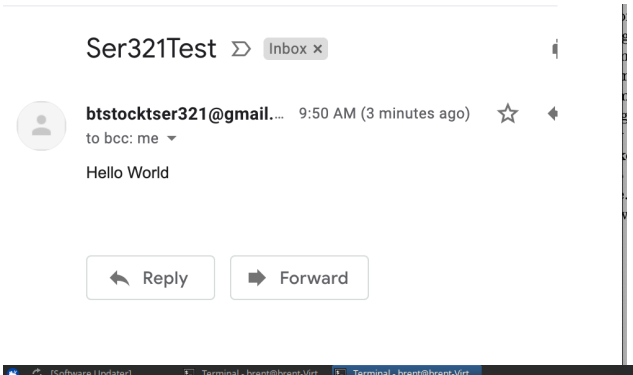
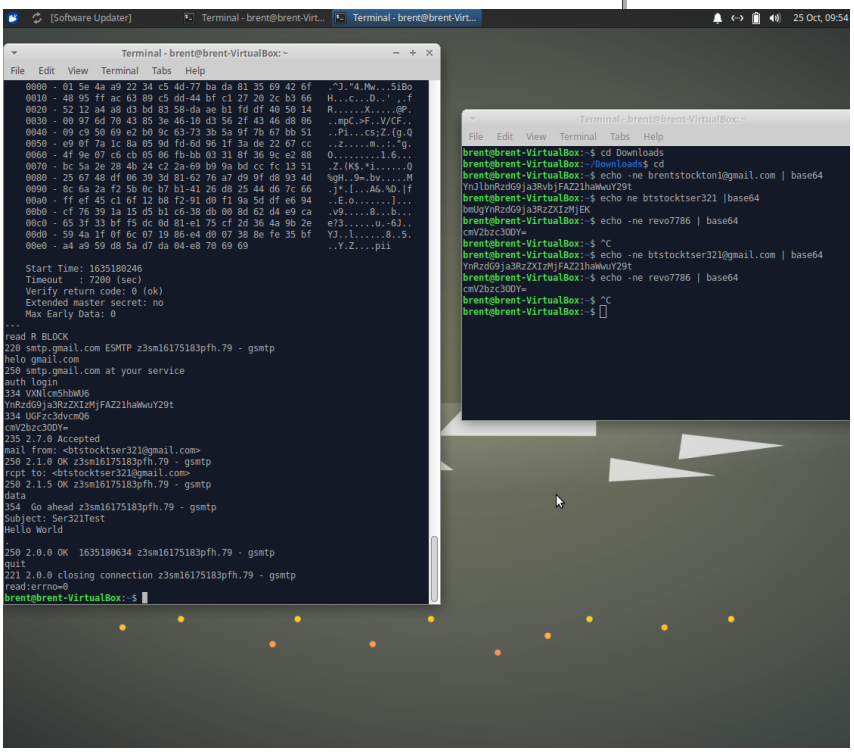
Brent Stockton
SER321
10/27/21

Assignment 2 SER321

1 Sending emails and SMTP (15 points)





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

I used a TCP, Arp and TLS filter to capture the traffic. I noticed all traffic started coming in through those protocols and was able to narrow down what to look at once I narrowed it down with those filter. I also could have added ip.addr==10.0.2.15 && ip.addr == 74.125.197.108 to further narrow down the traffic but It didn't seem necessary with the limited traffic I was receiving in my linux visual machine.

2. What is the standard port for SMTP and why do we use port 465 in the example above?
The standard port is normally Port 25. We use port 465 in the example above likely because we are using a legacy system that the application demands we use this port. Many ISPs and cloud hosting providers still support 465 for SMTP submission.

