Ivan Fernandez

SER 321 – Summer C

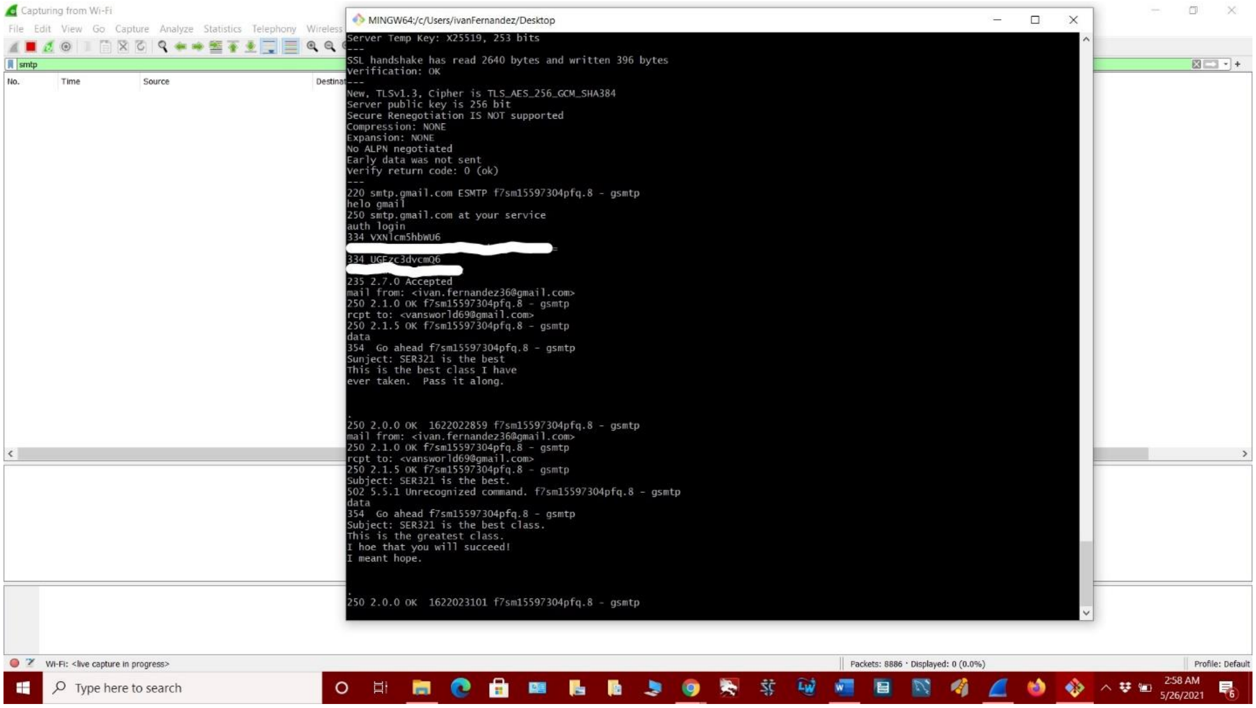
5/25/2021

Assignment 2

Upper Layers of the OSI-Model

1 Sending emails and SMTP

Screenshots:

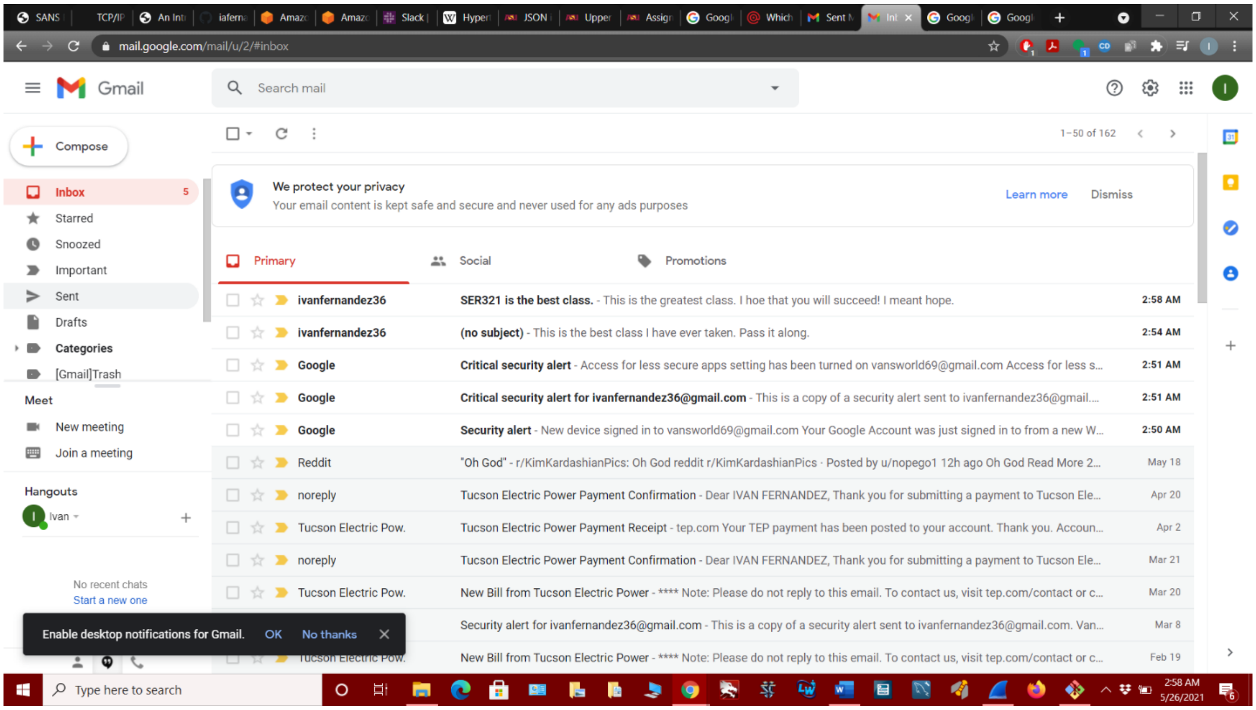


```
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 2640 bytes and written 396 bytes
Verification: OK
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 256 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---
220 smtp.gmail.com ESMTP f7sm15597304pfq.8 - gsmtp
helo gmail
250 smtp.gmail.com at your service
auth login
334 VXNlcm5hbWU6
[redacted]
334 UGFzc3dvcmQ6
[redacted]
235 2.7.0 Accepted
mail from: <ivan.fernandez36@gmail.com>
250 2.1.0 OK f7sm15597304pfq.8 - gsmtp
rcpt to: <vansworld69@gmail.com>
250 2.1.5 OK f7sm15597304pfq.8 - gsmtp
data
354  Go ahead f7sm15597304pfq.8 - gsmtp
Sunject: SER321 is the best
This is the best class I have
ever taken.  Pass it along.


.
250 2.0.0 OK  1622022859 f7sm15597304pfq.8 - gsmtp
mail from: <ivan.fernandez36@gmail.com>
250 2.1.0 OK f7sm15597304pfq.8 - gsmtp
rcpt to: <vansworld69@gmail.com>
250 2.1.5 OK f7sm15597304pfq.8 - gsmtp
Subject: SER321 is the best.
502 5.5.1 Unrecognized command. f7sm15597304pfq.8 - gsmtp
data
354  Go ahead f7sm15597304pfq.8 - gsmtp
Subject: SER321 is the best class.
This is the greatest class.
I hoe that you will succeed!
I meant hope.


.
250 2.0.0 OK  1622023101 f7sm15597304pfq.8 - gsmtp
```

Wi-Fi: <live capture in progress>
Packets: 8886 · Displayed: 0 (0.0%)
Profile: Default
Type here to search
2:58 AM
5/26/2021



