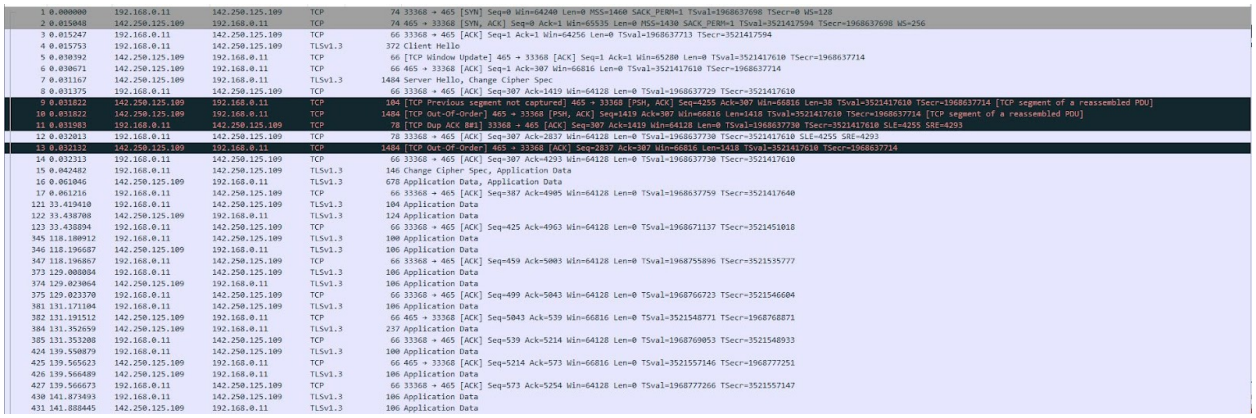
Assignment 2
Maura Peterson

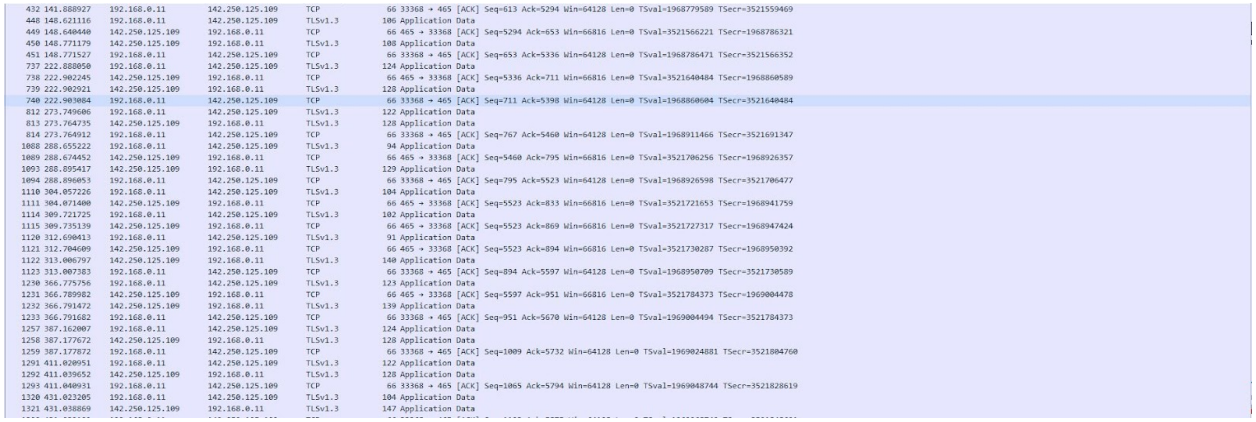
Part 1

1. tcp.port==465. I chose this filter because “smtp.gmail.com:465” specified port 465 because that is where the port is located.
2. Port 25 is the standard port for SMTP. This port is blocked by a lot of IPs because it is used for spam a lot. 465 is used to connect with SSL.
3.
    a. echo -ne username | base64
        i. This command is used to encode your username which will be needed because we are using SSL which is encrypted.
    b. echo -ne password | base64
        i. This command is used to encode your password which will be needed because we are using SSL which is encrypted.
    c. openssl s_client -crlf -ign_eof -connect smtp.gmail.com:465
        i. This line connects to smtp.gmail.com at port 465
        ii. The -crlf converts LF from the terminal into CRLF
        iii. The -ing_eof Ignores input of eof
    d. helo gmail.com
        i. Identifies the domain name of the sending host as gmail.com to smtp.
    e. auth login
        i. This allows you to login to your account.
        ii. You then copy your encoded username
        iii. Then your encoded password
    f. mail from: <username@gmail.com>
        i. Initiate a message transfer, from username@gmail.com
    g. rcpt to: <username@gmail.com>
        i. Identifies the addressee as username@gmail.com
    h. data
        i. Send the data line by line
    i. Subject:
        i. Enter the subject of the email
    j. Email text goes here
        i. Enter the body of the email
    k. .
        i. Finish and send the email
    l. quit
        i. end the connection.
4. 15 packets, starting with the initial connection Then the client hello, server hello/change cipher spec, application data from the server, then change cipher spec/ application data from the client, with acknowledgments.
5. Port: 33368, this port stayed the same the whole time.
6. I initiate the fin, then the server sends a fin ack and its own fin, then I send a fin ack and then we both terminate.

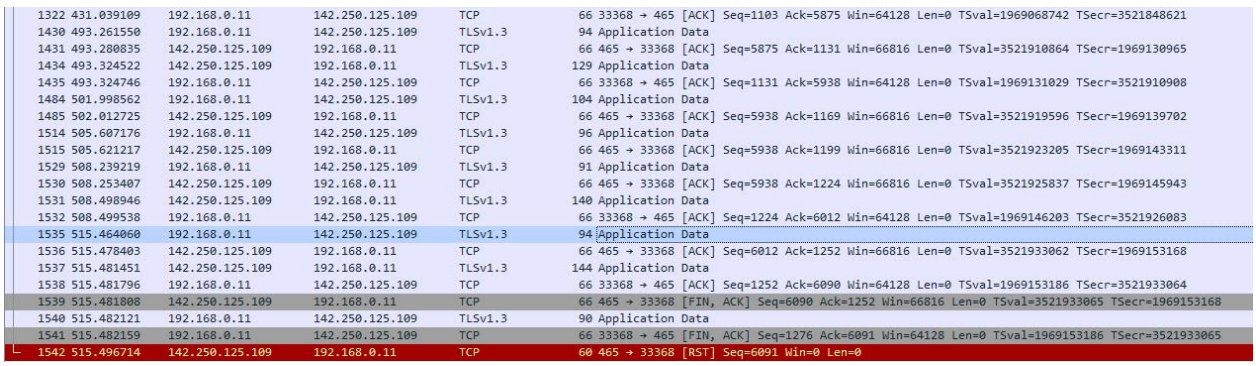
7.