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

The first two lines are basically creating a text encryption of your username and password through encoding binary strings into text representations using base64 coding format. This is needed to encode our certificate contents. Next, we use an exception library to connect to a remote host using SSL/TLS which is the gmail SMTP port 465. We then identify ourself using helo and authenticate on the smtp using auth login and providing our base64 username and password. Once authenticated we specify where we are mailing from and who we are mailing. We use data command to initiate our email text and provide the subject and email before closing and quitting to complete our process.

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

I did notice a fair amount of back and forth communication while establishing a connection.

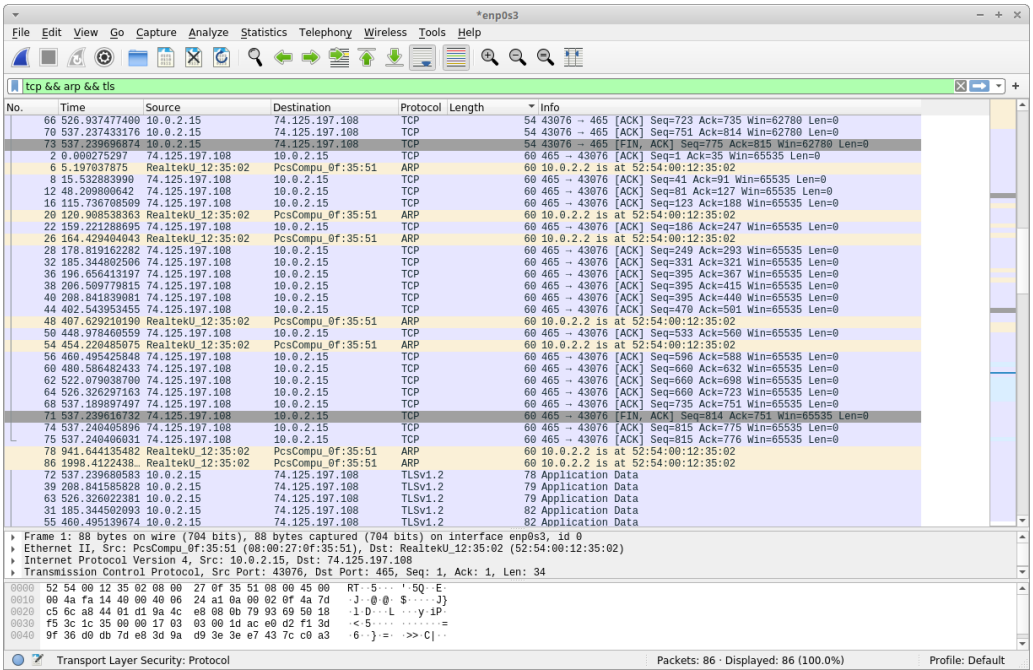
5. What is the port your local machine is using between sending the two emails when communicating with the SMTP server?

My local machine used 43076 and gmail used 465.

