**CS 433 Assignment 1:**

Team Members:
Zeeshan Snehil Bhagat 20110242
Ayush Chaudhari 20110042

## Note:

- For questions 1 & 2, tcpreplay will take around 35-40 seconds or at most 1 minute to play the pcap files. After that, we need to **`Ctrl + C`** in the terminal, where we are running the code.
- For images, if you cannot see them properly, do zoom in.

## Question 1

We first have to open two terminal windows.
In the first terminal window, run the command

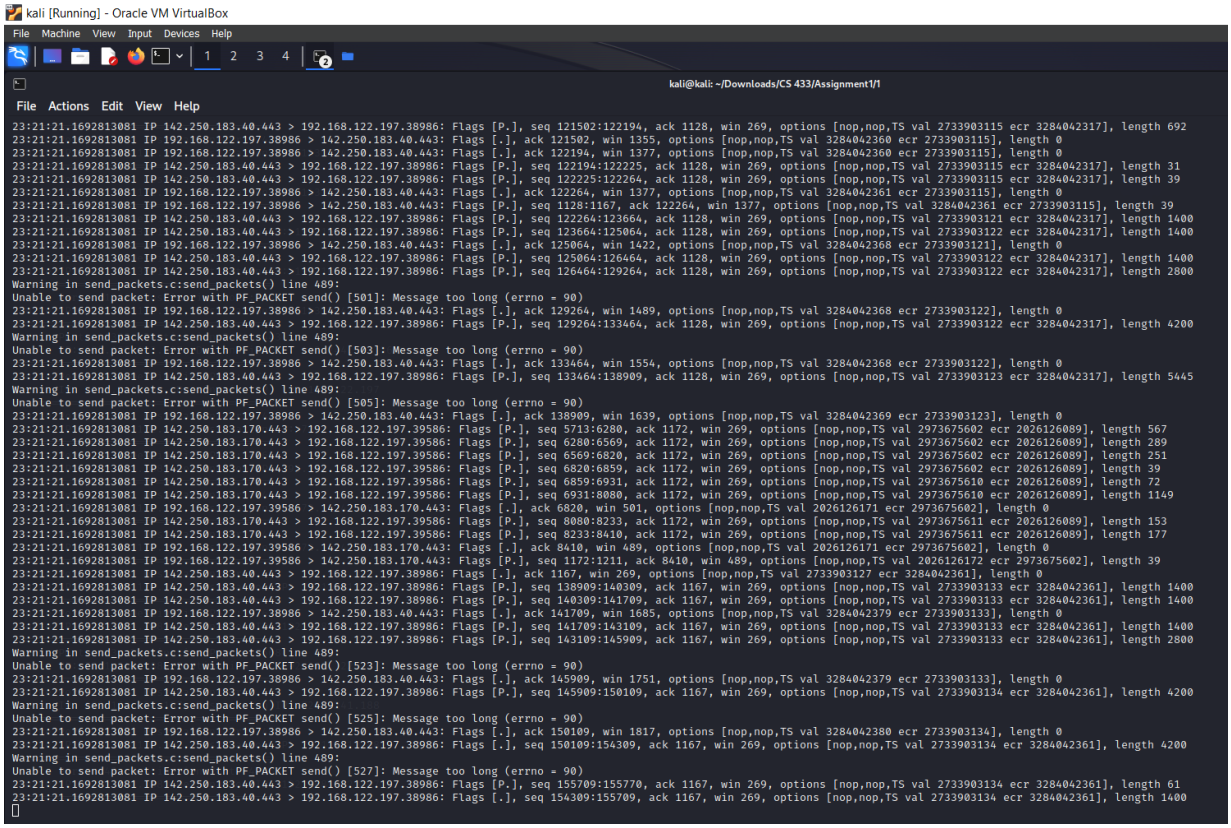
```
sudo python part1.py
```

Then, in the second terminal, immediately, run the command

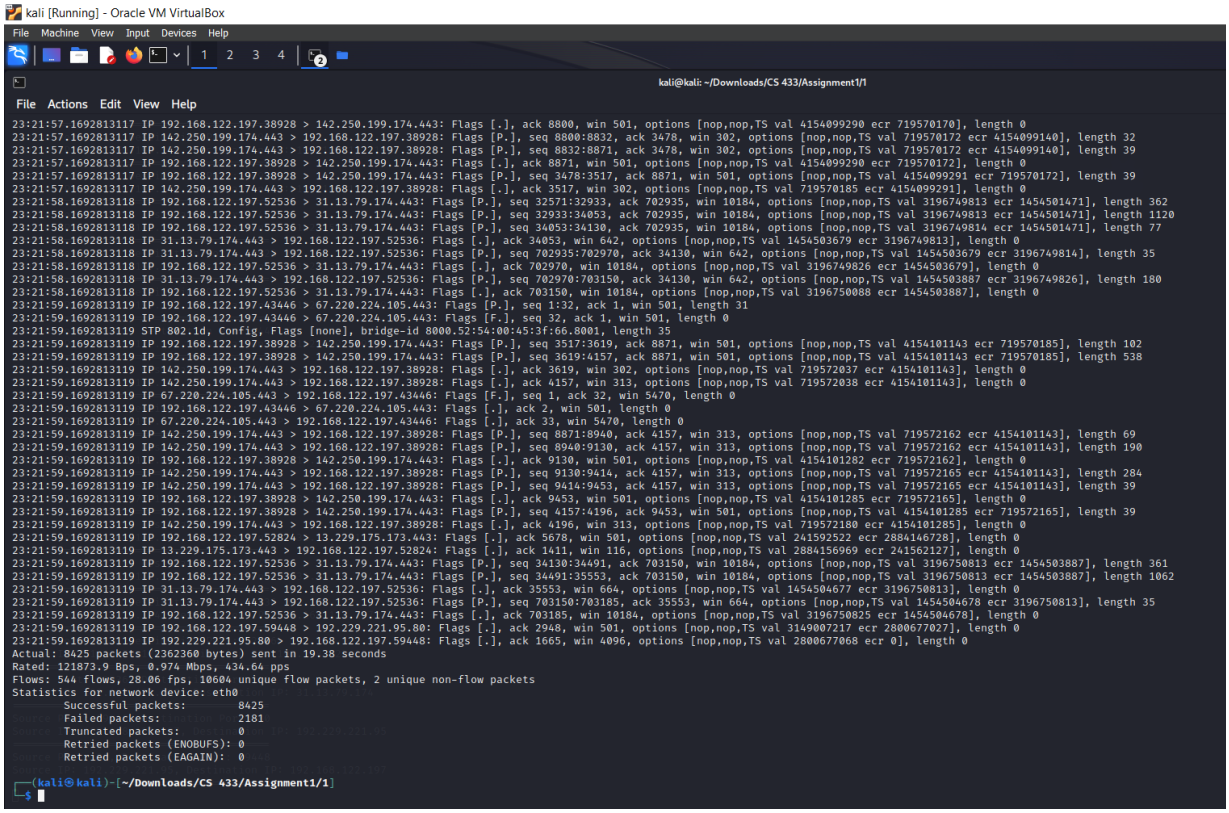
```
tcpreplay -i eth0 --mbps 1 -v <path_to_pcap_file>
```

For us, the network interface was **eth0**. Also, we kept the speed parameter at 1.
Make sure to give the proper path to the pcap file.
<result.pcap> is calculated as (<roll_no_of_member_1> + <roll_no_of_member_2>) % 3.
roll_no_of_member_1 -> 20110042
roll_no_of_member_2 -> 20110242
Hence, we get the file 1.pcap for ourselves. It is in the Part 1 folder.

