CS 433 Computer Networks (2023-24)

Assignment-01

Student 1 **Name**: Gali Sunny, **Roll No:** 20110067
Student 2 **Name:** Balu Karthik Ram **Roll No:** 20110036

Part I: Packet capture statistics: We have used 0.Pcap file

a. Explanation of the code for capturing and processing TCP packets on a network interface.

Brief Explanation:

The code continuously captures the TCP packets, extracts relevant information and group packets into flows. It saves the flow details and packet data to a file (flows.txt) when interrupted by user (CTRL+C).

Global Variables:

1. packetsCaptured: Stores the count of TCP packets captured.
2. BufferFlow: An array of type ‘struct packet’ that stores necessary packet information.

Struct packet:

The struct packet is a data structure which has different attributes like source IP & port, destination IP & port, TCP checksum, data, and the length of data.

Main Function:

- The Program starts by creating a (‘raw socket’) to capture ethernet frames using socket function
- It enters the infinite loop where it listens for the incoming packets using recvfrom() function
- When a packet is received, it passes the buffer and size to the process_packet() function

Process_packet Function:

- It extract the ip address from the ethernet frame
- Determines the protocol if it is tcp or not (iph->protocol)
- Store the details of the source and destination ip address
- Extract the tcp header and store the source port, destination port , tcp_check_sum and tcp payload data
- Stores all the packet details in the BufferFlow array.

signalhandler() Function:

- This function is signal handler for SIGINT (CTRL+C)
- If prints the no of packets captured
- It opens a file named “flows.txt” to store packet information
- It iterates throught the captured packets and group them into flows.
- The information regarding each flow is stored in the flows.txt
- It then call the function Q2() which will find write answers to a specified file
- Finally it prints the no of flows captured and closes the file

findstringindata function:

- This function searches for a specific string within the payload data of captured packets.
- It iterates through all captured packets and checks if the specified string is present in the packet's payload.
- If a match is found, the payload data is written to a specified file.

Q2_3 and Q2_4 functions:

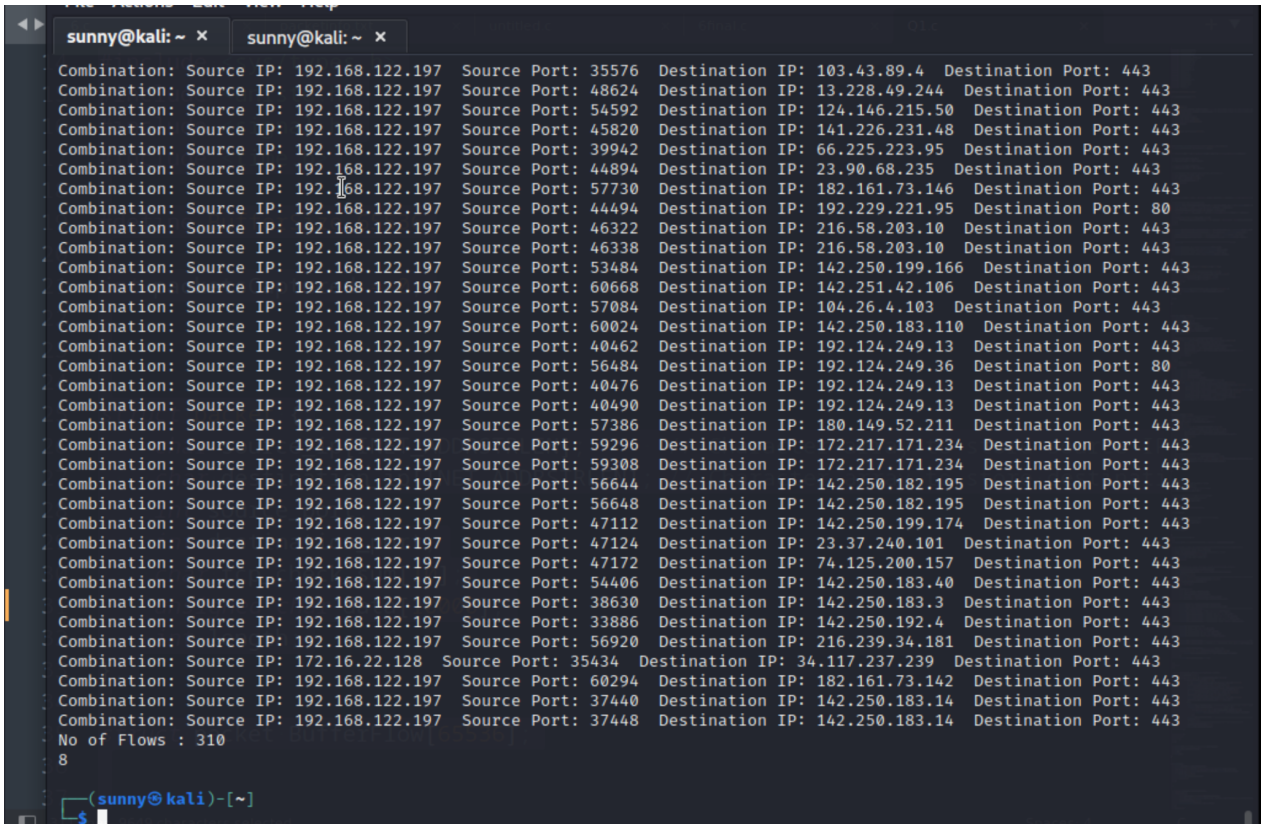
- These functions are designed to answer specific questions (presumably related to network traffic analysis) and write the results to a file.
- Q2_3 searches for packets with a specific TCP checksum and writes the matching packets' payload data to the file.
- Q2_4 looks for packets with a specific source IP address and port combination and writes the matching packets' payload data to the file.