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

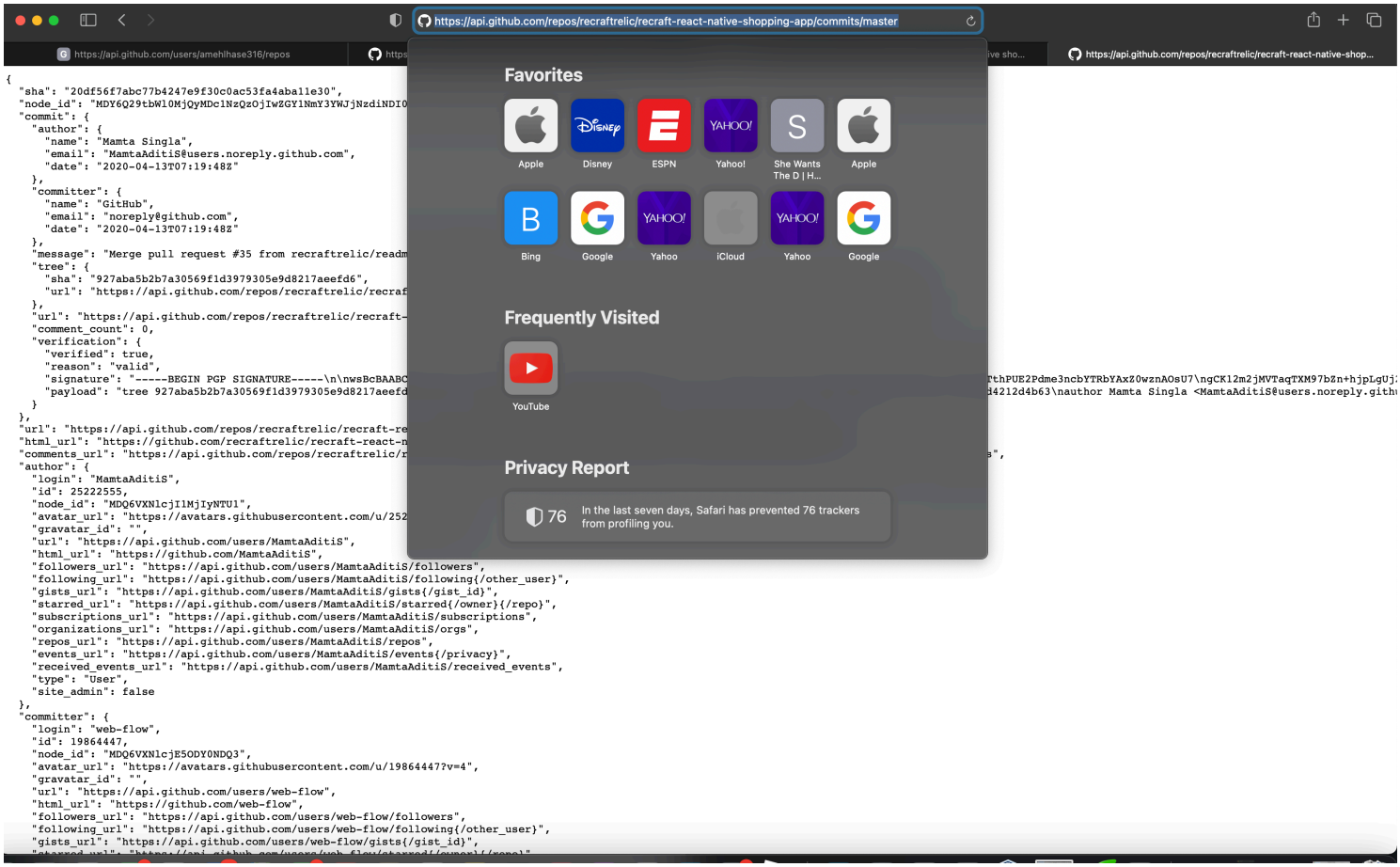
Port 465 sent the first fin flag and then sent two following acknowledgments to terminate the process.

