

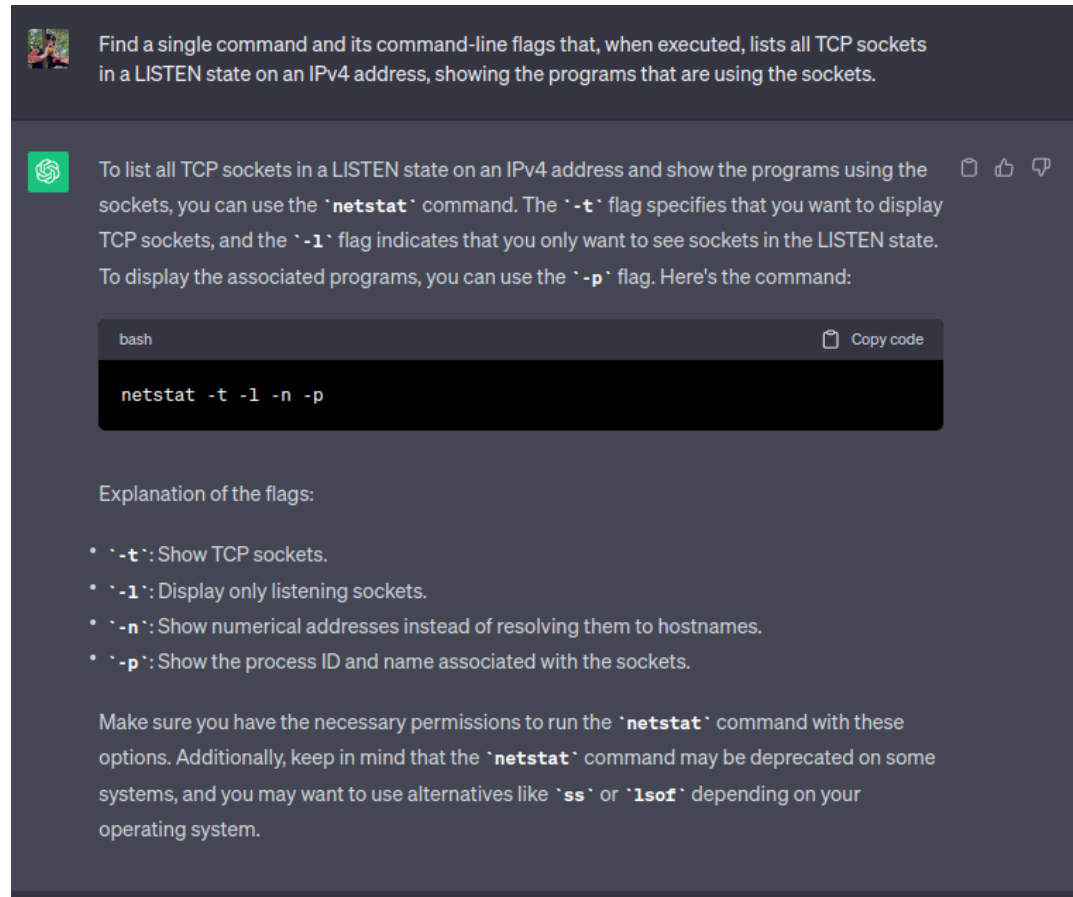
TABLE OF CONTENTS:

I. Lab 2.1 - TCP, HTTP.....	2
1. TCP #1: Sockets.....	2
2. LSOF.....	4
3. TCP #2: Throughput.....	5
4. - iperf.....	6
5. HTTP #3: Requests.....	7
6. - Asynchronous HTTP requests.....	12
II. Lab 2.2 - TODO.....	13
1. DNS reconnaissance #1 (dig).....	13
2. Iterative DNS lookups.....	16
3. Reverse DNS lookups.....	16
4. Host enumeration.....	18
5. Geographic DNS #2.....	18

I. Lab 2.1 - TCP, HTTP

1. TCP #1: Sockets

- Using ChatGPT, find a single command and its command-line flags that, when executed, lists all **TCP** sockets in a **LISTEN** state on an **IPv4** address, showing the **program** that is using it.
 - Take a screenshot of the prompt and the command that ChatGPT generates



- Run the command using **sudo** and take a screenshot of the output to include in your lab notebook.

```
meelz(amminer)@course-vm: ~
meelz(amminer)@course-vm:~$ sudo netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*                LISTEN      366/systemd-resolve
tcp        0      0 127.0.0.1:37149         0.0.0.0:*                LISTEN      453/containerd
tcp        0      0 0.0.0.0:22              0.0.0.0:*                LISTEN      794/sshd: /usr/sbin
tcp6       0      0 :::1:3350                :::*                    LISTEN      560/xrdp-sesman
tcp6       0      0 :::3389                  :::*                    LISTEN      644/xrdp
tcp6       0      0 :::22                    :::*                    LISTEN      794/sshd: /usr/sbin
meelz(amminer)@course-vm:~$
```

- List a service that can be contacted from any interface on the machine. List a service that can only be contacted by local processes.