We protect your privacy
Your email content is kept safe and secure and never used for any ads purposes

ivanfernandez36	**SER321 is the best class.** - This is the greatest class. I hoe that you will succeed! I meant hope.	2:58 AM
ivanfernandez36	**(no subject)** - This is the best class I have ever taken. Pass it along.	2:54 AM
Google	**Critical security alert** - Access for less secure apps setting has been turned on vansworld69@gmail.com Access for less s...	2:51 AM
Google	**Critical security alert for ivanfernandez36@gmail.com** - This is a copy of a security alert sent to ivanfernandez36@gmail....	2:51 AM
Google	**Security alert** - New device signed in to vansworld69@gmail.com Your Google Account was just signed in to from a new W...	2:50 AM
Reddit	"Oh God" - r/KimKardashianPics: Oh God reddit r/KimKardashianPics · Posted by u/nopego1 12h ago Oh God Read More 2...	May 18
noreply	Tucson Electric Power Payment Confirmation - Dear IVAN FERNANDEZ, Thank you for submitting a payment to Tucson Ele...	Apr 20
Tucson Electric Pow.	Tucson Electric Power Payment Receipt - tep.com Your TEP payment has been posted to your account. Thank you. Accoun...	Apr 2
noreply	Tucson Electric Power Payment Confirmation - Dear IVAN FERNANDEZ, Thank you for submitting a payment to Tucson Ele...	Mar 21
Tucson Electric Pow.	New Bill from Tucson Electric Power - **** Note: Please do not reply to this email. To contact us, visit tep.com/contact or c...	Mar 20
	Security alert for ivanfernandez36@gmail.com - This is a copy of a security alert sent to ivanfernandez36@gmail.com. Van...	Mar 8
Tucson Electric Pow.	New Bill from Tucson Electric Power - **** Note: Please do not reply to this email. To contact us, visit tep.com/contact or c...	Feb 19

1. What filter did you use to catch the traffic and explain why you chose that filter?

I chose the filter tcp.port == 465 because it is the port used to establish communications between the client and the gmail server.

2. What is the standard port for SMTP and why do we use port 465 in the example above?

The standard port for SMTP is port 587 for submissions by mail clients to mail servers. Port 25 is recommended for transmissions between mail servers. Port 465 was used because it uses SSL to encrypt transmissions over the internet; it is for SMTPS. It is also for mail client to mail server communications.

3. Explain each line used in the command line and what it does and why it is needed?

1. echo−ne username | base64 This command is used to encode the username to ensure that it will transmit across the network uncorrupted.

2. echo−ne password | base64 This command is used to encode the password to ensure that it will transmit across the network uncorrupted.

3. openssl s_client−c r l f−ign_eof−connect smtp . gmail . com:465 Creates the connection to the google gmail server via tcp port 465.

4. helo gmail . com Ensures that the connection was successful.

5. auth login Login into your gmail account, using base64 encrypted username and password.

6. mail from : username@gmail . com Begins email creation from owner's email account.

7. rcpt to : username@gmail . com The recipient's email account.

8. data This command precedes the creation of the email.

9. Subject : Your subject Will create the subject line for the email.

10. Email text goes here . The actual email text is typed into this section.

11. . This period is used to end the email message.

12. quit This closes the connection to the google gmail server.

4. How much back and forth communication do you see for establishing the connection?

We can easily see the three-way handshake, the local tcp port 14410 establishes communications with web server tcp port 465. Three packages can be observed, we can see a [SYN] , [SYN, ACK], and finally an [ACK] from the local port establishing communications with the web server.

5. What is the port your local machine is using between sending the two emails when communicating with the SMTP server?

The port being used is port 14410.

6. Explain who sends the first FIN flag and how the quitting process works.

It can be see that the web server sends the [FIN, ACK] flags first to the local port, the local port then responds with an [ACK]. Application data is sent to the local port, and then port 14410 sends back [FIN, ACK] to port 465. Port 465 then sends [RST] reset flag to port 14410.

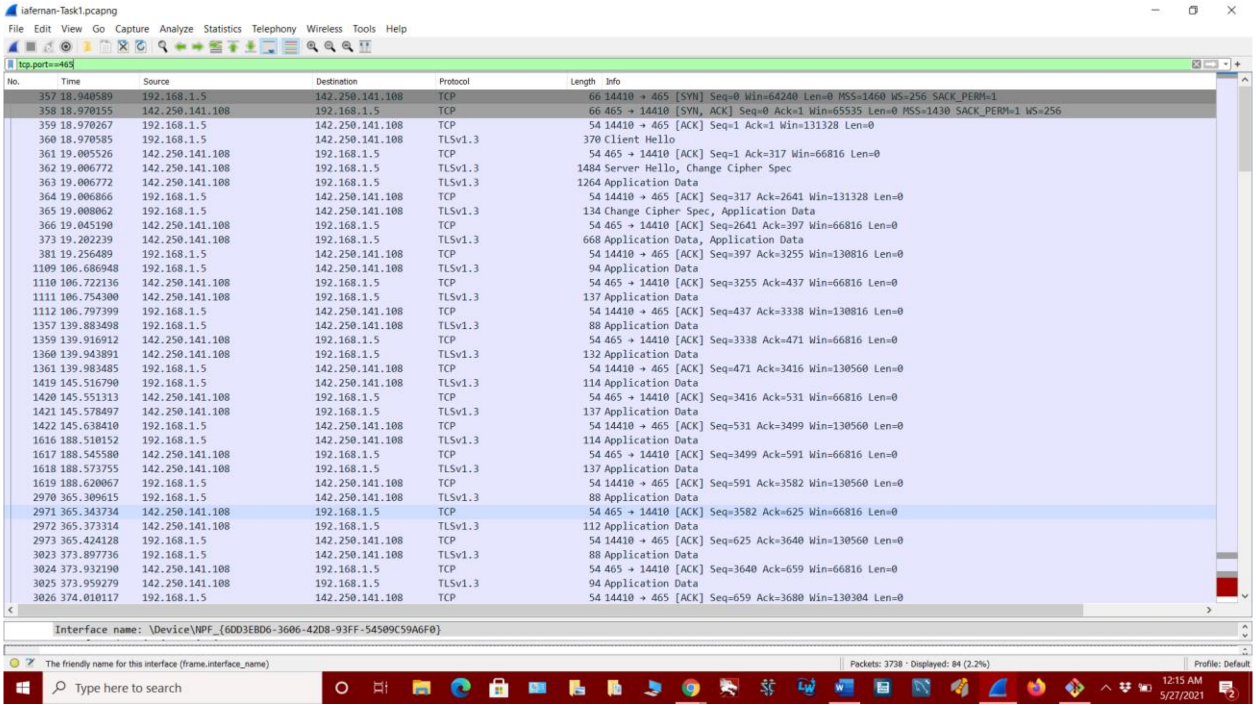
3652 531.130032 142.250.141.108 192.168.1.5 TCP 54 465 → 14410 [FIN, ACK] Seq=4194 Ack=1222 Win=66816 Len=0

3653 531.130126 192.168.1.5 142.250.141.108 TCP 54 14410 → 465 [ACK] Seq=1222 Ack=4195 Win=131328 Len=0

3654 531.130213 192.168.1.5 142.250.141.108 TLSv1.3 78 Application Data

3655 531.130286 192.168.1.5 142.250.141.108 TCP 54 14410 → 465 [FIN, ACK] Seq=1246 Ack=4195 Win=131328 Len=0

