

Assignment 2 SER 321 – Michael Krasnik {mkrasnik}

<https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik>

Task 1

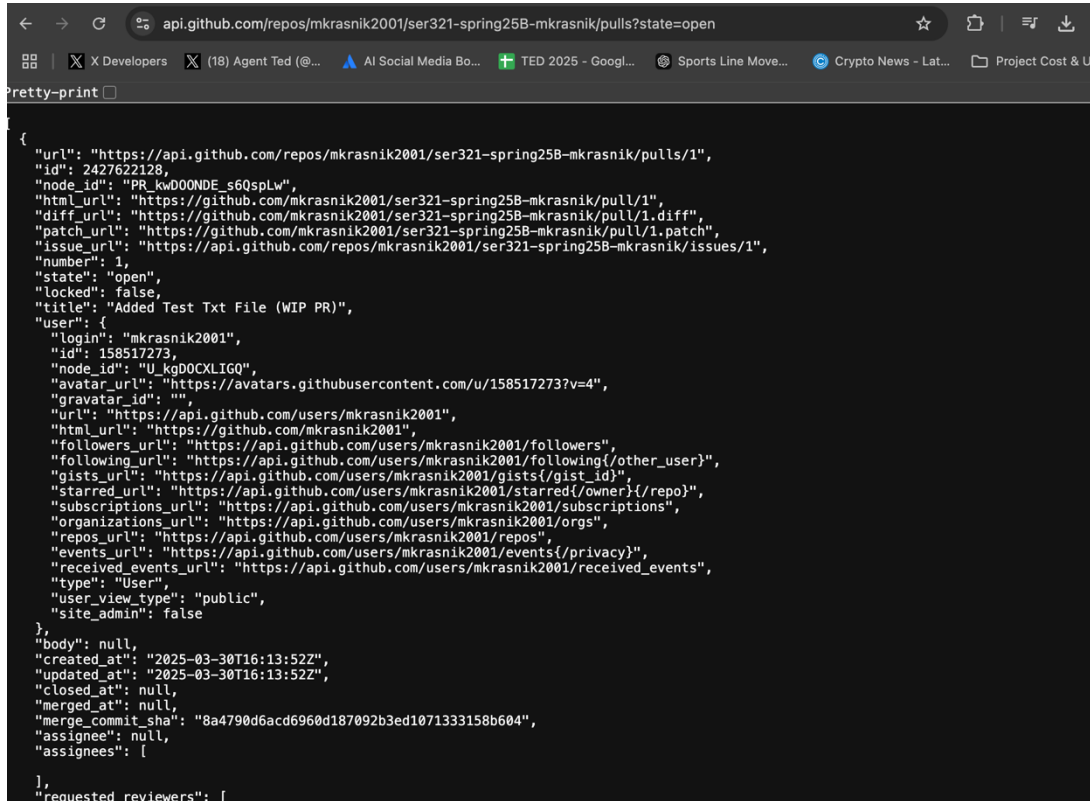
<https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits>

```
api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits
pretty-print
{
  "sha": "3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "node_id": "C_kw000NDU_toAKDMWjcmZmFjY2MyZmMxYjU5NWZlODkzZTVhN2VhMjc5ZTNlYTU3NzUz",
  "commit": {
    "author": {
      "name": "Michael",
      "email": "mike@credencelimited.com",
      "date": "2025-03-23T02:29:41Z"
    },
    "committer": {
      "name": "Michael",
      "email": "mike@credencelimited.com",
      "date": "2025-03-23T02:29:41Z"
    },
    "message": "Added pdf",
    "tree": {
      "sha": "628a8e3ab1c736ce201af8643424251f2aa3c8a0",
      "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/trees/628a8e3ab1c736ce201af8643424251f2aa3c8a0"
    },
    "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
    "comment_count": 0,
    "verification": {
      "verified": false,
      "reason": "unsigned",
      "signature": null,
      "payload": null,
      "verified_at": null
    }
  },
  "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/commit/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "comments_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775/comments",
  "author": {
    "login": "mkrasnik365",
    "id": 190394363,
    "node_id": "U_kgDQC1kv-w",
    "avatar_url": "https://avatars.githubusercontent.com/u/190394363?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mkrasnik365",
    "html_url": "https://github.com/mkrasnik365",
    "followers_url": "https://api.github.com/users/mkrasnik365/followers",
    "following_url": "https://api.github.com/users/mkrasnik365/following{other_user}",
    "gists_url": "https://api.github.com/users/mkrasnik365/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mkrasnik365/starred{/owner}/{repo}",
    "subscriptions_url": "https://api.github.com/users/mkrasnik365/subscriptions"
  }
}
```