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



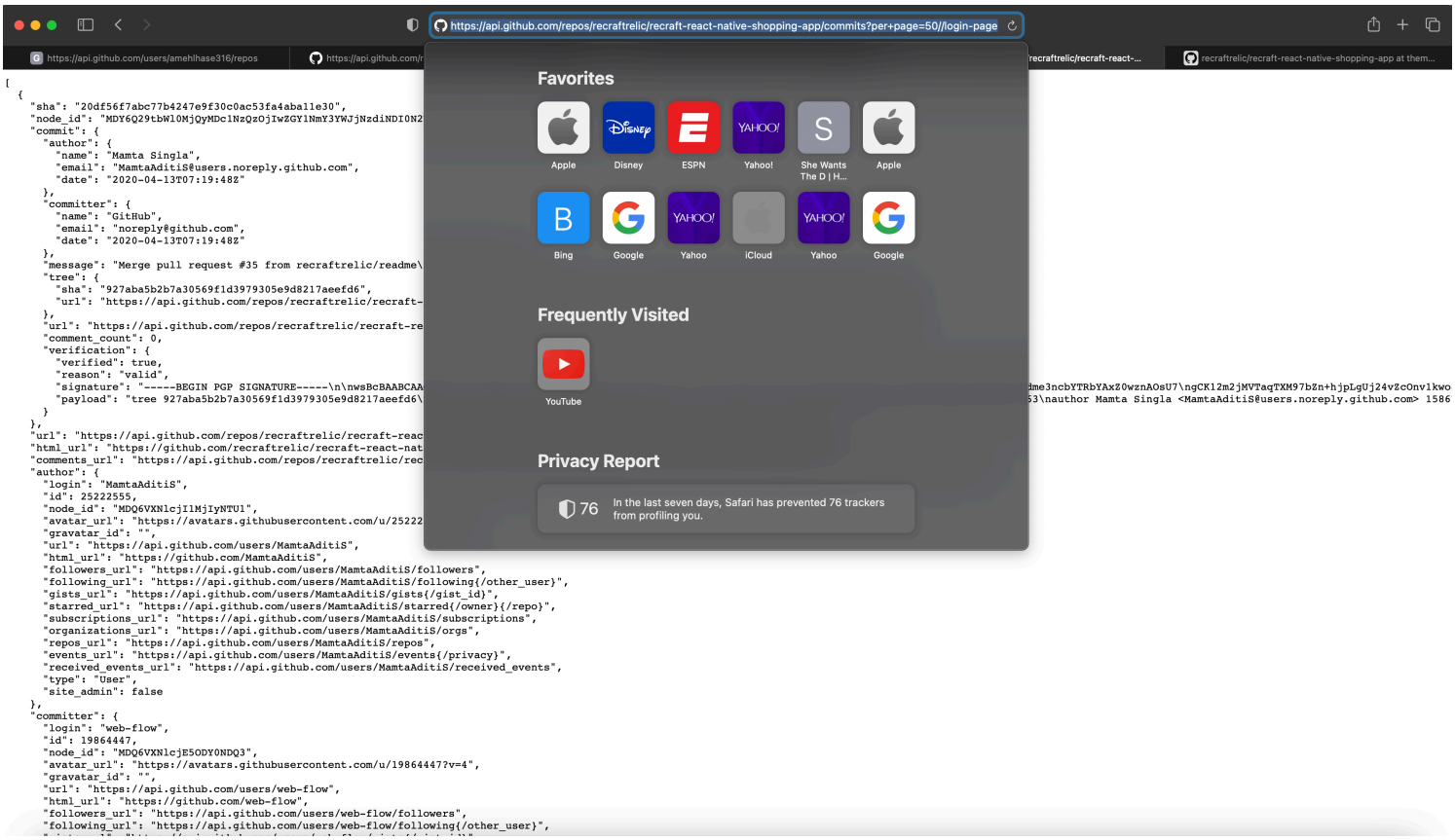
2 Understanding HTTP (20 points)

