Assignment 8
WRC078BEI048

A. **In the Programming language of you choice to write the web app that allows you to upload files.**

Ans:

App.py

```
from flask import Flask, request, redirect, url_for, render_template
from werkzeug.utils import secure_filename
import os

app = Flask(__name__)

# Configurations
UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/')
def upload_form():
    return render_template('upload.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return 'File successfully uploaded'
```

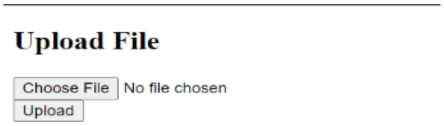
```python
    else:
        return 'Allowed file types are txt, pdf, png, jpg, jpeg, gif'

if __name__ == '__main__':
    app.run(debug=True)
```

upload.html

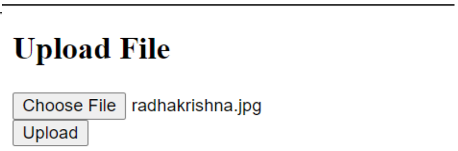
```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Upload File</title>
  </head>
  <body>
    <div class="container">
      <h2>Upload File</h2>
      <form method="POST" action="/upload" enctype="multipart/form-data">
        <div>
          <input type="file" name="file">
        </div>
        <div>
          <input type="submit" value="Upload">
        </div>
      </form>
    </div>
  </body>
</html>
```

Using this python and html code I built a web app that allows us to upload a file.



Upload File

Choose File No file chosen
Upload



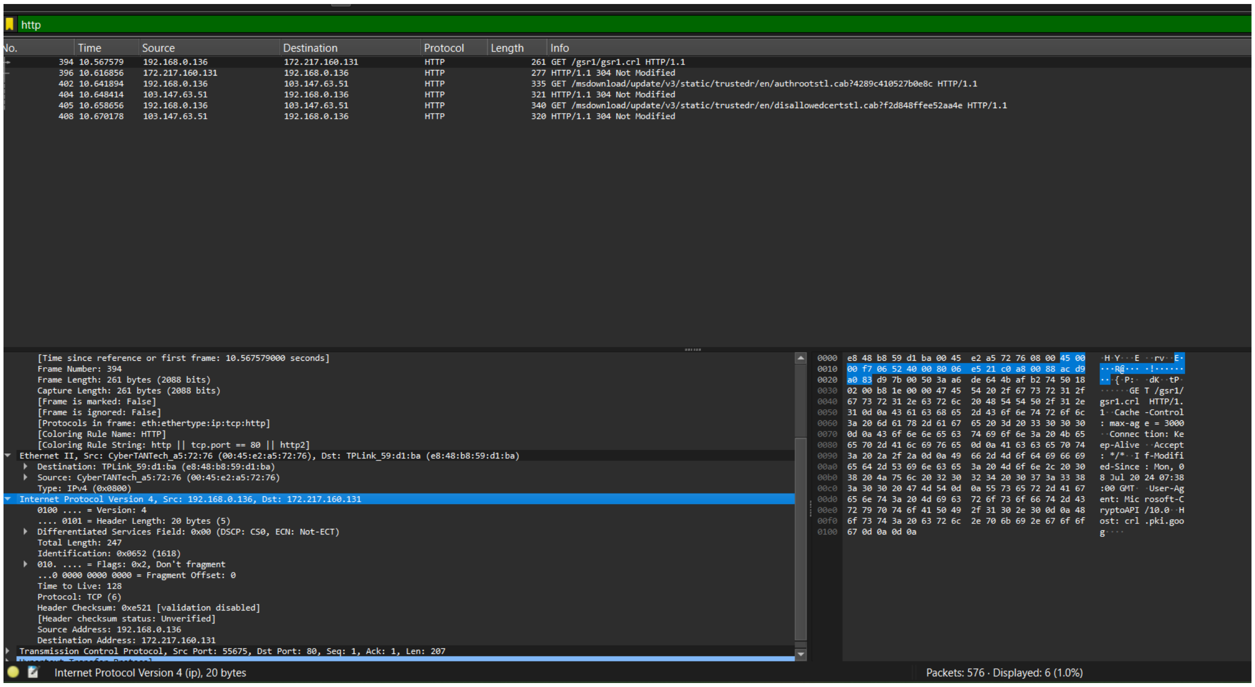
Upload File

Choose File radhakrishna.jpg
Upload

File successfully uploaded

B. Capture the traffic in wireshark while uploading the file.

Ans:



http

```
No.   Time       Source          Destination     Protocol Length Info
  394 10.567579  192.168.0.136   172.217.160.131 HTTP       261 GET /gsr1/gsr1.crl HTTP/1.1
  396 10.616856  172.217.160.131 192.168.0.136   HTTP       277 HTTP/1.1 304 Not Modified
  402 10.641894  192.168.0.136   103.147.63.51   HTTP       335 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?4289c410527b0e8c HTTP/1.1
  404 10.648414  103.147.63.51   192.168.0.136   HTTP       321 HTTP/1.1 304 Not Modified
  405 10.658656  192.168.0.136   103.147.63.51   HTTP       340 GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?f2d848ffee52aa4e HTTP/1.1
  408 10.670178  103.147.63.51   192.168.0.136   HTTP       320 HTTP/1.1 304 Not Modified
```

```
    [Time since reference or first frame: 10.567579000 seconds]
    Frame Number: 394
    Frame Length: 261 bytes (2088 bits)
    Capture Length: 261 bytes (2088 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
Ethernet II, Src: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76), Dst: TPLink_59:d1:ba (e8:48:b8:59:d1:ba)
  > Destination: TPLink_59:d1:ba (e8:48:b8:59:d1:ba)
  > Source: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.0.136, Dst: 172.217.160.131
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 247
    Identification: 0x0652 (1618)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0xe521 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.136
    Destination Address: 172.217.160.131
> Transmission Control Protocol, Src Port: 55675, Dst Port: 80, Seq: 1, Ack: 1, Len: 207
```