sshd can be contacted from any interface on the machine. containerd can only be contacted by local processes.

- ```

meelz@meelzBox: ~
meelzBox:~ > ssh amminer@linux.cs.pdx.edu
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-75-generic x86_64)

=====
This machine is for the exclusive use of those associated with
the Maseeh College of Engineering and Computer Science.

ALL ACTIVITY MAY BE RECORDED
=====
* CAT Support: https://cat.pdx.edu/
* Email: support@cat.pdx.edu
* Phone: 503-725-5420
* Chat: https://support.cat.pdx.edu
* Location: FAB 82-01

Last login: Sun Oct 8 07:10:54 2023 from 104.220.249.53
amminer@ada:~ > netstat -tlnp
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:39507 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.53:53 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:6100 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:5984 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:5953 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:5903 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp6 0 0 :::1:6100 :::* LISTEN -
tcp6 0 0 :::1:5903 :::* LISTEN -
tcp6 0 0 :::1:5953 :::* LISTEN -
tcp6 0 0 :::1:5984 :::* LISTEN -
tcp6 0 0 :::1:25 :::* LISTEN -
tcp6 0 0 :::1:631 :::* LISTEN -
tcp6 0 0 :::1719 :::* LISTEN -
tcp6 0 0 :::1717 :::* LISTEN -
tcp6 0 0 :::1716 :::* LISTEN -
tcp6 0 0 :::113 :::* LISTEN -
tcp6 0 0 :::22 :::* LISTEN -
amminer@ada:~ >

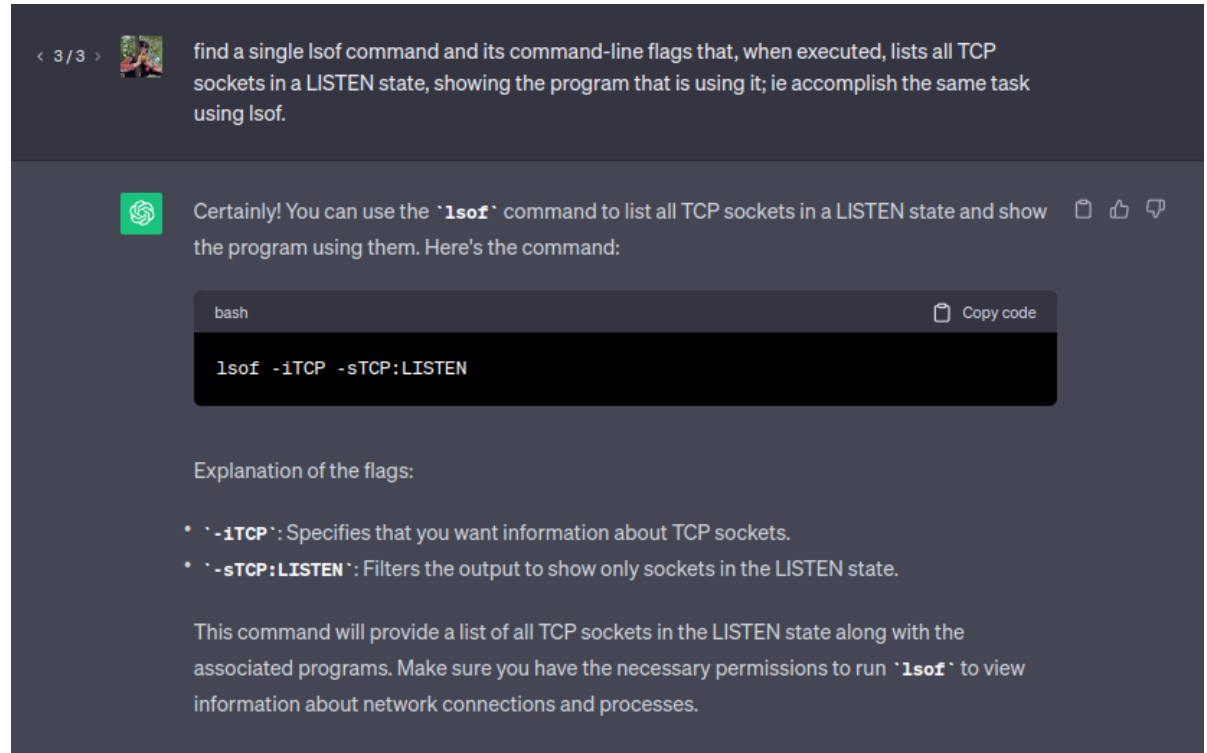
```

- List the services this machine provides for external access.

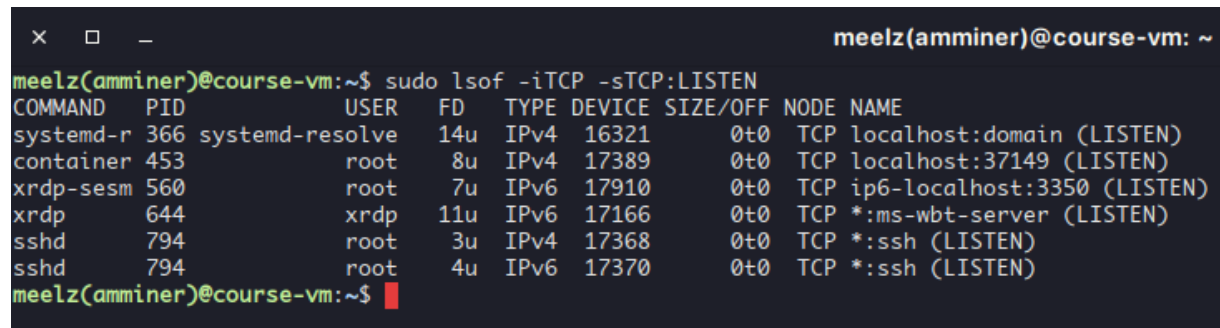
It looks like only ssh is available to external connections on linux.cs.pdx.edu.

## 2. LSOF

- Using ChatGPT, find a single `lsof` command and its command-line flags that, when executed, lists all `TCP` sockets in a `LISTEN` state on an `IPv4` address, showing the `program` that is using it. Note that you can leverage the conversation in the previous step and simply ask ChatGPT to repeat the task using `lsof`.
  - Take a screenshot of the prompt and the command that ChatGPT generates.