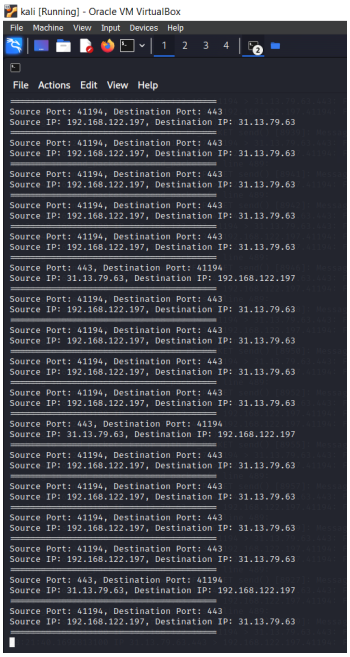
Ensure the browser and other internet-demanding software are closed as much as possible, but keep the internet on (for DNS lookup). Do **`Ctrl + C`** on the terminal containing Python output as soon as tcpreplay stops.
We can observe TCP flows consisting of 4 tuples of (client IP, client port, server IP, server port) [Here, if we take the tuple as (client=source IP, client=source port, server=destination IP, server=destination port), then the tuple(client=destination IP, client=destination port, server=source IP, server=source port) is also considered the same as previous tuple, since in a stream of tcp flows, only one ip address can be server while the other ip address can be client and their roles cannot be reversed; no matter whether the client is source or destination]. In our screenshot, we can observe 175 unique tuples. We also captured all observed IPs and did reverse DNS lookup on them. (We tried to make sure to only display those whose domains were found.)