7. Add a screenshot of your Wireshark output and add it to your document.


iafernan-Task1.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==465

```
No.   Time        Source           Destination      Protocol  Length Info
  357 18.940589   192.168.1.5      142.250.141.108  TCP         66 14410 → 465 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  358 18.970155   142.250.141.108  192.168.1.5      TCP         66 465 → 14410 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
  359 18.970267   192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=1 Ack=1 Win=131328 Len=0
  360 18.970585   192.168.1.5      142.250.141.108  TLSv1.3    370 Client Hello
  361 19.005526   142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=1 Ack=317 Win=66816 Len=0
  362 19.006772   142.250.141.108  192.168.1.5      TLSv1.3   1484 Server Hello, Change Cipher Spec
  363 19.006772   142.250.141.108  192.168.1.5      TLSv1.3   1264 Application Data
  364 19.006866   192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=317 Ack=2641 Win=131328 Len=0
  365 19.008062   192.168.1.5      142.250.141.108  TLSv1.3    134 Change Cipher Spec, Application Data
  366 19.045190   142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=2641 Ack=397 Win=66816 Len=0
  373 19.202239   142.250.141.108  192.168.1.5      TLSv1.3    668 Application Data, Application Data
  381 19.256489   192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=397 Ack=3255 Win=130816 Len=0
 1109 106.686948  192.168.1.5      142.250.141.108  TLSv1.3     94 Application Data
 1110 106.722136  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3255 Ack=437 Win=66816 Len=0
 1111 106.754300  142.250.141.108  192.168.1.5      TLSv1.3    137 Application Data
 1112 106.797399  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=437 Ack=3338 Win=130816 Len=0
 1357 139.883498  192.168.1.5      142.250.141.108  TLSv1.3     88 Application Data
 1359 139.916912  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3338 Ack=471 Win=66816 Len=0
 1360 139.943891  142.250.141.108  192.168.1.5      TLSv1.3    132 Application Data
 1361 139.983485  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=471 Ack=3416 Win=130560 Len=0
 1419 145.516790  192.168.1.5      142.250.141.108  TLSv1.3    114 Application Data
 1420 145.551313  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3416 Ack=531 Win=66816 Len=0
 1421 145.578497  142.250.141.108  192.168.1.5      TLSv1.3    137 Application Data
 1422 145.638410  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=531 Ack=3499 Win=130560 Len=0
 1616 188.510152  192.168.1.5      142.250.141.108  TLSv1.3    114 Application Data
 1617 188.545580  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3499 Ack=591 Win=66816 Len=0
 1618 188.573755  142.250.141.108  192.168.1.5      TLSv1.3    137 Application Data
 1619 188.620067  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=591 Ack=3582 Win=130560 Len=0
 2970 365.309615  192.168.1.5      142.250.141.108  TLSv1.3     88 Application Data
 2971 365.343734  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3582 Ack=625 Win=66816 Len=0
 2972 365.373314  142.250.141.108  192.168.1.5      TLSv1.3    112 Application Data
 2973 365.424128  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=625 Ack=3640 Win=130560 Len=0
 3023 373.897736  192.168.1.5      142.250.141.108  TLSv1.3     88 Application Data
 3024 373.932190  142.250.141.108  192.168.1.5      TCP         54 465 → 14410 [ACK] Seq=3640 Ack=659 Win=66816 Len=0
 3025 373.959279  142.250.141.108  192.168.1.5      TLSv1.3     94 Application Data
 3026 374.010117  192.168.1.5      142.250.141.108  TCP         54 14410 → 465 [ACK] Seq=659 Ack=3680 Win=130304 Len=0
```

```
Interface name: \Device\NPF_{6DD3EBD6-3606-42D8-93FF-54509C59A6F0}
```

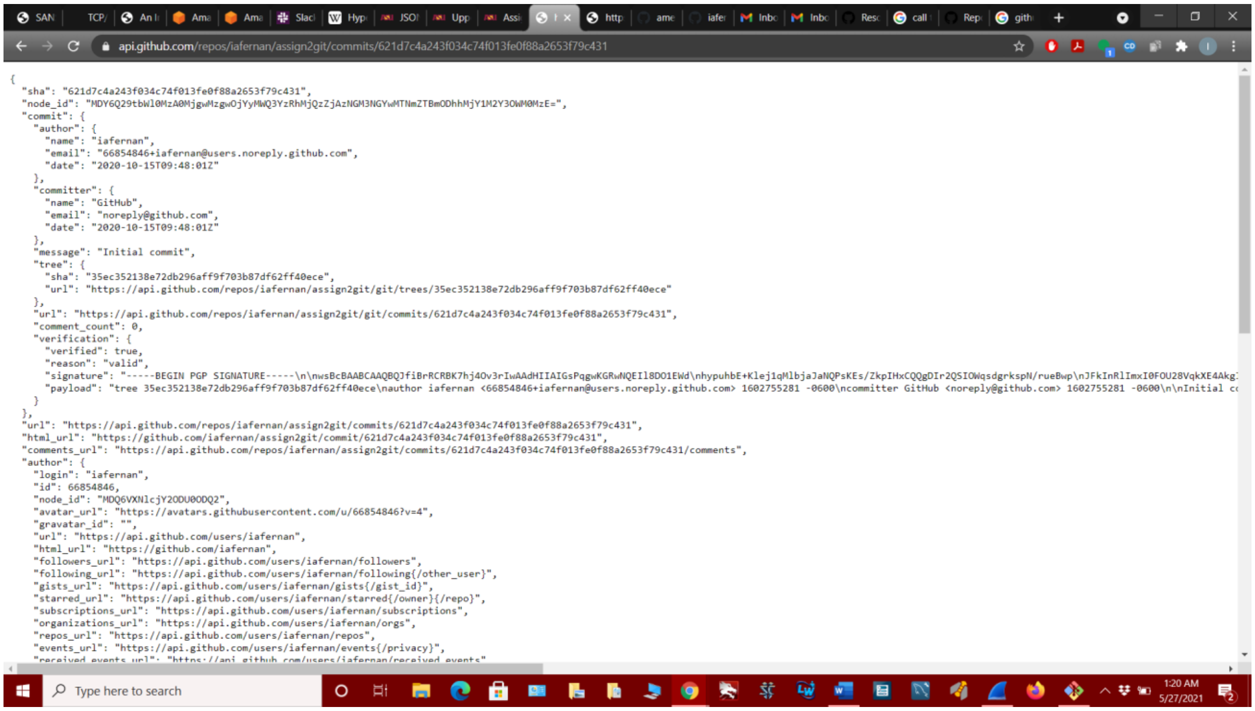
The friendly name for this interface (frame.interface_name)

Packets: 3738 · Displayed: 84 (2.2%)

Profile: Default

2 Understanding HTTP

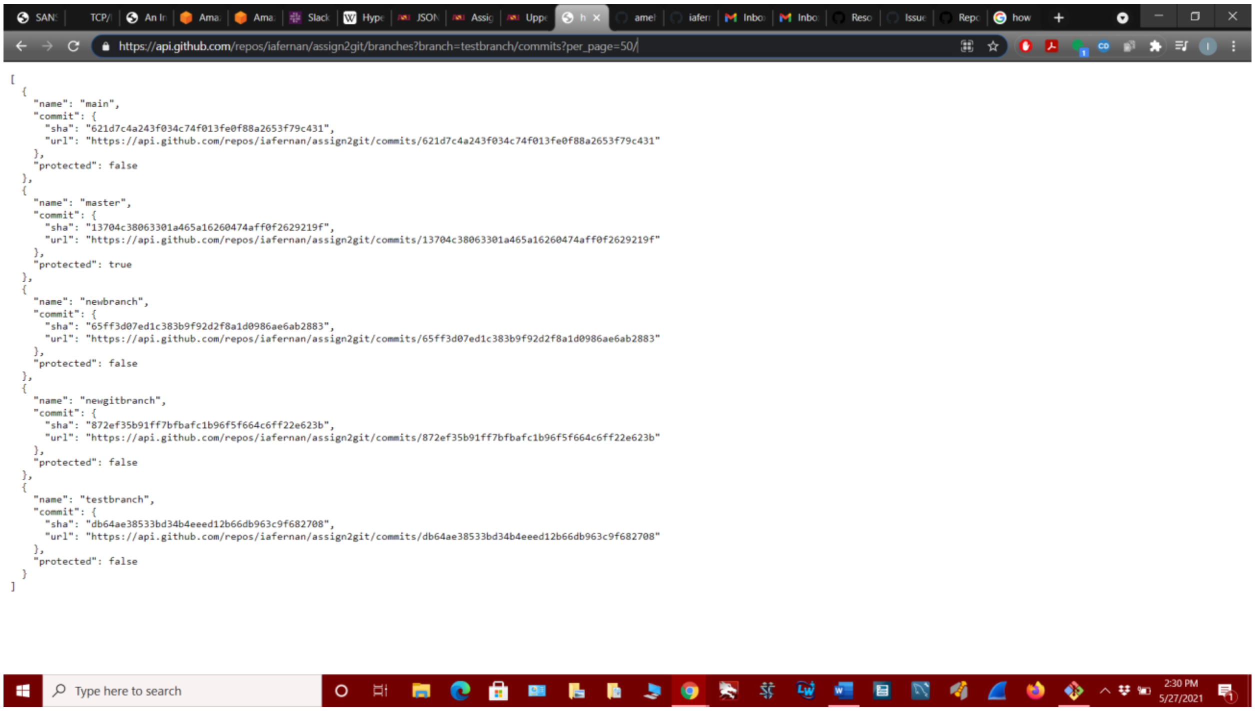
Screenshot 1:



api.github.com/repos/iafernan/assign2git/commits/621d7c4a243f034c74f013fe0f88a2653f79c431

```
{
  "sha": "621d7c4a243f034c74f013fe0f88a2653f79c431",
  "node_id": "MDY6Q29tbWl0MzA0MjgwMzgwOjYyMWQ3YzRhMjQzZjAzNGM3NGYwMTNmZTBmODhhMjY1M2Y3OWM0MzE=",
  "commit": {
    "author": {
      "name": "iafernan",
      "email": "66854846+iafernan@users.noreply.github.com",
      "date": "2020-10-15T09:48:01Z"
    },
    "committer": {
      "name": "GitHub",
      "email": "noreply@github.com",
      "date": "2020-10-15T09:48:01Z"
    },
    "message": "Initial commit",
    "tree": {
      "sha": "35ec352138e72db296aff9f703b87df62ff40ece",
      "url": "https://api.github.com/repos/iafernan/assign2git/git/trees/35ec352138e72db296aff9f703b87df62ff40ece"
    },
    "url": "https://api.github.com/repos/iafernan/assign2git/git/commits/621d7c4a243f034c74f013fe0f88a2653f79c431",
    "comment_count": 0,
    "verification": {
      "verified": true,
      "reason": "valid",
      "signature": "-----BEGIN PGP SIGNATURE-----\n\nwsBcBAABCAAQBQJfiBrRCRBK7hj4Ov3rIwAAdHIIAIGsPqgwKGRwNQEIl8DO1EWd\nhypuhbE+Klej1qMlbjaJaNQPsKEs/ZkpIHxCQQgDIr2QSIOWqsdgrkspN/rueBwp\nJFkInRlImxI0FOU28VqkXE4Akg
      "payload": "tree 35ec352138e72db296aff9f703b87df62ff40ece\nauthor iafernan <66854846+iafernan@users.noreply.github.com> 1602755281 -0600\ncommitter GitHub <noreply@github.com> 1602755281 -0600\n\nInitial co
    }
  },
  "url": "https://api.github.com/repos/iafernan/assign2git/commits/621d7c4a243f034c74f013fe0f88a2653f79c431",
  "html_url": "https://github.com/iafernan/assign2git/commit/621d7c4a243f034c74f013fe0f88a2653f79c431",
  "comments_url": "https://api.github.com/repos/iafernan/assign2git/commits/621d7c4a243f034c74f013fe0f88a2653f79c431/comments",
  "author": {
    "login": "iafernan",
    "id": 66854846,
    "node_id": "MDQ6VXNlcjY2ODU0ODQ2",
    "avatar_url": "https://avatars.githubusercontent.com/u/66854846?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/iafernan",
    "html_url": "https://github.com/iafernan",
    "followers_url": "https://api.github.com/users/iafernan/followers",
    "following_url": "https://api.github.com/users/iafernan/following{/other_user}",
    "gists_url": "https://api.github.com/users/iafernan/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/iafernan/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/iafernan/subscriptions",
    "organizations_url": "https://api.github.com/users/iafernan/orgs",
    "repos_url": "https://api.github.com/users/iafernan/repos",
    "events_url": "https://api.github.com/users/iafernan/events{/privacy}",
    "received_events_url": "https://api.github.com/users/iafernan/received_events",
```

Screenshot 2:



https://api.github.com/repos/iafernan/assign2git/branches?branch=testbranch/commits?per_page=50/

```
[
  {
    "name": "main",
    "commit": {
      "sha": "621d7c4a243f034c74f013fe0f88a2653f79c431",
      "url": "https://api.github.com/repos/iafernan/assign2git/commits/621d7c4a243f034c74f013fe0f88a2653f79c431"
    },
    "protected": false
  },
  {
    "name": "master",
    "commit": {
      "sha": "13704c38063301a465a16260474aff0f2629219f",
      "url": "https://api.github.com/repos/iafernan/assign2git/commits/13704c38063301a465a16260474aff0f2629219f"
    },
    "protected": true
  },
  {
    "name": "newbranch",
    "commit": {
      "sha": "65ff3d07ed1c383b9f92d2f8a1d0986ae6ab2883",
      "url": "https://api.github.com/repos/iafernan/assign2git/commits/65ff3d07ed1c383b9f92d2f8a1d0986ae6ab2883"
    },
    "protected": false
  },
  {
    "name": "newgitbranch",
    "commit": {
      "sha": "872ef35b91ff7bfbafc1b96f5f664c6ff22e623b",
      "url": "https://api.github.com/repos/iafernan/assign2git/commits/872ef35b91ff7bfbafc1b96f5f664c6ff22e623b"
    },
    "protected": false
  },
  {
    "name": "testbranch",
    "commit": {
      "sha": "db64ae38533bd34b4eeed12b66db963c9f682708",
      "url": "https://api.github.com/repos/iafernan/assign2git/commits/db64ae38533bd34b4eeed12b66db963c9f682708"
    },
    "protected": false
  }
]
```

1. Explain the specific API calls you used.

I used the calls:

`https://api.github.com/repos/iafernan/assign2git/commits/db64ae38533bd34b4eeed12b66db963c9f682708`

https://api.github.com/repos/iafernan/assign2git/branches?branch=testbranch/commits?per_page=50/

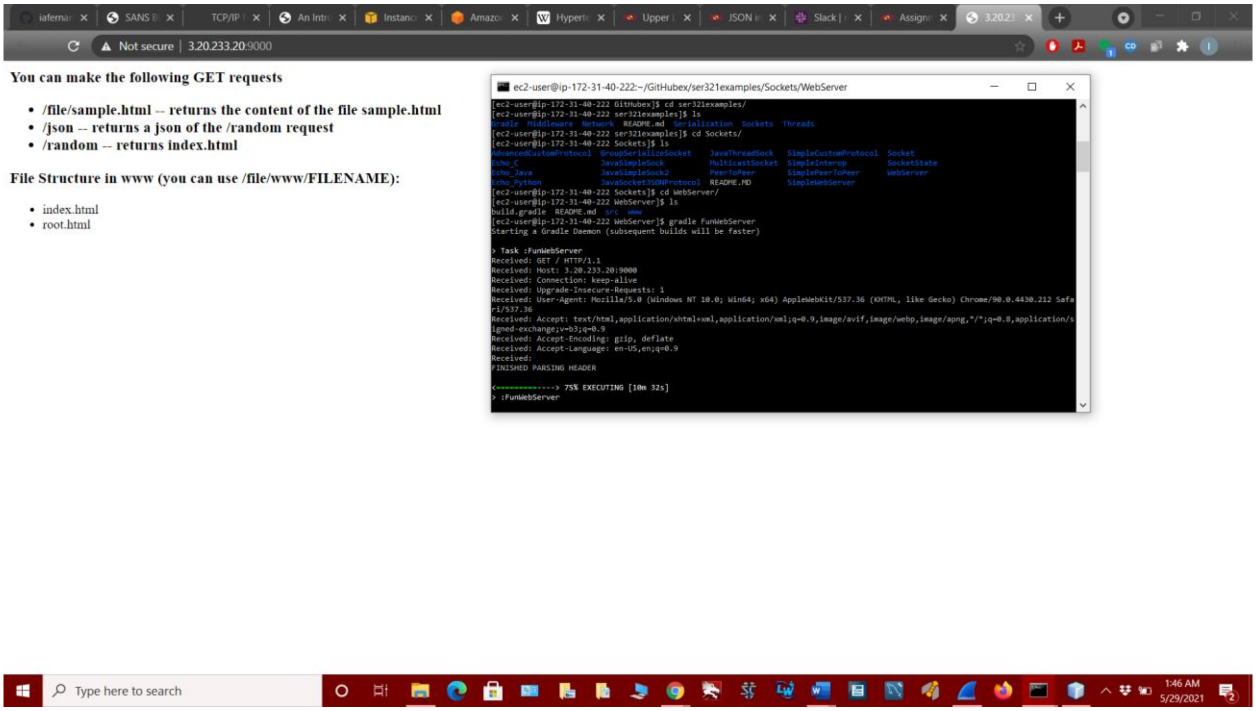
To get a listing of the commits for the test branch in my assign2git repo on GitHub.

2. Explain the difference between stateless and a state-full communication.

HTTP/HTTPS are stateless protocols, in which a user submits a request to a web browser for example, and HTTP/HTTPS does not save any states of the requests made to any web services. The web service, for example Google, may save information regarding your states and searches, as well as web browser cookies, but the HTTP/HTTPS protocols do not. In short, HTTP/HTTPS does not keep track of your previous web browsing requests or information. In contrast, FTP is a protocol that is stateful, and creates a connection to a server that keeps track and remembers requests that have been made during the session. FTP is unique because it uses two connections to the server to one establish and maintain the connection, and second to send and receive information.