- Run the command using `sudo` and take a screenshot of the output to include in your lab notebook.



### 3. TCP #2: Throughput

- VMs instantiated:

NAME: vm-europe-west1-d  
INTERNAL\_IP: 10.132.0.2  
EXTERNAL\_IP: 35.195.100.133

NAME: vm-us-west1-b  
INTERNAL\_IP: 10.138.0.6  
EXTERNAL\_IP: 34.168.52.191

NAME: vm-us-east1-b  
INTERNAL\_IP: 10.142.0.3  
EXTERNAL\_IP: 34.74.3.81

NAME: vm-australia-southeast1-b  
INTERNAL\_IP: 10.152.0.2  
EXTERNAL\_IP: 35.189.16.39

## 4. - iperf

- On each foreign machine, run `sudo iperf -s -p 80`.
- On the local machine, run `iperf -c <IP address> -p 80` for each foreign machine's internal IP.
- Show a screenshot of the measured bandwidth available between your us-west1-b VM and each of the other Compute Engine VMs. Explain the relative differences (or lack thereof) in your results.

```
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2
> do
> iperf -c $addr -p 80 | tee $addr.txt
> done

Client connecting to 10.132.0.2, TCP port 80
TCP window size: 85.0 KByte (default)

[1] local 10.138.0.6 port 46862 connected with 10.132.0.2 port 80
[ID] Interval Transfer Bandwidth
[1] 0.0000-10.2068 sec 185 MBytes 152 Mbits/sec

Client connecting to 10.142.0.3, TCP port 80
TCP window size: 85.0 KByte (default)

[1] local 10.138.0.6 port 47144 connected with 10.142.0.3 port 80
[ID] Interval Transfer Bandwidth
[1] 0.0000-10.0767 sec 435 MBytes 362 Mbits/sec

Client connecting to 10.152.0.2, TCP port 80
TCP window size: 85.0 KByte (default)

[1] local 10.138.0.6 port 38712 connected with 10.152.0.2 port 80
[ID] Interval Transfer Bandwidth
[1] 0.0000-10.2228 sec 173 MBytes 142 Mbits/sec
amminer@vm-us-west1-b:~$ ls
10.132.0.2.txt 10.142.0.3.txt 10.152.0.2.txt
amminer@vm-us-west1-b:~$
```

(in the same order as above: eu-west, us-east, aus-southeast).

Bandwidth values correspond roughly with discrepancies I would expect based on physical distances between the machines. Interestingly, when I run traceroute to these machines, discrepancies in the number of hops are minimal, presumably because I'm routing over gcloud's internal network which creates a sort of virtual data link over the actual routers that underlie it, like what we were talking about last week in lecture?

```
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr | wc -l; done
2
2
2
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr; done
traceroute to 10.132.0.2 (10.132.0.2), 30 hops max, 60 byte packets
 0 *
 1 vm-europe-west1-d.europe-west1-d.c.cloud-miner-amminer.internal (10.132.0.2) 135.685 ms 135.638 ms *
 2 *
 3 *
 4 *
 5 vm-us-east1-b.us-east1-b.c.cloud-miner-amminer.internal (10.142.0.3) 65.189 ms * *
 6 *
 7 *
 8 *
 9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr | wc -l; done
2
2
2
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr; done
traceroute to 10.152.0.2 (10.152.0.2), 30 hops max, 60 byte packets
 0 *
 1 vm-australia-southeast1-b.australia-southeast1-b.c.cloud-miner-amminer.internal (10.152.0.2) 149.845 ms * 149.791
 2 *
 3 *
 4 *
 5 *
 6 *
 7 *
 8 *
 9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
amminer@vm-us-west1-b:~$
```

## 5. HTTP #3: Requests

- Using chrome, visit <http://google.com> in incognito mode with quic/http3 enabled. Take a screenshot of the initial 3 requests that the browser makes:

The screenshot shows the Chrome DevTools Network tab with the 'All' filter selected. The first three requests are highlighted in blue. The table below shows the details of these requests:

| Name       | Status | Type     | Initiator | Size         | Time    | Waterfall |
|------------|--------|----------|-----------|--------------|---------|-----------|
| google.com | 307    | docum... | Other     | 0 B          | Pending |           |
| google.com | 301    | docum... | (index)   | 0 B          | Pending |           |
| google.com | 200    | docum... | (index)   | (disk ca...) | Pending |           |

The waterfall chart shows the timing of these requests. The first request is a 307 redirect, the second is a 301 redirect, and the third is a 200 OK response. The table also shows other resources loaded by the browser, including fonts, scripts, and stylesheets.

- For each of the initial 3 requests:
  - What is the URL being requested?