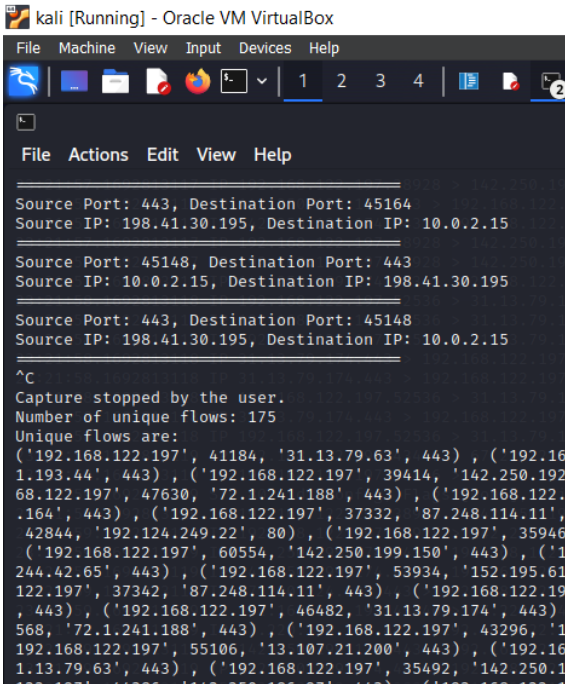

[In between while running the command **`tcpreplay -i eth0 --mbps 1 -v <path_to_pcap_file>`**]




[After the command **`tcpreplay -i eth0 --mbps 1 -v <path_to_pcap_file>`** has finished running]

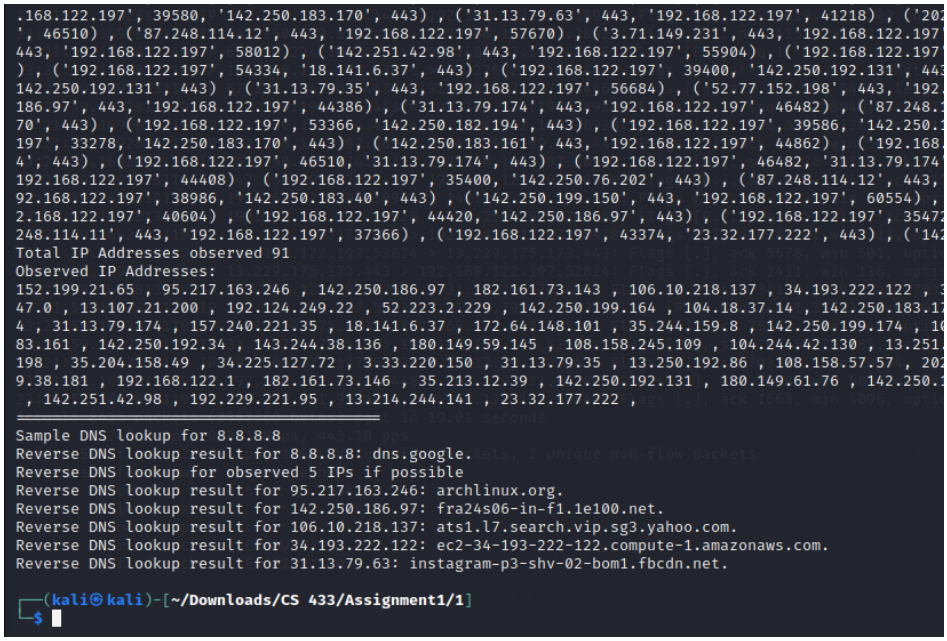



[In between while running the command **`sudo python part1.py`**]




[After the command **`sudo python part1.py`** has finished running]

In the image, we can see the number of unique flows captured and their IPs.




These are all the observed IPs and Reverse DNS lookups for some of the observed IPs.

## Question 2

We first have to open two terminal windows.