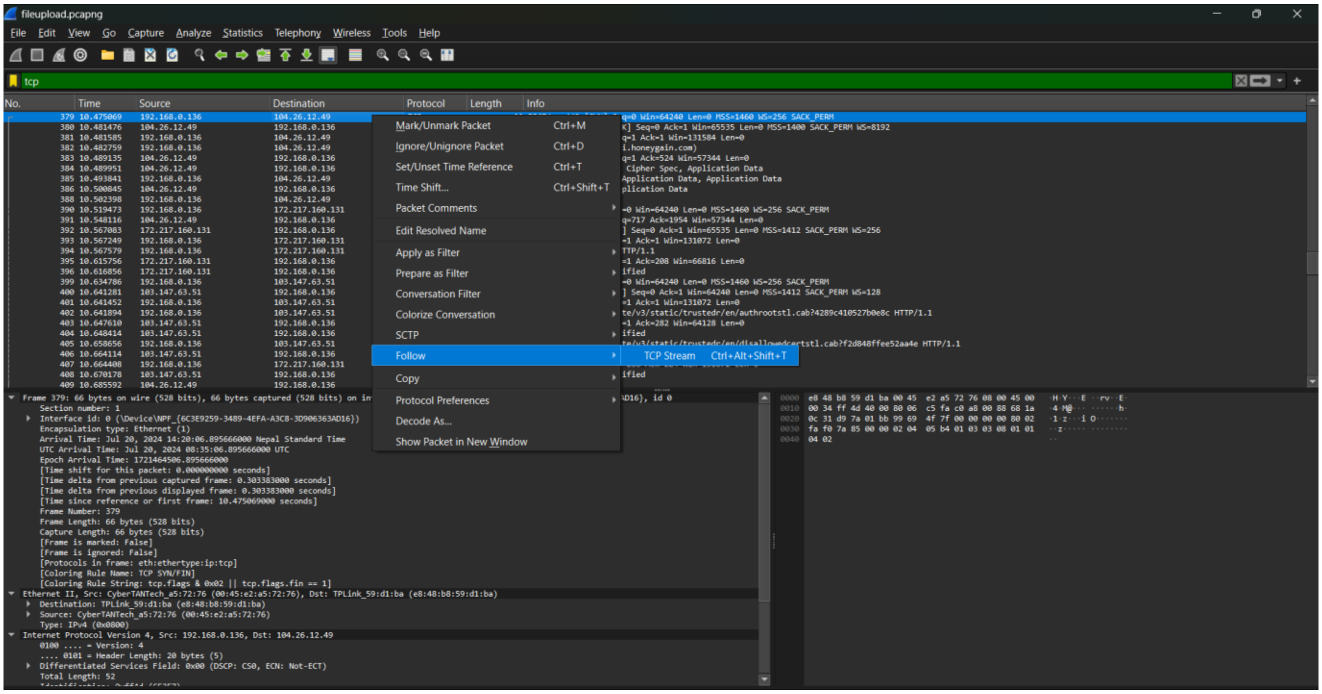
```
0000  e8 48 b8 59 d1 ba 00 45  e2 a5 72 76 08 00 45 00   ·H·Y···E ··rv··E·
0010  00 f7 06 52 40 00 80 06  e5 21 c0 a8 00 88 ac d9   ···R@··· ·!······
0020  a0 83 d9 7b 00 50 3a a6  de 64 4b af b2 74 50 18   ···{·P:· ·dK··tP·
0030  02 00 b8 1e 00 00 47 45  54 20 2f 67 73 72 31 2f   ······GE T /gsr1/
0040  67 73 72 31 2e 63 72 6c  20 48 54 54 50 2f 31 2e   gsr1.crl  HTTP/1.
0050  31 0d 0a 43 61 63 68 65  2d 43 6f 6e 74 72 6f 6c   1··Cache -Control
0060  3a 20 6d 61 78 2d 61 67  65 20 3d 20 33 30 30 30   : max-ag e = 3000
0070  0d 0a 43 6f 6e 6e 65 63  74 69 6f 6e 3a 20 4b 65   ··Connec tion: Ke
0080  65 70 2d 41 6c 69 76 65  0d 0a 41 63 63 65 70 74   ep-Alive ··Accept
0090  3a 20 2a 2f 2a 0d 0a 49  66 2d 4d 6f 64 69 66 69   : */*··I f-Modifi
00a0  65 64 2d 53 69 6e 63 65  3a 20 4d 6f 6e 2c 20 30   ed-Since : Mon, 0
00b0  38 20 4a 75 6c 20 32 30  32 34 20 30 37 3a 33 38   8 Jul 20 24 07:38
00c0  3a 30 30 20 47 4d 54 0d  0a 55 73 65 72 2d 41 67   :00 GMT· ·User-Ag
00d0  65 6e 74 3a 20 4d 69 63  72 6f 73 6f 66 74 2d 43   ent: Mic rosoft-C
00e0  72 79 70 74 6f 41 50 49  2f 31 30 2e 30 0d 0a 48   ryptoAPI /10.0··H
00f0  6f 73 74 3a 20 63 72 6c  2e 70 6b 69 2e 67 6f 6f   ost: crl .pki.goo
0100  67 0d 0a 0d 0a                                     g····
```

Internet Protocol Version 4 (ip), 20 bytes

Packets: 576 · Displayed: 6 (1.0%)