1	0.000000	192.168.0.11	142.250.125.109	TCP	74 33368 → 465 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1968637698 TSecr=0 WS=128
2	0.015048	142.250.125.109	192.168.0.11	TCP	74 465 → 33368 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=3521417594 TSecr=1968637698 WS=256
3	0.015247	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1968637713 TSecr=3521417594
4	0.015753	192.168.0.11	142.250.125.109	TLSv1.3	372 Client Hello
5	0.030392	142.250.125.109	192.168.0.11	TCP	66 [TCP Window Update] 465 → 33368 [ACK] Seq=1 Ack=1 Win=65280 Len=0 TSval=3521417610 TSecr=1968637714
6	0.030671	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=1 Ack=307 Win=66816 Len=0 TSval=3521417610 TSecr=1968637714
7	0.031167	142.250.125.109	192.168.0.11	TLSv1.3	1484 Server Hello, Change Cipher Spec
8	0.031375	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=307 Ack=1419 Win=64128 Len=0 TSval=1968637729 TSecr=3521417610
9	0.031822	142.250.125.109	192.168.0.11	TCP	104 [TCP Previous segment not captured] 465 → 33368 [PSH, ACK] Seq=4255 Ack=307 Win=66816 Len=38 TSval=3521417610 TSecr=1968637714 [TCP segment of a reassembled PDU]
10	0.031822	142.250.125.109	192.168.0.11	TCP	1484 [TCP Out-Of-Order] 465 → 33368 [PSH, ACK] Seq=1419 Ack=307 Win=66816 Len=1418 TSval=3521417610 TSecr=1968637714 [TCP segment of a reassembled PDU]
11	0.031983	192.168.0.11	142.250.125.109	TCP	78 [TCP Dup ACK 8#1] 33368 → 465 [ACK] Seq=307 Ack=1419 Win=64128 Len=0 TSval=1968637730 TSecr=3521417610 SLE=4255 SRE=4293
12	0.032013	192.168.0.11	142.250.125.109	TCP	78 33368 → 465 [ACK] Seq=307 Ack=2837 Win=64128 Len=0 TSval=1968637730 TSecr=3521417610 SLE=4255 SRE=4293
13	0.032132	142.250.125.109	192.168.0.11	TCP	1484 [TCP Out-Of-Order] 465 → 33368 [ACK] Seq=2837 Ack=307 Win=66816 Len=1418 TSval=3521417610 TSecr=1968637714
14	0.032313	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=307 Ack=4293 Win=64128 Len=0 TSval=1968637730 TSecr=3521417610
15	0.042482	192.168.0.11	142.250.125.109	TLSv1.3	146 Change Cipher Spec, Application Data
16	0.061046	142.250.125.109	192.168.0.11	TLSv1.3	678 Application Data, Application Data
17	0.061216	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=387 Ack=4905 Win=64128 Len=0 TSval=1968637759 TSecr=3521417640
121	33.419410	192.168.0.11	142.250.125.109	TLSv1.3	104 Application Data
122	33.438708	142.250.125.109	192.168.0.11	TLSv1.3	124 Application Data
123	33.438894	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=425 Ack=4963 Win=64128 Len=0 TSval=1968671137 TSecr=3521451018
345	118.180912	192.168.0.11	142.250.125.109	TLSv1.3	100 Application Data
346	118.196687	142.250.125.109	192.168.0.11	TLSv1.3	106 Application Data
347	118.196867	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=459 Ack=5003 Win=64128 Len=0 TSval=1968755896 TSecr=3521535777
373	129.008084	192.168.0.11	142.250.125.109	TLSv1.3	106 Application Data
374	129.023064	142.250.125.109	192.168.0.11	TLSv1.3	106 Application Data
375	129.023370	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=499 Ack=5043 Win=64128 Len=0 TSval=1968766723 TSecr=3521546604
381	131.171104	192.168.0.11	142.250.125.109	TLSv1.3	106 Application Data
382	131.191512	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5043 Ack=539 Win=66816 Len=0 TSval=3521548771 TSecr=1968768871
384	131.352659	142.250.125.109	192.168.0.11	TLSv1.3	237 Application Data
385	131.353208	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=539 Ack=5214 Win=64128 Len=0 TSval=1968769053 TSecr=3521548933
424	139.550879	192.168.0.11	142.250.125.109	TLSv1.3	100 Application Data
425	139.565623	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5214 Ack=573 Win=66816 Len=0 TSval=3521557146 TSecr=1968777251
426	139.566489	142.250.125.109	192.168.0.11	TLSv1.3	106 Application Data
427	139.566673	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=573 Ack=5254 Win=64128 Len=0 TSval=1968777266 TSecr=3521557147
430	141.873493	192.168.0.11	142.250.125.109	TLSv1.3	106 Application Data
431	141.888445	142.250.125.109	192.168.0.11	TLSv1.3	106 Application Data