In the first terminal window, run the command

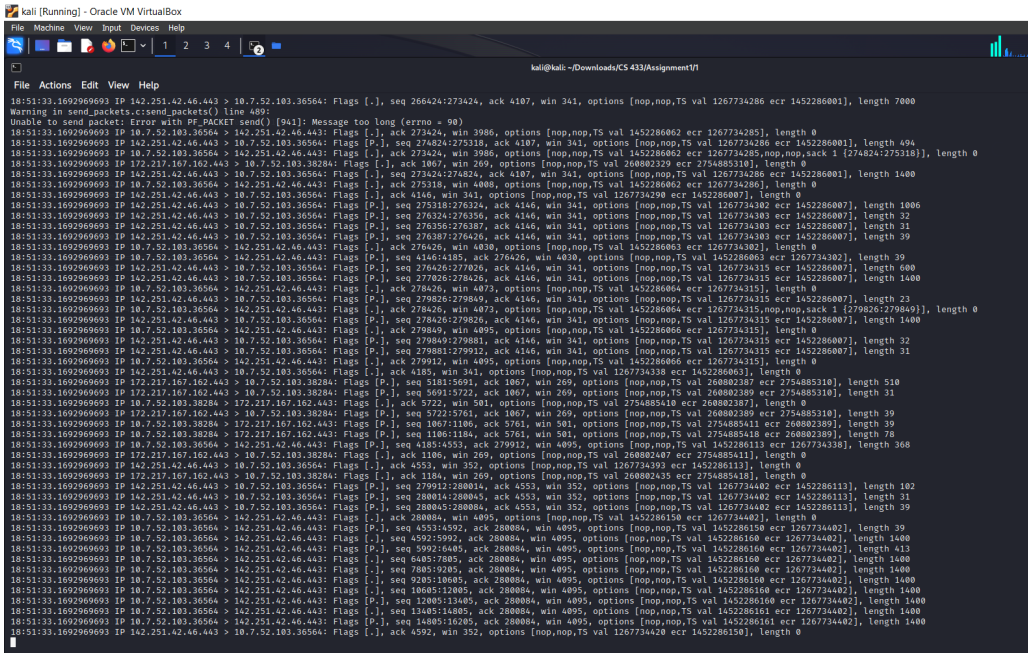
**`sudo python part2.py`**

Then, in the second terminal, immediately, run the command

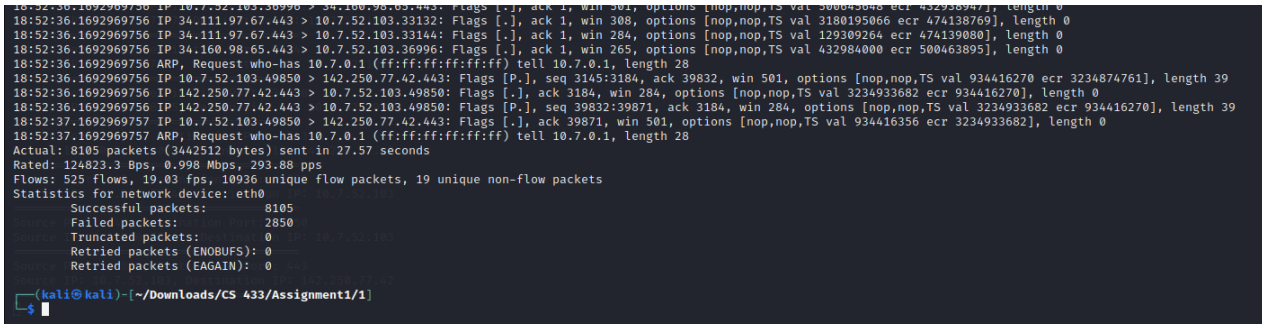
**`tcpreplay -i eth0 --mbps 1 -v <path_to_pcap_file>`**

For us, the network interface was **eth0**. Also, we kept the speed parameter at 1.
Make sure to give the proper path to the pcap file.
<result.pcap> is calculated as (<roll_no_of_member_1> + <roll_no_of_member_2>) % 4.
roll_no_of_member_1 -> 20110042
roll_no_of_member_2 -> 20110242
Hence, we get the file 0.pcap . It is in the Part 2 folder.

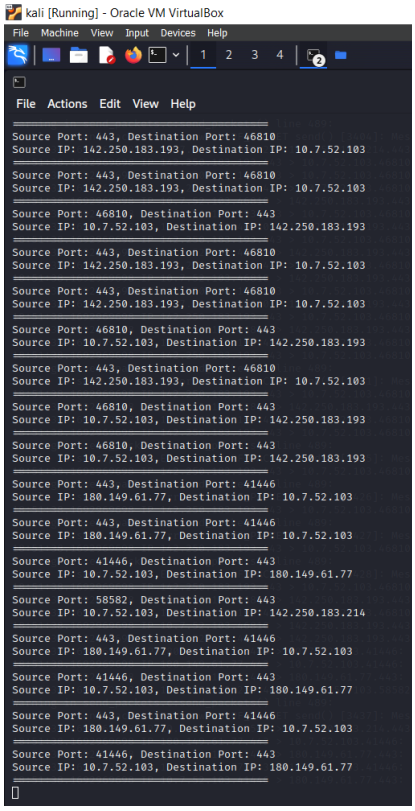
We ran our code multiple times and got the answers. Then we hardcoded those answers to find further answers.