3. Follow the TCP stream and explain connection initiation, connection maintenance and connection termination

Ans:


fileupload.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comments
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
TCP Stream Ctrl+Alt+Shift+T
Copy
Protocol Preferences
Decode As...
Show Packet in New Window

tcp

```
No.    Time       Source           Destination      Protocol  Length  Info
379 10.475069  192.168.0.136    104.26.12.49       q=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
380 10.481476  104.26.12.49     192.168.0.136      K] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM WS=8192
381 10.481585  192.168.0.136    104.26.12.49       q=1 Ack=1 Win=131584 Len=0
382 10.482759  192.168.0.136    104.26.12.49       i.honeygain.com)
383 10.489135  104.26.12.49     192.168.0.136      q=1 Ack=524 Win=57344 Len=0
384 10.489951  104.26.12.49     192.168.0.136       Cipher Spec, Application Data
385 10.493841  192.168.0.136    104.26.12.49       Application Data, Application Data
386 10.500845  104.26.12.49     192.168.0.136      plication Data
388 10.502398  192.168.0.136    104.26.12.49
390 10.519473  192.168.0.136    172.217.160.131    =0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
391 10.548116  104.26.12.49     192.168.0.136      q=717 Ack=1954 Win=57344 Len=0
392 10.567083  172.217.160.131  192.168.0.136      ] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
393 10.567249  192.168.0.136    172.217.160.131    =1 Ack=1 Win=131072 Len=0
394 10.567579  192.168.0.136    172.217.160.131    TTP/1.1
395 10.615756  172.217.160.131  192.168.0.136      =1 Ack=208 Win=66816 Len=0
396 10.616856  172.217.160.131  192.168.0.136      ified
399 10.634786  192.168.0.136    103.147.63.51      =0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
400 10.641281  103.147.63.51    192.168.0.136      ] Seq=0 Ack=1 Win=64240 Len=0 MSS=1412 SACK_PERM WS=128
401 10.641452  192.168.0.136    103.147.63.51      =1 Ack=1 Win=131072 Len=0
402 10.641894  192.168.0.136    103.147.63.51      te/v3/static/trustedr/en/authrootstl.cab?4289c410527b0e8c HTTP/1.1
403 10.647610  103.147.63.51    192.168.0.136      =1 Ack=282 Win=64128 Len=0
404 10.648414  103.147.63.51    192.168.0.136      ified
405 10.658656  192.168.0.136    103.147.63.51      te/v3/static/trustedr/en/disallowedcertstl.cab?f2d848ffee52aa4e HTTP/1.1
406 10.664114  103.147.63.51    192.168.0.136
407 10.664408  192.168.0.136    172.217.160.131
408 10.670178  103.147.63.51    192.168.0.136      ified
409 10.685592  104.26.12.49     192.168.0.136
```

```
Frame 379: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on in AD16}, id 0
    Section number: 1
    Interface id: 0 (\Device\NPF_{6C3E9259-3489-4EFA-A3C8-3D906363AD16})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jul 20, 2024 14:20:06.895666000 Nepal Standard Time
    UTC Arrival Time: Jul 20, 2024 08:35:06.895666000 UTC
    Epoch Arrival Time: 1721464506.895666000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.303383000 seconds]
    [Time delta from previous displayed frame: 0.303383000 seconds]
    [Time since reference or first frame: 10.475069000 seconds]
    Frame Number: 379
    Frame Length: 66 bytes (528 bits)
    Capture Length: 66 bytes (528 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
    [Coloring Rule Name: TCP SYN/FIN]
    [Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1]
Ethernet II, Src: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76), Dst: TPLink_59:d1:ba (e8:48:b8:59:d1:ba)
    Destination: TPLink_59:d1:ba (e8:48:b8:59:d1:ba)
    Source: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.0.136, Dst: 104.26.12.49
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
```

```
0000  e8 48 b8 59 d1 ba 00 45  e2 a5 72 76 08 00 45 00   ·H·Y···E ··rv··E·
0010  00 34 ff 4d 40 00 80 06  c5 fa c0 a8 00 88 68 1a   ·4·M@··· ······h·
0020  0c 31 d9 7a 01 bb 99 69  4f 7f 00 00 00 00 80 02   ·1·z···i O·······
0030  fa f0 7a 85 00 00 02 04  05 b4 01 03 03 08 01 01   ··z····· ········
0040  04 02                                              ··
```

1. Connection Initiation

Objective: To identify the TCP handshake that establishes the connection between the client and server.