3 Setup your second system and run Server on it

3.2 Running a simple Java Web Server



You can make the following GET requests

- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

File Structure in www (you can use /file/www/FILENAME):

- index.html
- root.html

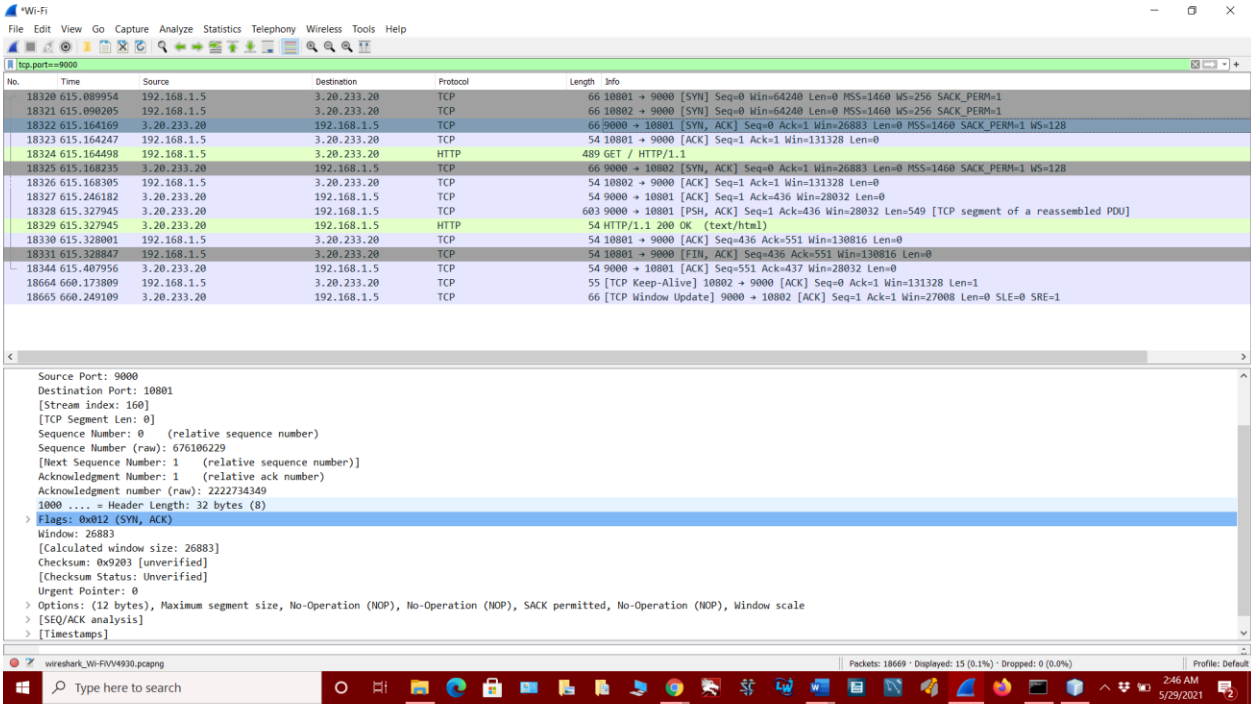
ec2-user@ip-172-31-40-222:~/GitHubex/ser321examples/Sockets/WebServer

```
[ec2-user@ip-172-31-40-222 GitHubex]$ cd ser321examples/
[ec2-user@ip-172-31-40-222 ser321examples]$ ls
Gradle  Middleware  Network  README.md  Serialization  Sockets  Threads
[ec2-user@ip-172-31-40-222 ser321examples]$ cd Sockets/
[ec2-user@ip-172-31-40-222 Sockets]$ ls
AdvancedCustomProtocol  GroupSerializeSocket    JavaThreadSock    SimpleCustomProtocol  Socket
Echo_C                  JavaSimpleSock          MulticastSocket   SimpleInterop         SocketState
Echo_Java               JavaSimpleSock2         PeerToPeer        SimplePeerToPeer      WebServer
Echo_Python             JavaSocketJSONProtocol  README.MD         SimpleWebServer
[ec2-user@ip-172-31-40-222 Sockets]$ cd WebServer/
[ec2-user@ip-172-31-40-222 WebServer]$ ls
build.gradle  README.md  src  www
[ec2-user@ip-172-31-40-222 WebServer]$ gradle FunWebServer
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :FunWebServer
Received: GET / HTTP/1.1
Received: Host: 3.20.233.20:9000
Received: Connection: keep-alive
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safa
ri/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

<=========----> 75% EXECUTING [10m 32s]
> :FunWebServer
```

3.3 Analyze what happens


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==9000

No.	Time	Source	Destination	Protocol	Length	Info
18320	615.089954	192.168.1.5	3.20.233.20	TCP	66	10801 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18321	615.090205	192.168.1.5	3.20.233.20	TCP	66	10802 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18322	615.164169	3.20.233.20	192.168.1.5	TCP	66	9000 → 10801 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=128
18323	615.164247	192.168.1.5	3.20.233.20	TCP	54	10801 → 9000 [ACK] Seq=1 Ack=1 Win=131328 Len=0
18324	615.164498	192.168.1.5	3.20.233.20	HTTP	489	GET / HTTP/1.1
18325	615.168235	3.20.233.20	192.168.1.5	TCP	66	9000 → 10802 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=128
18326	615.168305	192.168.1.5	3.20.233.20	TCP	54	10802 → 9000 [ACK] Seq=1 Ack=1 Win=131328 Len=0
18327	615.246182	3.20.233.20	192.168.1.5	TCP	54	9000 → 10801 [ACK] Seq=1 Ack=436 Win=28032 Len=0
18328	615.327945	3.20.233.20	192.168.1.5	TCP	603	9000 → 10801 [PSH, ACK] Seq=1 Ack=436 Win=28032 Len=549 [TCP segment of a reassembled PDU]
18329	615.327945	3.20.233.20	192.168.1.5	HTTP	54	HTTP/1.1 200 OK (text/html)
18330	615.328001	192.168.1.5	3.20.233.20	TCP	54	10801 → 9000 [ACK] Seq=436 Ack=551 Win=130816 Len=0
18331	615.328847	192.168.1.5	3.20.233.20	TCP	54	10801 → 9000 [FIN, ACK] Seq=436 Ack=551 Win=130816 Len=0
18344	615.407956	3.20.233.20	192.168.1.5	TCP	54	9000 → 10801 [ACK] Seq=551 Ack=437 Win=28032 Len=0
18664	660.173809	192.168.1.5	3.20.233.20	TCP	55	[TCP Keep-Alive] 10802 → 9000 [ACK] Seq=0 Ack=1 Win=131328 Len=1
18665	660.249109	3.20.233.20	192.168.1.5	TCP	66	[TCP Window Update] 9000 → 10802 [ACK] Seq=1 Ack=1 Win=27008 Len=0 SLE=0 SRE=1

