wireshark knew what my broswer was and had connection info while the telnet version had the barebones information. Here is the telnet request.

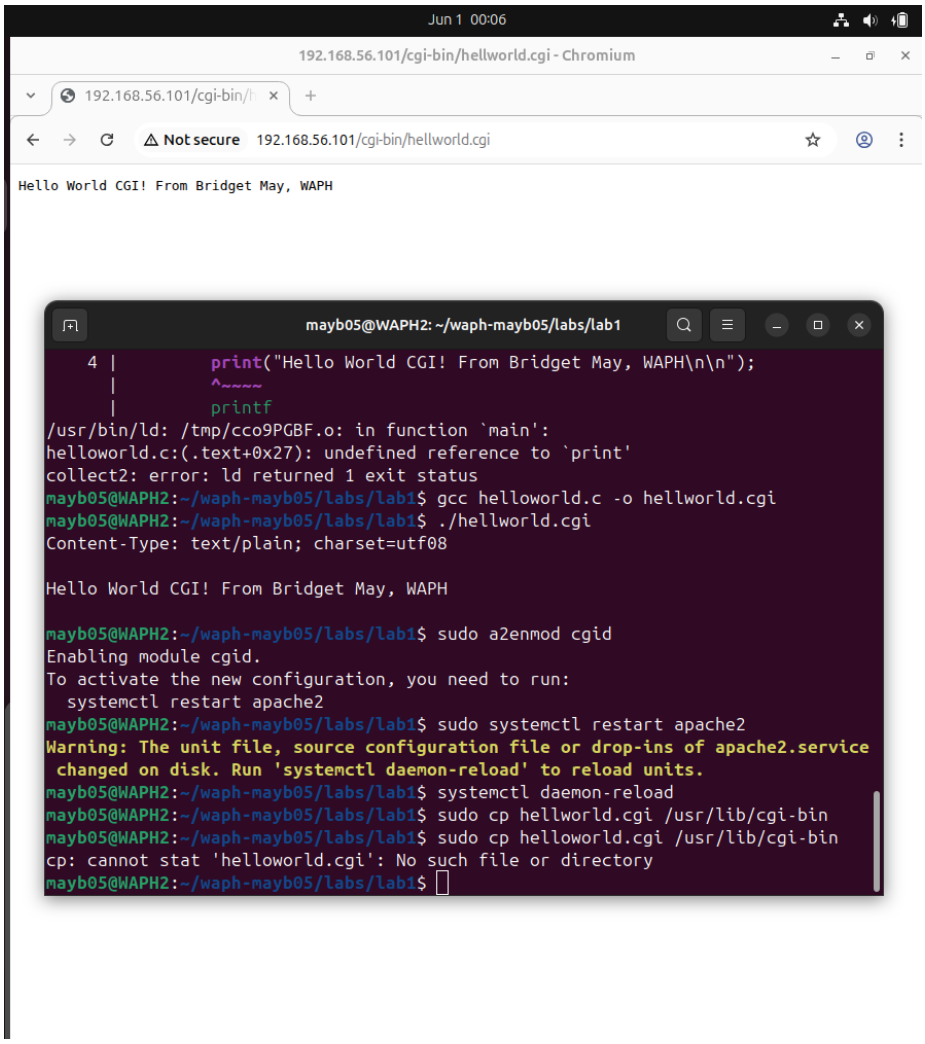


I analyzed the response as well. The differences in response were that the telnet version did not have the accept-ranges field and also did not have the date of con-