1. Commits for master default branch




```
{
  "sha": "20df56f7abc77b4247e9f30c0ac53fa4aba11e30",
  "node_id": "MDY6Q29tbWl0MjQyMDc1NzQzOjIwZGY1NmY3YWJjNzdiNDI0
  "commit": {
    "author": {
      "name": "Mamta Singla",
      "email": "MamtaAditiS@users.noreply.github.com",
      "date": "2020-04-13T07:19:48Z"
    },
    "committer": {
      "name": "GitHub",
      "email": "noreply@github.com",
      "date": "2020-04-13T07:19:48Z"
    },
    "message": "Merge pull request #35 from recraftrelic/readm
    "tree": {
      "sha": "927aba5b2b7a30569f1d3979305e9d8217aeefd6",
      "url": "https://api.github.com/repos/recraftrelic/recraf
    },
    "url": "https://api.github.com/repos/recraftrelic/recraft-
    "comment_count": 0,
    "verification": {
      "verified": true,
      "reason": "valid",
      "signature": "-----BEGIN PGP SIGNATURE-----\n\nwsBcBAABC
      "payload": "tree 927aba5b2b7a30569f1d3979305e9d8217aeefd
    }
  },
  "url": "https://api.github.com/repos/recraftrelic/recraft-re
  "html_url": "https://github.com/recraftrelic/recraft-react-n
  "comments_url": "https://api.github.com/repos/recraftrelic/r
  "author": {
    "login": "MamtaAditiS",
    "id": 25222555,
    "node_id": "MDQ6VXNlcjI1MjIyNTU1",
    "avatar_url": "https://avatars.githubusercontent.com/u/252
    "gravatar_id": "",
    "url": "https://api.github.com/users/MamtaAditiS",
    "html_url": "https://github.com/MamtaAditiS",
    "followers_url": "https://api.github.com/users/MamtaAditiS/followers",
    "following_url": "https://api.github.com/users/MamtaAditiS/following{/other_user}",
    "gists_url": "https://api.github.com/users/MamtaAditiS/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/MamtaAditiS/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/MamtaAditiS/subscriptions",
    "organizations_url": "https://api.github.com/users/MamtaAditiS/orgs",
    "repos_url": "https://api.github.com/users/MamtaAditiS/repos",
    "events_url": "https://api.github.com/users/MamtaAditiS/events{/privacy}",
    "received_events_url": "https://api.github.com/users/MamtaAditiS/received_events",
    "type": "User",
    "site_admin": false
  },
  "committer": {
    "login": "web-flow",
    "id": 19864447,
    "node_id": "MDQ6VXNlcjE5ODY0NDQ3",
    "avatar_url": "https://avatars.githubusercontent.com/u/19864447?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/web-flow",
    "html_url": "https://github.com/web-flow",
    "followers_url": "https://api.github.com/users/web-flow/followers",
    "following_url": "https://api.github.com/users/web-flow/following{/other_user}",
    "gists_url": "https://api.github.com/users/web-flow/gists{/gist_id}",
```

Commits for log-in branch per_page limit to 50



1. Explain the specify API calls you used.

Firstly, I made a call to view the commits on the default branch. Using the Github API I called the https://api.github.com/users/recraftrelic/repos to access the recraftrelic repository JSON data. I then chose to video the react-native-shopping-app. Then I looked over the data to locate the call to find the commits to the master default branch using https://api.github.com/repos/recraftrelic/recraft-react-native-shopping-app/commits/master.

Next I wanted to specify commits to another branch (log-in) using a per_page limit of 50. For this I adjusted my call to https://api.github.com/repos/recraftrelic/recraft-react-native-shopping-app/commits?per+page=50//login-page where the "?" is the end of URL resource path and start of query parameters and per page is set to 50 to show 50 commits.

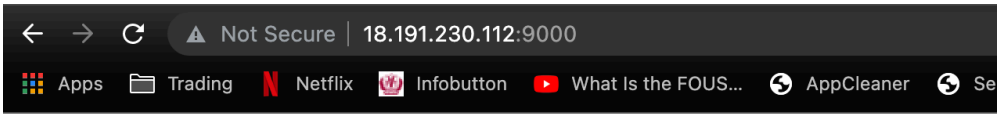
2. Explain the difference between stateless and a state-full communication.

