CS-301

Computer Networks Lab Report

Assignment - 2

BTech 5th Semester 2021

Department of Computer Science and Engineering,

National Institute of Technology Karnataka, Surathkal

October 2021

Submitted By:
Deepta Devkota - 191CS117
Akanksha More - 191CS106

Q1. Using TCP socket, implement HTTP server and client.

An **HTTP server** processes requests via HTTP, a network protocol used to exchange information on the World Wide Web (WWW). The main function of an HTTP server is to store, process, and deliver web pages to clients.
The **client** submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client.
The response contains completion status information about the request and may also contain requested content in its message body.
The HTTP protocol uses TCP in the transport layer.
An **HTTP client** initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server typically **port 80** or occasionally **port 8080**
SERVER program:

```
import socket


server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
HOST_IP='localhost'
PORT=8000
server.bind((HOST_IP,PORT))
server.listen(3)

print('HTTP server is listening..')


def handle_request(request):
    """Handles the HTTP request."""

    headers = request.split('\n')
    filename1 = headers[0].split()[1]


    filename=filename1[1:len(filename1)]


    try:
        fin = open(filename)
        content = fin.read()
        fin.close()

        response = 'HTTP/1.0 200 OK\n\n' + content
    except FileNotFoundError:
        response = 'HTTP/1.0 404 NOT FOUND\n\nFile Not Found'

    return response

while True:
    client,addr=server.accept()
    request=client.recv(1024).decode()
    response=handle_request(request)
    client.sendall(response.encode())
    client.close()

server.close()
```

CLIENT PROGRAM:

```python
import socket


client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client.connect(("localhost", 8000))


request='GET /index.html HTTP/1.0'
client.sendall(request.encode())

response=client.recv(1024).decode()
print(response)

client.close()
```

OUTPUT:

```
dell@dell-Inspiron-3593:~/Documents/Btech 5th sem/CN lab/lab/Week_2/Q1$ python3 server.py
HTTP server is listening..
GET /index.html HTTP/1.0
```

Once the server receives the GET request it proceeds to send the required web page to the client.

```
dell@dell-Inspiron-3593:~/Documents/Btech 5th sem/CN lab/lab/Week_2/Q1$ python3 client.py
HTTP/1.0 200 OK

<html>
<head>
    <title>Document</title>
</head>
<body>
    <h1>Hello World!</h1>
    <p>This is a HTML page</p>
</body>
</html>
```

On the client-side, the HTTP 200 OK success status response code indicates that the request has succeeded and hence we get the contents of the required webpage.

Q2. Write a program to translate a Domain name or hostname to its IP address and vice versa.

IP address:

It is a 32-bit address used for identifying any device on the internet or local network.

Domain name:

These are human-readable names to identify internet devices or web servers.

Domain Name System(DNS):

Conversion for the domain name to IP address is done by Domain Name System(DNS) resolution. It uses a distributed database and is implemented in the hierarchy of many name servers such as **root servers**, **TLD servers**, **authoritative servers.**

The given experiment is implemented in python which uses the socket library to convert the domain name to the corresponding IP address.

Socket library consists of a **gethostbyname()** method which takes in a parameter i.e. host name and returns the corresponding IPv4 address.

ip_address = socket.gethostbyname(host_name)

*In case no parameter is provided, then the default parameter is “localhost”.

CODE:

```python
import socket

# function get host id from host name
def get_HOST_Name_from_IP_Address(hostName):
    try:
        host_IP = socket.gethostbyname(hostName)
        print("IP Address: ",host_IP,"\n")
    except:
        print("Unable to get IP address from host name!","\n")

#take input for host name
hostName = input("\nEnter domain/host name: ")
get_HOST_Name_from_IP_Address(hostName)
```

OUTPUT:

```
(base) PS F:\sem5\cn_lab_git\Computer-Network-Lab\Week_2\Q2> python dns.py

Enter domain/host name: imap.gmail.com
IP Address:  74.125.130.109

(base) PS F:\sem5\cn_lab_git\Computer-Network-Lab\Week_2\Q2> python dns.py

Enter domain/host name: localhost
IP Address:  127.0.0.1
```

Q3. Develop a program to view the data of the top 50 movies in IMDB. (Movie name, actors, IMDB ratings).

Required information from a website can be acquired by either using the API of the website or extracting information from the HTML of the required page which is known as web scraping.
The above experiment uses the second method for implementation i.e. web scraping.
Web Scraping: The process of gathering information from the Internet.

Steps involved:

1. Requesting the required HTML page to the server using an HTTP request.
2. The server responds to the request by returning the HTML content of the web page.
3. The received HTML content is then parsed using a parser, eg. lxml, html.parser, html5lib, etc. The parser generates a tree structure of the HTML data.
4. Navigate through the tree to obtain the required information.

Libraries used:

Requests(requests) :
Python library for making HTTP requests. It abstracts the complexities of making requests behind a beautiful, simple API so that one can focus on interacting with services and consuming data in the application.

BeautifulSoup(bs4) :
Python library for parsing structured data such as HTML and XML content. It allows us to interact with the HTML data using simple development tools.
Steps involved in the lab experiment:

1. Sending the HTTP request to the URL to get the required HTML page and storing the response object.
2. Creating an instance of BeautifulSoup says “soup” by passing response.text and lxml parser as the parameters.
3. Using the select method on “soup” which is a CSS selector which finds and returns a list of multiple instances of matching elements passed as an argument, we obtain the required data.