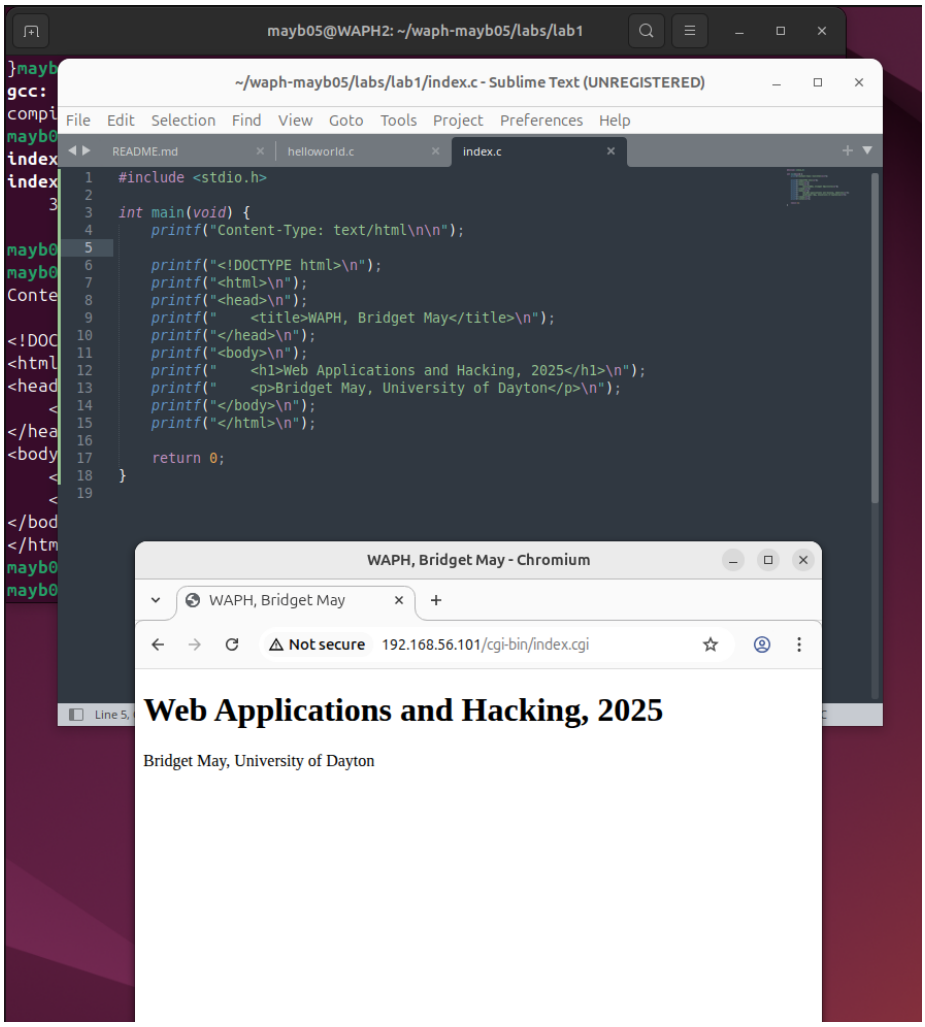


nection.

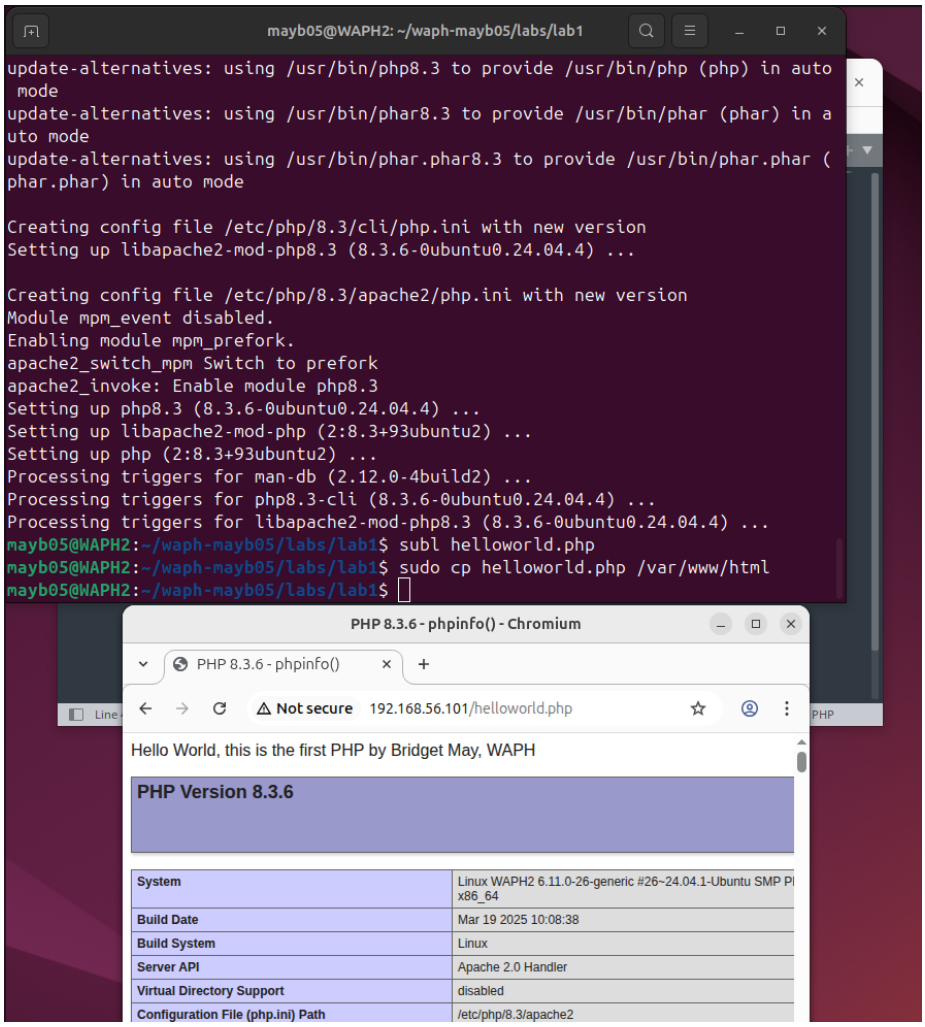
Part 2 - Basic Web Apllication Programming ##### Task 1 - CGI Web Apps in C The first step was developing a program in C to print helloworld then deploy it with CGI. I used the code provided to write the program in C. After doing that, I set the program up to be deployed with CGI. To be able to see the webpage, I had to copy over the code to the cgi-bin.