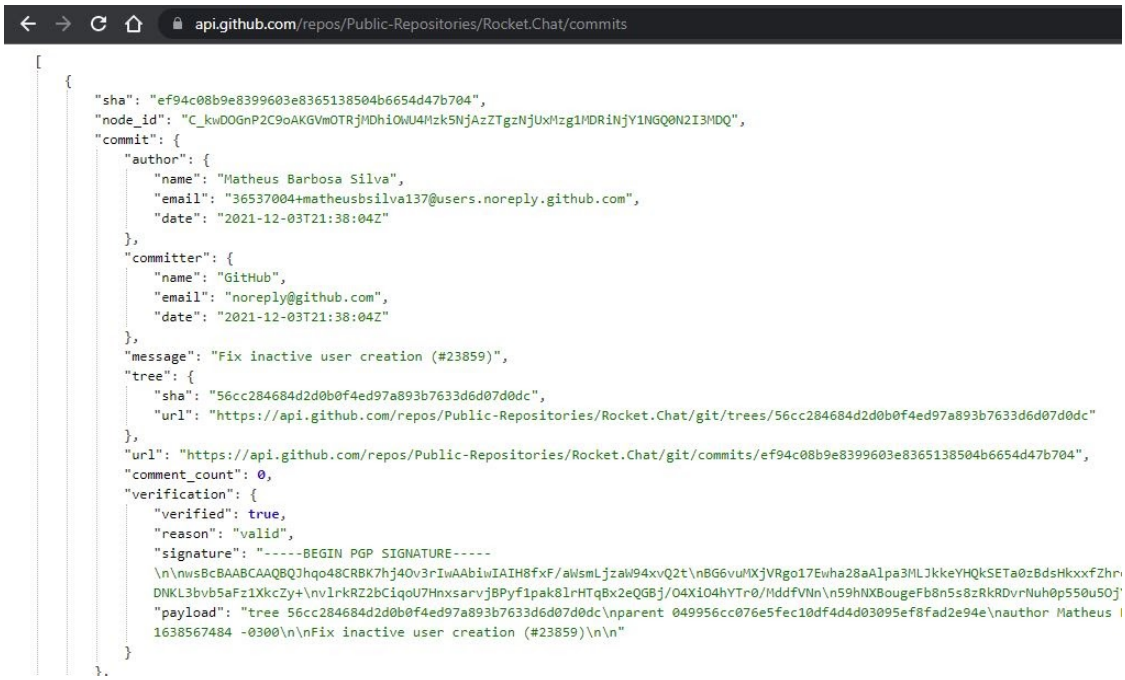
432	141.888927	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=613 Ack=5294 Win=64128 Len=0 TSval=1968779589 TSecr=3521559469
448	148.621116	192.168.0.11	142.250.125.109	TLSv1.3	106 Application Data
449	148.640440	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5294 Ack=653 Win=66816 Len=0 TSval=3521566221 TSecr=1968786321
450	148.771179	142.250.125.109	192.168.0.11	TLSv1.3	108 Application Data
451	148.771527	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=653 Ack=5336 Win=64128 Len=0 TSval=1968786471 TSecr=3521566352
737	222.888050	192.168.0.11	142.250.125.109	TLSv1.3	124 Application Data
738	222.902245	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5336 Ack=711 Win=66816 Len=0 TSval=3521640484 TSecr=1968860589
739	222.902921	142.250.125.109	192.168.0.11	TLSv1.3	128 Application Data
740	222.903084	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=711 Ack=5398 Win=64128 Len=0 TSval=1968860604 TSecr=3521640484
812	273.749606	192.168.0.11	142.250.125.109	TLSv1.3	122 Application Data
813	273.764735	142.250.125.109	192.168.0.11	TLSv1.3	128 Application Data
814	273.764912	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=767 Ack=5460 Win=64128 Len=0 TSval=1968911466 TSecr=3521691347
1088	288.655222	192.168.0.11	142.250.125.109	TLSv1.3	94 Application Data
1089	288.674452	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5460 Ack=795 Win=66816 Len=0 TSval=3521706256 TSecr=1968926357
1093	288.895417	142.250.125.109	192.168.0.11	TLSv1.3	129 Application Data
1094	288.896055	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=795 Ack=5523 Win=64128 Len=0 TSval=1968926598 TSecr=3521706477
1110	304.057226	192.168.0.11	142.250.125.109	TLSv1.3	104 Application Data
1111	304.071400	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5523 Ack=833 Win=66816 Len=0 TSval=3521721653 TSecr=1968941759
1114	309.721725	192.168.0.11	142.250.125.109	TLSv1.3	102 Application Data
1115	309.735139	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5523 Ack=869 Win=66816 Len=0 TSval=3521727317 TSecr=1968947424
1120	312.690413	192.168.0.11	142.250.125.109	TLSv1.3	91 Application Data
1121	312.704609	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5523 Ack=894 Win=66816 Len=0 TSval=3521730287 TSecr=1968950392
1122	313.006797	142.250.125.109	192.168.0.11	TLSv1.3	140 Application Data
1123	313.007383	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=894 Ack=5597 Win=64128 Len=0 TSval=1968950709 TSecr=3521730589
1230	366.775756	192.168.0.11	142.250.125.109	TLSv1.3	123 Application Data
1231	366.789982	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5597 Ack=951 Win=66816 Len=0 TSval=3521784373 TSecr=1969004478
1232	366.791472	142.250.125.109	192.168.0.11	TLSv1.3	139 Application Data
1233	366.791682	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=951 Ack=5670 Win=64128 Len=0 TSval=1969004494 TSecr=3521784373
1257	387.162007	192.168.0.11	142.250.125.109	TLSv1.3	124 Application Data
1258	387.177672	142.250.125.109	192.168.0.11	TLSv1.3	128 Application Data
1259	387.177872	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1009 Ack=5732 Win=64128 Len=0 TSval=1969024881 TSecr=3521804760
1291	411.020951	192.168.0.11	142.250.125.109	TLSv1.3	122 Application Data
1292	411.039652	142.250.125.109	192.168.0.11	TLSv1.3	128 Application Data
1293	411.040931	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1065 Ack=5794 Win=64128 Len=0 TSval=1969048744 TSecr=3521828619
1320	431.023205	192.168.0.11	142.250.125.109	TLSv1.3	104 Application Data
1321	431.038869	142.250.125.109	192.168.0.11	TLSv1.3	147 Application Data



