

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
6. Wireshark Lab #3.....	20
7. -.....	21
8. Network Recap Lab #4.....	22
9. Collect trace.....	23
10. Analyze trace.....	23
11. Clean up.....	24

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



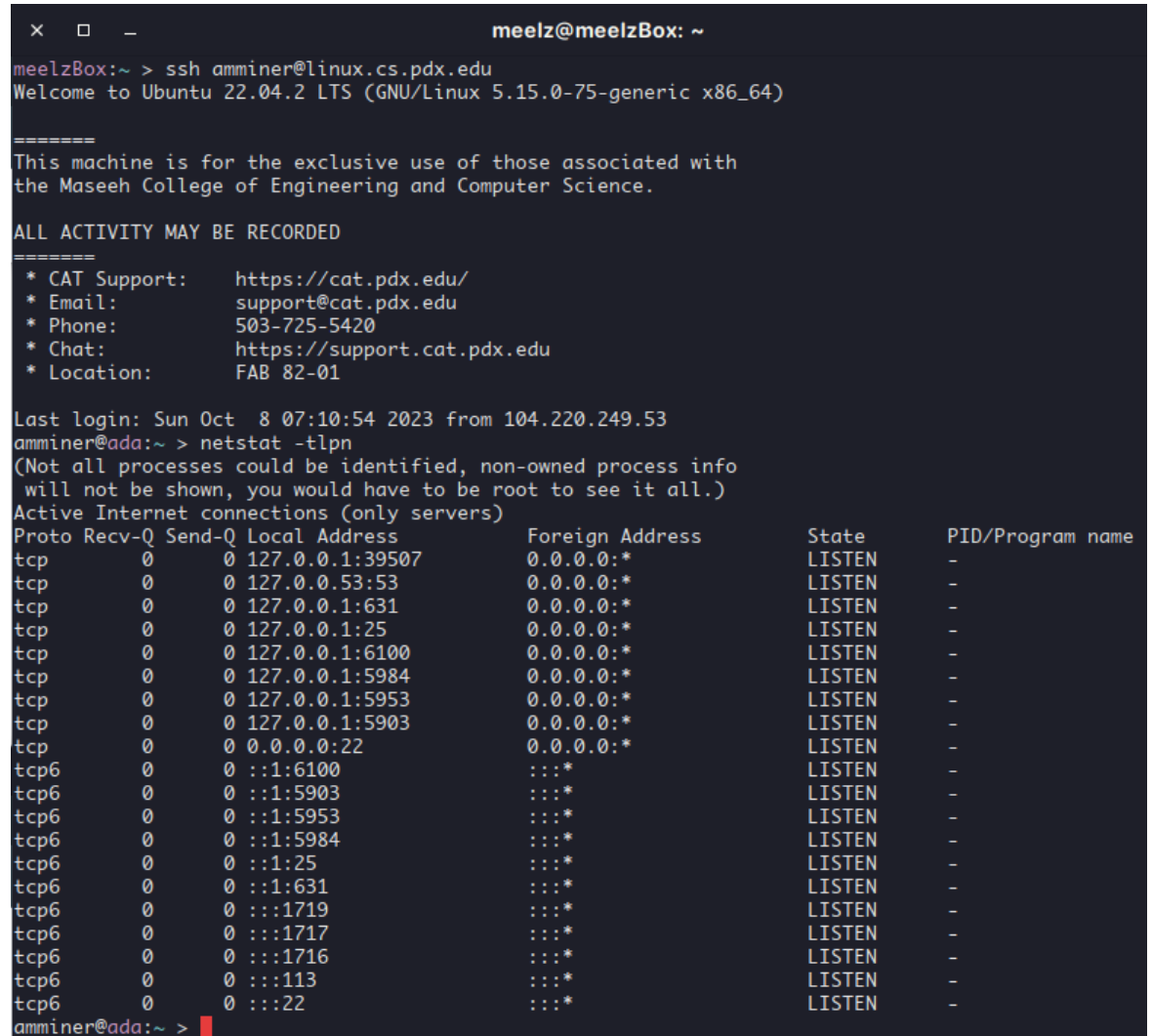
- Run the command using **sudo** and take a screenshot of the output to include in your lab notebook.

```
meelz(amminer)@course-vm: ~$ sudo netstat -tlnp
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 :::3350                  :::*                    LISTEN      560/xrdp-sesman
tcp6       0      0 :::3389                  :::*                    LISTEN      644/xrdp
tcp6       0      0 :::22                    :::*                    LISTEN      794/sshd: /usr/sbin
```

- 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.

- Login to linux.cs.pdx.edu
 - Run the command again, but do not use `sudo` as this is a machine managed by CAT. Include a screenshot of the output.



```
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	-

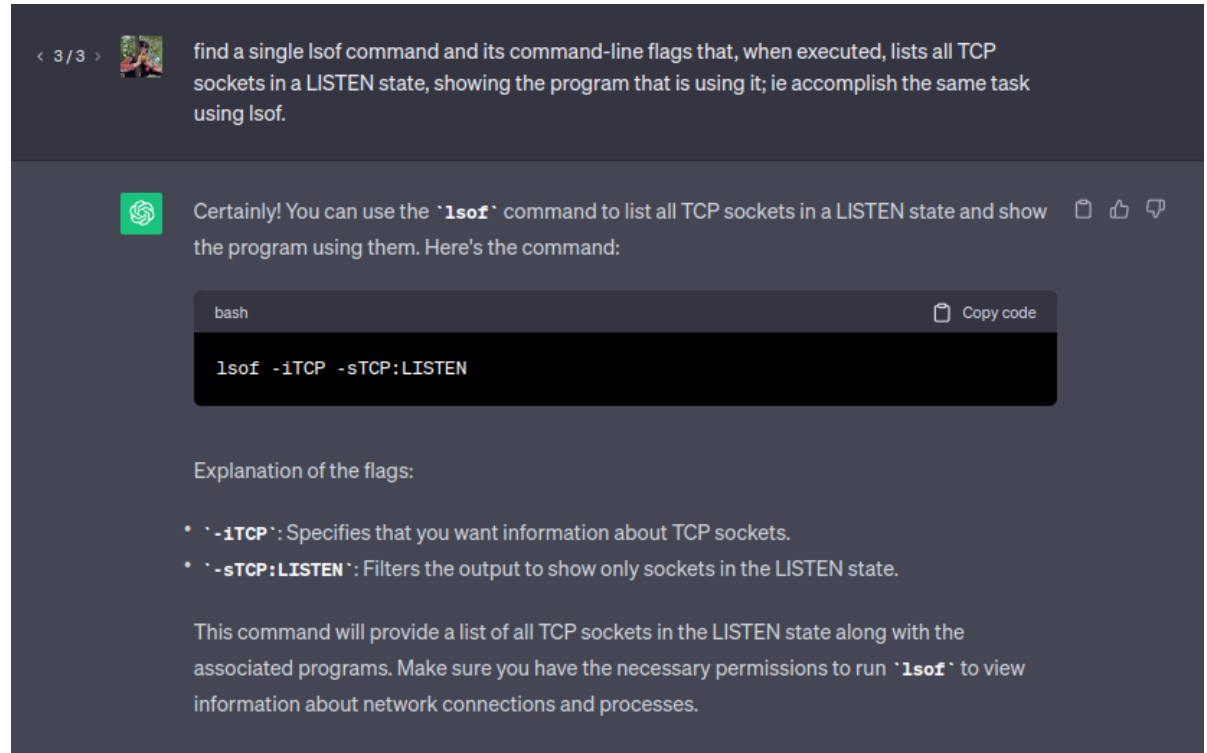
```
amminer@ada:~ >
```

- 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.