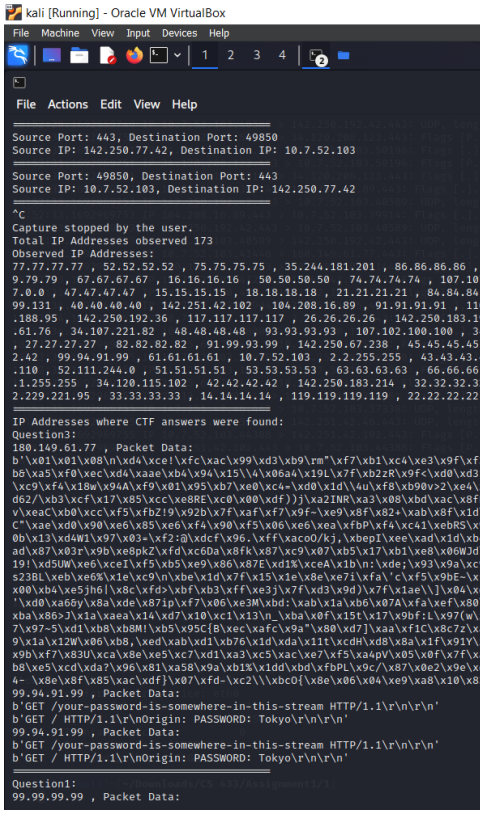



[Running tcpreplay for the pcap file]