1322	431.039109	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1103 Ack=5875 Win=64128 Len=0 TSval=1969068742 TSecr=3521848621
1430	493.261550	192.168.0.11	142.250.125.109	TLSv1.3	94 Application Data
1431	493.280835	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5875 Ack=1131 Win=66816 Len=0 TSval=3521910864 TSecr=1969130965
1434	493.324522	142.250.125.109	192.168.0.11	TLSv1.3	129 Application Data
1435	493.324746	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1131 Ack=5938 Win=64128 Len=0 TSval=1969131029 TSecr=3521910908
1484	501.998562	192.168.0.11	142.250.125.109	TLSv1.3	104 Application Data
1485	502.012725	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5938 Ack=1169 Win=66816 Len=0 TSval=3521919596 TSecr=1969139702
1514	505.607176	192.168.0.11	142.250.125.109	TLSv1.3	96 Application Data
1515	505.621217	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5938 Ack=1199 Win=66816 Len=0 TSval=3521923205 TSecr=1969143311
1529	508.239219	192.168.0.11	142.250.125.109	TLSv1.3	91 Application Data
1530	508.253407	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=5938 Ack=1224 Win=66816 Len=0 TSval=3521925837 TSecr=1969145943
1531	508.498946	142.250.125.109	192.168.0.11	TLSv1.3	140 Application Data
1532	508.499538	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1224 Ack=6012 Win=64128 Len=0 TSval=1969146203 TSecr=3521926083
1535	515.464060	192.168.0.11	142.250.125.109	TLSv1.3	94 Application Data
1536	515.478403	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [ACK] Seq=6012 Ack=1252 Win=66816 Len=0 TSval=3521933062 TSecr=1969153168
1537	515.481451	142.250.125.109	192.168.0.11	TLSv1.3	144 Application Data
1538	515.481796	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [ACK] Seq=1252 Ack=6090 Win=64128 Len=0 TSval=1969153186 TSecr=3521933064
1539	515.481808	142.250.125.109	192.168.0.11	TCP	66 465 → 33368 [FIN, ACK] Seq=6090 Ack=1252 Win=66816 Len=0 TSval=3521933065 TSecr=1969153168
1540	515.482121	192.168.0.11	142.250.125.109	TLSv1.3	90 Application Data
1541	515.482159	192.168.0.11	142.250.125.109	TCP	66 33368 → 465 [FIN, ACK] Seq=1276 Ack=6091 Win=64128 Len=0 TSval=1969153186 TSecr=3521933065
1542	515.496714	142.250.125.109	192.168.0.11	TCP	60 465 → 33368 [RST] Seq=6091 Win=0 Len=0

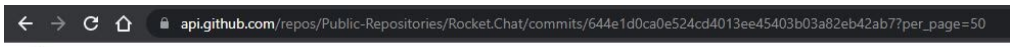
Part 2

1.




```
[
  {
    "sha": "ef94c08b9e8399603e8365138504b6654d47b704",
    "node_id": "C_kwDOGnP2C9oAKGVmOTRjMDhiOWU4Mzk5NjAzZTgzNjUxMzg1MDRiNjY1NGQ0N2I3MDQ",
    "commit": {
      "author": {
        "name": "Matheus Barbosa Silva",
        "email": "36537004+matheusbsilva137@users.noreply.github.com",
        "date": "2021-12-03T21:38:04Z"
      },
      "committer": {
        "name": "GitHub",
        "email": "noreply@github.com",
        "date": "2021-12-03T21:38:04Z"
      },
      "message": "Fix inactive user creation (#23859)",
      "tree": {
        "sha": "56cc284684d2d0b0f4ed97a893b7633d6d07d0dc",
        "url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/git/trees/56cc284684d2d0b0f4ed97a893b7633d6d07d0dc"
      },
      "url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/git/commits/ef94c08b9e8399603e8365138504b6654d47b704",
      "comment_count": 0,
      "verification": {
        "verified": true,
        "reason": "valid",
        "signature": "-----BEGIN PGP SIGNATURE-----
        \n\nwsBcBAABCAAQBQJhqo48CRBK7hj4Ov3rIwAAbiwIAIH8fxF/aWsmLjzaW94xvQ2t\nBG6vuMXjVRgo17Ewha28aAlpa3MLJkkeYHQkSETa0zBdsHkxxfZhr
        DNKL3bvb5aFz1XkcZy+\nvlrkRZ2bCiqoU7HnxsarvjBPyf1pak8lrHTqBx2eQGBj/O4XiO4hYTr0/MddfVNn\n59hNXBougeFb8n5s8zRkRDvrNuh0p550u5Oj
        "payload": "tree 56cc284684d2d0b0f4ed97a893b7633d6d07d0dc\nparent 049956cc076e5fec10df4d4d03095ef8fad2e94e\nauthor Matheus
        1638567484 -0300\n\nFix inactive user creation (#23859)\n\n"
      }
    },
```


To get to the commits on the default branch all I had to do was add /commits to the repository API.