```
meelz(amminer)@course-vm:~$ sudo lsof -iTCP -sTCP:LISTEN
COMMAND PID  USER   FD  TYPE DEVICE SIZE/OFF NODE NAME
systemd-r 366 systemd-resolve 14u  IPv4 16321      0t0  TCP localhost:domain (LISTEN)
container 453    root      8u  IPv4 17389      0t0  TCP localhost:37149 (LISTEN)
xrdp-sesm 560    root      7u  IPv6 17910      0t0  TCP ip6-localhost:3350 (LISTEN)
xrdp      644    xrdp     11u  IPv6 17166      0t0  TCP *:ms-wbt-server (LISTEN)
sshd      794    root      3u  IPv4 17368      0t0  TCP *:ssh (LISTEN)
sshd      794    root      4u  IPv6 17370      0t0  TCP *:ssh (LISTEN)
meelz(amminer)@course-vm:~$
```

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 $addr; done
traceroute to 10.132.0.2 (10.132.0.2), 30 hops max, 60 byte packets
 1 vm-europe-west1-d.europe-west1-4.c.cloud-miner-amminer.internal (10.132.0.2)  135.685 ms  135.638 ms *
traceroute to 10.142.0.3 (10.142.0.3), 30 hops max, 60 byte packets
 1 * * *
 2 * * *
 3 * * *
 4 * * *
 5 vm-us-east1-b.us-east1-b.c.cloud-miner-amminer.internal (10.142.0.3)  65.189 ms * *
traceroute to 10.152.0.2 (10.152.0.2), 30 hops max, 60 byte packets
 1 vm-australia-southeast1-b.australia-southeast1-b.c.cloud-miner-amminer.internal (10.152.0.2)  149.845 ms * 149.791
amminer@vm-us-west1-b:~$ 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 $addr | wc -l; done
2
2
2
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr | wc -l; done
2
2
2
amminer@vm-us-west1-b:~$ for addr in 10.132.0.2 10.142.0.3 10.152.0.2; do traceroute $addr | wc -l; done
2
2
2
amminer@vm-us-west1-b:~$
```

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, indicating they are the initial requests. The table below summarizes the data for 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, with the first request (307) starting at approximately 1000 ms and the second (301) starting at approximately 1500 ms. The third request (200) is still pending at the time of the screenshot.

- 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

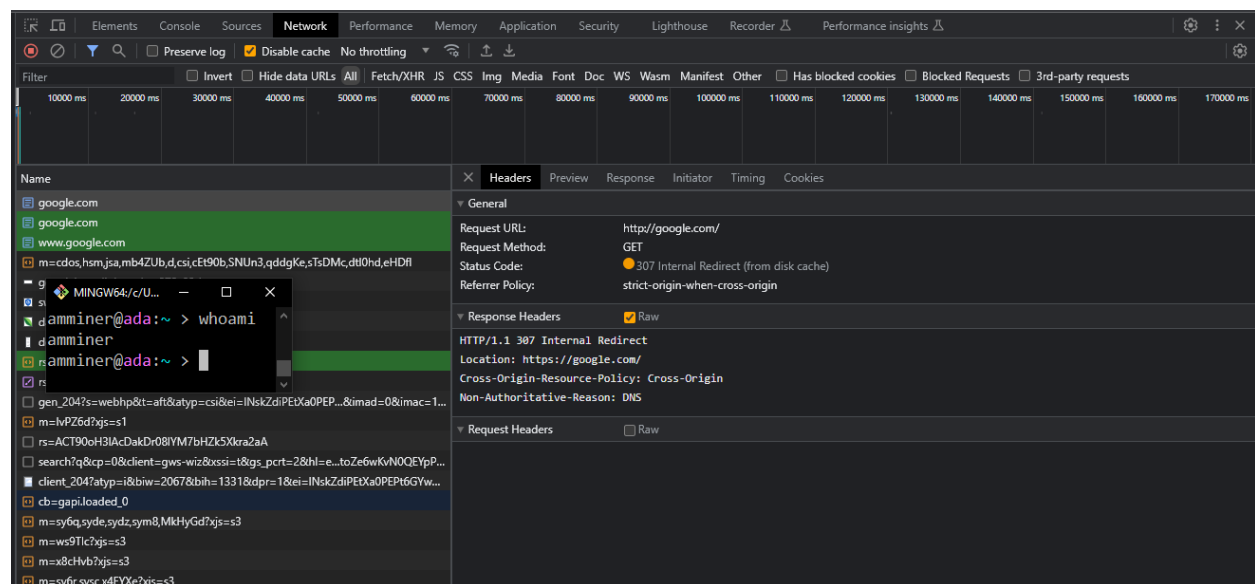
307-temporary redirect. The response includes a new address to which the resource has been “temporarily” moved.