CODE:

```python
import requests
import bs4

#IMDB URL for top 50 movies
requestURL = "https://www.imdb.com/chart/top"
response = requests.get(requestURL)

#get the required text from response using lxml parser
soup = bs4.BeautifulSoup(response.text, 'lxml')

movieName = soup.select(".titleColumn > a",limit=50)
imdbRating = soup.select(".imdbRating > strong",limit=50)
actors = soup.select(".titleColumn > a",limit=50)

#display top 50 movies from IMDB
print('Top 50 movies are:')
for i in range(50):
    print('\nRank: ', i+1)
    print('Movie Name: ' , movieName[i].text,' -|- Movie Rating: ',imdbRating[i].text, ' -|- Movie Actors: ', actors[i].attrs.get("title"))
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                                    python

F:\sem5\cn_lab_git\Computer-Network-Lab\Week_2\Q3>python get_movies.py
Top 50 movies are:

Rank:  1
Movie Name:  The Shawshank Redemption  -|- Movie Rating:  9.2  -|- Movie Actors:  Frank Darabont (dir.), Tim Robbins, Morgan Freeman

Rank:  2
Movie Name:  The Godfather  -|- Movie Rating:  9.1  -|- Movie Actors:  Francis Ford Coppola (dir.), Marlon Brando, Al Pacino

Rank:  3
Movie Name:  The Godfather: Part II  -|- Movie Rating:  9.0  -|- Movie Actors:  Francis Ford Coppola (dir.), Al Pacino, Robert De Niro

Rank:  4
Movie Name:  The Dark Knight  -|- Movie Rating:  9.0  -|- Movie Actors:  Christopher Nolan (dir.), Christian Bale, Heath Ledger

Rank:  5
Movie Name:  12 Angry Men  -|- Movie Rating:  8.9  -|- Movie Actors:  Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb

Rank:  6
Movie Name:  Schindler's List  -|- Movie Rating:  8.9  -|- Movie Actors:  Steven Spielberg (dir.), Liam Neeson, Ralph Fiennes

Rank:  7
Movie Name:  The Lord of the Rings: The Return of the King  -|- Movie Rating:  8.9  -|- Movie Actors:  Peter Jackson (dir.), Elijah Wood, Viggo Mortensen

Rank:  8
Movie Name:  Pulp Fiction  -|- Movie Rating:  8.8  -|- Movie Actors:  Quentin Tarantino (dir.), John Travolta, Uma Thurman

Rank:  9
Movie Name:  Il buono, il brutto, il cattivo  -|- Movie Rating:  8.8  -|- Movie Actors:  Sergio Leone (dir.), Clint Eastwood, Eli Wallach

Rank:  10
Movie Name:  The Lord of the Rings: The Fellowship of the Ring  -|- Movie Rating:  8.8  -|- Movie Actors:  Peter Jackson (dir.), Elijah Wood, Ian McKellen

Rank:  11
Movie Name:  Fight Club  -|- Movie Rating:  8.8  -|- Movie Actors:  David Fincher (dir.), Brad Pitt, Edward Norton

Rank:  12
Movie Name:  Forrest Gump  -|- Movie Rating:  8.7  -|- Movie Actors:  Robert Zemeckis (dir.), Tom Hanks, Robin Wright
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
python
Rank: 13
Movie Name: Inception -|- Movie Rating: 8.7 -|- Movie Actors: Christopher Nolan (dir.), Leonardo DiCaprio, Joseph Gordon-Levitt
Rank: 14
Movie Name: The Lord of the Rings: The Two Towers -|- Movie Rating: 8.7 -|- Movie Actors: Peter Jackson (dir.), Elijah Wood, Ian McKellen
Rank: 15
Movie Name: Star Wars: Episode V - The Empire Strikes Back -|- Movie Rating: 8.7 -|- Movie Actors: Irvin Kershner (dir.), Mark Hamill, Harrison Ford
Rank: 16
Movie Name: The Matrix -|- Movie Rating: 8.6 -|- Movie Actors: Lana Wachowski (dir.), Keanu Reeves, Laurence Fishburne
Rank: 17
Movie Name: Goodfellas -|- Movie Rating: 8.6 -|- Movie Actors: Martin Scorsese (dir.), Robert De Niro, Ray Liotta
Rank: 18
Movie Name: One Flew Over the Cuckoo's Nest -|- Movie Rating: 8.6 -|- Movie Actors: Milos Forman (dir.), Jack Nicholson, Louise Fletcher
Rank: 19
Movie Name: Shichinin no samurai -|- Movie Rating: 8.6 -|- Movie Actors: Akira Kurosawa (dir.), Toshirô Mifune, Takashi Shimura
Rank: 20
Movie Name: Se7en -|- Movie Rating: 8.6 -|- Movie Actors: David Fincher (dir.), Morgan Freeman, Brad Pitt
Rank: 21
Movie Name: The Silence of the Lambs -|- Movie Rating: 8.6 -|- Movie Actors: Jonathan Demme (dir.), Jodie Foster, Anthony Hopkins
Rank: 22
Movie Name: Cidade de Deus -|- Movie Rating: 8.6 -|- Movie Actors: Fernando Meirelles (dir.), Alexandre Rodrigues, Leandro Firmino
Rank: 23
Movie Name: La vita è bella -|- Movie Rating: 8.6 -|- Movie Actors: Roberto Benigni (dir.), Roberto Benigni, Nicoletta Braschi