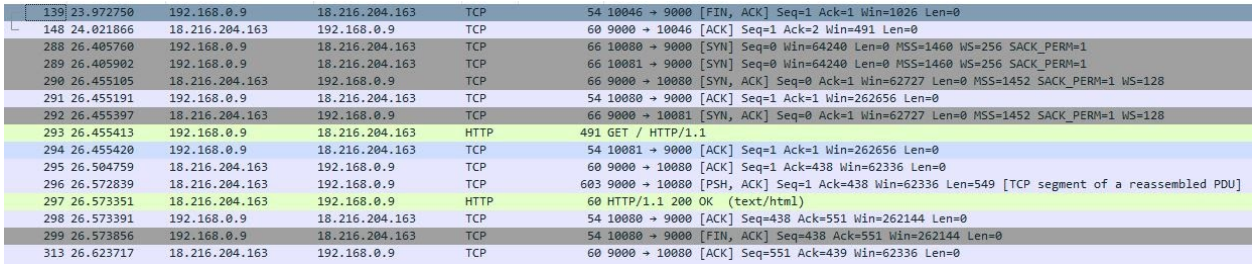



```
{
  "sha": "644e1d0ca0e524cd4013ee45403b03a82eb42ab7",
  "node_id": "MDY6Q29tbWl0NDQzODA3MjQzOjY0NGUxZDBjYTBlNTI0Y2Q0MDEzZWU0NTQwM2IwM2E4MmViNDJhYjc=",
  "commit": {
    "author": {
      "name": "Pierre H. Lehnen",
      "email": "Hudell@users.noreply.github.com",
      "date": "2019-05-07T19:19:40Z"
    },
    "committer": {
      "name": "GitHub",
      "email": "noreply@github.com",
      "date": "2019-05-07T19:19:40Z"
    },
    "message": "Merge branch 'develop' into 2fa-on-password-reset",
    "tree": {
      "sha": "62972e75de3a34271b5b5c0cbe52fe62718c7f8c",
      "url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/git/trees/62972e75de3a34271b5b5c0cbe52fe62718c7f8c"
    },
    "url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/git/commits/644e1d0ca0e524cd4013ee45403b03a82eb42ab7",
    "comment_count": 0,
    "verification": {
      "verified": true,
      "reason": "valid",
      "signature": "-----BEGIN PGP SIGNATURE-----
      \n\nwsBcBAABCAAQBQJc0dpMCRBK7hj4Ov3rIwAAdHIIABh8KgZ+FFIKF2Q5xP3gpzel\nI6tR4Q/rsppuS1iAcWJwpclGwESB3vkNRhPgk90DdobybyJoEHYI
      c47Zkg0e7jb7OiD\nvNI+3ZpGWw/tcUHmACfrpyNYfihbbVjfzOsEw55J5FGZZAhS5qEkGnWQ9sgO+zcI\nTExG2JioEl5+wK65iGLDyIqYxF6uFCD2RjcerhOi
      "payload": "tree 62972e75de3a34271b5b5c0cbe52fe62718c7f8c\nparent a6796f0a365522f982ec8e1de667faff43d6ea79\nparent aad74af
      <noreply@github.com> 1557256780 -0300\n\nMerge branch 'develop' into 2fa-on-password-reset"
    }
  },
  "url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/commits/644e1d0ca0e524cd4013ee45403b03a82eb42ab7",
  "html_url": "https://github.com/Public-Repositories/Rocket.Chat/commit/644e1d0ca0e524cd4013ee45403b03a82eb42ab7",
  "comments_url": "https://api.github.com/repos/Public-Repositories/Rocket.Chat/commits/644e1d0ca0e524cd4013ee45403b03a82eb42ab7/com
  "author": {
```


To get to a specific branch I went back to the repository API then added /branches then used the URL for the second branch in the list. This took me to the commits for that branch. Then I added ?per_page=50 to set the commits per page to 50.

2. In stateful communication, a connection is established and maintained until the end. Information from the connection is saved and used later in the communication, an example of this is FTP. Stateless communication does not do this, the client sends a request and receives a response then it is done. Each new response is completely separate from the response before it and the server does not need to store information on the previous request. HTTP is an example of stateless communication.

Part 3
3.2 and 3.3



139	23.972750	192.168.0.9	18.216.204.163	TCP	54 10046 → 9000 [FIN, ACK] Seq=1 Ack=1 Win=1026 Len=0
148	24.021866	18.216.204.163	192.168.0.9	TCP	60 9000 → 10046 [ACK] Seq=1 Ack=2 Win=491 Len=0
288	26.405760	192.168.0.9	18.216.204.163	TCP	66 10080 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
289	26.405902	192.168.0.9	18.216.204.163	TCP	66 10081 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
290	26.455105	18.216.204.163	192.168.0.9	TCP	66 9000 → 10080 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1452 SACK_PERM=1 WS=128
291	26.455191	192.168.0.9	18.216.204.163	TCP	54 10080 → 9000 [ACK] Seq=1 Ack=1 Win=262656 Len=0
292	26.455397	18.216.204.163	192.168.0.9	TCP	66 9000 → 10081 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1452 SACK_PERM=1 WS=128
293	26.455413	192.168.0.9	18.216.204.163	HTTP	491 GET / HTTP/1.1
294	26.455420	192.168.0.9	18.216.204.163	TCP	54 10081 → 9000 [ACK] Seq=1 Ack=1 Win=262656 Len=0
295	26.504759	18.216.204.163	192.168.0.9	TCP	60 9000 → 10080 [ACK] Seq=1 Ack=438 Win=62336 Len=0
296	26.572839	18.216.204.163	192.168.0.9	TCP	603 9000 → 10080 [PSH, ACK] Seq=1 Ack=438 Win=62336 Len=549 [TCP segment of a reassembled PDU]
297	26.573351	18.216.204.163	192.168.0.9	HTTP	60 HTTP/1.1 200 OK (text/html)
298	26.573391	192.168.0.9	18.216.204.163	TCP	54 10080 → 9000 [ACK] Seq=438 Ack=551 Win=262144 Len=0
299	26.573856	192.168.0.9	18.216.204.163	TCP	54 10080 → 9000 [FIN, ACK] Seq=438 Ack=551 Win=262144 Len=0
313	26.623717	18.216.204.163	192.168.0.9	TCP	60 9000 → 10080 [ACK] Seq=551 Ack=439 Win=62336 Len=0

1. I used tcp.port==9000, because that is the port the server is running on so it should only show the server.

2. The command line in the server says GET HTTP/1.1 for the main page and for /random it says GET /random HTTP/1.1. In Wireshark it shows:
   homepage:
   me: GET HTTP/1.1
   Server: HTTP/1.1 200 OK (Text/html)
   Random:
   Me: GET /random HTTP/1.1
   Server: HTTP/1.1 200 OK (Text/html)
   Me: GET /json HTTP/1.1
   Server: HTTP/1.1 200 OK, JavaScript Object Notation (application/json)

3. Doing /json gets:
   Server: HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
   Me: GET /favicon.ico HTTP/1.1
   Server: HTTP/1.1 400 Bad Request