A stateless communication is a communication where the receiver must not retain the previous session state from previous requests. A stateful communication is where all user activities in a session are performed in a single context

3.2 Running a simple Java WebServer (10 points)

```
The FunWebServer does a little more than the SimpleWebServer. Check out what it does :-)[ec2-user@ip-172-31-30-213 We
bServer]$ gradle FunWebServer
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :FunWebServer
Received: GET / HTTP/1.1
Received: Host: 18.191.230.112:9000
Received: Connection: keep-alive
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/9
4.0.4606.81 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
plication/signed-exchange;v=b3;q=0.9
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER
          > :FunWebServer                <=========----> 75% EXECUTING [3m 13s]
<=========-<=========----> 75% EXECUTING [3m 15s]
```



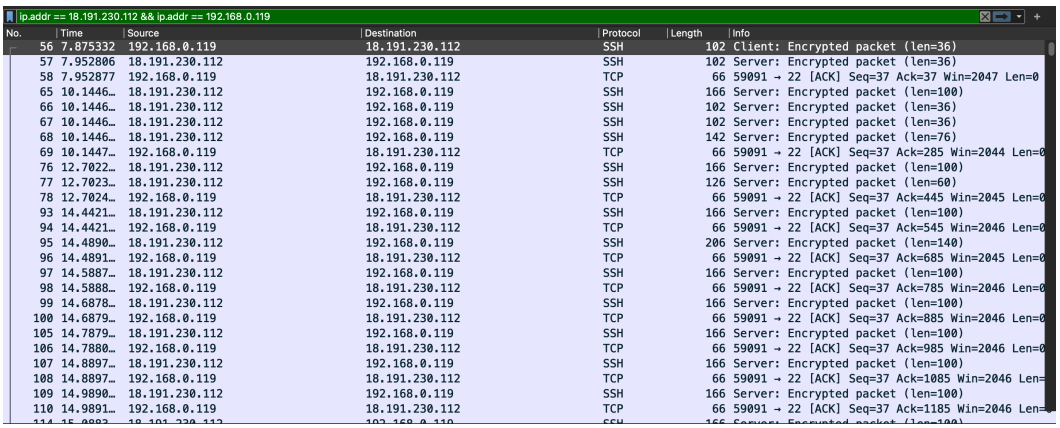
You can make the following GET requests

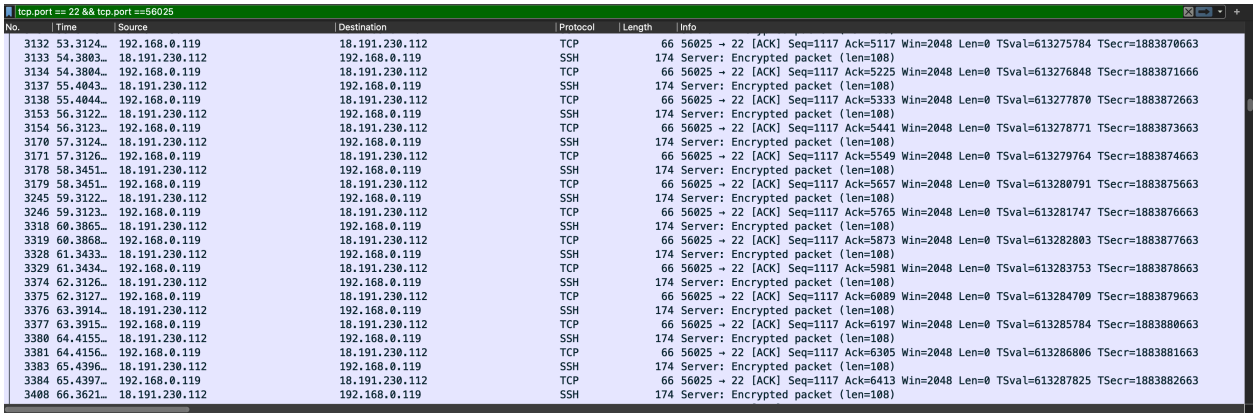
- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