Rank: 24
Movie Name: It's a Wonderful Life -|- Movie Rating: 8.6 -|- Movie Actors: Frank Capra (dir.), James Stewart, Donna Reed
Rank: 25
Movie Name: Star Wars -|- Movie Rating: 8.6 -|- Movie Actors: George Lucas (dir.), Mark Hamill, Harrison Ford


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Rank: 26
Movie Name: Saving Private Ryan -|- Movie Rating: 8.5 -|- Movie Actors: Steven Spielberg (dir.), Tom Hanks, Matt Damon
Rank: 27
Movie Name: Interstellar -|- Movie Rating: 8.5 -|- Movie Actors: Christopher Nolan (dir.), Matthew McConaughey, Anne Hathaway
Rank: 28
Movie Name: Sen to Chihiro no kamikakushi -|- Movie Rating: 8.5 -|- Movie Actors: Hayao Miyazaki (dir.), Daveigh Chase, Suzanne Pleshette
Rank: 29
Movie Name: The Green Mile -|- Movie Rating: 8.5 -|- Movie Actors: Frank Darabont (dir.), Tom Hanks, Michael Clarke Duncan
Rank: 30
Movie Name: Gisaengchung -|- Movie Rating: 8.5 -|- Movie Actors: Bong Joon Ho (dir.), Kang-ho Song, Sun-kyun Lee
Rank: 31
Movie Name: Léon -|- Movie Rating: 8.5 -|- Movie Actors: Luc Besson (dir.), Jean Reno, Gary Oldman
Rank: 32
Movie Name: Seppuku -|- Movie Rating: 8.5 -|- Movie Actors: Masaki Kobayashi (dir.), Tatsuya Nakadai, Akira Ishihama
Rank: 33
Movie Name: The Pianist -|- Movie Rating: 8.5 -|- Movie Actors: Roman Polanski (dir.), Adrien Brody, Thomas Kretschmann
Rank: 34
Movie Name: The Usual Suspects -|- Movie Rating: 8.5 -|- Movie Actors: Bryan Singer (dir.), Kevin Spacey, Gabriel Byrne
Rank: 35
Movie Name: Terminator 2: Judgment Day -|- Movie Rating: 8.5 -|- Movie Actors: James Cameron (dir.), Arnold Schwarzenegger, Linda Hamilton
Rank: 36
Movie Name: Back to the Future -|- Movie Rating: 8.5 -|- Movie Actors: Robert Zemeckis (dir.), Michael J. Fox, Christopher Lloyd
Rank: 37
Movie Name: Psycho -|- Movie Rating: 8.5 -|- Movie Actors: Alfred Hitchcock (dir.), Anthony Perkins, Janet Leigh
Rank: 38
Movie Name: The Lion King -|- Movie Rating: 8.5 -|- Movie Actors: Roger Allers (dir.), Matthew Broderick, Jeremy Irons


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Rank: 39
Movie Name: Modern Times -|- Movie Rating: 8.5 -|- Movie Actors: Charles Chaplin (dir.), Charles Chaplin, Paulette Goddard
Rank: 40
Movie Name: American History X -|- Movie Rating: 8.5 -|- Movie Actors: Tony Kaye (dir.), Edward Norton, Edward Furlong
Rank: 41
Movie Name: City Lights -|- Movie Rating: 8.5 -|- Movie Actors: Charles Chaplin (dir.), Charles Chaplin, Virginia Cherrill
Rank: 42
Movie Name: Hotaru no haka -|- Movie Rating: 8.5 -|- Movie Actors: Isao Takahata (dir.), Tsutomu Tatsumi, Ayano Shiraishi
Rank: 43
Movie Name: Whiplash -|- Movie Rating: 8.5 -|- Movie Actors: Damien Chazelle (dir.), Miles Teller, J.K. Simmons
Rank: 44
Movie Name: Gladiator -|- Movie Rating: 8.5 -|- Movie Actors: Ridley Scott (dir.), Russell Crowe, Joaquin Phoenix
Rank: 45
Movie Name: The Departed -|- Movie Rating: 8.5 -|- Movie Actors: Martin Scorsese (dir.), Leonardo DiCaprio, Matt Damon
Rank: 46
Movie Name: The Intouchables -|- Movie Rating: 8.5 -|- Movie Actors: Olivier Nakache (dir.), François Cluzet, Omar Sy
Rank: 47
Movie Name: The Prestige -|- Movie Rating: 8.5 -|- Movie Actors: Christopher Nolan (dir.), Christian Bale, Hugh Jackman
Rank: 48
Movie Name: Casablanca -|- Movie Rating: 8.4 -|- Movie Actors: Michael Curtiz (dir.), Humphrey Bogart, Ingrid Bergman
Rank: 49
Movie Name: Once Upon a Time in the West -|- Movie Rating: 8.4 -|- Movie Actors: Sergio Leone (dir.), Henry Fonda, Charles Bronson
Rank: 50
Movie Name: Rear Window -|- Movie Rating: 8.4 -|- Movie Actors: Alfred Hitchcock (dir.), James Stewart, Grace Kelly
F:\sem5\cn_lab_git\Computer-Network-Lab\Week_2\Q3>

Q4. Write a program to display the details of an input URL (status code, headers,
history, encoding, reason, cookies, elapsed, request)

The get the required information as mentioned in the question above **request library of Python** was used. The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a simple API. It makes us focus on interacting with services and consuming data in your application.

CODE:

```
import requests

try:

    url = input('Enter a valid url: ')

    r = requests.get(url)

    print('STATUS CODE:')
    print(r.status_code)
    print('HEADERS:')
    print(r.headers)
    print('HISTORY:')
    print(r.history)
    print('\n')
    print('ENCODING:')
    print(r.encoding)
    print('REASON:')
    print(r.reason)
    print('COOKIES:')
    print(r.cookies)
    print('ELAPSED:')
    print(r.elapsed)
    print('REQUEST:')
    print(r.request)


except requests.ConnectionError:
    print("failed to connect")
```

Details printed by the above program are:

Status code: HTTP status codes are standard response codes given by website servers on the internet. It helps identify the cause of the problem when a web page or other resource does not load properly.

Headers: HTTP headers let the client and the server pass additional information with an HTTP request or response.

Encoding: Encoding format of the URL, example: ISO-8859-1

Reason: OK

Cookies: It has the cookies information

Elapsed: It gives the elapsed time, for example, 0:00:00.919588

Request: It gives the type of request, example: <PreparedRequest [GET]>

OUTPUT:

Q5. Capture HTTP packets by visiting an HTTP Website, analyze the packets and significance of their various fields. Do the same for HTTPS packets and compare both.

HTTP:

The Hypertext Transfer Protocol is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems. It uses TCP as the transport layer protocol and the connection is stateless.
There are two types of HTTP connections:

1. **Persistent HTTP:** In persistent connection function that allows the channel to remain open rather than be closed after a requested exchange of data.
2. **Non-persistent HTTP**: In a non-persistent connection function that allows the channel to remain open rather than be closed after a requested exchange of data.

For the experiment, we analyzed the HTTP packets of the URL:
http://scratchpads.org/explore/sites-list

No.	Time	Source	Destination	Protocol	Length	Info
137	3.214414248	192.168.1.105	157.140.2.32	HTTP	533	GET /explore/sites-list HTTP/1.1
173	3.571688447	157.140.2.32	192.168.1.105	HTTP	2166	HTTP/1.1 200 OK (text/html)
176	3.580638757	192.168.1.105	157.140.2.32	HTTP	406	GET /css/main.css HTTP/1.1
178	3.580891606	192.168.1.105	157.140.2.32	HTTP	415	GET /css/summary-stats.css HTTP/1.1
206	3.919608257	157.140.2.32	192.168.1.105	HTTP	997	HTTP/1.1 200 OK (text/css)
212	3.922168993	192.168.1.105	157.140.2.32	HTTP	406	GET /css/page.css HTTP/1.1
213	3.931938093	157.140.2.32	192.168.1.105	HTTP	1325	HTTP/1.1 200 OK (text/css)
216	3.934661665	192.168.1.105	157.140.2.32	HTTP	468	GET /assets/logo/logo-explore.png HTTP/1.1
233	4.256253851	157.140.2.32	192.168.1.105	HTTP	946	HTTP/1.1 200 OK (text/css)
235	4.258110444	192.168.1.105	157.140.2.32	HTTP	471	GET /assets/sidebar/shrimp-202px.png HTTP/1.1
237	4.271181214	192.168.1.105	157.140.2.32	HTTP	466	GET /assets/external-link-explore.png HTTP/1.1
245	4.311452615	157.140.2.32	192.168.1.105	HTTP	4189	HTTP/1.1 200 OK (PNG)
290	4.613110215	157.140.2.32	192.168.1.105	HTTP	1064	HTTP/1.1 200 OK (PNG)
294	4.615122446	192.168.1.105	157.140.2.32	HTTP	468	GET /assets/sponsor-logos/ner.png HTTP/1.1
309	4.631509214	157.140.2.32	192.168.1.105	HTTP	2324	HTTP/1.1 200 OK (PNG)
315	4.637064509	192.168.1.105	157.140.2.32	HTTP	471	GET /assets/sponsor-logos/einfra.png HTTP/1.1
318	4.637254675	192.168.1.105	157.140.2.32	HTTP	473	GET /assets/sponsor-logos/emonocot.png HTTP/1.1
322	4.642067342	192.168.1.105	157.140.2.32	HTTP	472	GET /assets/sponsor-logos/vibrant.png HTTP/1.1
325	4.645623870	192.168.1.105	157.140.2.32	HTTP	468	GET /assets/sponsor-logos/nhm.png HTTP/1.1
348	4.958929278	157.140.2.32	192.168.1.105	HTTP	221	HTTP/1.1 200 OK (PNG)
351	4.959890932	192.168.1.105	157.140.2.32	HTTP	466	GET /assets/external-link-develop.png HTTP/1.1
360	4.976269429	157.140.2.32	192.168.1.105	HTTP	2323	HTTP/1.1 200 OK (PNG)
368	4.979269989	192.168.1.105	157.140.2.32	HTTP	466	GET /assets/external-link-support.png HTTP/1.1
370	4.981184336	157.140.2.32	192.168.1.105	HTTP	1537	HTTP/1.1 200 OK (PNG)
374	4.981184629	157.140.2.32	192.168.1.105	HTTP	10154	HTTP/1.1 200 OK (PNG)
382	4.991309022	157.140.2.32	192.168.1.105	HTTP	80	HTTP/1.1 200 OK (PNG)
408	5.282641527	157.140.2.32	192.168.1.105	HTTP	1080	HTTP/1.1 200 OK (PNG)
410	5.316691132	157.140.2.32	192.168.1.105	HTTP	1083	HTTP/1.1 200 OK (PNG)
412	5.331096985	192.168.1.105	157.140.2.32	HTTP	494	GET /favicon.ico HTTP/1.1
443	5.722279551	157.140.2.32	192.168.1.105	HTTP	974	HTTP/1.1 200 OK (image/vnd.microsoft.icon)

Fig.1

In the image attached above, we can see the **time**, **source** and **destination IP address**, **Protocol**, the **length** of the packer in bytes, and the **request line** of the HTTP message. The source IP address and the destination IP address can also be observed in the table,

The source IP address for the GET request is the user IP address in this case **192.168.1.105** and the destination IP address is **157.140.2.32**

As seen above the very first line is the HTTP GET method. The **GET method** is used to retrieve the data from the website.
As the URL is http://scratchpads.org/explore/sites-list in the path it is **/explore/sites-list.**
The HTTP version in the request line is seen to be HTTP 1.1, which is a **persistent connection.**

Fig. 2

In the above-attached picture, we can see the HTTP request message. The request line is followed by the **Host: scratchpads.org**, with various additional information including the **Connection: keep-alive**, **encoding: gzip**, **language: en-US**, **user-agent: Mozilla/5.0,** and **much more.**

Upon further analyzing the GET request further, we see the various fields:
In the IPv4 section, as shown in the attached picture below, The protocol field is seen to be **TCP**, the **Time to Live (TTL)** is 64, the flag fields gives further information about the sets and unset fields, here the **more fragments are set to zero** which implies that the TCP message is not split into more fragments.