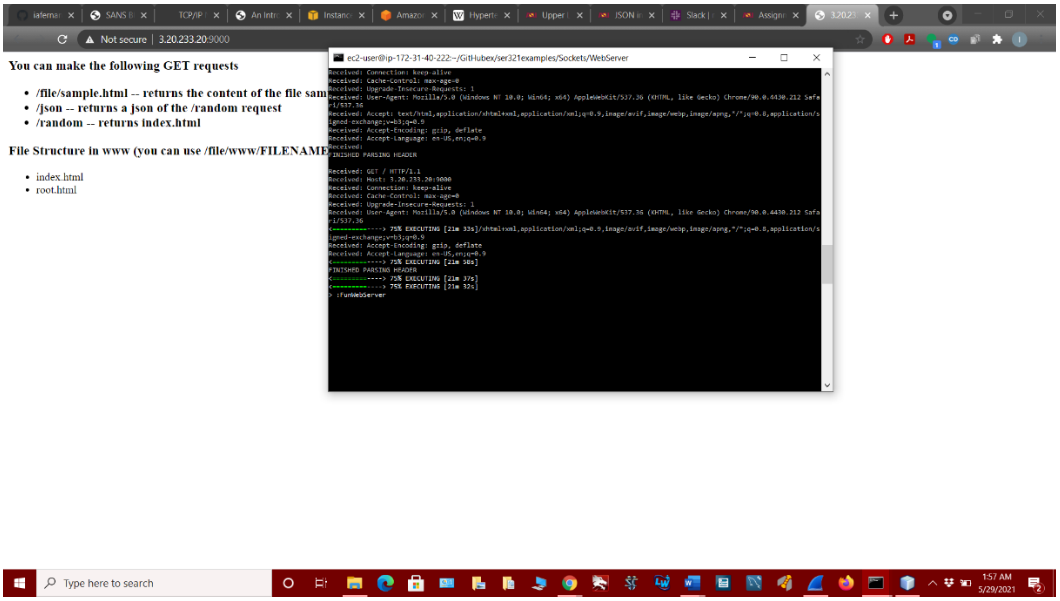
```
    Source Port: 9000
    Destination Port: 10801
    [Stream index: 160]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 676106229
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2222734349
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
    Window: 26883
    [Calculated window size: 26883]
    Checksum: 0x9203 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  > [SEQ/ACK analysis]
  > [Timestamps]
```

wireshark_Wi-FiVV4930.pcapng | Packets: 18669 · Displayed: 15 (0.1%) · Dropped: 0 (0.0%) | Profile: Default

1. What filter did you use? Explain why you chose that filter.

I chose the filter tcp.port==9000 because that is the port of the Web Server on the second system.

2. What happens when you are on /random and click the refresh button compared to the browser refresh (you can also use the command line output that the WebServer generates to answer this)?



You can make the following GET requests

- **/file/sample.html -- returns the content of the file sam**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

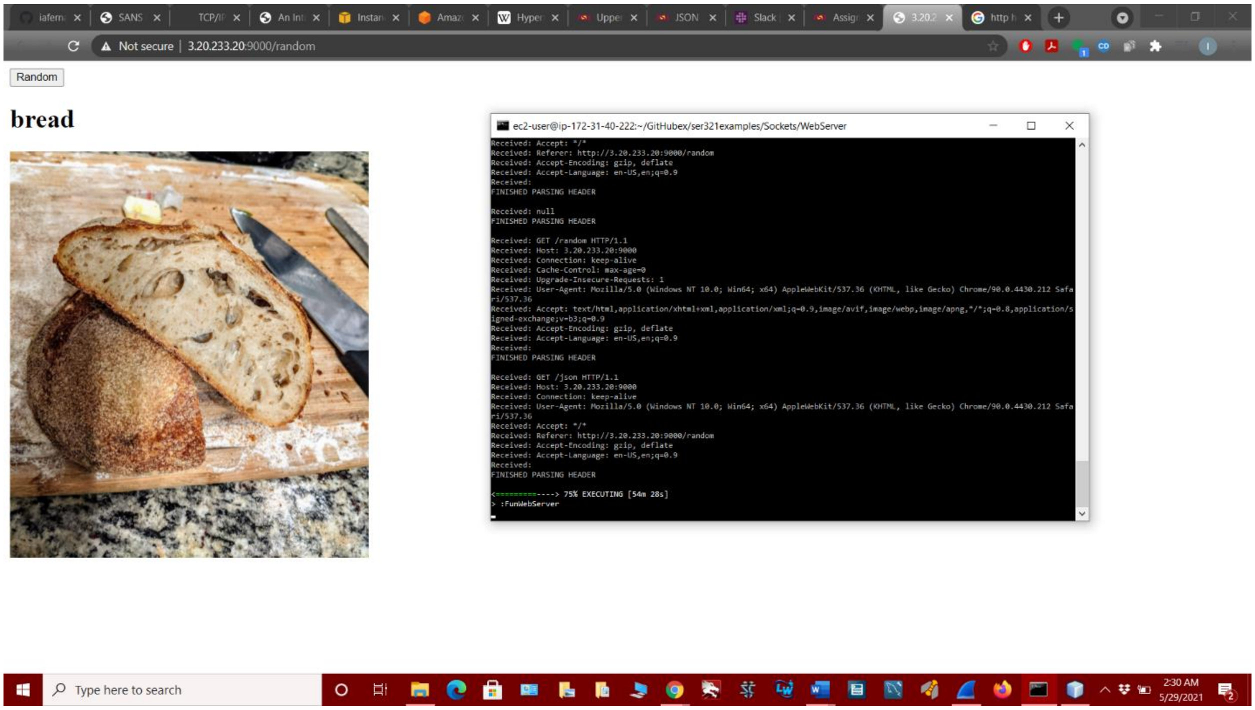
File Structure in www (you can use /file/www/FILENAME

- index.html
- root.html

ec2-user@ip-172-31-40-222:~/GitHubex/ser321examples/Sockets/WebServer

```
Received: Connection: keep-alive
Received: Cache-Control: max-age=0
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safa
ri/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: GET / HTTP/1.1
Received: Host: 3.20.233.20:9000
Received: Connection: keep-alive
Received: Cache-Control: max-age=0
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safa
ri/537.36
<==========---> 75% EXECUTING [21m 33s]/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
<==========---> 75% EXECUTING [21m 58s]
FINISHED PARSING HEADER
<==========---> 75% EXECUTING [21m 37s]
<==========---> 75% EXECUTING [21m 32s]
> :FunWebServer
```



bread

ec2-user@ip-172-31-40-222:~/GitHubex/ser321examples/Sockets/WebServer

```
Received: Accept: */*
Received: Referer: http://3.20.233.20:9000/random
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: null
FINISHED PARSING HEADER

Received: GET /random HTTP/1.1
Received: Host: 3.20.233.20:9000
Received: Connection: keep-alive
Received: Cache-Control: max-age=0
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safa
ri/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: GET /json HTTP/1.1
Received: Host: 3.20.233.20:9000
Received: Connection: keep-alive
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safa
ri/537.36
Received: Accept: */*
Received: Referer: http://3.20.233.20:9000/random
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

<=========----> 75% EXECUTING [54m 28s]
> :FunWebServer
```

The WebServer Received: GET /json HTTP/1.1

449	59.808087	192.168.0.19	3.20.233.20	HTTP	372	GET /json HTTP/1.1

3. What kinds of response codes are you able to get through different requests to your server?

355	46.752836	192.168.0.19	3.20.233.20	TCP	66	11284 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

357	46.830405	3.20.233.20	192.168.0.19	TCP	66	9000 → 11284 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=128

359	46.830580	192.168.0.19	3.20.233.20	TCP	54	11284 → 9000 [ACK] Seq=1 Ack=1 Win=65536 Len=0

360	46.830737	192.168.0.19	3.20.233.20	HTTP	515	GET / HTTP/1.1

441	59.680668	192.168.0.19	3.20.233.20	HTTP	495	GET /random HTTP/1.1

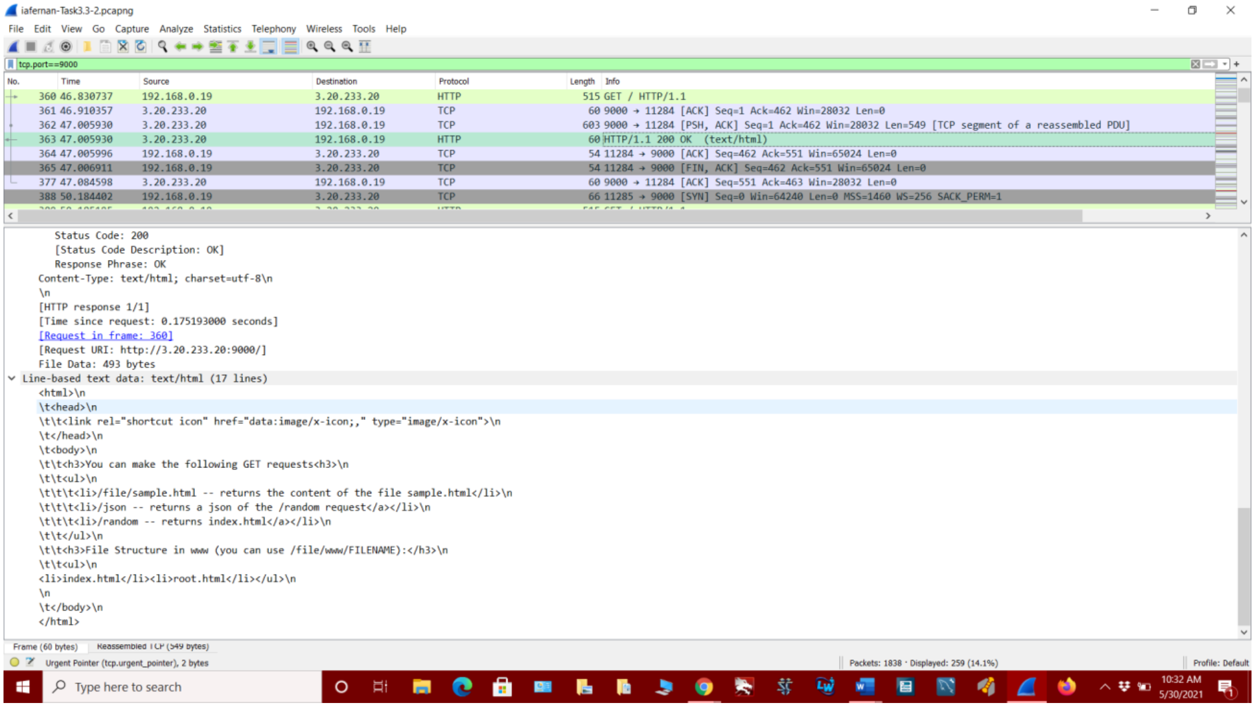
449	59.808087	192.168.0.19	3.20.233.20	HTTP	372	GET /json HTTP/1.1