<http://google.com>,

<https://google.com>,

and <http://google.com>

- Explain the HTTP status code that is returned and what the code indicates

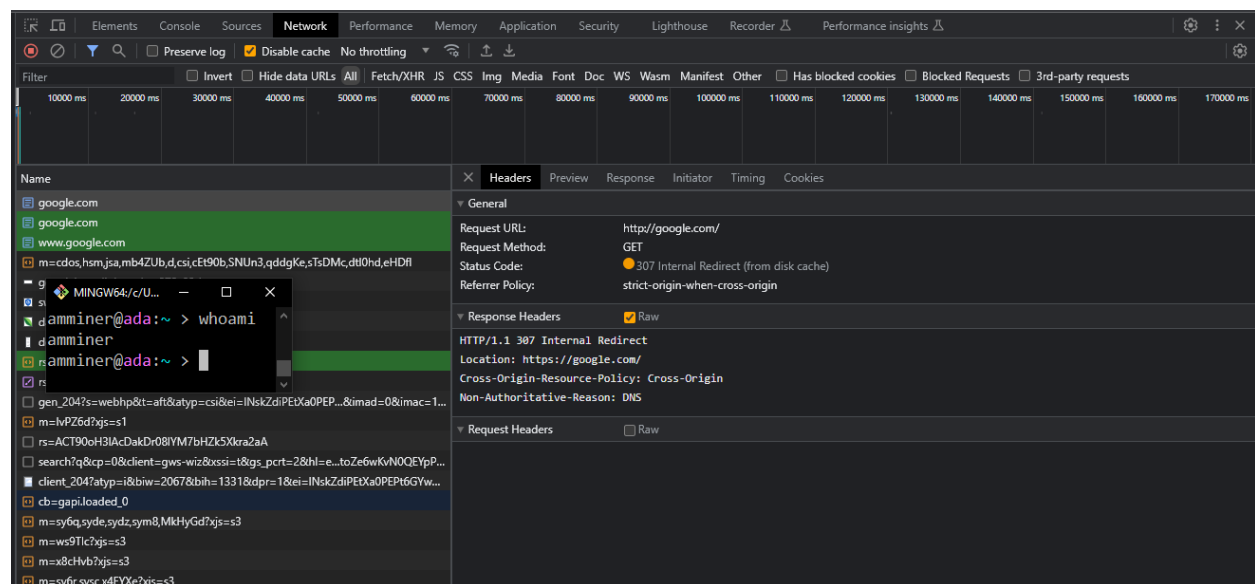
307-temporary redirect. The response includes a new address to which the resource has been “temporarily” moved.

301-moved permanently. The response includes a new address to which the resource has been “permanently” moved.

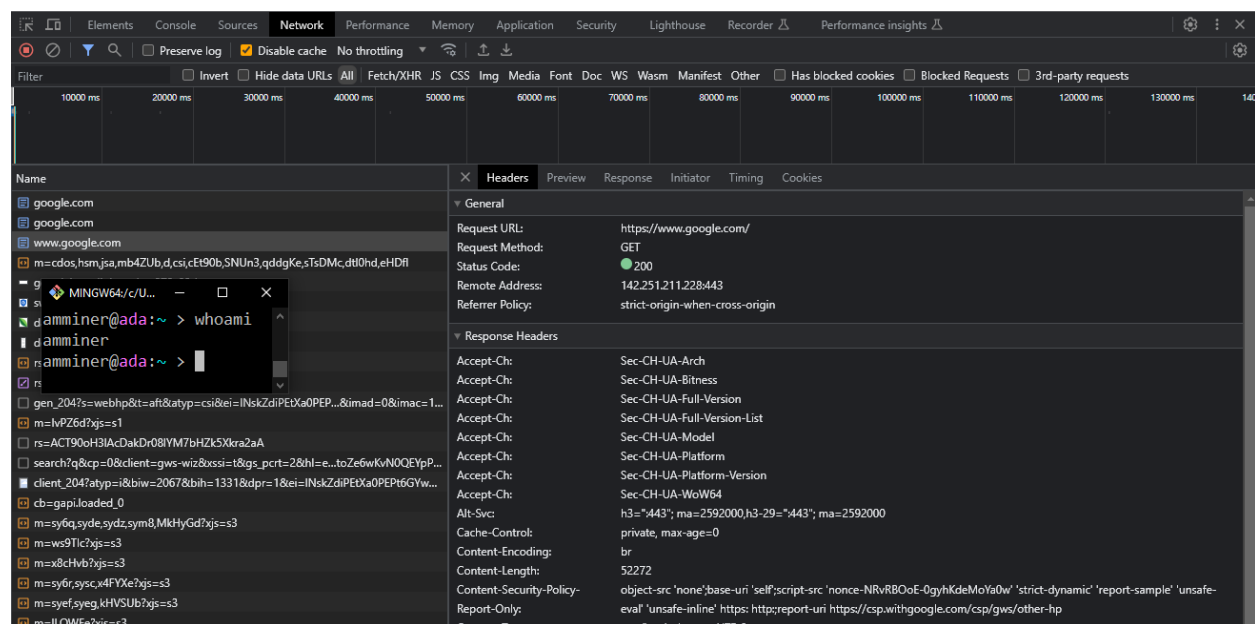
200-ok. The request was successful - here’s those bits you asked for.

- Take a screenshot indicating the version of the HTTP protocol that is used for each request. (Hint: look at the response status line and `alt-svc`: HTTP response headers indicating HTTP/2 or HTTP/3).

My first request shows http/1.1. The other requests aren’t showing any version info in their headers. `alt-svc` in the responses doesn’t mean anything to me - what’s wrong with my setup? Chrome also occasionally mixes up the wrong status code/name combinations, like 200-internal redirect, etc.







- Find the **Location** response header for the two redirections.
  - What URL does the first redirection send the browser to?

`https://google.com`

- What URL does the second redirection send the browser to?