```
▾ Transmission Control Protocol, Src Port: 43156, Dst Port: 80, Seq: 1, Ack: 1, Len: 467
    Source Port: 43156
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 467]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 499841864
    [Next Sequence Number: 468    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3387188790
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 502
    [Calculated window size: 64256]
    [Window size scaling factor: 128]
    Checksum: 0x63b7 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▾ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▸ TCP Option - No-Operation (NOP)
    ▸ TCP Option - No-Operation (NOP)
    ▸ TCP Option - Timestamps: TSval 3505276703, TSecr 3962180959
  ▾ [SEQ/ACK analysis]
      [iRTT: 0.326414764 seconds]
      [Bytes in flight: 467]
      [Bytes sent since last PSH flag: 467]
  ▾ [Timestamps]
      [Time since first frame in this TCP stream: 0.326660686 seconds]
      [Time since previous frame in this TCP stream: 0.000245922 seconds]
    TCP payload (467 bytes)
```

In the TCP section, we can observe the various TCP fields as listed below:

Source Port,

Destination Port,

Sequence Number: To ensure that the packets arrive on order,

Acknowledgment number: Cumulative acknowledgment,

Flags: To get additional information for troubleshooting and various other purposes,

Window size: This field is used by the receiver to indicate to the sender the amount of data that it is able to accept,

Checksum: simple error detection mechanism to determine the integrity of the data,

Options: Timestamps, Nop

As seen in figure 1 the response message first line is **HTTP/1.1 200 OK**

```
  TCP payload (2100 bytes)
  TCP segment data (2100 bytes)
[7 Reassembled TCP Segments (21668 bytes): #161(1368), #163(4200), #165(8400), #167(2800), #169(1400), #171(1400), #173(2100)]
  [Frame: 161, payload: 0-1367 (1368 bytes)]
  [Frame: 163, payload: 1368-5567 (4200 bytes)]
  [Frame: 165, payload: 5568-13967 (8400 bytes)]
  [Frame: 167, payload: 13968-16767 (2800 bytes)]
  [Frame: 169, payload: 16768-18167 (1400 bytes)]
  [Frame: 171, payload: 18168-19567 (1400 bytes)]
  [Frame: 173, payload: 19568-21667 (2100 bytes)]
  [Segment count: 7]
  [Reassembled TCP length: 21668]
  [Reassembled TCP Data: 485454502f312e3120323030204f4b0d0a446174653a204d6f6e2c203034204f63742032…]
```

On analyzing the packets it is seen that the response message has arrived in **7 TCP segments** as mentioned in the attached picture above.