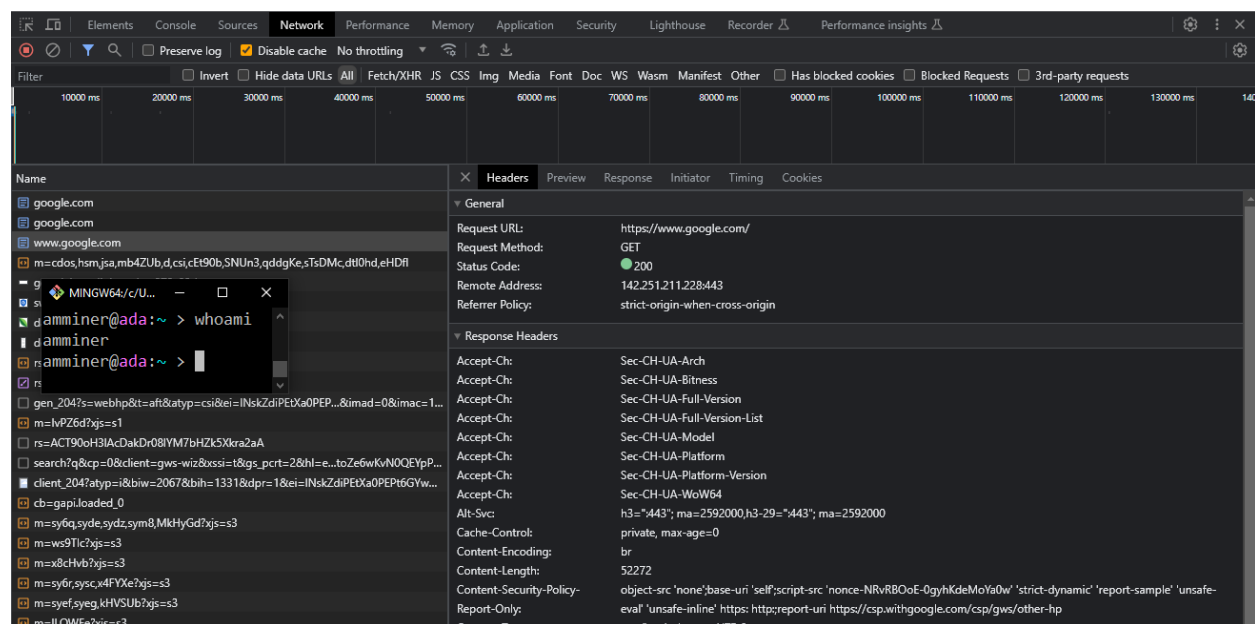
301-moved permanently. The response includes a new address to which the resource has been “permanently” moved.

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.





- <https://google.com>

- <https://www.google.com/>

- The screenshot displays a web browser's developer tools interface. The top toolbar includes buttons for search, filter, and various network-related actions. The 'Network' tab is active, showing a list of requests. The selected request is 'set-cookie' from 'www.google.com'. The 'Headers' panel on the right shows the response headers for this request. The 'Response' panel on the right shows the response body, which is a 200 status code. The 'Console' panel at the bottom shows a terminal window with a 'whoami' command being executed, returning 'amminer@ada:~'.

Network Tab:

 - Filter: ☐ Invert ☐ Hide data URLs ☐ All ☐ Fetch/XHR ☐ JS ☐ CSS ☐ Img ☐ Media ☐ Font ☐ Doc ☐ WS ☐ Wasm ☐ Manifest ☐ Other ☐ Has blocked cookies ☐ Blocked Requests
 - client_204 - www.google.com/client_204...
set-cookie: 1P_JAR=2023-10-10-05; ...
 - gen_204 - www.google.com/gen_204...
set-cookie: 1P_JAR=2023-10-10-05; ...
 - hpbpa - www.google.com/async/hpbpa...
set-cookie: 1P_JAR=2023-10-10-05; ...
 - www.google.com/ - www.google.com...
set-cookie: 1P_JAR=2023-10-10-05; ...

Headers Panel:

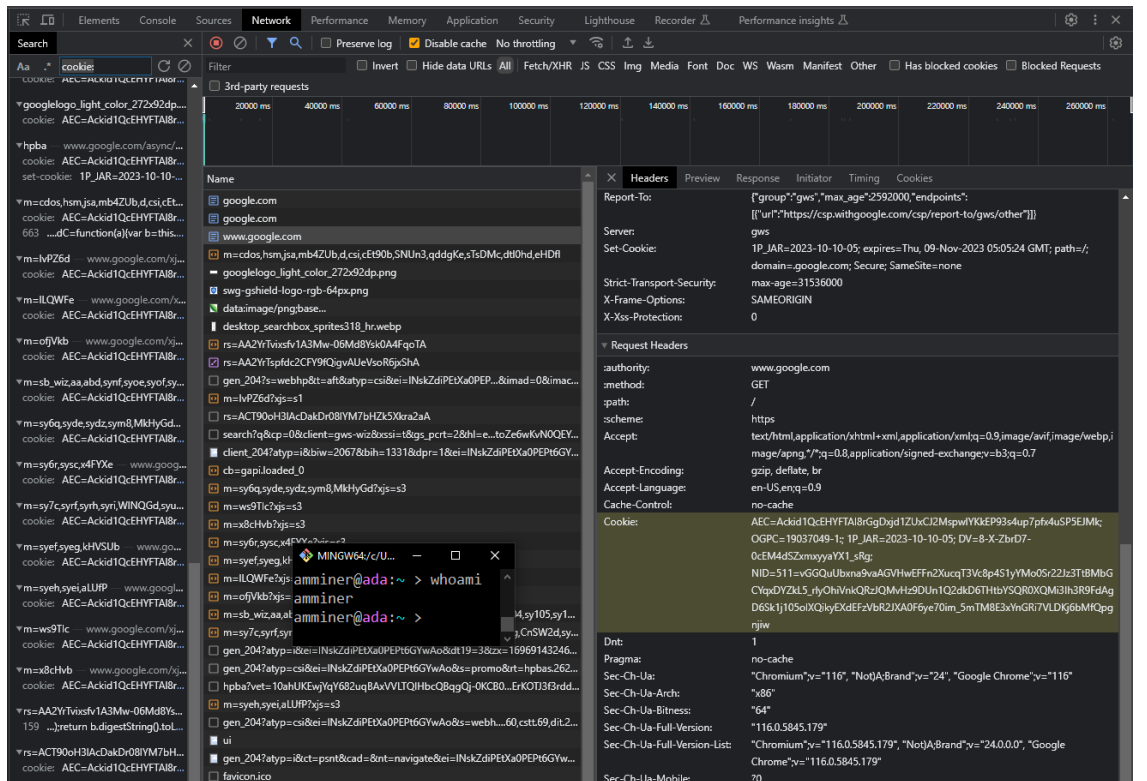
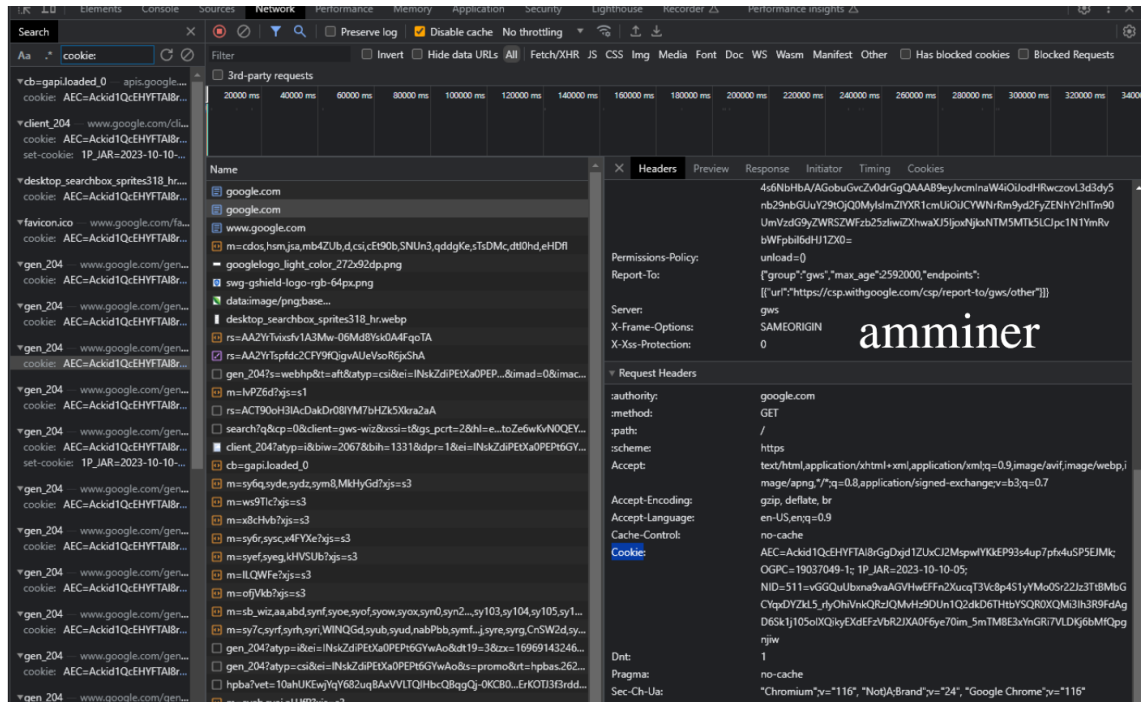
 - Status Code: 200
 - Remote Address: 142.251.2.11228:443
 - Referrer Policy: strict-origin-when-cross-origin
 - Response Headers:
 - Accept-Ch: Sec-CH-UA-Arch
 - Accept-Ch: Sec-CH-UA-Bitness
 - Accept-Ch: Sec-CH-UA-Full-Version
 - Accept-Ch: Sec-CH-UA-Full-Version-List
 - Accept-Ch: Sec-CH-UA-Model
 - Accept-Ch: Sec-CH-UA-Platform
 - Accept-Ch: Sec-CH-UA-Platform-Version
 - Accept-Ch: Sec-CH-UA-WoW64
 - Alt-Svc: h3="443"; ma=2592000, h3-29="443"; ma=2592000
 - Cache-Control: private, max-age=0
 - Content-Encoding: br
 - Content-Length: 52272
 - Content-Security-Policy: object-src 'none'; base-uri 'self'; script-src 'nonce-NRyRBOe-0gYhKdeMoYa0w' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https://httpreporter.https://csp.withgoogle.com/csp/gws/other-hp
 - Report-Only: text/html; charset=UTF-8
 - Content-Type: true, 10 Oct 2023 05:05:24 GMT
 - Expires: -1
 - Origin-Trial: Ap-qNlnLzDKSMHJmSila908GuehlqGbb6eMESlkelh2QvZvf6zPmQ3l odoeuzPhaAolmhnP88w4AlAAABfeyJvcmlnaW4iOiJodHRwczovLjZ3d3Y5bnB2 9nbGUuY290Q0MyISMzYXR1cmUiOiJ0Z2UtaXNzaW9uc1BvbGJleVubG9h ZCUslmV4GlyeSj6MTYNTYXZMzk5O0=
 - Origin-Trial: AvudjZmZqL7335p1KL2VlHo1KxMeINodU115d0CPz9dovVLCeXk8QaQhio1DX 6n29nHbAAGobuGvzd0drgGQAAAB9eyJvcmlnaW4iOiJodHRwczovLjZ3d3Y5 bnB29nbGUuY290Q0MyISMzYXR1cmUiOiJ0Z2UtaXNzaW9uc1BvbGJleVubG9h ZCUslmV4GlyeSj6MTYNTYXZMzk5O0=
 - Permissions-Policy: unload=()
 - Report-To: {"group":"gws","max_age":2592000,"endpoints": [{"url":"https://csp.withgoogle.com/csp/report-to/gws/other"}]}
 - Server: gws
 - Set-Cookie: 1P_JAR=2023-10-10-05; expires=Thu, 09-Nov-2023 05:05:24 GMT; path=/; domain=.google.com; Secure; SameSite=none
 - Strict-Transport-Security: max-age=31536000

Console Panel:

```

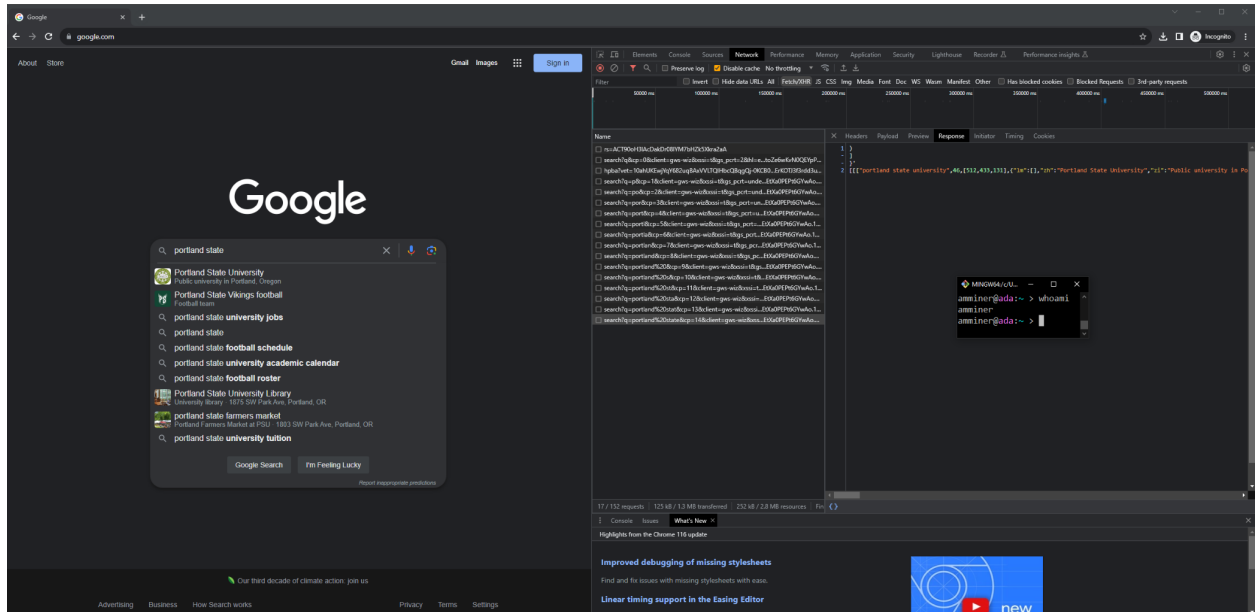
MINGW64/c/ U...
amminer@ada:~$ whoami
amminer@ada:~$
  