4. Explain the response codes you get and why you get them?

Within wireshark, we are able to see multiple local ports connect to the web server during the course of the web browsing session. The TCP three-way handshake is viewable for every port that connects to the web server. We can see various GET requests from the local ports to the server tinted in lime green color, as well as 200 OK HTTP success statuses from the web browsing session.

5. When you do aipOfSecondMachine:9000 take a look what Wireshark generates as a server response. Are you able to find the data that the server sends back to you?

We are able to view the line-based text data:


iafernan-Task3.3-2.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==9000
Frame (60 bytes)
Reassembled TCP (549 bytes)
Urgent Pointer (tcp.urgent_pointer), 2 bytes
Packets: 1838 · Displayed: 259 (14.1%)
Profile: Default
Type here to search
10:32 AM
5/30/2021

```
No.  Time       Source        Destination   Protocol  Length Info
 360 46.830737  192.168.0.19  3.20.233.20   HTTP      515 GET / HTTP/1.1
 361 46.910357  3.20.233.20   192.168.0.19  TCP        60 9000 → 11284 [ACK] Seq=1 Ack=462 Win=28032 Len=0
 362 47.005930  3.20.233.20   192.168.0.19  TCP       603 9000 → 11284 [PSH, ACK] Seq=1 Ack=462 Win=28032 Len=549 [TCP segment of a reassembled PDU]
 363 47.005930  3.20.233.20   192.168.0.19  HTTP       60 HTTP/1.1 200 OK  (text/html)
 364 47.005996  192.168.0.19  3.20.233.20   TCP        54 11284 → 9000 [ACK] Seq=462 Ack=551 Win=65024 Len=0
 365 47.006911  192.168.0.19  3.20.233.20   TCP        54 11284 → 9000 [FIN, ACK] Seq=462 Ack=551 Win=65024 Len=0
 377 47.084598  3.20.233.20   192.168.0.19  TCP        60 9000 → 11284 [ACK] Seq=551 Ack=463 Win=28032 Len=0
 388 50.184402  192.168.0.19  3.20.233.20   TCP        66 11285 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
```

```
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Content-Type: text/html; charset=utf-8\n
    \n
    [HTTP response 1/1]
    [Time since request: 0.175193000 seconds]
    [Request in frame: 360]
    [Request URI: http://3.20.233.20:9000/]
    File Data: 493 bytes
v Line-based text data: text/html (17 lines)
    <html>\n
    \t<head>\n
    \t\t<link rel="shortcut icon" href="data:image/x-icon;," type="image/x-icon">\n
    \t</head>\n
    \t<body>\n
    \t\t<h3>You can make the following GET requests<h3>\n
    \t\t<ul>\n
    \t\t\t<li>/file/sample.html -- returns the content of the file sample.html</li>\n
    \t\t\t<li>/json -- returns a json of the /random request</a></li>\n
    \t\t\t<li>/random -- returns index.html</a></li>\n
    \t\t</ul>\n
    \t\t<h3>File Structure in www (you can use /file/www/FILENAME):</h3>\n
    \t\t<ul>\n
    <li>index.html</li><li>root.html</li></ul>\n
    \n
    \t</body>\n
    </html>
```

6. Based on the above question explain why HTTPs is now more common than HTTP.

Data with using HTTPS is encrypted, it uses a secure protocol known as Transport Layer Security.

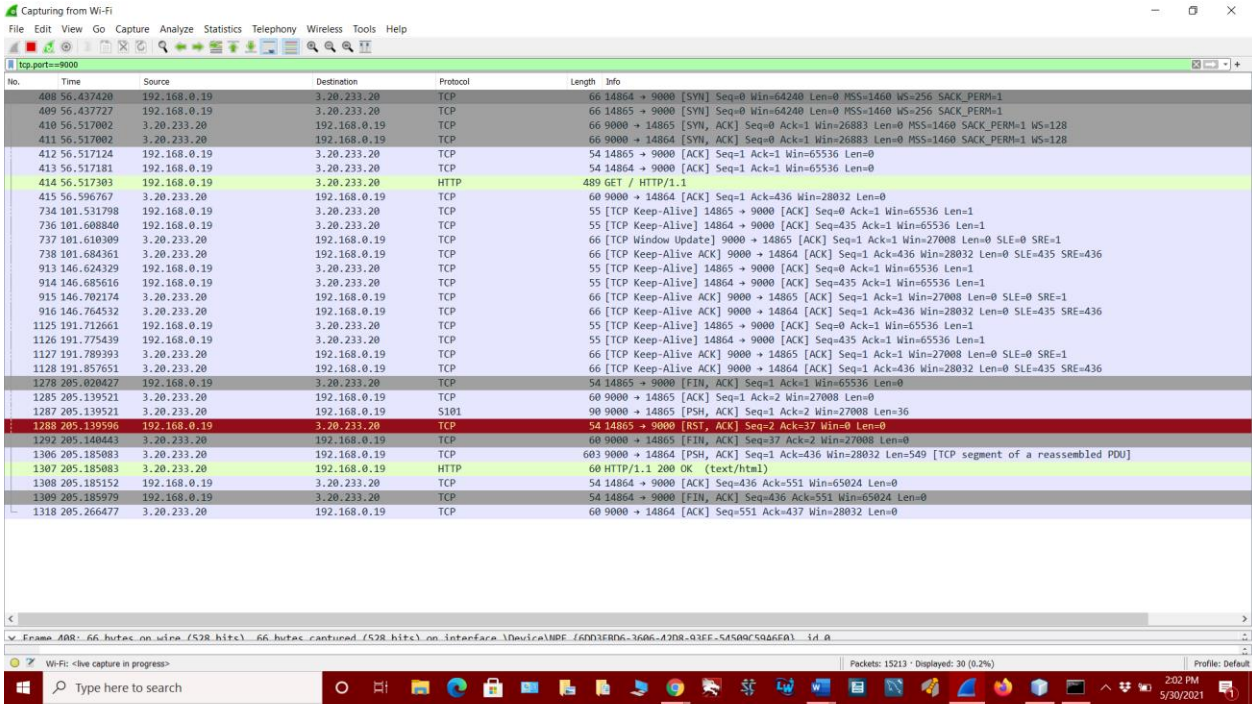
7. What port does the server listen to for HTTP requests in our case and is that the most common port for HTTP?

During the web browsing it is interesting to see that multiple local ports connect to the destination web server port 9000. The destination ip is of course the same. This signifies the various processes that are running concurrently as requests are being made to the web server during the web browser session. For example, keeping the home page “alive” while navigating towards other pages/tabs of a website and the various interactions being made within them.

8. What local port is used when sending different requests to the WebServer? How does it differ to the traffic to your SMTP server from part 1?

SMTP made use of one local port for communications whereas http communicated with various local ports when making requests to the webserver.