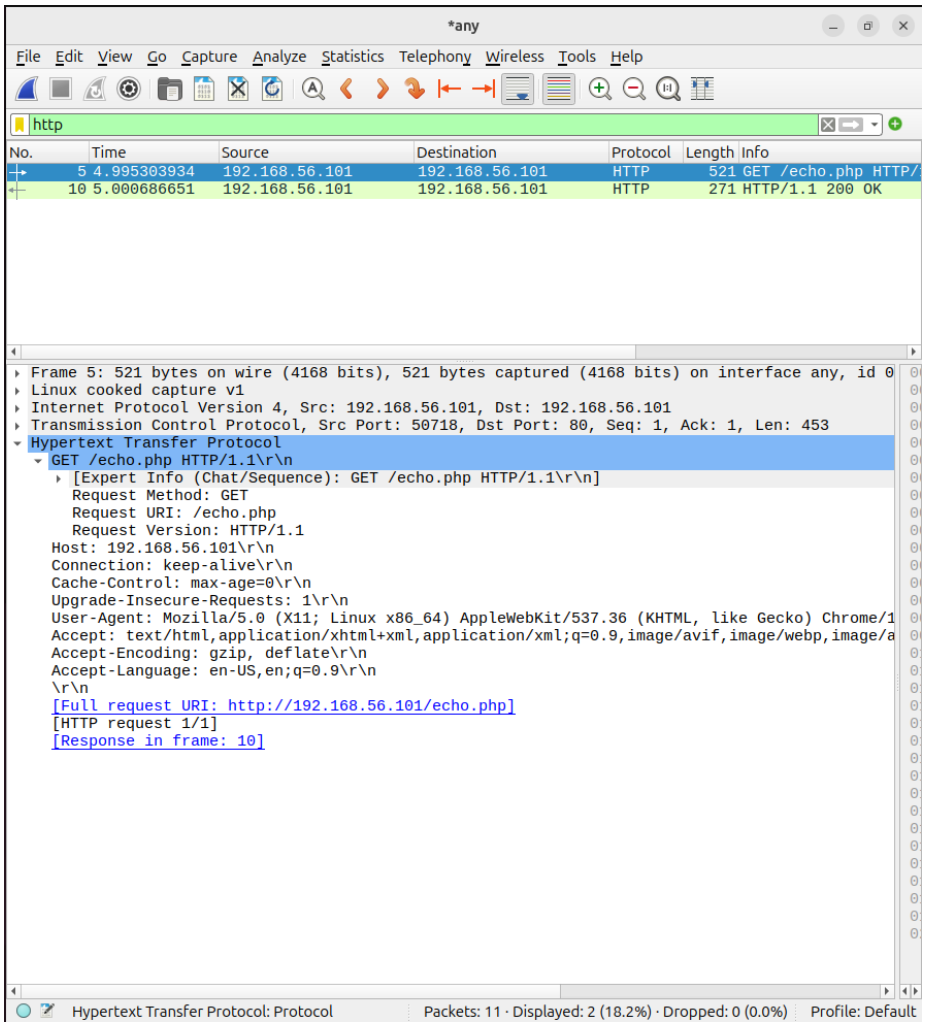
Next, I added html code to display the course info. Each statement required the "print" line. I deployed this one the same way.