`https://www.google.com/`

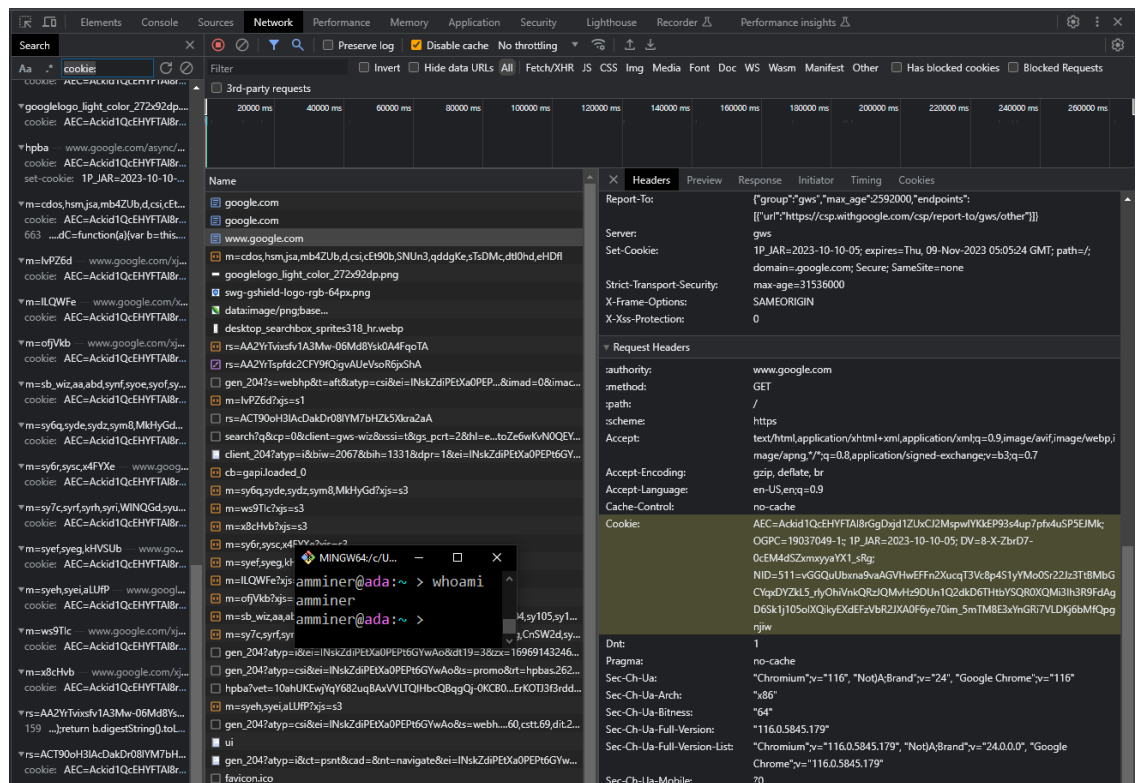
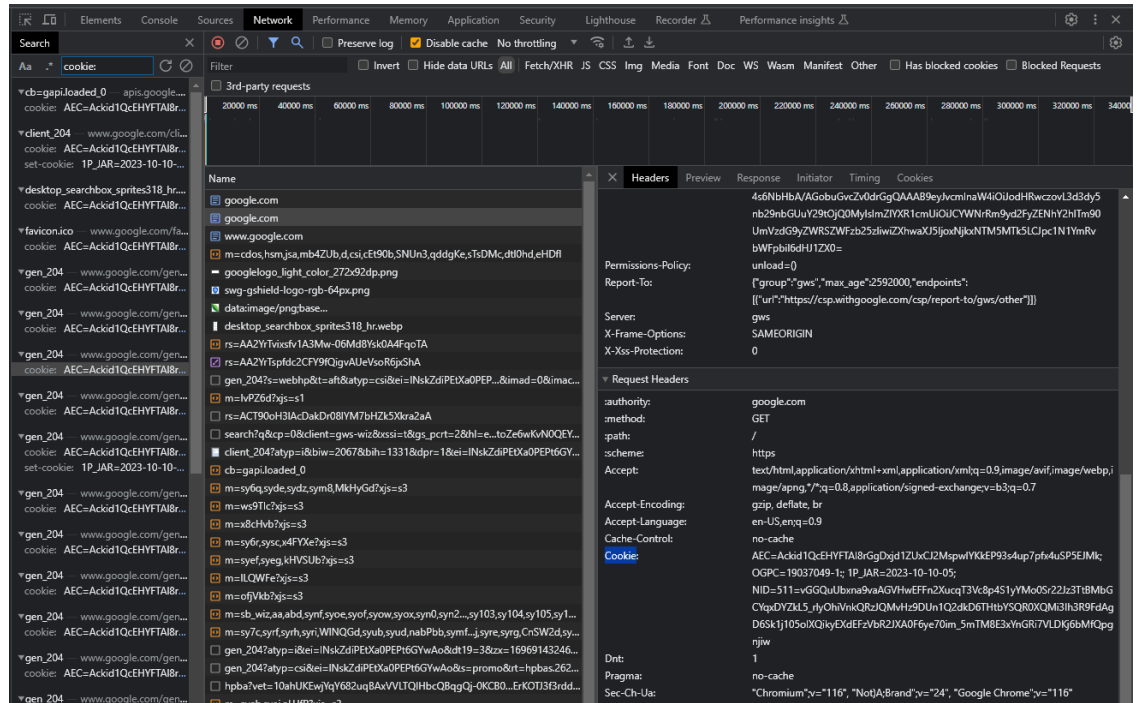
- Examine the HTTP request and response headers for cookies throughout the requests.