3.4 Setting up a "real" Web server

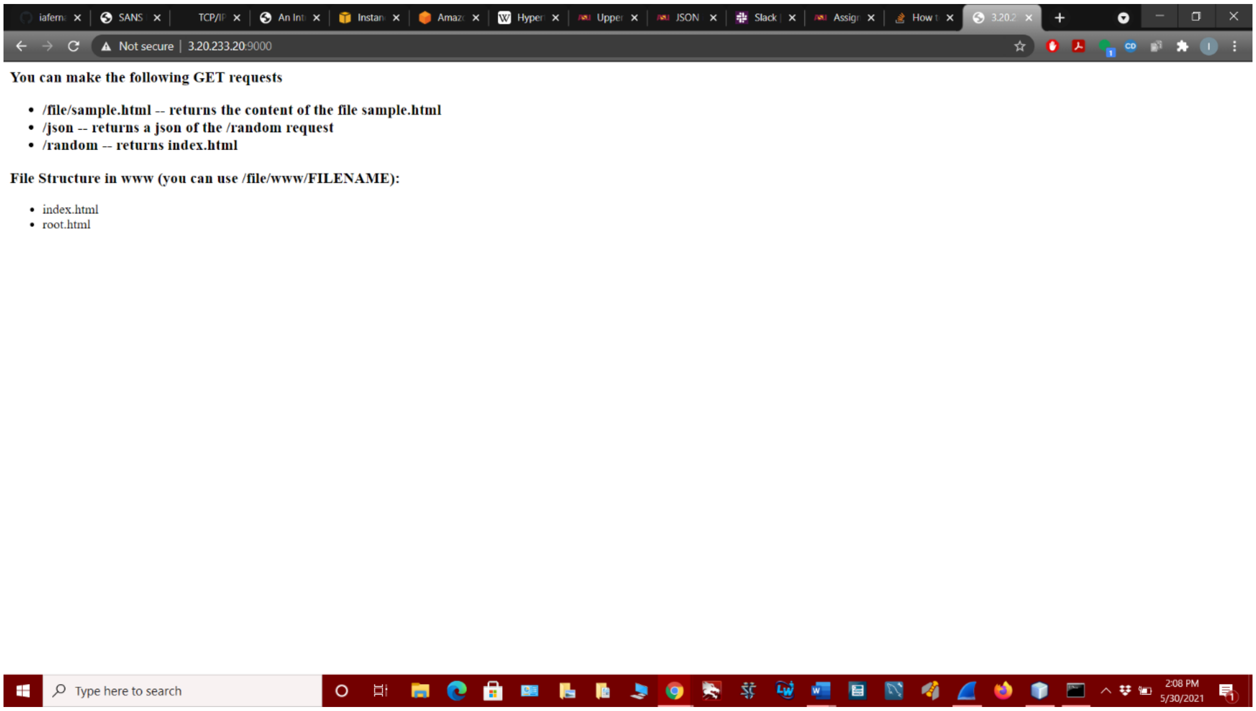

Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==9000

```
No.   Time        Source         Destination    Protocol  Length Info
 408  56.437420   192.168.0.19   3.20.233.20    TCP       66 14864 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 409  56.437727   192.168.0.19   3.20.233.20    TCP       66 14865 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 410  56.517002   3.20.233.20    192.168.0.19   TCP       66 9000 → 14865 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=128
 411  56.517002   3.20.233.20    192.168.0.19   TCP       66 9000 → 14864 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=128
 412  56.517124   192.168.0.19   3.20.233.20    TCP       54 14865 → 9000 [ACK] Seq=1 Ack=1 Win=65536 Len=0
 413  56.517181   192.168.0.19   3.20.233.20    TCP       54 14864 → 9000 [ACK] Seq=1 Ack=1 Win=65536 Len=0
 414  56.517303   192.168.0.19   3.20.233.20    HTTP     489 GET / HTTP/1.1
 415  56.596767   3.20.233.20    192.168.0.19   TCP       60 9000 → 14864 [ACK] Seq=1 Ack=436 Win=28032 Len=0
 734  101.531798  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14865 → 9000 [ACK] Seq=0 Ack=1 Win=65536 Len=1
 736  101.608840  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14864 → 9000 [ACK] Seq=435 Ack=1 Win=65536 Len=1
 737  101.610309  3.20.233.20    192.168.0.19   TCP       66 [TCP Window Update] 9000 → 14865 [ACK] Seq=1 Ack=1 Win=27008 Len=0 SLE=0 SRE=1
 738  101.684361  3.20.233.20    192.168.0.19   TCP       66 [TCP Keep-Alive ACK] 9000 → 14864 [ACK] Seq=1 Ack=436 Win=28032 Len=0 SLE=435 SRE=436
 913  146.624329  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14865 → 9000 [ACK] Seq=0 Ack=1 Win=65536 Len=1
 914  146.685616  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14864 → 9000 [ACK] Seq=435 Ack=1 Win=65536 Len=1
 915  146.702174  3.20.233.20    192.168.0.19   TCP       66 [TCP Keep-Alive ACK] 9000 → 14865 [ACK] Seq=1 Ack=1 Win=27008 Len=0 SLE=0 SRE=1
 916  146.764532  3.20.233.20    192.168.0.19   TCP       66 [TCP Keep-Alive ACK] 9000 → 14864 [ACK] Seq=1 Ack=436 Win=28032 Len=0 SLE=435 SRE=436
1125  191.712661  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14865 → 9000 [ACK] Seq=0 Ack=1 Win=65536 Len=1
1126  191.775439  192.168.0.19   3.20.233.20    TCP       55 [TCP Keep-Alive] 14864 → 9000 [ACK] Seq=435 Ack=1 Win=65536 Len=1
1127  191.789393  3.20.233.20    192.168.0.19   TCP       66 [TCP Keep-Alive ACK] 9000 → 14865 [ACK] Seq=1 Ack=1 Win=27008 Len=0 SLE=0 SRE=1
1128  191.857651  3.20.233.20    192.168.0.19   TCP       66 [TCP Keep-Alive ACK] 9000 → 14864 [ACK] Seq=1 Ack=436 Win=28032 Len=0 SLE=435 SRE=436
1278  205.020427  192.168.0.19   3.20.233.20    TCP       54 14865 → 9000 [FIN, ACK] Seq=1 Ack=1 Win=65536 Len=0
1285  205.139521  3.20.233.20    192.168.0.19   TCP       60 9000 → 14865 [ACK] Seq=1 Ack=2 Win=27008 Len=0
1287  205.139521  3.20.233.20    192.168.0.19   S101      90 9000 → 14865 [PSH, ACK] Seq=1 Ack=2 Win=27008 Len=36
1288  205.139596  192.168.0.19   3.20.233.20    TCP       54 14865 → 9000 [RST, ACK] Seq=2 Ack=37 Win=0 Len=0
1292  205.140443  3.20.233.20    192.168.0.19   TCP       60 9000 → 14865 [FIN, ACK] Seq=37 Ack=2 Win=27008 Len=0
1306  205.185083  3.20.233.20    192.168.0.19   TCP      603 9000 → 14864 [PSH, ACK] Seq=1 Ack=436 Win=28032 Len=549 [TCP segment of a reassembled PDU]
1307  205.185083  3.20.233.20    192.168.0.19   HTTP      60 HTTP/1.1 200 OK  (text/html)
1308  205.185152  192.168.0.19   3.20.233.20    TCP       54 14864 → 9000 [ACK] Seq=436 Ack=551 Win=65024 Len=0
1309  205.185979  192.168.0.19   3.20.233.20    TCP       54 14864 → 9000 [FIN, ACK] Seq=436 Ack=551 Win=65024 Len=0
1318  205.266477  3.20.233.20    192.168.0.19   TCP       60 9000 → 14864 [ACK] Seq=551 Ack=437 Win=28032 Len=0

Frame 408: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{6DD3FBD6-3606-42D8-93FF-54509C5946F0}, id 0

Wi-Fi: <live capture in progress>          Packets: 15213 · Displayed: 30 (0.2%)          Profile: Default
```



You can make the following GET requests

- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

File Structure in www (you can use /file/www/FILENAME):

- index.html
- root.html

1. Check your traffic to your Webserver now. What port is the traffic going to now? Is it the same as previously used or is it and should it be different?

It is the same as before.

2. Is it still HTTP or is it now HTTPs? Why?

It is still HTTP.