Steps:

1. **Find the SYN Packet**:
   - Look for a packet with the SYN flag set. This is typically the first packet sent by the client to initiate the connection.
   - **Example Packet Details**:
     - **Flags**: SYN
     - **Sequence Number**: `X`
2. **Find the SYN-ACK Packet**:
   - Find the packet with both SYN and ACK flags set. This is the server's response to the client's SYN request.
   - **Example Packet Details**:
     - **Flags**: SYN, ACK
     - **Sequence Number**: `Y`
     - **Acknowledgment Number**: `X + 1` (acknowledges the client's SYN)
3. **Find the ACK Packet**:
   - Look for the final packet in the handshake with the ACK flag set. This is sent by the client to acknowledge the server's SYN-ACK.
   - **Example Packet Details**:
     - **Flags**: ACK

        - **Sequence Number**: `X + 1`
        - **Acknowledgment Number**: `Y + 1` (acknowledges the server's SYN)

2. Connection Maintenance

Objective: To analyze the data transfer and how TCP maintains the connection.

Steps:

1. **Identify Data Packets**:
    - Look at the TCP packets with the PSH (Push) flag set, which indicates data being pushed to the application layer.
    - **Example Packet Details**:
        - **Flags**: PSH, ACK
        - **Sequence Number**: This is the starting byte of the data segment.
        - **Acknowledgment Number**: Indicates the next expected byte from the other side.
2. **Check Flow Control**:
    - Observe the window size in the TCP header to understand how much data the receiver is willing to accept.
    - **Example Field**:
        - **Window Size**: Indicates the amount of buffer space available.
3. **Monitor Congestion Control**:
    - Look for signs of congestion control mechanisms like Slow Start or Congestion Avoidance in TCP headers, though detailed congestion control analysis may require deeper inspection of the TCP behavior over time.
4. **Check Keep-Alive Packets (Optional)**:
    - If configured, check for occasional TCP keep-alive packets sent to ensure the connection remains active.
    - **Example Packet Details**:
        - **Flags**: ACK (with no payload)

3. Connection Termination

Objective: To identify the sequence of packets that close the TCP connection.

Steps:

1. **Find the FIN Packet**:
    - Look for a packet with the FIN flag set. This indicates that one side wants to close the connection.
    - **Example Packet Details**:
        - **Flags**: FIN
        - **Sequence Number**: `U`
2. **Find the ACK Packet**:
    - Locate the packet that acknowledges the FIN request from the other side.

- **Example Packet Details**:
  - **Flags**: ACK
  - **Acknowledgment Number**: `U + 1` (acknowledges the FIN)

3. **Find the Second FIN Packet**:
   - The side that acknowledged the initial FIN now sends its own FIN packet to close its side of the connection.
   - **Example Packet Details**:
     - **Flags**: FIN
     - **Sequence Number**: `W`
4. **Find the Final ACK Packet**:
   - Look for the final ACK packet that acknowledges the second FIN request.
   - **Example Packet Details**:
     - **Flags**: ACK
     - **Acknowledgment Number**: `W + 1`

How to View This in Wireshark:

1. **Start Capture**: Begin capturing traffic on the appropriate network interface.
2. **Follow TCP Stream**:
   - Right-click on a TCP packet related to the connection you want to analyze.
   - Select "Follow" > "TCP Stream" to view the entire conversation.
3. **Examine Packets**:
   - Use the filters and packet details to find SYN, SYN-ACK, and ACK packets for initiation.
   - Look for data packets, flow control, and keep-alive packets during maintenance.
   - Identify FIN and ACK packets for termination.