- Take a screenshot of when cookies are set via Set-Cookie:

In the third request:

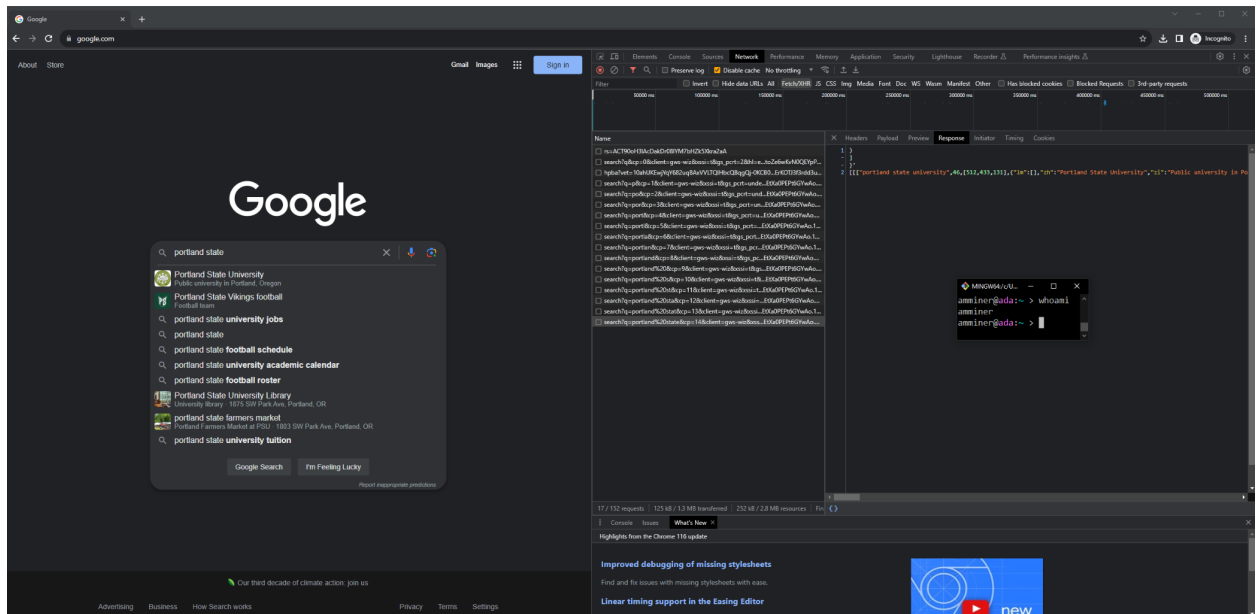
The screenshot shows the Chrome DevTools Network tab with the third request selected. The Headers panel is expanded, showing the Set-Cookie header: `TP_JAR=2023-10-10-05; expires=Thu, 09-Nov-2023 05:05:24 GMT; path=/; domain=.google.com; Secure; SameSite=none`. The Cookies panel shows the cookie is set.

- Take a screenshot of when cookies are attached via Cookie:  
In the second and third requests:



## 6. - Asynchronous HTTP requests

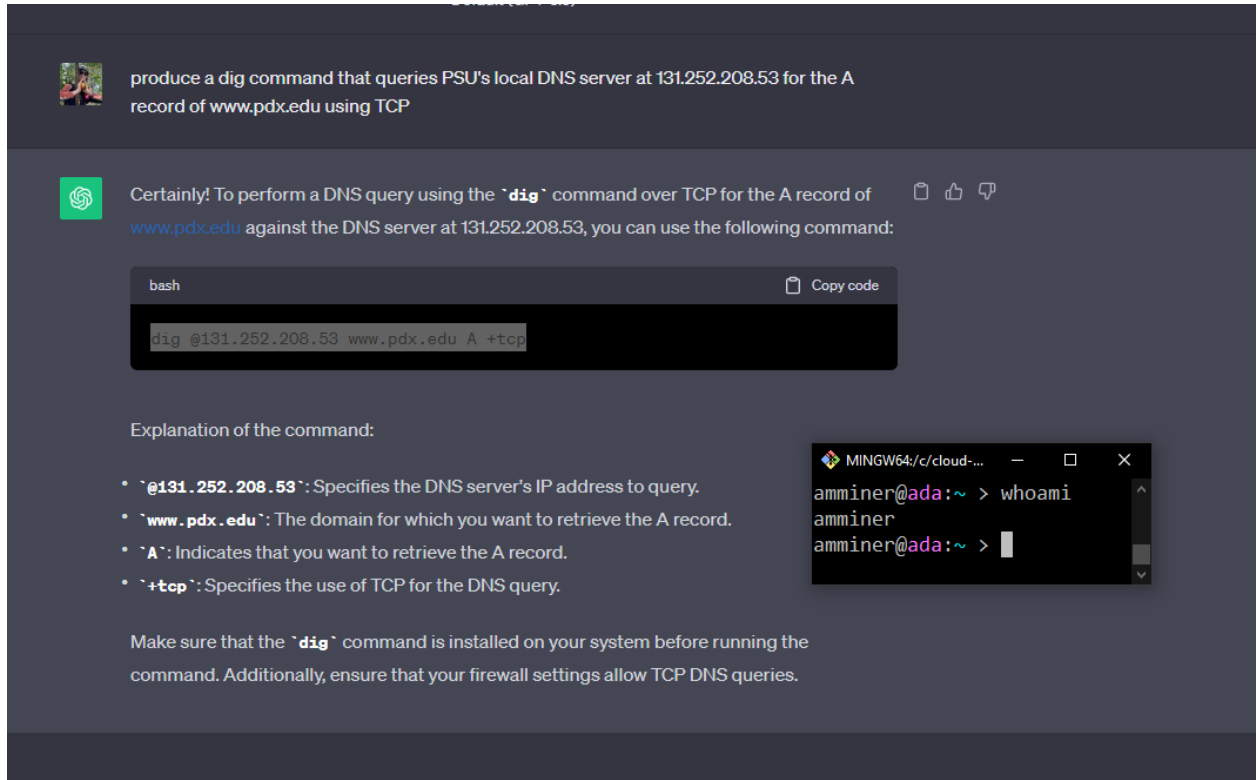
- Show the asynchronous HTTP requests made by typing in the search bar:



## II. Lab 2.2 - TODO

### 1. DNS reconnaissance #1 (dig)

- Using ChatGPT, produce a `dig` command that queries PSU's local DNS server at 131.252.208.53 for the `A` record of `www.pdx.edu` using TCP.