https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits?sha=dev&per_page=40

```
api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits?sha=dev&per_page=40
Pretty-print
[
  {
    "sha": "f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "node_id": "C_kw000NDU_toAKGY2ZDhUNWU4ZDJlMDZKYjYjZjQ0ZjNhNzg5NjYlMTI0YmZm",
    "commit": {
      "author": {
        "name": "Michael",
        "email": "mike@credencelimited.com",
        "date": "2025-03-30T16:09:35Z"
      },
      "committer": {
        "name": "Michael",
        "email": "mike@credencelimited.com",
        "date": "2025-03-30T16:09:35Z"
      },
      "message": "added test file",
      "tree": {
        "sha": "a57b3e457bb68a08e333723a847752b6640e0661",
        "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/trees/a57b3e457bb68a08e333723a847752b6640e0661"
      },
      "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
      "comment_count": 0,
      "verification": {
        "verified": false,
        "reason": "unsigned",
        "signature": null,
        "payload": null,
        "verified_at": null
      }
    },
    "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/commit/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "comments_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3/comments",
    "author": {
      "login": "mkrasnik365",
      "id": 190394363,
      "node_id": "U_kgDQC1kv-w",
      "avatar_url": "https://avatars.githubusercontent.com/u/190394363?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/mkrasnik365",
      "html_url": "https://github.com/mkrasnik365",
      "followers_url": "https://api.github.com/users/mkrasnik365/followers",
      "following_url": "https://api.github.com/users/mkrasnik365/following{other_user}",
      "gists_url": "https://api.github.com/users/mkrasnik365/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/mkrasnik365/starred{/owner}/{repo}",
      "subscriptions_url": "https://api.github.com/users/mkrasnik365/subscriptions"
    }
  }
]
```

<https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/pulls?state=open>



The screenshot shows a web browser window with the address bar displaying the URL: `api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/pulls?state=open`. The browser's developer tools are open, showing the JSON response of the API call. The response is a single object representing a pull request, with fields such as `url`, `id`, `node_id`, `html_url`, `diff_url`, `patch_url`, `issue_url`, `number`, `state`, `locked`, `title`, `user`, `body`, `created_at`, `updated_at`, `closed_at`, `merged_at`, `merge_commit_sha`, `assignee`, `assignees`, and `requested_reviewers`.

```
{
  "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/pulls/1",
  "id": 2427622128,
  "node_id": "PR_kw000NDE_s60spLw",
  "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1",
  "diff_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1.diff",
  "patch_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1.patch",
  "issue_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/issues/1",
  "number": 1,
  "state": "open",
  "locked": false,
  "title": "Added Test Txt File (WIP PR)",
  "user": {
    "login": "mkrasnik2001",
    "id": 158517273,
    "node_id": "U_kgD0CXLIQ0",
    "avatar_url": "https://avatars.githubusercontent.com/u/158517273?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mkrasnik2001",
    "html_url": "https://github.com/mkrasnik2001",
    "followers_url": "https://api.github.com/users/mkrasnik2001/followers",
    "following_url": "https://api.github.com/users/mkrasnik2001/following{/other_user}",
    "gists_url": "https://api.github.com/users/mkrasnik2001/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mkrasnik2001/starred{/owner}/{repo}",
    "subscriptions_url": "https://api.github.com/users/mkrasnik2001/subscriptions",
    "organizations_url": "https://api.github.com/users/mkrasnik2001/orgs",
    "repos_url": "https://api.github.com/users/mkrasnik2001/repos",
    "events_url": "https://api.github.com/users/mkrasnik2001/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mkrasnik2001/received_events",
    "type": "User",
    "user_view_type": "public",
    "site_admin": false
  },
  "body": null,
  "created_at": "2025-03-30T16:13:52Z",
  "updated_at": "2025-03-30T16:13:52Z",
  "closed_at": null,
  "merged_at": null,
  "merge_commit_sha": "8a4790d6acd6960d187092b3ed1071333158b604",
  "assignee": null,
  "assignees": [
  ],
  "requested_reviewers": [
  ]
}
```