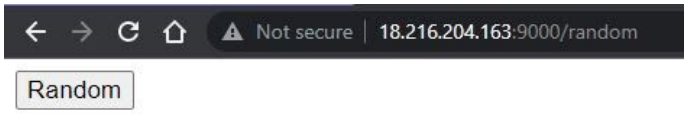
   Doing /file/www/index.html puts me on the random street page and Wireshark gets:
   Server: HTTP/1.1 200 OK (Text/html)
   Me: GET /json HTTP/1.1
   Server: HTTP/1.1 200 OK, JavaScript Object Notation (application/json)

   Doing /file/www/root.html puts me on the main page with ${links} instead of files
   Server: HTTP/1.1 200 OK (Text/html)
4. 200 indicates a successful response because I correctly asked for the right thing

400 indicates a bad request which means invalid syntax and this is in response to the GET /favicon.ico HTTP/1.1
404 means not found and URL is not recognized, I got it because there was a typo.

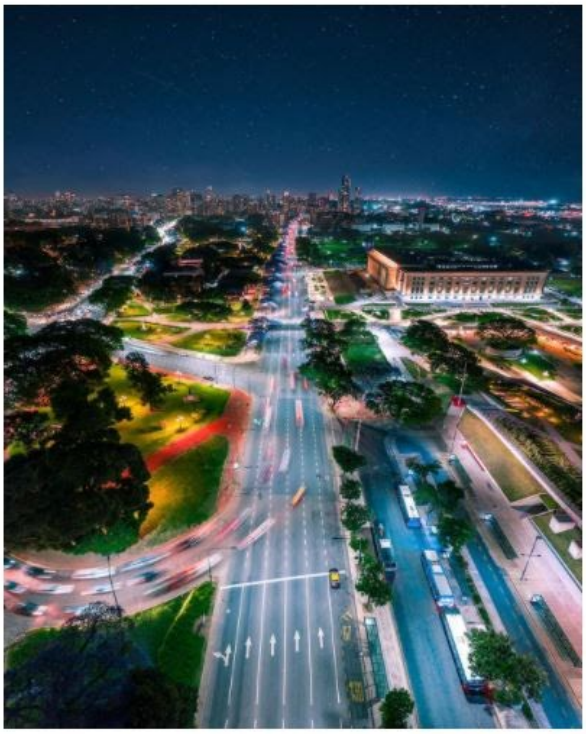
5. If you open the packet that the server sends you it shows the HTML contents of the page with all the words and addresses of images.
6. Because in HTTPs all that information is encrypted and someone else can not read everything you and the website are sending to each other, including personal data.
7. The server is listening to port 9000, the most common port for HTTP is 80, and 443 is most common for HTTPs.
8. The local port changes frequently compared to the consistently port: 33368 for the SMTP.