```
Line-based text data: text/html (9210 lines)
    <!doctype html>\n
    \n
    <html lang="en">\n
    <head>\n
      <meta charset="utf-8">\n
      <meta name="viewport" content="width=device-width, initial-scale=1">\n
      <link rel="stylesheet" href="/css/main.css">\n
      \n
      <link rel="stylesheet" href="/css/summary-stats.css">\n
      \n
      \n
      <link rel="stylesheet" href="/css/page.css">\n
      \n
      <link type="application/atom+xml" rel="alternate" href="http://scratchpads.org/feed.xml" title="scratchpads.eu" />\n
      \n
      <script>\n
        function skipToMain () {\n
          var main = document.getElementById('main');\n
          main.setAttribute('tabindex', -1);\n
          main.focus();\n
        }\n
      </script>\n
    \n
```

In the TCP response message above, we can see the text data of the web page is transmitted.

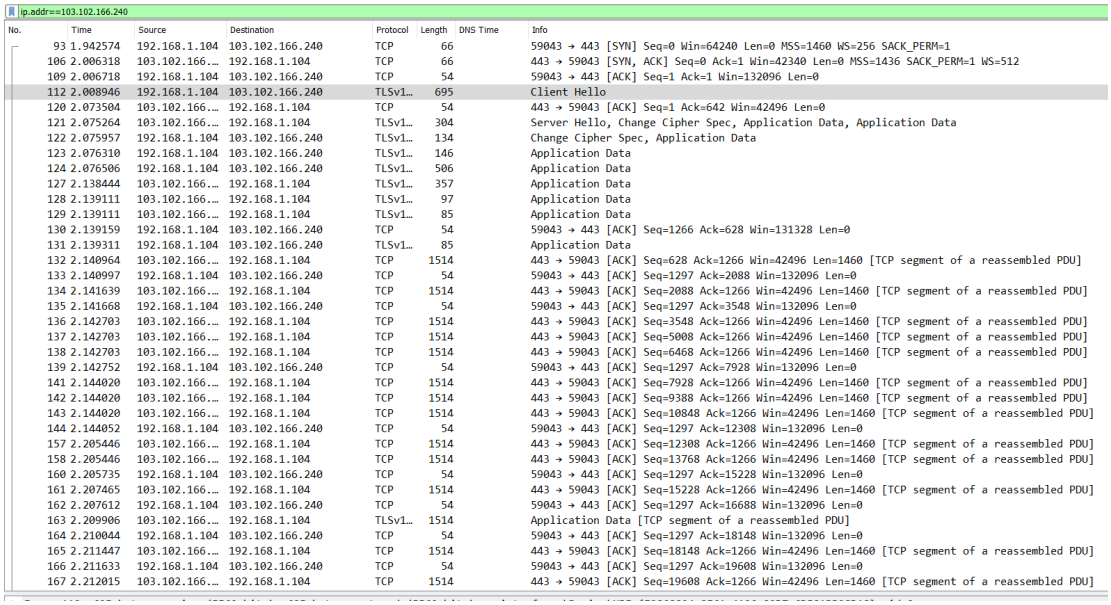
HTTPS:

Hypertext Transfer Protocol Secure is an **extension of HTTP** which is used for secure communication over a computer network and is the most widely used internet protocol. It is encrypted in order to increase the security of data transfer. It uses **Transport Layer Security(TLS)** / **Secure Socket Layer(SSL)** protocol to encrypt communication. It uses two types of keys for encryption:

1. **Private Key**: This key is controlled by the owner of a website and it's kept, as the reader may have speculated, private.
2. **Public Key**: This key is available to everyone who wants to interact with the server in a way that's secure.

For the following experiment we will be analyzing **https://en.wikiversity.org.**

We can observe the captured packets for required destination bellow:

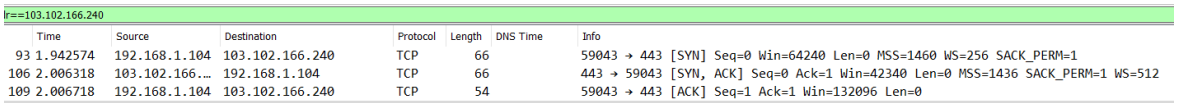


ip.addr==103.102.166.240

No.	Time	Source	Destination	Protocol	Length	DNS Time	Info
93	1.942574	192.168.1.104	103.102.166.240	TCP	66		59043 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
106	2.006318	103.102.166.…	192.168.1.104	TCP	66		443 → 59043 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1436 SACK_PERM=1 WS=512
109	2.006718	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0
112	2.008946	192.168.1.104	103.102.166.240	TLSv1…	695		Client Hello
120	2.073504	103.102.166.…	192.168.1.104	TCP	54		443 → 59043 [ACK] Seq=1 Ack=642 Win=42496 Len=0
121	2.075264	103.102.166.…	192.168.1.104	TLSv1…	304		Server Hello, Change Cipher Spec, Application Data, Application Data
122	2.075957	192.168.1.104	103.102.166.240	TLSv1…	134		Change Cipher Spec, Application Data
123	2.076310	192.168.1.104	103.102.166.240	TLSv1…	146		Application Data
124	2.076506	192.168.1.104	103.102.166.240	TLSv1…	506		Application Data
127	2.138444	103.102.166.…	192.168.1.104	TLSv1…	357		Application Data
128	2.139111	103.102.166.…	192.168.1.104	TLSv1…	97		Application Data
129	2.139111	103.102.166.…	192.168.1.104	TLSv1…	85		Application Data
130	2.139159	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1266 Ack=628 Win=131328 Len=0
131	2.139311	192.168.1.104	103.102.166.240	TLSv1…	85		Application Data
132	2.140964	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=628 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
133	2.140997	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=2088 Win=132096 Len=0
134	2.141639	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=2088 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
135	2.141668	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=3548 Win=132096 Len=0
136	2.142703	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=3548 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
137	2.142703	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=5008 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
138	2.142703	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=6468 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
139	2.142752	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=7928 Win=132096 Len=0
141	2.144020	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=7928 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
142	2.144020	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=9388 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
143	2.144020	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=10848 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
144	2.144052	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=12308 Win=132096 Len=0
157	2.205446	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=12308 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
158	2.205446	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=13768 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
160	2.205735	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=15228 Win=132096 Len=0
161	2.207465	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=15228 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
162	2.207612	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=16688 Win=132096 Len=0
163	2.209906	103.102.166.…	192.168.1.104	TLSv1…	1514		Application Data [TCP segment of a reassembled PDU]
164	2.210044	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=18148 Win=132096 Len=0
165	2.211447	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=18148 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
166	2.211633	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1297 Ack=19608 Win=132096 Len=0
167	2.212015	103.102.166.…	192.168.1.104	TCP	1514		443 → 59043 [ACK] Seq=19608 Ack=1266 Win=42496 Len=1460 [TCP segment of a reassembled PDU]

We can observe that the destination IP address is 103.102.166.240

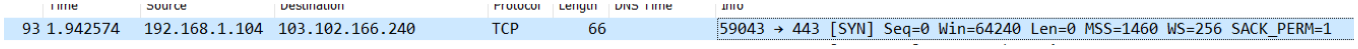
TCP Connection Traffic:



ip.addr==103.102.166.240

	Time	Source	Destination	Protocol	Length	DNS Time	Info
93	1.942574	192.168.1.104	103.102.166.240	TCP	66		59043 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
106	2.006318	103.102.166.…	192.168.1.104	TCP	66		443 → 59043 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1436 SACK_PERM=1 WS=512
109	2.006718	192.168.1.104	103.102.166.240	TCP	54		59043 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0

Above we can observe that the first three packets correspond to the **TCP three-way handshake**: **TCP SYN**, **TCP SYN / ACK**, **TCK ACK**.



	Time	Source	Destination	Protocol	Length	DNS Time	Info
93	1.942574	192.168.1.104	103.102.166.240	TCP	66		59043 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

If we analyze the above packet we get the following fields:

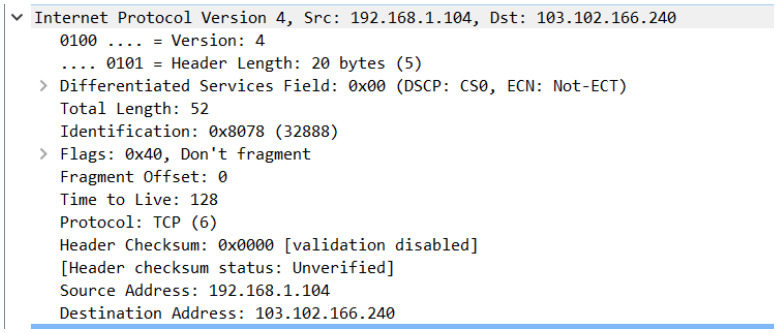
```
Frame 93: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{E28C8894-8E61-4A9C-89D7-6D581BB98DA8}, id 0
Ethernet II, Src: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2), Dst: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
Internet Protocol Version 4, Src: 192.168.1.104, Dst: 103.102.166.240
Transmission Control Protocol, Src Port: 59043, Dst Port: 443, Seq: 0, Len: 0
```

On examining the Ethernet II field we get:

```
▾ Destination: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
      Address: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
▾ Source: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
      Address: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

1. **Destination address**: Default Gateway's MAC address
2. **Source address**: Host's MAC address.

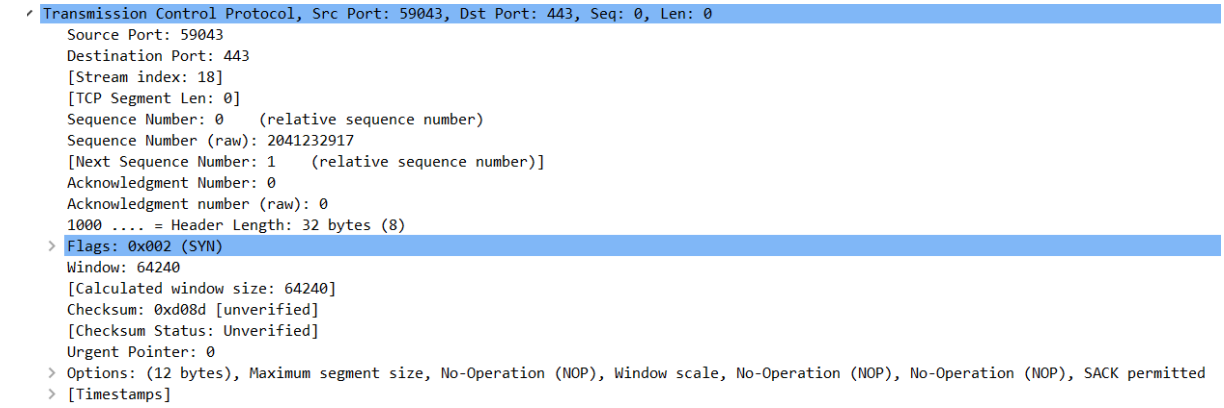
On examining the Internet Protocol Version 4 field we get:



It mainly consists of:

1. **Source Address:** Host's IP address
2. **Destination Address:** IP address of the HTTPS server

On examining **Transmission Control Protocol** field we get:

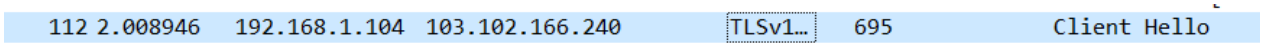


We can observe that:

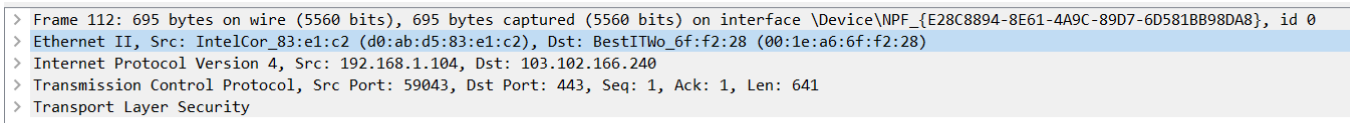
1. **Source Port**: Dynamic port selected for the HTTPS connection.
2. **Destination Port**: HTTPS port number i.e. 443

SSL/TLS Client Hello Traffic:

We can observe the following packet:



The above packet consists of the following fields:



On expanding **Transport Layer Security** we can observe the following:

```
Transport Layer Security
▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 636
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 632
      Version: TLS 1.2 (0x0303)
      Random: 5d4cf93a3783da302be8728037641b65f34dd765c0855e23b58da97d4cc5b6b4
      Session ID Length: 32
      Session ID: a88299193d3ac1a0778945826e490ca0d53093bb4236cfbb810fb20132f4b25f
      Cipher Suites Length: 32
    ▸ Cipher Suites (16 suites)
      Compression Methods Length: 1
    ▸ Compression Methods (1 method)
      Extensions Length: 527
    ▸ Extension: Reserved (GREASE) (len=0)
    ▸ Extension: server_name (len=25)
    ▸ Extension: extended_master_secret (len=0)
    ▸ Extension: renegotiation_info (len=1)
    ▸ Extension: supported_groups (len=10)
    ▸ Extension: ec_point_formats (len=2)
    ▸ Extension: session_ticket (len=0)
    ▸ Extension: application_layer_protocol_negotiation (len=14)
    ▸ Extension: status_request (len=5)
    ▸ Extension: signature_algorithms (len=18)
    ▸ Extension: signed_certificate_timestamp (len=0)
    ▸ Extension: key_share (len=43)
    ▸ Extension: psk_key_exchange_modes (len=2)
    ▸ Extension: supported_versions (len=11)
    ▸ Extension: compress_certificate (len=3)
    ▸ Extension: Unknown type 17513 (len=5)
    ▸ Extension: Reserved (GREASE) (len=1)
    ▸ Extension: pre_shared_key (len=315)
```