1. The first API call used was a GET request to view all the commits of a repo on the default branch. There are no query parameters used in this request, just the `/commit` endpoint was used. Doc link: <https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28> . The second API call was also a GET request to view the commits of a specific branch while limiting the amount of commits per page to 40. The query parameters used was specifying the branch as well as the page limit. Doc link: <https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28> . The third API call used was another GET request to the pull requests endpoint which returns a list of PRs that are still open. The query parameter used was specifying the state of the PR (open in this case). Doc link: <https://docs.github.com/en/rest/pulls/pulls?apiVersion=2022-11-28> .
2. Stateless communication refers to communication between the client and server where each request is independent. The server does not maintain any knowledge of past requests by the client and whenever a client sends a request, it needs to provide all the required info within the same request for it to be fulfilled. In

other words, there is no “state” or context maintained of that logical connection between the client and the server. A stateless communication protocol example is HTTP. Stateful communication is the opposite, this is when the server does store context and states of the conversation with the client and during requests the client doesn’t need to provide info that was provided earlier (like authentication). An example protocol could be SSH.

Task 2

2.2 Running a Simple Java WebServer (10 points)

The screenshot displays a terminal window on the left and a web browser on the right, illustrating a simple Java web server running on an AWS instance.

Terminal Window:

```
Received:
FINISHED PARSING HEADER
Received: null
FINISHED PARSING HEADER
Received: GET / HTTP/1.1
Received: Host: 98.84.143.8:9000
Received: Connection: keep-alive
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_1_1; rv:37.0) Gecko/20100101 Firefox/37.0
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8,application/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
Received: null
FINISHED PARSING HEADER
Received: GET / HTTP/1.1
Received: Host: 98.84.143.8:9000
Received: Connection: keep-alive
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_1_1; rv:37.0) Gecko/20100101 Firefox/37.0
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8,application/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 17s]
> :FunWebServer
i-00bb6e8fa7bc40bb9 (SER321 Instance)
PublicIPs: 98.84.143.8 PrivateIPs: 172.31.31.97
Feedback Privacy Terms
© 2025, Amazon Web Services, Inc. or its affiliates.
```

Web Browser:

The browser shows the URL `98.84.143.8:9000`. The page content includes:

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`

2.3 Analyze what happens (10 points)

No.	Time	Source	Destination	Protocol	Length	Info
200	6.140724	192.168.1.157	98.84.143.8	TCP	78	57702 → 9000 [SYN] Seq=0 Win=65535 Len=0...
201	6.153138	98.84.143.8	192.168.1.157	TCP	74	9000 → 57702 [SYN, ACK] Seq=0 Ack=1 Win=...
202	6.153206	192.168.1.157	98.84.143.8	TCP	66	57702 → 9000 [ACK] Seq=1 Ack=1 Win=13171...
203	6.153270	192.168.1.157	98.84.143.8	HTTP	529	GET / HTTP/1.1
204	6.172030	98.84.143.8	192.168.1.157	TCP	66	9000 → 57702 [ACK] Seq=1 Ack=464 Win=622...
205	6.173106	98.84.143.8	192.168.1.157	TCP	615	9000 → 57702 [PSH, ACK] Seq=1 Ack=464 Wi...
206	6.173107	98.84.143.8	192.168.1.157	HTTP	66	HTTP/1.1 200 OK (text/html)
207	6.173147	192.168.1.157	98.84.143.8	TCP	66	57702 → 9000 [ACK] Seq=464 Ack=550 Win=1...
208	6.173174	192.168.1.157	98.84.143.8	TCP	66	57702 → 9000 [ACK] Seq=464 Ack=551 Win=1...
209	6.173481	192.168.1.157	98.84.143.8	TCP	66	57702 → 9000 [FIN, ACK] Seq=464 Ack=551 ...
210	6.186995	98.84.143.8	192.168.1.157	TCP	66	9000 → 57702 [ACK] Seq=551 Ack=465 Win=6...
759	25.970865	192.168.1.157	98.84.143.8	TCP	78	57712 → 9000 [SYN] Seq=0 Win=65535 Len=0...
760	25.985974	98.84.143.8	192.168.1.157	TCP	74	9000 → 57712 [SYN, ACK] Seq=0 Ack=1 Win=...
761	25.986073	192.168.1.157	98.84.143.8	TCP	66	57712 → 9000 [ACK] Seq=1 Ack=1 Win=13171...
762	25.986132	192.168.1.157	98.84.143.8	HTTP	529	GET / HTTP/1.1
763	26.006007	98.84.143.8	192.168.1.157	TCP	66	9000 → 57712 [ACK] Seq=1 Ack=464 Win=622...
764	26.006008	98.84.143.8	192.168.1.157	TCP	615	9000 → 57712 [PSH, ACK] Seq=1 Ack=464 Wi...
765	26.006009	98.84.143.8	192.168.1.157	HTTP	66	HTTP/1.1 200 OK (text/html)
766	26.006060	192.168.1.157	98.84.143.8	TCP	66	57712 → 9000 [ACK] Seq=464 Ack=550 Win=1...
767	26.006084	192.168.1.157	98.84.143.8	TCP	66	57712 → 9000 [ACK] Seq=464 Ack=551 Win=1...
768	26.006377	192.168.1.157	98.84.143.8	TCP	66	57712 → 9000 [FIN, ACK] Seq=464 Ack=551 ...
769	26.020915	98.84.143.8	192.168.1.157	TCP	66	9000 → 57712 [ACK] Seq=551 Ack=465 Win=6...

Hex	ASCII
0000 3c bd c5 6d dc 15 26 31 10 10 53 62 08 00 45 00	<...m...&1...Sb...E...
0010 00 40 00 00 40 00 40 06 87 16 c0 a8 01 9d 62 54	...@...@...bT...
0020 8f 08 e1 66 23 28 1a e8 f4 2d 00 00 00 00 b0 02	...f#(...
0030 ff ff 0c 20 00 00 02 04 05 b4 01 03 03 06 01 01
0040 08 0a 94 a9 ce eb 00 00 00 00 04 02 00 00 00

1. On Wireshark I used the following filter: `ip.addr == 98.84.143.8 && tcp.port == 9000`. The ip address filter isolates traffic to and from my EC2 instance and the port filter isolates only TCP traffic to 9000 which is the port that the FunWebServer app was listening on. HTTP uses TCP as it's transfer layer protocol.
2. When we click the "Random" button on the /random page the client sends a new HTTP GET request to the server that is read, processed and shows a new image, however if we refresh the browser, there are more frames coming through because the client first sends a request to load the /random page and then calls the random button to load an image effectively doing the same thing as pressing the button but with an extra step at the beginning when loading the page.
3. The most common types of response codes are 200 which means the request was processed successfully, 400 level codes which means the client did something wrong with the request and 500 level codes where the error is on the server side.
4. The response codes I see at the moment are 200, which means the request was processed successfully and 400 which is a "Bad Request" and means the server didn't understand