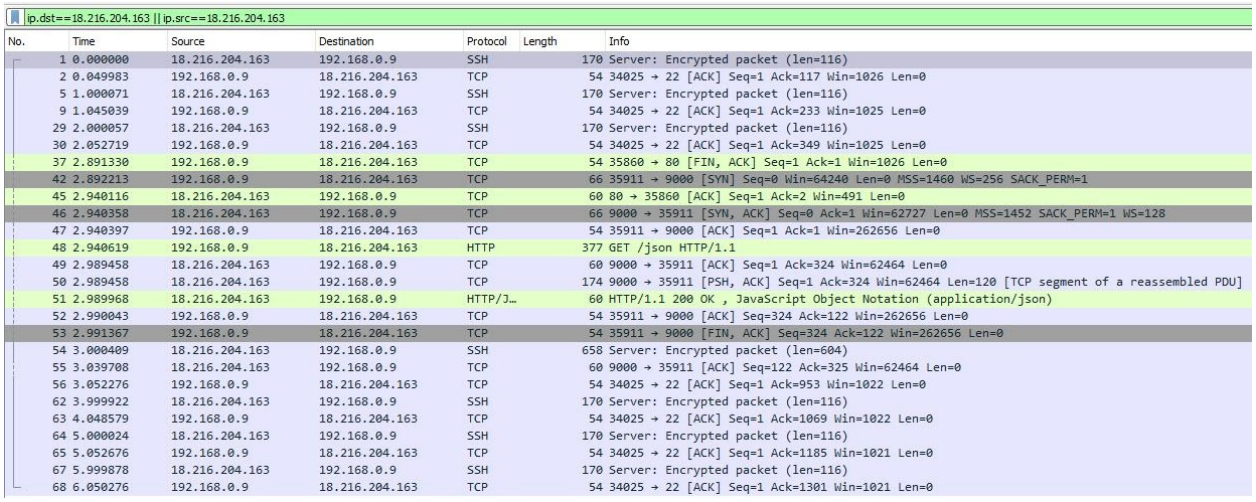
3.4



Random

streets



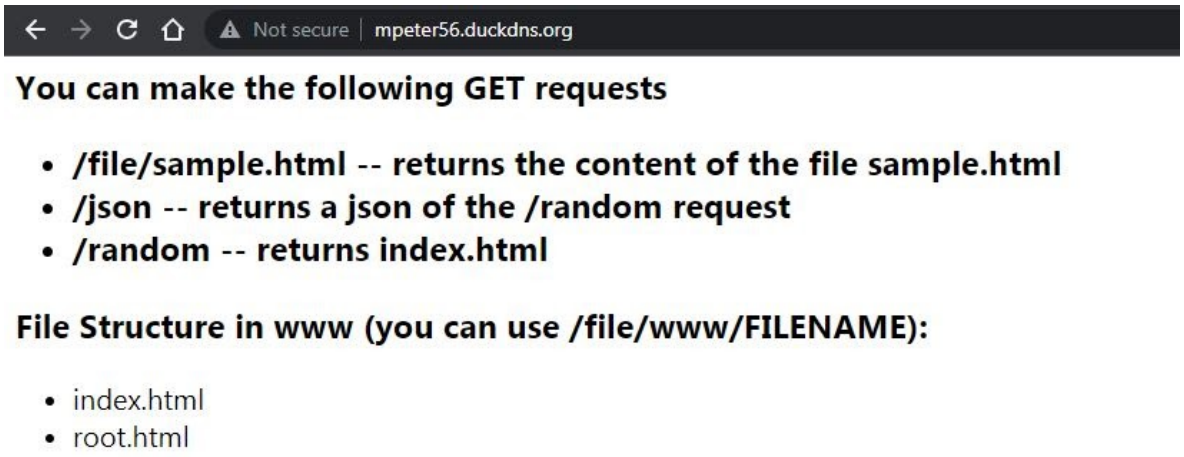


ip.dst==18.216.204.163 || ip.src==18.216.204.163

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
2	0.049983	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=117 Win=1026 Len=0
5	1.000071	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
9	1.045039	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=233 Win=1025 Len=0
29	2.000057	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
30	2.052719	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=349 Win=1025 Len=0
37	2.891330	192.168.0.9	18.216.204.163	TCP	54	35860 → 80 [FIN, ACK] Seq=1 Ack=1 Win=1026 Len=0
42	2.892213	192.168.0.9	18.216.204.163	TCP	66	35911 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
45	2.940116	18.216.204.163	192.168.0.9	TCP	60	80 → 35860 [ACK] Seq=1 Ack=2 Win=491 Len=0
46	2.940358	18.216.204.163	192.168.0.9	TCP	66	9000 → 35911 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1452 SACK_PERM=1 WS=128
47	2.940397	192.168.0.9	18.216.204.163	TCP	54	35911 → 9000 [ACK] Seq=1 Ack=1 Win=262656 Len=0
48	2.940619	192.168.0.9	18.216.204.163	HTTP	377	GET /json HTTP/1.1
49	2.989458	18.216.204.163	192.168.0.9	TCP	60	9000 → 35911 [ACK] Seq=1 Ack=324 Win=62464 Len=0
50	2.989458	18.216.204.163	192.168.0.9	TCP	174	9000 → 35911 [PSH, ACK] Seq=1 Ack=324 Win=62464 Len=120 [TCP segment of a reassembled PDU]
51	2.989968	18.216.204.163	192.168.0.9	HTTP/J…	60	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
52	2.990043	192.168.0.9	18.216.204.163	TCP	54	35911 → 9000 [ACK] Seq=324 Ack=122 Win=262656 Len=0
53	2.991367	192.168.0.9	18.216.204.163	TCP	54	35911 → 9000 [FIN, ACK] Seq=324 Ack=122 Win=262656 Len=0
54	3.000409	18.216.204.163	192.168.0.9	SSH	658	Server: Encrypted packet (len=604)
55	3.039708	18.216.204.163	192.168.0.9	TCP	60	9000 → 35911 [ACK] Seq=122 Ack=325 Win=62464 Len=0
56	3.052276	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=953 Win=1022 Len=0
62	3.999922	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
63	4.048579	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=1069 Win=1022 Len=0
64	5.000024	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
65	5.052676	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=1185 Win=1021 Len=0
67	5.999878	18.216.204.163	192.168.0.9	SSH	170	Server: Encrypted packet (len=116)
68	6.050276	192.168.0.9	18.216.204.163	TCP	54	34025 → 22 [ACK] Seq=1 Ack=1301 Win=1021 Len=0

1. Now all of the requests sent to the server are sent to port 80, port 22, or port 9000. Previously all the traffic went to port 9000 exclusively.
2. Yes, it is still HTTP. To get HTTPs you must get SSL certified and have encryption for your communication. The HTTP packets are still unencrypted and HTTP.

3.5



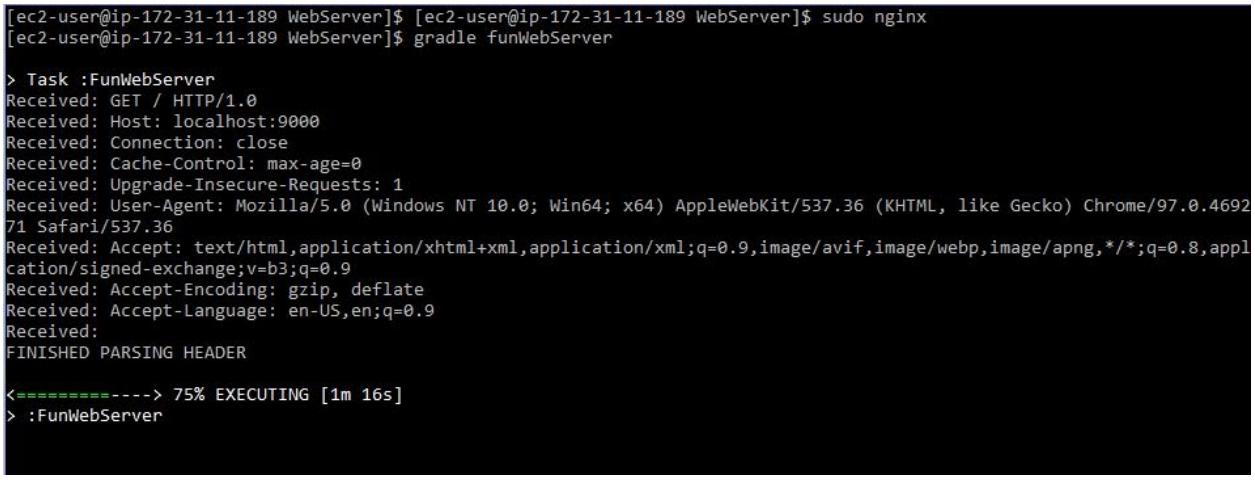
Not secure | mpeter56.duckdns.org

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):