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

- index.html
- root.html

3. 3 Analyze what happens (10 points)





tcp.port == 22 && tcp.port ==56025

No.	Time	Source	Destination	Protocol	Length	Info
3132	53.3124…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5117 Win=2048 Len=0 TSval=613275784 TSecr=1883870663
3133	54.3803…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3134	54.3804…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5225 Win=2048 Len=0 TSval=613276848 TSecr=1883871666
3137	55.4043…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3138	55.4044…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5333 Win=2048 Len=0 TSval=613277870 TSecr=1883872663
3153	56.3122…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3154	56.3123…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5441 Win=2048 Len=0 TSval=613278771 TSecr=1883873663
3170	57.3124…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3171	57.3126…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5549 Win=2048 Len=0 TSval=613279764 TSecr=1883874663
3178	58.3451…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3179	58.3451…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5657 Win=2048 Len=0 TSval=613280791 TSecr=1883875663
3245	59.3122…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3246	59.3123…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5765 Win=2048 Len=0 TSval=613281747 TSecr=1883876663
3318	60.3865…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3319	60.3868…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5873 Win=2048 Len=0 TSval=613282803 TSecr=1883877663
3328	61.3433…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3329	61.3434…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=5981 Win=2048 Len=0 TSval=613283753 TSecr=1883878663
3374	62.3126…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3375	62.3127…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=6089 Win=2048 Len=0 TSval=613284709 TSecr=1883879663
3376	63.3914…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3377	63.3915…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=6197 Win=2048 Len=0 TSval=613285784 TSecr=1883880663
3380	64.4155…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3381	64.4156…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=6305 Win=2048 Len=0 TSval=613286806 TSecr=1883881663
3383	65.4396…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
3384	65.4397…	192.168.0.119	18.191.230.112	TCP	66	56025 → 22 [ACK] Seq=1117 Ack=6413 Win=2048 Len=0 TSval=613287825 TSecr=1883882663
3408	66.3621…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)

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

I chose filtering commands: ip.addr == 18.191.230.112 && ip.addr == 192.168.0.119 as well as tcp.port == 22 && tcp.port ==56025 . The first filtered for the I.P address of my local network with the public IPv4address of my EC2 instance. The second filter I used examined TCP ports 22 and 56025 to capture traffic. This allowed me to show my Network traffic to and from my WebServer.

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

When you are on random and click “Random” the server receives a GET /json HTTP/1.1 compared to the refresh that receives a GET /random HTTP/1.1 then a GET /json HTTP/1.1.
Seemingly you go two levels when refreshing on the random page.

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

200 OK, 400 Bad Request, 404 Not Found

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

200 OK - request succeeded. This is because server successfully processed you request.
400 Bad Request - Client Error response that you get if the server could not understand your request because of invalid syntax.
404 not Found. The server cannot find the requested resource.

5. When you do a ipOfSecondMachine: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?

No

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

Https is more secure. Https has SSL to encrypt information and TLS transport layer security which HTTP doesn’t.

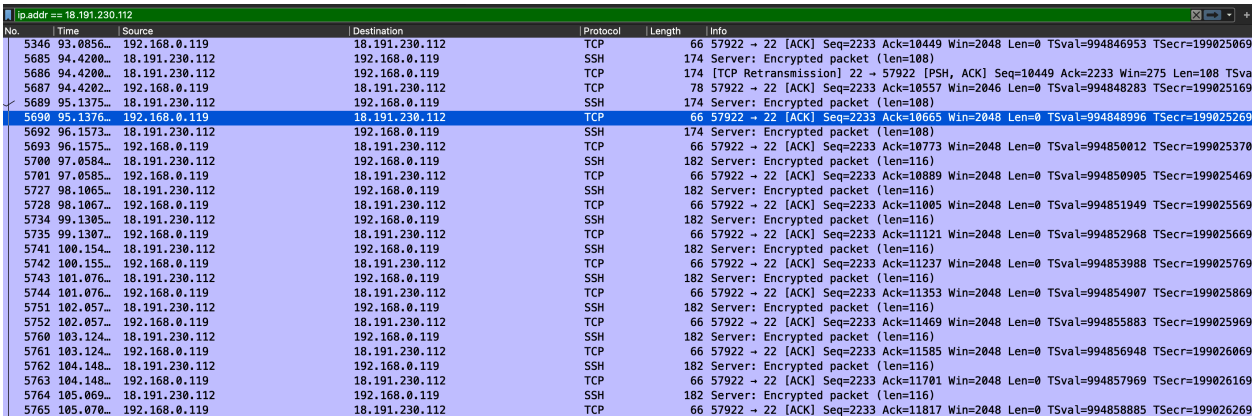
7. What port does the server listen to for HTTP requests in our case and is that the most common port for HTTP?