Q2 function:

- This function coordinates the execution of the previously mentioned functions to answer a set of questions and store the results in a file.

b. An analysis of different flows while performing tcpreplay using the provided packet capture (pcap file). The analysis should contain at least the following

i) The number of flows observed by your program and their 4-tuple.



No of tcp flows captured = 310

All the related flow information is stored in the flows.txt file

ii) A reverse DNS lookup of 5 observed IP addresses.

(i) For Destination IP: 52.11.69.53

ec2-52-11-69-53.us-west-2.compute.amazonaws.com.

```
balukarthikram@balukarthikram-G3-3500:~$ dig -x 52.11.69.53

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> -x 52.11.69.53
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18482
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;53.69.11.52.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
53.69.11.52.in-addr.arpa. 300   IN      PTR     ec2-52-11-69-53.us-west-2.compute.amazonaws.com.
```

(ii) For Destination IP: 34.223.124.45

ec2-34-223-124-45.us-west-2.compute.amazonaws.com.

```
balukarthikram@balukarthikram-G3-3500:~$ dig -x 34.223.124.45

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> -x 34.223.124.45
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55986
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;45.124.223.34.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
45.124.223.34.in-addr.arpa. 300 IN      PTR     ec2-34-223-124-45.us-west-2.compute.amazonaws.com.
```

(iii) Destination IP: 18.164.237.50

server-18-164-237-50.del54.r.cloudfront.net.

```
balukarthikram@balukarthikram-G3-3500:~$ dig -x 18.164.237.50

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> -x 18.164.237.50
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62670
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;50.237.164.18.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
50.237.164.18.in-addr.arpa. 12683 IN    PTR     server-18-164-237-50.del54.r.cloudfront.net.
```

(iv) Destination IP: 52.16.32.113

Ec2-52-16-32-113.eu-west-1.compute.amazonaws.com.

```
balukarthikram@balukarthikram-G3-3500:~$ dig -x 52.16.32.113

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> -x 52.16.32.113
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11444
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;113.32.16.52.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
113.32.16.52.in-addr.arpa. 300  IN      PTR     ec2-52-16-32-113.eu-west-1.compute.amazonaws.com.
```