The screenshot shows a ChatGPT interface. The user's prompt is: "produce a dig command that queries PSU's local DNS server at 131.252.208.53 for the A record of www.pdx.edu using TCP". The AI's response is: "Certainly! To perform a DNS query using the `dig` command over TCP for the A record of `www.pdx.edu` against the DNS server at 131.252.208.53, you can use the following command: `dig @131.252.208.53 www.pdx.edu A +tcp`". Below the command, there is an "Explanation of the command:" section with four bullet points: 

- `@131.252.208.53`: Specifies the DNS server's IP address to query.
- `www.pdx.edu`: The domain for which you want to retrieve the A record.
- `A`: Indicates that you want to retrieve the A record.
- `+tcp`: Specifies the use of TCP for the DNS query.

 At the bottom, a note says: "Make sure that the `dig` command is installed on your system before running the command. Additionally, ensure that your firewall settings allow TCP DNS queries." To the right of the explanation is a terminal window snippet showing a user running `whoami` and getting the output `amminer`.

- Run the command to find the record. Then, use `dig` to do the same for the `MX` record of `pdx.edu`.

```
MINGW64/c/cloud-miner-amminer
amminer@ada:~ > dig @131.252.208.53 www.pdx.edu A +tcp

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @131.252.208.53 www.pdx.edu A +
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27036
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: 6be0bb9ac38971c6010000006524e1a787ff7611cc8e0942 (good)
;; QUESTION SECTION:
;www.pdx.edu. IN A

;; ANSWER SECTION:
www.pdx.edu. 60 IN A 18.161.6.84
www.pdx.edu. 60 IN A 18.161.6.120
www.pdx.edu. 60 IN A 18.161.6.112
www.pdx.edu. 60 IN A 18.161.6.96

;; Query time: 15 msec
;; SERVER: 131.252.208.53#53(131.252.208.53) (TCP)
;; WHEN: Mon Oct 09 22:31:19 PDT 2023
;; MSG SIZE rcvd: 132

amminer@ada:~ > dig @131.252.208.53 pdx.edu MX

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @131.252.208.53 pdx.edu MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3088
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: 6c40b37c9a25fe72010000006524e1aa46cdc55930328b08 (good)
;; QUESTION SECTION:
;pdx.edu. IN MX

;; ANSWER SECTION:
pdx.edu. 61457 IN MX 10 alt4.aspmx.l.google.com.
pdx.edu. 61457 IN MX 10 alt3.aspmx.l.google.com.
pdx.edu. 61457 IN MX 1 aspmx.l.google.com.
pdx.edu. 61457 IN MX 5 alt1.aspmx.l.google.com.
pdx.edu. 61457 IN MX 5 alt2.aspmx.l.google.com.

;; Query time: 0 msec
;; SERVER: 131.252.208.53#53(131.252.208.53) (UDP)
;; WHEN: Mon Oct 09 22:31:22 PDT 2023
;; MSG SIZE rcvd: 182

amminer@ada:~ >
```

Using the IP addresses contained in these records, utilize IP address information services at <https://www.iplocation.net/> to answer the following questions

- What cloud provider hosts the web site for [www.pdx.edu](http://www.pdx.edu)?

Amazon.

- What cloud provider handles mail for pdx.edu?

Google.

- Use `dig` to find the authoritative server (NS record type, AUTHORITY section response) for [mashimaro.cs.pdx.edu](http://mashimaro.cs.pdx.edu) and then query that server for the A record of [mashimaro.cs.pdx.edu](http://mashimaro.cs.pdx.edu).

```
MINGW64/c/cloud-miner-amminer
amminer@ada:~ > dig mashimaro.cs.pdx.edu NS

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> mashimaro.cs.pdx.edu NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49205
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;mashimaro.cs.pdx.edu. IN NS

;; AUTHORITY SECTION:
cs.pdx.edu. 300 IN SOA walt.ee.pdx.edu. support.cat.pdx.edu. 2023100302 600 300 1209600 300

;; Query time: 7 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Oct 09 22:34:00 PDT 2023
;; MSG SIZE rcvd: 105

amminer@ada:~ > dig @walt.ee.pdx.edu mashimaro.cs.pdx.edu A

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @walt.ee.pdx.edu mashimaro.cs.pdx.edu A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12901
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 15def9c0f80e1d9f010000006524e27945012ec1b5840806 (good)
;; QUESTION SECTION:
;mashimaro.cs.pdx.edu. IN A

;; ANSWER SECTION:
mashimaro.cs.pdx.edu. 14400 IN A 131.252.220.66

;; Query time: 0 msec
;; SERVER: 131.252.208.38#53(walt.ee.pdx.edu) (UDP)
;; WHEN: Mon Oct 09 22:34:49 PDT 2023
;; MSG SIZE rcvd: 93

amminer@ada:~ >
```