Server listens on port 22. No that is not the most common port for HTTP. Port 80 is the most common

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?

Localhost:9000 is used when sending different requests to the WebServer. The traffic sent to my SMTP server from part 1 was sent using port 465. Port 465 is used for implicit TLS while TCP port 9000 is the mail communication port of the Linux/PnScan.

3.4 Setting up a "real" Web server (10 points)



ip.addr == 18.191.230.112

No.	Time	Source	Destination	Protocol	Length	Info
5346	93.0856…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=10449 Win=2048 Len=0 TSval=994846953 TSecr=199025069
5685	94.4200…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
5686	94.4200…	18.191.230.112	192.168.0.119	TCP	174	[TCP Retransmission] 22 → 57922 [PSH, ACK] Seq=10449 Ack=2233 Win=275 Len=108 TSva
5687	94.4202…	192.168.0.119	18.191.230.112	TCP	78	57922 → 22 [ACK] Seq=2233 Ack=10557 Win=2046 Len=0 TSval=994848283 TSecr=199025169
5689	95.1375…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
5690	95.1376…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=10665 Win=2048 Len=0 TSval=994848996 TSecr=199025269
5692	96.1573…	18.191.230.112	192.168.0.119	SSH	174	Server: Encrypted packet (len=108)
5693	96.1575…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=10773 Win=2048 Len=0 TSval=994850012 TSecr=199025370
5700	97.0584…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5701	97.0585…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=10889 Win=2048 Len=0 TSval=994850905 TSecr=199025469
5727	98.1065…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5728	98.1067…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11005 Win=2048 Len=0 TSval=994851949 TSecr=199025569
5734	99.1305…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5735	99.1307…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11121 Win=2048 Len=0 TSval=994852968 TSecr=199025669
5741	100.154…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5742	100.155…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11237 Win=2048 Len=0 TSval=994853988 TSecr=199025769
5743	101.076…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5744	101.076…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11353 Win=2048 Len=0 TSval=994854907 TSecr=199025869
5751	102.057…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5752	102.057…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11469 Win=2048 Len=0 TSval=994855883 TSecr=199025969
5760	103.124…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5761	103.124…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11585 Win=2048 Len=0 TSval=994856948 TSecr=199026069
5762	104.148…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5763	104.148…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11701 Win=2048 Len=0 TSval=994857969 TSecr=199026169
5764	105.069…	18.191.230.112	192.168.0.119	SSH	182	Server: Encrypted packet (len=116)
5765	105.070…	192.168.0.119	18.191.230.112	TCP	66	57922 → 22 [ACK] Seq=2233 Ack=11817 Win=2048 Len=0 TSval=994858885 TSecr=199026269

```
    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;


    server {
        server_name  18.191.230.112;
        root         /usr/share/nginx/html;
        location / {
        proxy_pass http://localhost:9000/;
        }

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html;
        location = /40x.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?**

Im not sure if my configuration was correct after following directions but it seems that my web server traffic is going through port 22. I believe after configuring it should be different from previous but I am getting the same after following directions.

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

I believe after configuring it should be HTTPs.

3.6.1 Multiply

I used a try catch block with exception e which handled the situation where the url was wrong. This covered if the user used a wrong character for the number, only provided one number, or mistyped the URL. I used error code 400 because in these cases the server cannot process the request due to client error or malformed request syntax.