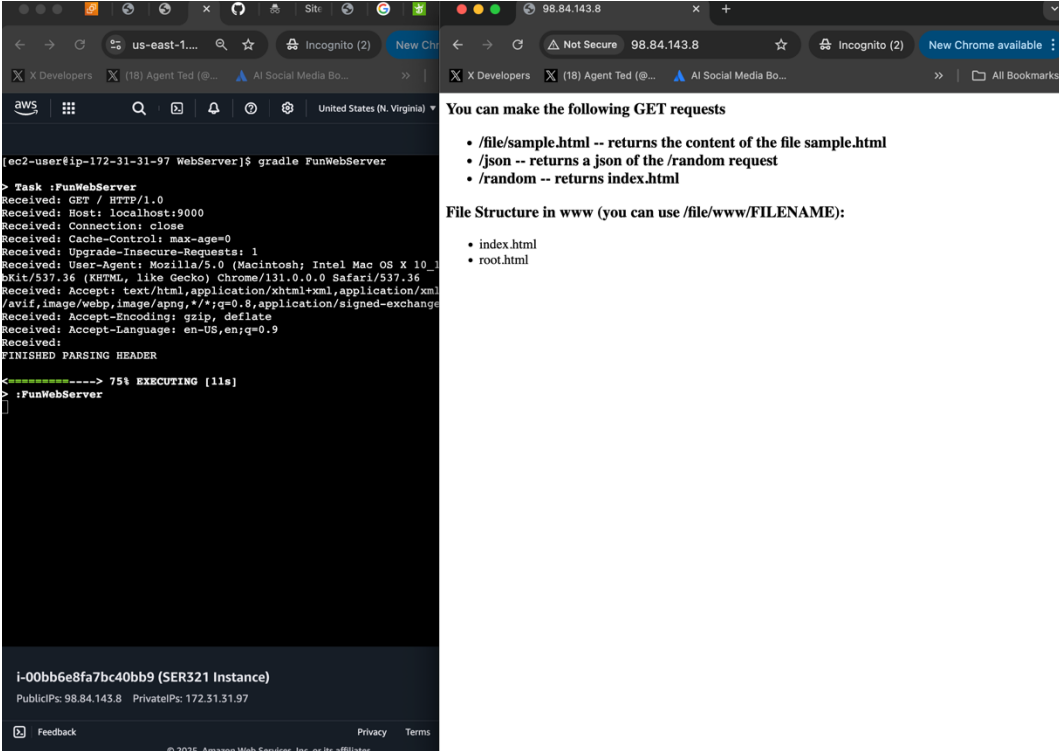
the request. When I went to the /json url I got a 400 response code which shows that the server didn't know what to do with this request.

5. Yes I can see the data that's sent back. When you click on the info section you can see the JSON returned with the data (in the /random example it's links to jpg files).
6. HTTPS is more common, because you won't be able to "read" the data if you capture the packets coming in. This is because HTTPS is encrypted and if a malicious node intercepts the packets they still won't be able to understand the data because it'll be encrypted.
7. Our server by default listens to 9000, this is not a common HTTP port. The common HTTP port is usually port 80.
8. The OS assigns temporary ports to use when sending outgoing requests and change all the time. In our case I see ports 59180, 59230 and 59179 used on the local side for outgoing requests.

2.4 Setting up a "real" WebServer (10 points)

1. The URL is <http://98.84.143.8>
2. Traffic is now going to port 80. It is not the same as before (before it was 9000) and yes it should be different.
3. We're still using HTTP because the Transport Layer Security section in the conf file is commented out and therefore nginx is using the normal port 80 and only serving HTTP.
4. Yes you can, because in the ruleset we still have port 80 to allow for inbound TCP traffic, but we don't need anymore the 8000-9999 range rule for inbound TCP traffic. It's a good security practice to update this to avoid exploitation of these ports.

5.



The screenshot shows a terminal window on the left and a web browser on the right. The terminal window displays the output of a `gradle FunWebServer` command, showing the server's response to a GET request on port 9000. The browser window shows the URL `http://98.84.143.8` and the page content, which includes a list of GET requests and a file structure.

Terminal Output:

```
[ec2-user@ip-172-31-31-97 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 (Macintosh; Intel Mac OS X 10_15_7; rv:81.0) Gecko/20100101 Firefox/81.0
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/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 [11s]
> :FunWebServer
```

Web Browser Content:

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

Capturing from Wi-Fi: en0

ip.addr == 98.84.143.8 && tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
38	2.181114	192.168.1.157	98.84.143.8	TCP	78	60660 → 80 [SYN] Seq=0 Win=65535 Len=0 M...
39	2.194062	98.84.143.8	192.168.1.157	TCP	74	80 → 60660 [SYN, ACK] Seq=0 Ack=1 Win=62...
40	2.194155	192.168.1.157	98.84.143.8	TCP	66	60660 → 80 [ACK] Seq=1 Ack=1 Win=131712 ...
41	2.194241	192.168.1.157	98.84.143.8	HTTP	524	GET / HTTP/1.1
42	2.212056	98.84.143.8	192.168.1.157	TCP	66	80 → 60660 [ACK] Seq=1 Ack=459 Win=62208...
43	2.214057	98.84.143.8	192.168.1.157	HTTP	741	HTTP/1.1 200 OK (text/html)
44	2.214111	192.168.1.157	98.84.143.8	TCP	66	60660 → 80 [ACK] Seq=459 Ack=676 Win=131...
80	2.728671	192.168.1.157	98.84.143.8	HTTP	524	GET / HTTP/1.1
81	2.747551	98.84.143.8	192.168.1.157	HTTP	741	HTTP/1.1 200 OK (text/html)
82	2.747608	192.168.1.157	98.84.143.8	TCP	66	60660 → 80 [ACK] Seq=917 Ack=1351 Win=13...
210	5.411499	192.168.1.157	98.84.143.8	TCP	66	60660 → 80 [FIN, ACK] Seq=917 Ack=1351 W...
224	5.425103	98.84.143.8	192.168.1.157	TCP	66	80 → 60660 [FIN, ACK] Seq=1351 Ack=918 W...
225	5.425185	192.168.1.157	98.84.143.8	TCP	66	60660 → 80 [ACK] Seq=918 Ack=1352 Win=13...

> Frame 38: 78 bytes on wire (624 bits) captured (78 bytes captured on interface en0) containing 66 bytes of protocol data on interface en0

> Ethernet II, Src: 26:31:00:00:00:00, Dst: 08:00:27:00:00:00

> Internet Protocol Version 4, Src: 192.168.1.157, Dst: 98.84.143.8

> Transmission Control Protocol, Src Port: 60660, Dst Port: 80, Seq: 0, Win: 65535, Len: 0

2.5 Setting up HTTPS (5 points)

```

Documents - ec2-user@ip-172-31-31-97:~/ser321-spring258-m
Received: Host: localhost:9000
Received: Connection: close
Received: Sec-Fetch-Dest: document
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Version/18.1.1 Safari/605.1.15
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Received: Sec-Fetch-Site: none
Received: Sec-Fetch-Mode: navigate
Received: Accept-Language: en-US,en;q=0.9
Received: Priority: u=0, i
Received: Accept-Encoding: gzip, deflate, br
Received:
FINISHED PARSING HEADER

Received: GET / HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: Sec-Fetch-Dest: document
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Version/18.1.1 Safari/605.1.15
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Received: Sec-Fetch-Site: none
Received: Sec-Fetch-Mode: navigate
Received: Accept-Language: en-US,en;q=0.9
Received: Priority: u=0, i
Received: Accept-Encoding: gzip, deflate, br
Received:
FINISHED PARSING HEADER

Received: GET /random HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: Sec-Fetch-Dest: document
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Version/18.1.1 Safari/605.1.15
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Received: Sec-Fetch-Site: none
Received: Sec-Fetch-Mode: navigate
Received: Accept-Language: en-US,en;q=0.9
Received: Priority: u=0, i
Received: Accept-Encoding: gzip, deflate, br
Received:
FINISHED PARSING HEADER

Received: GET /json HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: Sec-Fetch-Dest: empty
Received: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Version/18.1.1 Safari/605.1.15
Received: Accept: */*
Received: Referer: https://ser321-mkrasnik.duckdns.org/random
Received: Sec-Fetch-Site: same-origin
Received: Sec-Fetch-Mode: cors
Received: Accept-Language: en-US,en;q=0.9
Received: Priority: u=3, i
Received: Accept-Encoding: gzip, deflate, br
Received:
FINISHED PARSING HEADER

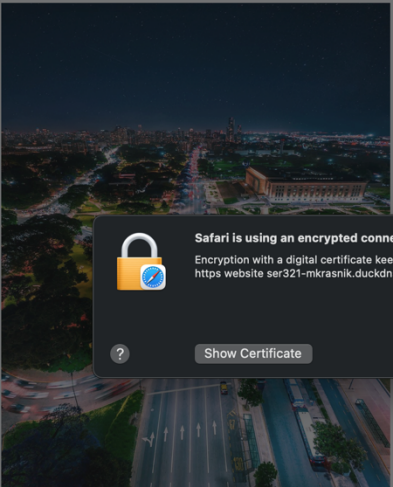
<===== 75% EXECUTING [1m 41s]
> :FunWebServer


```

ser321-mkrasnik.duckdns.org

Random

streets





Safari is using an encrypted connection to ser321-mkrasnik.duckdns.org.