## 2. Iterative DNS lookups

- dig told me the f root server is named f.root-servers.net.
- dig f.root-servers.net told me the f root server's IP address is 192.5.5.241. Now we can really start. I'll resolve example.com:
- dig @192.5.5.241 example.com NS +tcp points us toward several .com TLD servers and conveniently includes their A records in addition to the requested NS records. I'll go with f.gtld-servers.net. 172800 IN A 192.35.51.30.
- dig @192.35.51.30 example.com NS +tcp points us down another layer to a server owned by IANA, as we would expect for example.com. Annoyingly we don't get free A records back this time. An additional query to the same TLD server for the A record associated with a.iana-servers.net yields 199.43.135.53 - this looks to be the authoritative server.
- Finally:

```
amminer@ada:~ > dig @199.43.135.53 example.com A +tcp

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @199.43.135.53 example.com A +tcp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27290
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
;; QUESTION SECTION:
;example.com. IN A

;; ANSWER SECTION:
example.com. 86400 IN A 93.184.216.34

;; Query time: 27 msec
;; SERVER: 199.43.135.53#53(199.43.135.53) (TCP)
;; WHEN: Mon Oct 09 22:47:32 PDT 2023
;; MSG SIZE rcvd: 56

amminer@ada:~ >
```

## 3. Reverse DNS lookups

- Using ChatGPT, produce a single command line with commands `dig`, `egrep`, and `awk`, to list all IPv4 addresses that `espn.go.com` points to.





Using ChatGPT, produce a single command line with commands `dig`, `egrep`, and `awk`, to list all IPv4 addresses that `espn.go.com` points to.



Certainly! Here's a command line that uses `dig`, `egrep`, and `awk` to list all IPv4 addresses associated with `espn.go.com`:



bash

Copy code

```
dig +short espn.go.com | egrep '([0-9]{1,3}\.){3}[0-9]{1,3}' | awk '{print $1}'
```

This command performs the following steps:

1. `dig +short espn.go.com`: Uses `dig` to query the DNS for the IPv4 addresses associated with `espn.go.com` and outputs them in a short format.
2. `egrep '([0-9]{1,3}\.){3}[0-9]{1,3}'`: Uses `egrep` to filter out lines containing IPv4 addresses using a regular expression.
3. `awk '{print $1}'`: Uses `awk` to print the first field (IPv4 address) from the filtered output.

This should give you a list of IPv4 addresses associated with `espn.go.com`.

- Run the command

MINGW64/c/cloud-miner-amminer

```
amminer@ada:~ > dig +short espn.go.com | egrep '([0-9]{1,3}\.){3}[0-9]{1,3}' | awk '{print $1}'
18.161.6.80
18.161.6.38
18.161.6.94
18.161.6.89
amminer@ada:~ >
```

- Find the flag for `dig` that allows one to perform reverse lookups on IPv4 addresses. Take that list of addresses and create a single for loop in the shell that iterates over the list and performs a reverse lookup of each IP address to find each address's associated DNS name. Pipe output of the loop to `egrep` and `awk` so that the output consists only of the DNS names.

MINGW64/c/cloud-miner-amminer

```
amminer@ada:~ > addrs=$(dig +short espn.go.com | egrep '([0-9]{1,3}\.){3}[0-9]{1,3}' | awk '{print $1}')
amminer@ada:~ > echo $addrs
18.161.6.89 18.161.6.80 18.161.6.94 18.161.6.38
amminer@ada:~ > for a in $addrs; do dig -x $a | egrep "$a.+" | awk '{print $5}'; done
server-18-161-6-89.hio52.r.cloudfront.net.
server-18-161-6-80.hio52.r.cloudfront.net.
server-18-161-6-94.hio52.r.cloudfront.net.
server-18-161-6-38.hio52.r.cloudfront.net.
amminer@ada:~ >
```

## 4. Host enumeration

- Using a `for` loop, perform a reverse DNS lookup for each IP address on the `131.252.220.0/24` subnet. Note that some addresses on the subnet do not have names bound to them and will not return a record. Take the output of the loop and pipe it to `egrep` and `awk` to list just the names of the hosts, then redirect the final output to a file called `220hosts.txt`, using the `>` character to perform output redirection to a file.
- Within the range of hosts is a set of car manufacturer names. Using the `head` and `tail` commands, craft a command in the format below that returns their names.

```
cat 220hosts.txt | head -<number_of_lines_1> | tail -<number_of_lines_2>
```

```
MINGW64/c/cloud-miner-amminer
amminer@ada:~ > for ip in {1..254}; do dig -x 131.252.220.$ip +short; done | awk '{print $1}' > 220hosts.txt
amminer@ada:~ > cat 220hosts.txt | head -185 | tail -30
acura.cs.pdx.edu.
astonmartin.cs.pdx.edu.
audi.cs.pdx.edu.
bentley.cs.pdx.edu.
bmw.cs.pdx.edu.
cadillac.cs.pdx.edu.
ferrari.cs.pdx.edu.
fiat.cs.pdx.edu.
ford.cs.pdx.edu.
honda.cs.pdx.edu.
hummer.cs.pdx.edu.
jaguar.cs.pdx.edu.
jeep.cs.pdx.edu.
lamborghini.cs.pdx.edu.
landrover.cs.pdx.edu.
lexus.cs.pdx.edu.
lotus.cs.pdx.edu.
maserati.cs.pdx.edu.
mazda.cs.pdx.edu.
mclaren.cs.pdx.edu.
mercedes.cs.pdx.edu.
nissan.cs.pdx.edu.
panoz.cs.pdx.edu.
porsche.cs.pdx.edu.
subaru.cs.pdx.edu.
toyota.cs.pdx.edu.
tvr.cs.pdx.edu.
ultima.cs.pdx.edu.
volvo.cs.pdx.edu.
vw.cs.pdx.edu.
amminer@ada:~ >
```

## 5. Geographic DNS #2

TODO...