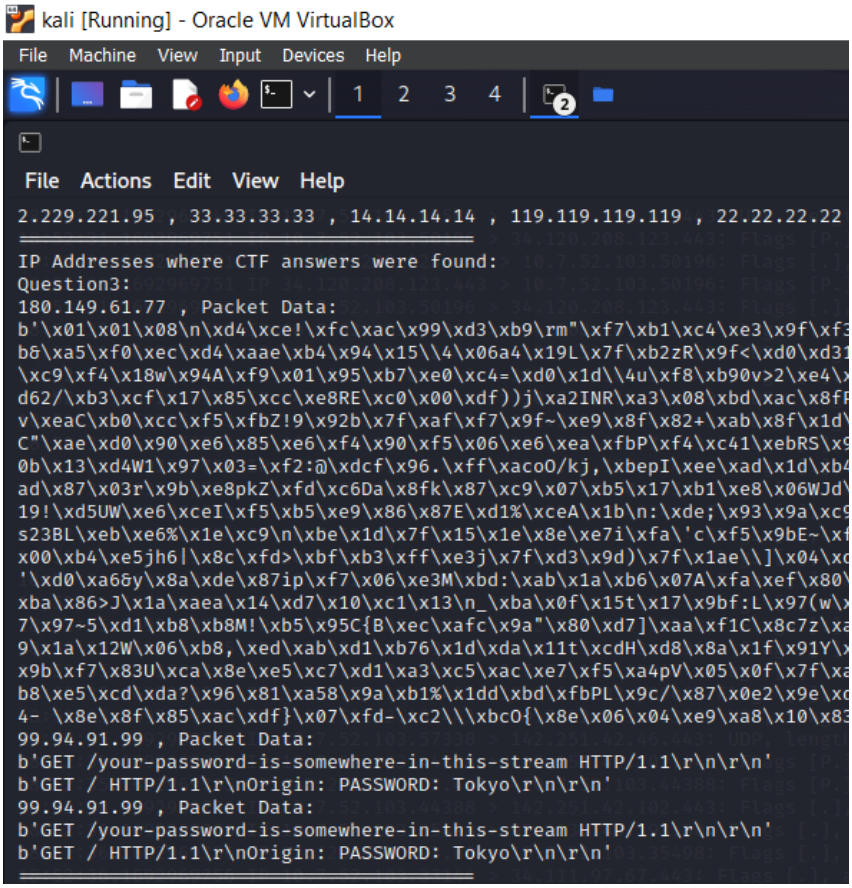


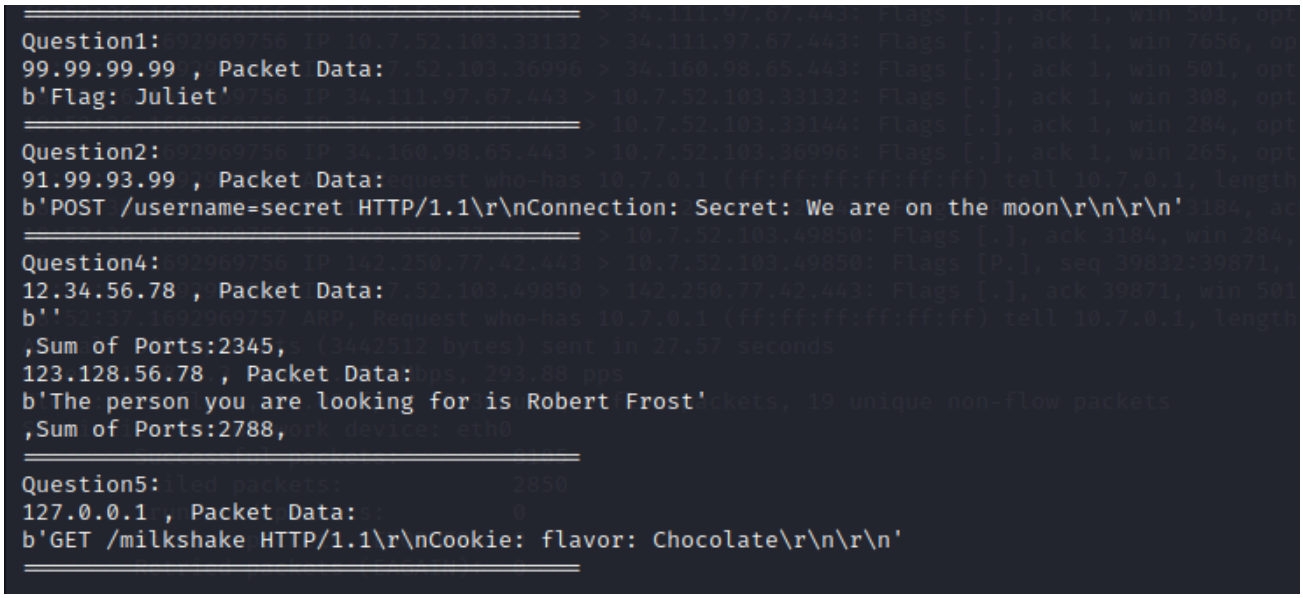

[After tcpreplay has finished running for the pcap file]




[List of IP addresses where CTF answers were found]




[CTF answers for the 5 questions]

**Answer 1:** Juliet
**Answer 2:** We are on the moon
**Answer 3:** Tokyo

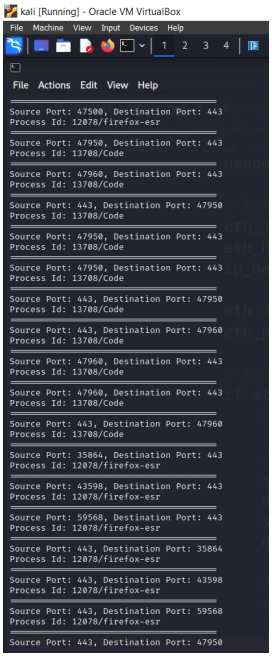
**Answer 4:** Robert Frost
**Answer 5:** Chocolate

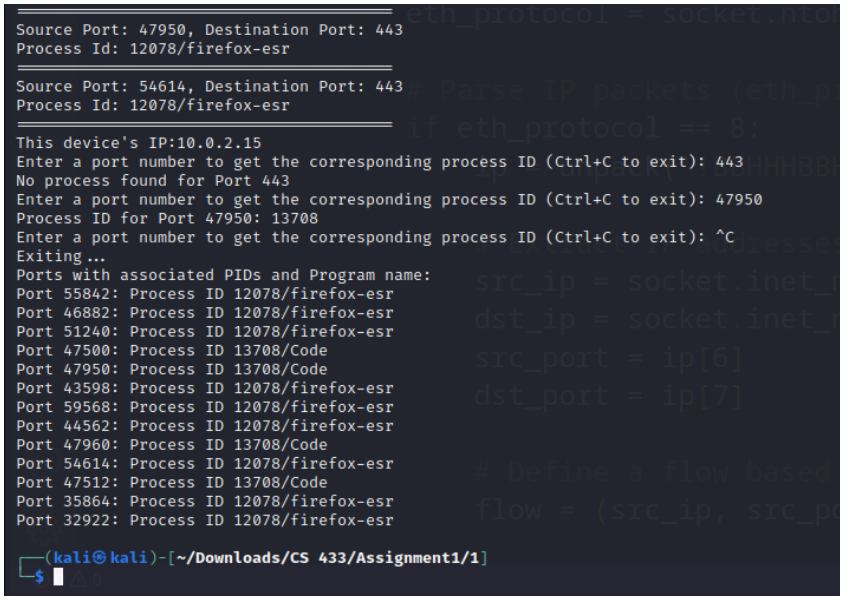
## Question 3

In the terminal window, run the command
**`sudo python part3.py`**

As per the question requirement, we will sniff the packets only for a duration of 30 seconds and process them.

No need to play pcap files. Just open any software such as browsers, IDEs and so on which can make our device create processes involving our local ips and ports. Or you can reload some pages on the browser.