Encryption with a digital certificate keeps information private as it's sent to or from the https website ser321-mkrasnik.duckdns.org.

Show Certificate

OK

Capturing from Wi-Fi: en0

ip.addr == 98.84.143.8 && tcp.port == 443

No.	Time	Source	Destination	Protocol	Length	Info
43	2.296764	192.168.1.157	98.84.143.8	TCP	78	61401 → 443 [SYN] Seq=0 Win=65535 Len=0 ...
44	2.308218	98.84.143.8	192.168.1.157	TCP	74	443 → 61401 [SYN, ACK] Seq=0 Ack=1 Win=6...
45	2.308274	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [ACK] Seq=1 Ack=1 Win=131712...
46	2.308578	192.168.1.157	98.84.143.8	TCP	1514	61401 → 443 [ACK] Seq=1 Ack=1 Win=131712...
47	2.308585	192.168.1.157	98.84.143.8	TLSv1...	434	Client Hello (SNI=ser321-mkrasnik.duckdn...
48	2.333116	192.168.1.157	98.84.143.8	TCP	1514	[TCP Retransmission] 61401 → 443 [PSH, A...
49	2.334776	98.84.143.8	192.168.1.157	TCP	66	443 → 61401 [ACK] Seq=1 Ack=1449 Win=613...
50	2.334777	98.84.143.8	192.168.1.157	TCP	66	443 → 61401 [ACK] Seq=1 Ack=1817 Win=610...
51	2.334778	98.84.143.8	192.168.1.157	TLSv1...	310	Server Hello, Change Cipher Spec, Applic...
52	2.334838	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [ACK] Seq=1817 Ack=245 Win=1...
53	2.335062	192.168.1.157	98.84.143.8	TLSv1...	130	Change Cipher Spec, Application Data
54	2.335170	192.168.1.157	98.84.143.8	TLSv1...	795	Application Data
56	2.346777	98.84.143.8	192.168.1.157	TCP	78	[TCP Dup ACK 50#1] 443 → 61401 [ACK] Seq...
57	2.348058	98.84.143.8	192.168.1.157	TCP	66	443 → 61401 [ACK] Seq=245 Ack=1881 Win=6...
58	2.348059	98.84.143.8	192.168.1.157	TLSv1...	145	Application Data
59	2.348089	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [ACK] Seq=2610 Ack=324 Win=1...
60	2.352094	98.84.143.8	192.168.1.157	TLSv1...	763	Application Data
61	2.352148	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [ACK] Seq=2610 Ack=1021 Win=...
79	3.412262	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [FIN, ACK] Seq=2610 Ack=1021...
84	3.426295	98.84.143.8	192.168.1.157	TCP	66	443 → 61401 [FIN, ACK] Seq=1021 Ack=2611...
85	3.426364	192.168.1.157	98.84.143.8	TCP	66	61401 → 443 [ACK] Seq=2611 Ack=1022 Win=...

Frame 43: 78 bytes on wire (624 bits)	Ethernet II, Src: 26:31:00:00:00:00, Dst: 08:00:00:00:00:00	Internet Protocol Version 4, Src: 192.168.1.157, Dst: 98.84.143.8	Transmission Control Protocol, Src Port: 61401, Dst Port: 443
0000	3c bd c5 6d dc 15 26 31 10 10 53 62 08 00 45 00		<...m...&1...Sb...E...
0010	00 40 00 00 40 00 40 06 87 16 c0 a8 01 9d 62 54		@...@...@...bT...
0020	8f 08 ef d9 01 bb 88 79 80 7c 00 00 00 00 b0 02	y...
0030	ff ff 91 d5 00 00 02 04 05 b4 01 03 03 06 01 01	p.....
0040	08 0a 27 70 cf 89 00 00 00 00 04 02 00 00		

Wi-Fi: en0: <live capture in progress> Packets: 3111 · Displayed: 21 (0.7%) Profile: Default

1. Traffic is now going through the standard TCP port 443.
2. No all I can see is the TCP payload and the packet data is basically gibberish and encrypted.

2.6.4

Ser321-mkrasnik.duckdns.org

(Posted on #server channel as well)