Handshake Protocol consists of:

1. **Handshake Type:** Client Hello message initiates the handshake.
2. **Cipher suites:** Sets of instructions that enable secure network connections through Transport Layer Security (TLS).
3. **Extensions:** They are used to extend the TLS capabilities without actually modifying the protocol itself.

The following package is received as TCP acknowledgment to client hello request:

120	2.073504	103.102.166.…	192.168.1.104	TCP	54	443 → 59043 [ACK] Seq=1 Ack=642 Win=42496 Len=0

SSL/TLS Server Hello Traffic:

We can observe the following packet:

121	2.075264	103.102.166.…	192.168.1.104	TLSv1…	304	Server Hello, Change Cipher Spec, Application Data, Application Data

Above packet consists of following fields:

```
▸ Frame 121: 304 bytes on wire (2432 bits), 304 bytes captured (2432 bits) on interface \Device\NPF_{E28C8894-8E61-4A9C-89D7-6D581BB98DA8}, id 0
▸ Ethernet II, Src: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28), Dst: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
▸ Internet Protocol Version 4, Src: 103.102.166.240, Dst: 192.168.1.104
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 59043, Seq: 1, Ack: 642, Len: 250
▸ Transport Layer Security
```

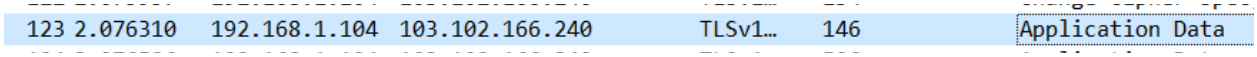
On expanding **Transport Layer Security** we can observe the following:

```
Transport Layer Security
▾ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 128
  ▾ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 124
      Version: TLS 1.2 (0x0303)
      Random: 106fe7a468fd314acab5e3b786be54da984b964e77f1f2eb4140dbf608f495c6
      Session ID Length: 32
      Session ID: a88299193d3ac1a0778945826e490ca0d53093bb4236cfbb810fb20132f4b25f
      Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
      Compression Method: null (0)
      Extensions Length: 52
    › Extension: supported_versions (len=2)
    › Extension: key_share (len=36)
    › Extension: pre_shared_key (len=2)
▾ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 32
    Encrypted Application Data: 317984a4a51bd68b686da712bfa67593b83bc9634daad3024872be7ad7750190
    [Application Data Protocol: http-over-tls]
▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 69
    Encrypted Application Data: 2dc3c723fc9cbe9b5f72f6567234bfc083678152c32a02807cb29580c897afa1d52bf772…
    [Application Data Protocol: http-over-tls]
```

Similar to the client hello package, it consists of the handshake protocol subfield. Along with that, it consists of information about the **Application Layer protocol** and **Change Cipher Spec protocol**. It also consists of the **session id** that is then used to encrypt session traffic.

HTTPs Encrypted data exchange:

We can observe the following packet:



It consists of the following fields:

```
› Frame 123: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface \Device\NPF_{E28C8894-8E61-4A9C-89D7-6D581BB98DA8}, id 0
› Ethernet II, Src: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2), Dst: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
› Internet Protocol Version 4, Src: 192.168.1.104, Dst: 103.102.166.240
› Transmission Control Protocol, Src Port: 59043, Dst Port: 443, Seq: 722, Ack: 251, Len: 92
› Transport Layer Security
```

Here we can observe the **encrypted application data** and its protocol is HTTP.

```
Transport Layer Security
▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 87
    Encrypted Application Data: 3376ecd7a4e6626802f8f2b43f5f121a14524eea99367ce1b971038f43e46c5ee21abaaf…
    [Application Data Protocol: http-over-tls]
```

Encrypted application data:

```
0000  00 1e a6 6f f2 28 d0 ab  d5 83 e1 c2 08 00 45 00   ···o·(·· ·····E·
0010  00 84 80 7c 40 00 80 06  00 00 c0 a8 01 68 67 66   ···|@··· ·····hgf
0020  a6 f0 e6 a3 01 bb 79 aa  c0 e7 30 86 07 9d 50 18   ·····y· ··0···P·
0030  02 03 d0 dd 00 00 17 03  03 00 57 33 76 ec d7 a4   ········ ··W3v···
0040  e6 62 68 02 f8 f2 b4 3f  5f 12 1a 14 52 4e ea 99   ·bh····? _···RN··
0050  36 7c e1 b9 71 03 8f 43  e4 6c 5e e2 1a ba af 60   6|··q··C ·l^····`
0060  d9 d2 87 96 5b a2 e4 43  e9 53 24 47 df d5 e7 86   ····[··C ·S$G····
0070  87 b1 13 11 3f d9 50 90  c1 71 4c 13 aa 04 d6 a8   ····?·P· ·qL·····
0080  da f7 b4 07 32 db c8 66  15 cc 3d b0 d6 5f 0f a1   ····2··f ··=··_··
0090  82 0b                                             ··
```