```

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



6. - Asynchronous HTTP requests

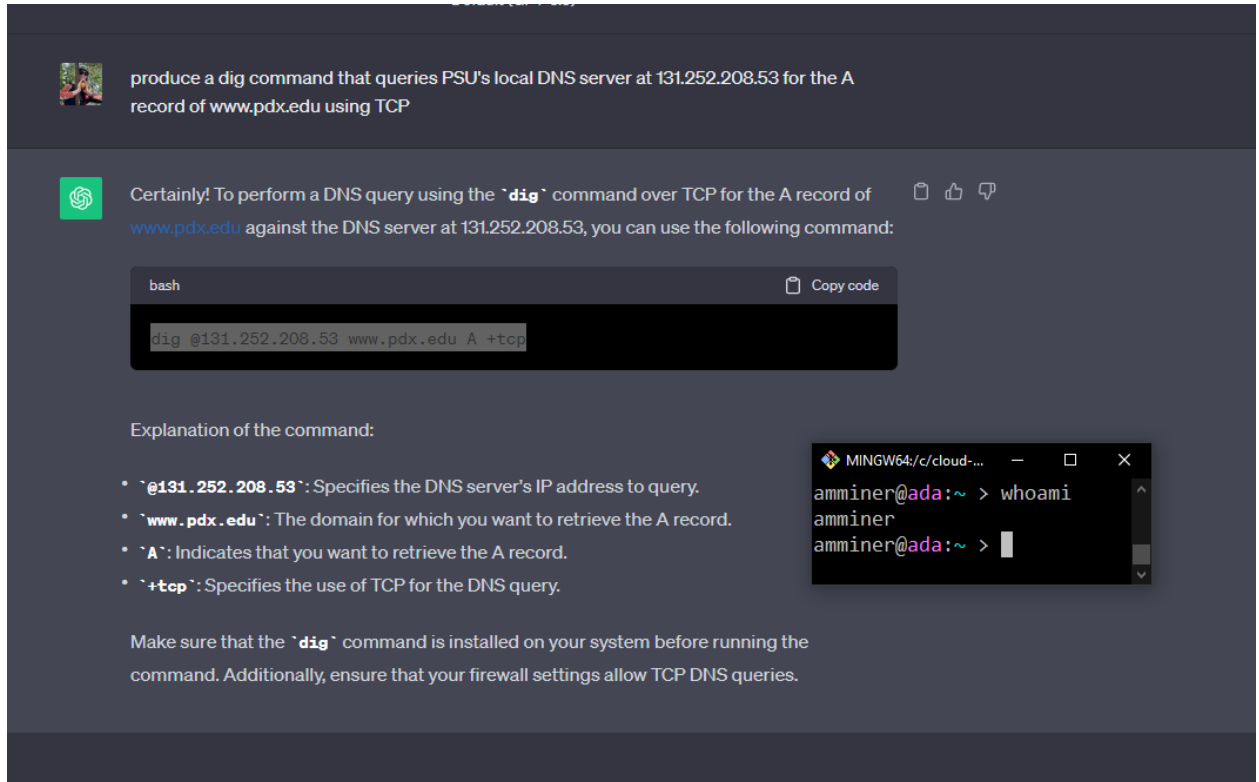
- Show the asynchronous HTTP requests made by typing in the search bar:



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.



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

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:

```
bash
```

```
dig @131.252.208.53 www.pdx.edu A +tcp
```

Explanation of the command:

- `@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.

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.



```
MINGW64:/c:/cloud-...  
amminer@ada:~ > whoami  
amminer  
amminer@ada:~ >
```

- 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?

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 and then query that server for the A record of 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:~ > addr=$(dig +short espn.go.com | egrep '([0-9]{1,3}\.){3}[0-9]{1,3}' | awk '{print $1}')  
amminer@ada:~ > echo $addr  
18.161.6.89 18.161.6.80 18.161.6.94 18.161.6.38  
amminer@ada:~ > for a in $addr; 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:~ >
```

One could automate this easily if the first and last manufacturers are known.

5. Geographic DNS #2

- Visit <https://www.iplocation.net/> and look up the geographical location of the following DNS servers: `131.252.208.53` and `198.82.247.66`. What geographic locations do ipinfo.io and DB-IP return?

ipinfo.io says Portland, OR. DB-IP is not on the results page.

- Then, using `dig`, resolve `www.google.com` from each of the DNS servers. As Google forward deploys its web content close to its users, its name will resolve differently in different geographic locations.

```
dig @<DNS_server_IP> www.google.com
```

- Record one address for `www.google.com` from each result for your lab notebook.

@131.252.208.53: `www.google.com` resolves to 142.251.215.228.

@198.82.247.66: `www.google.com` resolves to 142.251.163.106.

- What are the geographic coordinates of each DNS server and the IP address it resolves for `www.google.com`?

142.251.215.228 looks to be in Seattle.

142.251.163.106 looks to be in Mountain View, CA, but ipinfo.io seems to think it's in Ashburn, VA. My inclination is to expect google to obfuscate things through their mountain view facilities, so I would guess Ashburn, final answer...

- traceroute the DNS servers and the IPs you found for `google.com`:

```
MINGW64/c/cloud-miner-ammner
amminer@ada:~$ > for addr in 131.252.208.53 198.82.247.66 142.251.215.228 142.251.163.106; do traceroute $addr; done
traceroute to 131.252.208.53 (131.252.208.53), 30 hops max, 60 byte packets
 1 rdns.cat.pdx.edu (131.252.208.53) 0.573 ms 0.427 ms 0.290 ms
traceroute to 198.82.247.66 (198.82.247.66), 30 hops max, 60 byte packets
 1 radiant.seas.pdx.edu (131.252.208.212) 1.712 ms 1.688 ms 1.655 ms
 2 CORE1.net.pdx.edu (131.252.5.142) 3.177 ms 3.007 ms 2.845 ms
 3 131.252.5.213 (131.252.5.213) 0.539 ms 0.420 ms 0.363 ms
 4 port-psu-pe-01.net.linkoregon.org (199.165.177.48) 0.430 ms 0.334 ms 0.380 ms
 5 eugn-oh-vpn-01.net.linkoregon.org (207.98.126.3) 12.352 ms eugn-oh-pe-01.net.linkoregon.org (207.98.126.55) 12.330 ms eugn-oh-vpn-01.net.linkoregon.org (207.98.126.3) 12.119 ms
 6 eugn-oh-pe-02.net.linkoregon.org (207.98.126.57) 12.358 ms 12.321 ms 12.538 ms
 7 bois-gtwy-pe-01-loren.net.linkoregon.org (163.253.5.65) 12.285 ms 12.190 ms 12.076 ms
 8 hundredge-0-0-0-24.703.core1.bois.net.internet2.edu (163.253.5.64) 13.759 ms 13.632 ms 13.571 ms
 9 fourhundredge-0-0-0-0.4079.core2.salt.net.internet2.edu (163.253.1.249) 66.709 ms 69.047 ms 68.960 ms
10 fourhundredge-0-0-0-23.4079.core1.salt.net.internet2.edu (163.253.1.32) 66.818 ms fourhundredge-0-0-0-0.4079.core2.denv.net.internet2.edu (163.253.1.168) 69.122 ms 69.063 ms
11 fourhundredge-0-0-0-0.4079.core1.denv.net.internet2.edu (163.253.1.170) 68.312 ms 68.237 ms fourhundredge-0-0-0-0.4079.core2.kans.net.internet2.edu (163.253.1.251) 67.583 ms
12 fourhundredge-0-0-0-22.4079.core1.kans.net.internet2.edu (163.253.1.54) 67.573 ms 67.496 ms fourhundredge-0-0-0-0.4079.core1.kans.net.internet2.edu (163.253.1.243) 67.430 ms
13 fourhundredge-0-0-0-3.4079.core2.chic.net.internet2.edu (163.253.1.244) 68.083 ms 68.012 ms 68.002 ms
14 fourhundredge-0-0-0-3.4079.core2.eqch.net.internet2.edu (163.253.2.19) 69.118 ms 66.444 ms 67.553 ms
15 fourhundredge-0-0-0-0.4079.core2.clev.net.internet2.edu (163.253.2.16) 66.832 ms 66.742 ms 67.257 ms
16 fourhundredge-0-0-0-3.4079.core2.ashb.net.internet2.edu (163.253.1.138) 65.865 ms 68.474 ms 68.417 ms
17 192.122.175.14 (192.122.175.14) 65.548 ms 65.447 ms 65.430 ms
18 vtacs-1.msap.cns.vt.edu (192.70.187.18) 67.044 ms 66.938 ms 66.906 ms
19 lab-core-7-0-0-0.cns.vt.edu (128.173.0.202) 69.106 ms 69.026 ms 69.127 ms
20 cas-core-100.2000.cns.vt.edu (198.82.1.143) 68.280 ms 68.151 ms 68.082 ms
21 jeru.cns.vt.edu (198.82.247.66) 68.886 ms 68.705 ms 68.831 ms
traceroute to 142.251.215.228 (142.251.215.228), 30 hops max, 60 byte packets
 1 radiant.seas.pdx.edu (131.252.208.212) 1.022 ms 1.102 ms 1.794 ms
 2 CORE1.net.pdx.edu (131.252.5.142) 6.566 ms 6.479 ms 6.358 ms
 3 131.252.5.213 (131.252.5.213) 0.613 ms 0.496 ms 0.378 ms
 4 google.nwax.net (198.32.195.34) 4.091 ms 3.918 ms 3.857 ms
 5 108.170.245.97 (108.170.245.97) 5.432 ms 108.170.245.113 (108.170.245.113) 4.328 ms 108.170.245.97 (108.170.245.97) 5.188 ms
 6 216.239.56.223 (216.239.56.223) 4.596 ms 142.251.241.137 (142.251.241.137) 4.385 ms 4.313 ms
 7 sea09s35-in-f4.1e100.net (142.251.215.228) 3.888 ms 4.521 ms 4.611 ms
traceroute to 142.251.163.106 (142.251.163.106), 30 hops max, 60 byte packets
 1 radiant.seas.pdx.edu (131.252.208.212) 1.101 ms 2.032 ms 2.037 ms
 2 CORE1.net.pdx.edu (131.252.5.142) 0.872 ms 0.759 ms 0.648 ms
 3 131.252.5.213 (131.252.5.213) 0.845 ms 0.730 ms 0.618 ms
 4 google.nwax.net (198.32.195.34) 4.246 ms 3.865 ms 4.299 ms
 5 74.125.243.198 (74.125.243.198) 5.176 ms 74.125.243.189 (74.125.243.189) 4.813 ms 74.125.243.178 (74.125.243.178) 4.745 ms
 6 172.253.77.194 (172.253.77.194) 10.212 ms 216.239.50.20 (216.239.50.20) 11.493 ms 11.351 ms
 7 192.178.74.218 (192.178.74.218) 55.010 ms 192.178.74.216 (192.178.74.216) 95.606 ms 192.178.74.222 (192.178.74.222) 48.992 ms
 8 192.178.72.200 (192.178.72.200) 59.634 ms 192.178.72.195 (192.178.72.195) 58.157 ms 192.178.72.197 (192.178.72.197) 58.600 ms
 9 * * 192.178.81.238 (192.178.81.238) 73.298 ms
10 142.251.66.133 (142.251.66.133) 72.812 ms 172.253.67.44 (172.253.67.44) 73.704 ms 216.239.57.97 (216.239.57.97) 71.219 ms
11 142.250.235.95 (142.250.235.95) 72.105 ms 71.306 ms 142.250.235.87 (142.250.235.87) 74.728 ms
12 * * *
13 * * *
14 * * *
15 * * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 vv-in-f106.1e100.net (142.251.163.106) 70.114 ms * 72.549 ms
amminer@ada:~$ >
amminer@ada:~$ >
```

6. Wireshark Lab #3

- RDP into your gcloud linux VM. In a terminal, using commands from prior labs, find the addresses and interfaces on the VM. Make a note of:
 - The IP address of the VM

10.138.0.2

- The name of the local virtual ethernet interface

eth0

- The IP address of the default router

10.138.0.1

7. -

- Capture ICMP on eth0 and send 3 pings to www.google.com.
- Click on the first packet in the top window of the Wireshark UI. Then, in the middle window, expand the data-link layer packet and click on the destination hardware addresses. See which bytes in the payload window this corresponds to.

The screenshot shows a Kali Linux virtual machine. In the foreground, the Wireshark network protocol analyzer is running, capturing traffic on the eth0 interface. The packet list pane shows three ICMP Echo (ping) requests and their corresponding replies. The packet details pane for the first packet (No. 1) is expanded, showing the Ethernet II header with the destination MAC address 42:01:0a:8a:00:01. The packet bytes pane shows the raw data of the frame, with the destination MAC address highlighted in blue. In the background, a terminal window shows the results of a ping command to www.google.com, indicating that the ping was successful.

- Does the destination MAC address correspond to an interface on the VM, an interface on the default router or an interface on Google's web site?

The destination MAC is the MAC of the default router:

```
amminer@kali-vm:~$ netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask        Flags   MSS Window  irtt Iface
0.0.0.0          10.138.0.1     0.0.0.0        UG      0 0        0 eth0
10.138.0.1       0.0.0.0        255.255.255.255 UH      0 0        0 eth0
172.17.0.0       0.0.0.0        255.255.0.0    U       0 0        0 docker0
amminer@kali-vm:~$ arp -a 10.138.0.1
? (10.138.0.1) at 42:01:0a:8a:00:01 [ether] on eth0
amminer@kali-vm:~$
```

- Click on the next packet in the trace. Does the destination MAC address correspond to an interface on the VM, an interface on the default router or an interface on Google's

web site?

The next packet is another ping, same destination address. There are 3 pings and 3 replies. The replies have a destination MAC address corresponding with my VM's eth0 interface.

8. Network Recap Lab #4

- Find the IP address of <OdinId>.oregonctf.org, replacing <OdinId> with your OdinId

```
amminer@kali-vm:~$ dig amminer.oregonctf.org

; <<>> DiG 9.19.17-1-Debian <<>> amminer.oregonctf.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30593
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

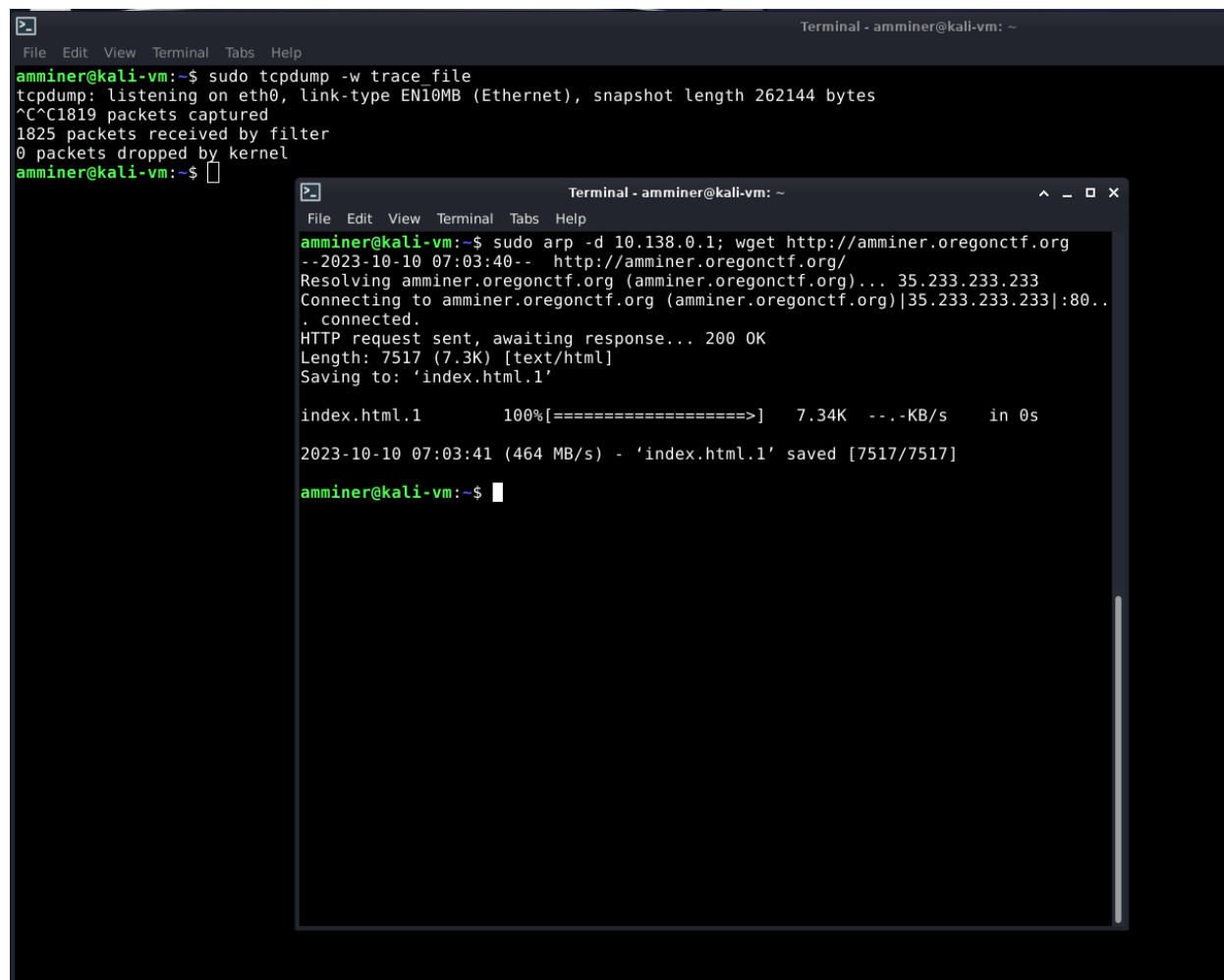
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;amminer.oregonctf.org.      IN      A

;; ANSWER SECTION:
amminer.oregonctf.org.  3600    IN      A      35.233.233.233

;; Query time: 127 msec
;; SERVER: 169.254.169.254#53(169.254.169.254) (UDP)
;; WHEN: Tue Oct 10 06:55:36 UTC 2023
;; MSG SIZE  rcvd: 66

amminer@kali-vm:~$ arp -an
? (10.138.0.1) at 42:01:0a:8a:00:01 [ether] on eth0
amminer@kali-vm:~$ sudo arp -d 10.138.0.1; arp -an
amminer@kali-vm:~$
```

9. Collect trace