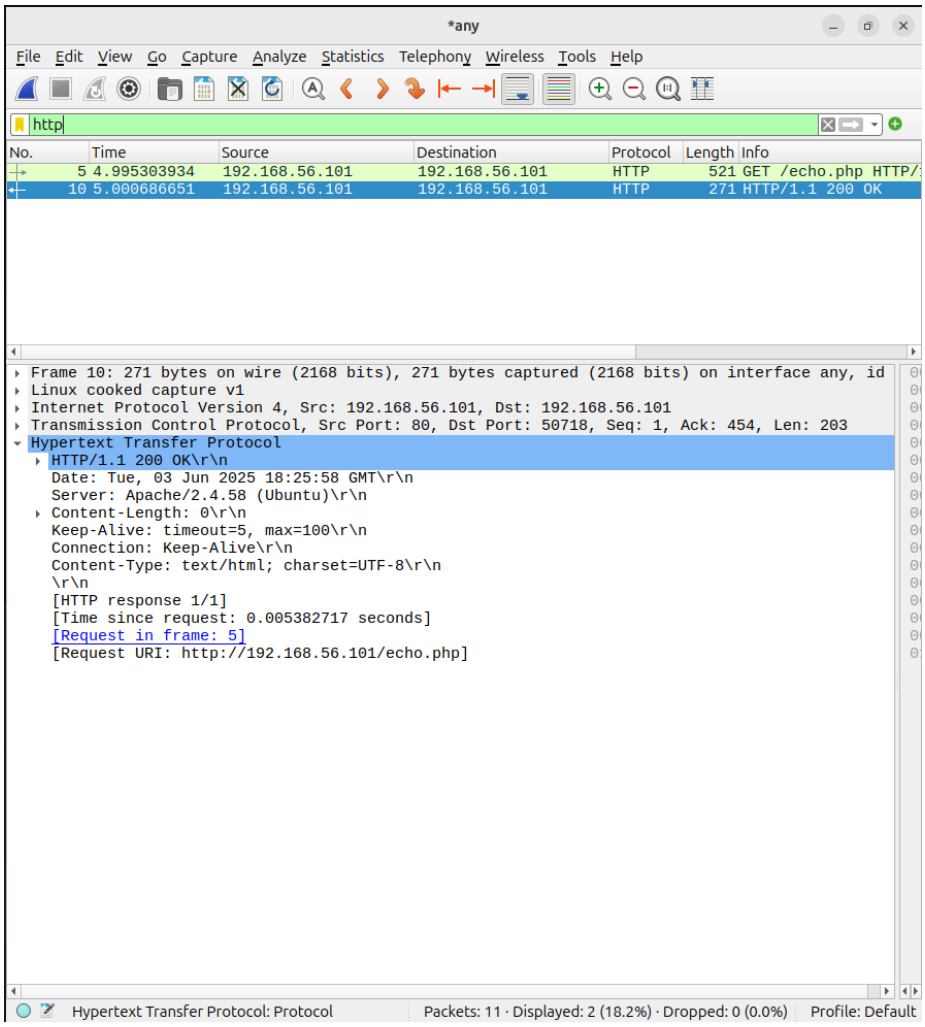


Task 2 - PHP with user input For task 2, I used PHP to create another hello world program. I had to install php in order to get started.



I then did some tests on echo.php for the first part. I had to deploy it first. This code does have some risks as the data is not sanitized in anticipation of attack.
Task 3 - HTTP GET and POST requests
I did the test to the data of echo.php with just sending my name. I then compared the two request/responses against each other to notice the differences.


*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
5 4.995303934 192.168.56.101 192.168.56.101 HTTP 521 GET /echo.php HTTP/
10 5.000686651 192.168.56.101 192.168.56.101 HTTP 271 HTTP/1.1 200 OK
Frame 5: 521 bytes on wire (4168 bits), 521 bytes captured (4168 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.101
Transmission Control Protocol, Src Port: 50718, Dst Port: 80, Seq: 1, Ack: 1, Len: 453
Hypertext Transfer Protocol
GET /echo.php HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /echo.php HTTP/1.1\r\n]
Request Method: GET
Request URI: /echo.php
Request Version: HTTP/1.1
Host: 192.168.56.101\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/1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
\r\n
[Full request URI: http://192.168.56.101/echo.php]
[HTTP request 1/1]
[Response in frame: 10]
Hypertext Transfer Protocol: Protocol
Packets: 11 · Displayed: 2 (18.2%) · Dropped: 0 (0.0%)
Profile: Default


*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
5 4.995303934 192.168.56.101 192.168.56.101 HTTP 521 GET /echo.php HTTP/
10 5.000686651 192.168.56.101 192.168.56.101 HTTP 271 HTTP/1.1 200 OK
Frame 10: 271 bytes on wire (2168 bits), 271 bytes captured (2168 bits) on interface any, id
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.101
Transmission Control Protocol, Src Port: 80, Dst Port: 50718, Seq: 1, Ack: 454, Len: 203
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Tue, 03 Jun 2025 18:25:58 GMT\r\n
Server: Apache/2.4.58 (Ubuntu)\r\n
Content-Length: 0\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: 0.005382717 seconds]
[Request in frame: 5]
[Request URI: http://192.168.56.101/echo.php]
Hypertext Transfer Protocol: Protocol
Packets: 11 · Displayed: 2 (18.2%) · Dropped: 0 (0.0%)
Profile: Default

I then tested the program using curl. The user agent changed.