[As per the question, after 30 seconds of packet processing, we enter any port number and get the process IDs accordingly]

After **`Ctrl + C`** for the final time (to exit "Enter a port number…" mode), it displays the process ID and program name of all the ports it had captured till then. You can also **`Ctrl + C`** before 30 seconds to enter the mode where we can find process IDs through port numbers.

## Question 4

1. Network Protocols captured in Wireshark are as follows:

**OCSP (Online Certificate Status Protocol):**

- Operation/Usage: OCSP checks digital certificates' validity and revocation status in real-time. It helps ensure the security of HTTPS connections by verifying that a certificate has not been revoked.
- Layer of Operation: OCSP operates at the OSI model's Application Layer (Layer 7).
- Associated RFC: RFC 6960

**ARP (Address Resolution Protocol):**

- While using Mozilla Firefox.
- Operation/Usage: ARP maps an IP address to a corresponding physical MAC (Media Access Control) address in a local network. It is essential for resolving IP addresses to their associated hardware addresses.
- Layer of Operation: ARP operates at the OSI model's Link Layer (Layer 2).
- Associated RFC: RFC 826

**TLSv1.2 (Transport Layer Security version 1.2):**

- While using Mozilla Firefox.

- Operation/Usage: TLSv1.2 is an older version of TLS that provides similar security features as TLSv1.3 but with some differences in encryption algorithms and security mechanisms.
- Layer of Operation: TLS operates between the OSI model's Transport Layer (Layer 4) and Application Layer (Layer 7).
- Associated RFC: RFC 5246

**TLSv1.3 (Transport Layer Security version 1.3):**

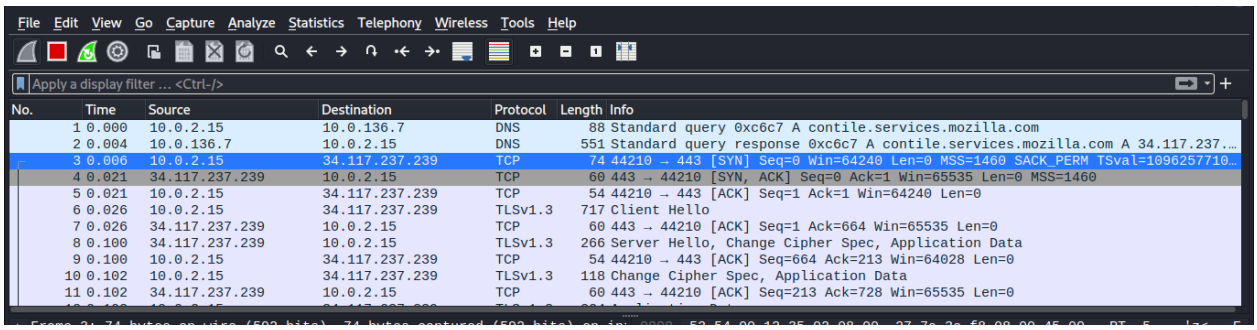
- While using Mozilla Firefox.
- Operation/Usage: TLSv1.3 is a cryptographic protocol that secures communication over the internet. It provides encryption, authentication, and data integrity for secure web browsing, email, and other applications.
- Layer of Operation: TLS operates between the OSI model's Transport Layer (Layer 4) and Application Layer (Layer 7).
- Associated RFC: RFC 8446

**ICMPv6 (Internet Control Message Protocol version 6):**

- While taking Snapshots.
- Operation/Usage: ICMPv6 is used for network error reporting, diagnostic messages, and network management in IPv6 networks. It serves functions like ping, traceroute, and neighbor discovery in IPv6.
- Layer of Operation: ICMPv6 operates at the OSI model's Network Layer (Layer 3).
- Associated RFC: RFC 4443

**RTCP (Real-Time Control Protocol):**

- While using Google Meet.
- Operation/Usage: RTCP works in conjunction with RTP (Real-Time Transport Protocol) to provide control and feedback for real-time multimedia streaming sessions. It allows participants in a session to monitor the quality of the transmission, synchronize streams, and facilitate other control functions necessary for effective real-time communication.
- Layer of Operation: RTCP operates at the Application Layer (Layer 7) of the OSI model.
- Associated RFC: RFC 3550



The RTT time is 0.021 - 0.006 = 0.015 ms.