```
amminer@kali-vm:~$ sudo tcpdump -w trace file
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C^C1819 packets captured
1825 packets received by filter
0 packets dropped by kernel
amminer@kali-vm:~$

amminer@kali-vm:~$ sudo arp -d 10.138.0.1; wget http://amminer.oregonctf.org
--2023-10-10 07:03:40-- http://amminer.oregonctf.org/
Resolving amminer.oregonctf.org (amminer.oregonctf.org)... 35.233.233.233
Connecting to amminer.oregonctf.org (amminer.oregonctf.org)|35.233.233.233|:80..
. connected.
HTTP request sent, awaiting response... 200 OK
Length: 7517 (7.3K) [text/html]
Saving to: 'index.html.1'

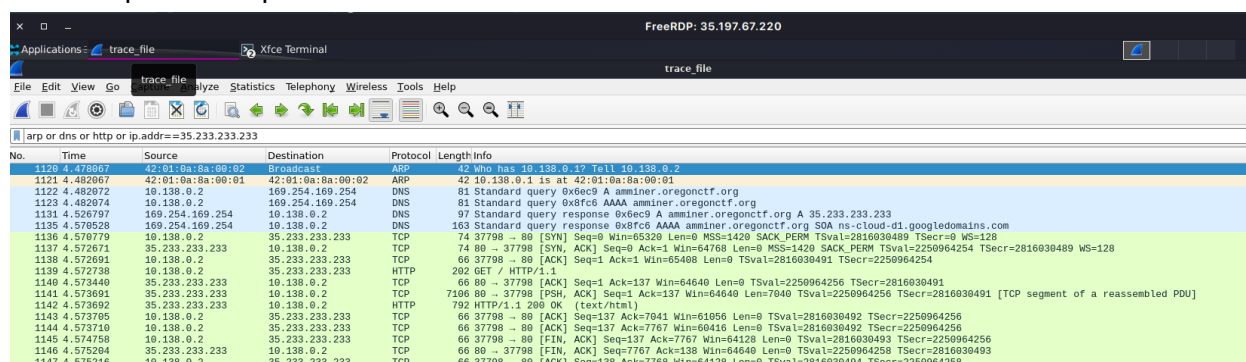
index.html.1      100%[=====] 7.34K  --.-KB/s   in 0s

2023-10-10 07:03:41 (464 MB/s) - 'index.html.1' saved [7517/7517]

amminer@kali-vm:~$
```

10. Analyze trace

- Open the cap in wireshark and filter as shown:



No.	Time	Source	Destination	Protocol	Length	Info
1120	4.482067	42:01:0a:8a:00:02	Broadcast	ARP	42	Who has 10.138.0.1? Tell 10.138.0.2
1121	4.482067	42:01:0a:8a:00:01	42:01:0a:8a:00:02	ARP	42	10.138.0.1 is at 42:01:0a:8a:00:01
1122	4.482072	10.138.0.2	169.254.169.254	DNS	81	Standard query 0x6ec9 A amminer.oregonctf.org
1123	4.482074	10.138.0.2	169.254.169.254	DNS	81	Standard query response 0x0fc6 AAAA amminer.oregonctf.org
1131	4.526797	169.254.169.254	10.138.0.2	DNS	97	Standard query response 0x6ec9 A amminer.oregonctf.org A 35.233.233.233
1135	4.570528	169.254.169.254	10.138.0.2	DNS	163	Standard query response 0x8fc6 AAAA amminer.oregonctf.org SOA ns-cloud-d1.google.com
1136	4.570779	10.138.0.2	35.233.233.233	TCP	74	37798 -> 80 [SYN] Seq=0 Win=65536 Len=0 MSS=1420 SACK_PERM TSval=2816030489 TSecr=0 WS=128
1137	4.572671	35.233.233.233	10.138.0.2	TCP	74	80 -> 37798 [SYN, ACK] Seq=0 Ack=1 Win=64768 Len=0 MSS=1420 SACK_PERM TSval=2250964254 TSecr=2816030489 WS=128
1138	4.572691	10.138.0.2	35.233.233.233	TCP	66	37798 -> 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2816030491 TSecr=2250964254
1139	4.572738	10.138.0.2	35.233.233.233	HTTP	202	GET / HTTP/1.1
1140	4.573440	35.233.233.233	10.138.0.2	TCP	66	80 -> 37798 [ACK] Seq=1 Ack=137 Win=64640 Len=0 TSval=2250964256 TSecr=2816030491
1141	4.573691	35.233.233.233	10.138.0.2	TCP	7106	80 -> 37798 [PSH, ACK] Seq=1 Ack=137 Win=64640 Len=7040 TSval=2250964256 TSecr=2816030491 [TCP segment of a reassembled PDU]
1142	4.573692	35.233.233.233	10.138.0.2	HTTP	792	HTTP/1.1 200 OK (text/html)
1143	4.573705	10.138.0.2	35.233.233.233	TCP	66	37798 -> 80 [ACK] Seq=137 Ack=7041 Win=61056 Len=0 TSval=2816030492 TSecr=2250964256
1144	4.573710	10.138.0.2	35.233.233.233	TCP	66	37798 -> 80 [ACK] Seq=137 Ack=7767 Win=60416 Len=0 TSval=2816030492 TSecr=2250964256
1145	4.574758	10.138.0.2	35.233.233.233	TCP	66	37798 -> 80 [FIN, ACK] Seq=137 Ack=7767 Win=64128 Len=0 TSval=2816030493 TSecr=2250964256
1146	4.575204	35.233.233.233	10.138.0.2	TCP	66	80 -> 37798 [FIN, ACK] Seq=7767 Ack=138 Win=64640 Len=0 TSval=2250964258 TSecr=2816030493
1147	4.575216	10.138.0.2	35.233.233.233	TCP	66	37798 -> 80 [ACK] Seq=138 Ack=7768 Win=64128 Len=0 TSval=2816030494 TSecr=2250964258

- ARP
 - What packet numbers in the trace are the result of the VM attempting to get the hardware address of the default router?
1120, 1121
 - What is this hardware address?
42:01:0a:8a:00:01
- DNS
 - What packet numbers in the trace correspond to the DNS request for the web site?
1122, 1123
 - What is the IP address of the local DNS server being queried?
169.254.169.254
- TCP
 - What packet numbers in the trace correspond to the initial TCP handshake for the web request?
1136, 1137, 1138 (syn, syn/ack, ack)
 - How long does it take to perform the initial TCP handshake?
A little under 0.002 seconds.
- HTTP
 - What packet numbers in the trace correspond to the actual HTTP request and response?
1139 is the GET request and 1142 is the 200-OK response.
 - How long does it take to process the HTTP request after the handshake?
Just under 0.001 seconds.

11. Clean up

Done!