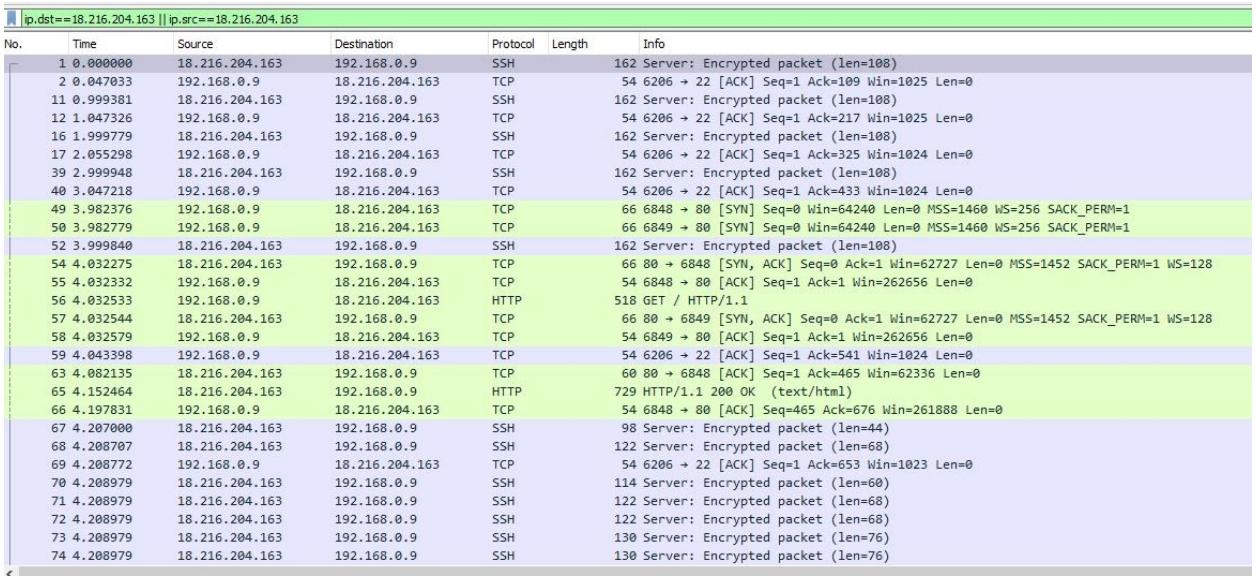
- index.html
- root.html



```
[ec2-user@ip-172-31-11-189 WebServer]$ [ec2-user@ip-172-31-11-189 WebServer]$ sudo nginx
[ec2-user@ip-172-31-11-189 WebServer]$ gradle funWebServer

> Task :FunWebServer
Received: GET / HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
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/97.0.4692
71 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appl
cation/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

<=========----> 75% EXECUTING [1m 16s]
> :FunWebServer
```



ip.dst==18.216.204.163 || ip.src==18.216.204.163

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	18.216.204.163	192.168.0.9	SSH	162	Server: Encrypted packet (len=108)
2	0.047033	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=109 Win=1025 Len=0
11	0.999381	18.216.204.163	192.168.0.9	SSH	162	Server: Encrypted packet (len=108)
12	1.047326	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=217 Win=1025 Len=0
16	1.999779	18.216.204.163	192.168.0.9	SSH	162	Server: Encrypted packet (len=108)
17	2.055298	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=325 Win=1024 Len=0
39	2.999948	18.216.204.163	192.168.0.9	SSH	162	Server: Encrypted packet (len=108)
40	3.047218	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=433 Win=1024 Len=0
49	3.982376	192.168.0.9	18.216.204.163	TCP	66	6848 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
50	3.982779	192.168.0.9	18.216.204.163	TCP	66	6849 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
52	3.999840	18.216.204.163	192.168.0.9	SSH	162	Server: Encrypted packet (len=108)
54	4.032275	18.216.204.163	192.168.0.9	TCP	66	80 → 6848 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1452 SACK_PERM=1 WS=128
55	4.032332	192.168.0.9	18.216.204.163	TCP	54	6848 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
56	4.032533	192.168.0.9	18.216.204.163	HTTP	518	GET / HTTP/1.1
57	4.032544	18.216.204.163	192.168.0.9	TCP	66	80 → 6849 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1452 SACK_PERM=1 WS=128
58	4.032579	192.168.0.9	18.216.204.163	TCP	54	6849 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
59	4.043398	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=541 Win=1024 Len=0
63	4.082135	18.216.204.163	192.168.0.9	TCP	60	80 → 6848 [ACK] Seq=1 Ack=465 Win=62336 Len=0
65	4.152464	18.216.204.163	192.168.0.9	HTTP	729	HTTP/1.1 200 OK (text/html)
66	4.197831	192.168.0.9	18.216.204.163	TCP	54	6848 → 80 [ACK] Seq=465 Ack=676 Win=261888 Len=0
67	4.207000	18.216.204.163	192.168.0.9	SSH	98	Server: Encrypted packet (len=44)
68	4.208707	18.216.204.163	192.168.0.9	SSH	122	Server: Encrypted packet (len=68)
69	4.208772	192.168.0.9	18.216.204.163	TCP	54	6206 → 22 [ACK] Seq=1 Ack=653 Win=1023 Len=0
70	4.208979	18.216.204.163	192.168.0.9	SSH	114	Server: Encrypted packet (len=60)
71	4.208979	18.216.204.163	192.168.0.9	SSH	122	Server: Encrypted packet (len=68)
72	4.208979	18.216.204.163	192.168.0.9	SSH	122	Server: Encrypted packet (len=68)
73	4.208979	18.216.204.163	192.168.0.9	SSH	130	Server: Encrypted packet (len=76)
74	4.208979	18.216.204.163	192.168.0.9	SSH	130	Server: Encrypted packet (len=76)

1. I could not get the HTTPs set up but I was able to do the duckdns part. Now there is no port 9000 and it all goes to 80 or 22.
2. Yes unfortunately because HTTPs was not set up.

3.6.1

I decided to print out:

“Please use format /multiply?num1=#&num2=#

E.g. /multiply?num1=3&num2=4”

I did not want to use default values because someone may not realize they did it wrong and could be confused why the result is the way it is. I used a try-catch block with StringIndexOutOfBoundsException to catch errors in extracting path parameters and a try-catch block inside the other for catching NumberFormatException from errors in extracting required fields from parameters. I used error cod 400 Bad Request because the error would be a syntax error.

.