2. The application layer protocols used when visiting the following websites are:
    1. **github.com:** HTTP/2
    2. **netflix.com:** HTTP/3
    3. **google.com:** HTTP/3

- HTTP/2 and HTTP/3 are both protocols used for transmitting data over the internet. HTTP/2 is the successor to HTTP/1.1, while HTTP/3 is the successor to HTTP/2.
- The major difference between HTTP/2 and HTTP/3 is the transport layer protocol used. HTTP/2 uses TCP (Transmission Control Protocol) to handle streams in the HTTP layer, while HTTP/3 uses QUIC (Quick UDP Internet Connections) as a transport layer to handle stream.
- HTTP/3 has a much quicker handshake to establish a secure session compared to HTTP/2, which achieves this using TCP and TLS (Transport Layer Security) 2. In addition, HTTP/3 offers 0-RTT support, which means that subsequent connections can start up much faster by eliminating the TLS acknowledgment from the server when setting up the connection. This means the client can start requesting data much quicker than with a full TLS negotiation, meaning the website starts loading earlier.
- Another advantage of HTTP/3 over HTTP/2 is that it is UDP-based. This means that if a packet gets dropped, it only interrupts that one stream, not all of them. In contrast, with HTTP/2, any interruption (packet loss) in the TCP connection blocks all streams (Head of line blocking).
- The similarities between HTTP/2 and HTTP/3 include:
    - Both protocols make use of the server push mechanisms.
    - They offer multiplexing which is made over a single connection via streams.
    - Resource prioritization is also done based on streams over the two protocols.
    - Both versions utilize header compression as HPACK and QPACK which are tied to packet order 2.

3. Name: The identifier of the cookie is "**AIT**."
    - Value: The specific value associated with the cookie is "ff2d794d29ef4b4addc9a43e4e955d97."
    - Created: The cookie was created on Thu, 07 Sep 2023, at 17:16:29 GMT.

- Domain: The cookie is associated with the domain "eoffice.iitgn.ac.in."
- Expires / Max-Age: The cookie expires on Sat, 07 Oct 2023, at 17:17:00 GMT.
- HostOnly: The cookie is specific to the domain "eoffice.iitgn.ac.in."
- HttpOnly: It is set as HttpOnly, meaning it cannot be accessed by client-side JavaScript.
- Last Accessed: The cookie was last accessed on Thu, 07 Sep 2023, at 17:17:00 GMT.
- Path: It's valid for all paths within the domain ("/").
- SameSite: The cookie attribute is set to "None," allowing it to be included in cross-origin requests.
- Secure: The cookie is marked as secure, indicating that it can only be sent over secure (HTTPS) connections.
- Size: The size of the cookie is 35 bytes.

Name: The identifier of the cookie is "**PHPSESSID**."

- Cookie is used by PHP.
- Functionality: To provide functions across pages.
- Value: The specific value associated with the cookie is "gcaslpp7d8dpsqdmk6c609pnmo."
- Created: The cookie was created on Thu, 07 Sep 2023, at 17:16:29 GMT.
- Domain: The cookie is associated with the domain "eoffice.iitgn.ac.in."
- Expires / Max-Age: It's a session cookie, meaning it expires when the browser session ends.
- HostOnly: The cookie is specific to the domain "eoffice.iitgn.ac.in."
- HttpOnly: It is not set as HttpOnly, so it can be accessed by client-side JavaScript.
- Last Accessed: The cookie was last accessed on Thu, 07 Sep 2023, at 17:16:29 GMT.
- Path: It's valid for all paths within the domain ("/").
- SameSite: The cookie attribute is set to "None," allowing it to be included in cross-origin requests.
- Secure: The cookie is not marked as secure, so it can be sent over non-HTTPS connections.
- Size: The size of the cookie is 35 bytes.

References:


- https://gist.github.com/DiabloHorn/fb43f3555f60454a729ddf66f642fefb
- https://github.com/ShyamsundarDas/packet-sniffers-in-python-using-raw-sockets
- *Black Hat Python: Python Programming for Hackers and Pentesters (2014),* https://apprize.best/python/black/3.html
- *socket — Low-level networking interface,* https://docs.python.org/3/library/socket.html
- *Raw Socket Programming in Python on Linux – Code Examples,* https://www.binarytides.com/raw-socket-programming-in-python-linux/
- *Packet sniffer in Python,* https://www.uv.mx/personal/angelperez/files/2018/10/sniffers_texto.pdf
- *Raw Sockets with Python: Sniffing and network packet injections.* https://medium.com/nerd-for-tech/raw-sockets-with-python-sniffing-and-network-packet-injections-486043061bd5
- *Online Certificate Status Protocol (OCSP),* FORTINET, https://www.fortinet.com/resources/cyberglossary/ocsp
- *System TLS enhancements to the TLSv1.3 and TLSv1.2 protocols,* IBM Support, https://www.ibm.com/support/pages/system-tls-enhancements-tlsv13-and-tlsv12-protocols
- *cookiedatabase*, https://cookiedatabase.org/cookie/php/phpsessid/
- *Comparing HTTP/3 vs. HTTP/2 Performance,* https://blog.cloudflare.com/http-3-vs-http-2/