(v) Destination IP: 34.120.115.102

. 102.115.120.34.bc.googleusercontent.com.

```
balukarthikram@balukarthikram-G3-3500:~$ dig -x 34.120.115.102

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> -x 34.120.115.102
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15008
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;102.115.120.34.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
102.115.120.34.in-addr.arpa. 120 IN     PTR     102.115.120.34.bc.googleusercontent.com.
```

Part II: Capture the Flag (20 points) : we have used 3.pcap file

The required output for the Part will be available in Q2.txt file after running Q1.c

1.There is a Flag in a TCP Packet. Identify the flag. (Hint: Search for the keyword Flag)

Data: Flag: Adam

```
95 Flow 132 : Source IP: 18.192.188.189  Source Port: 1911  Destination IP: 111.121.13.114  Destination Port: 11120
96 ----------------------------------------
97 Packet 0 - TCP Check Sum: 50059
98         Data: Flag: Adam
99 ****************************************************************************************************
```

My username is secret, Identify my secret

2. My username is secret, Identify my secret.

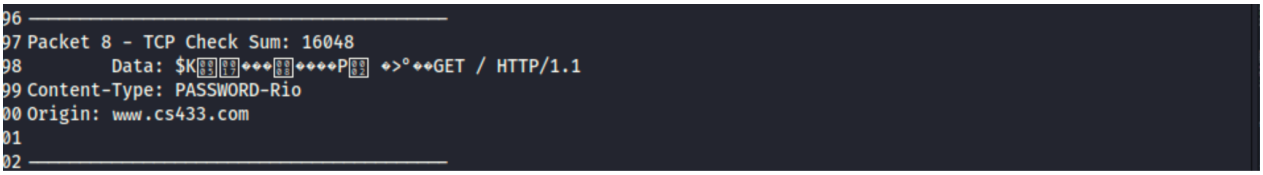
Data: POST /username=secret HTTP/1.1

Connection: Secret: I can do this all day

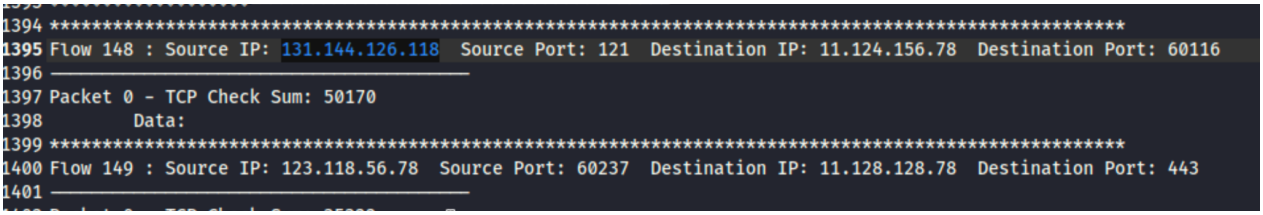
```
29 Flow 136 : Source IP: 181.123.13.192  Source Port: 937  Destination IP: 119.112.128.123  Destination Port: 10293
30 ----------------------------------------
31 Packet 0 - TCP Check Sum: 58839
32         Data: POST /username=secret HTTP/1.1
33 Connection: Secret: I can do this all day
34
35
```

3. I have a TCP checksum “0x46a4”. I have instructions in my path.

Password - Rio



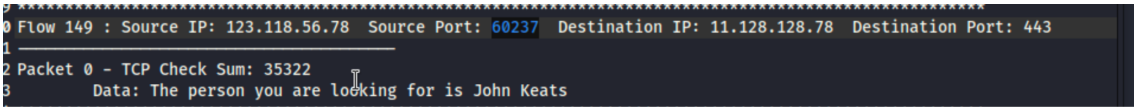
4. **My device has an IP Address “131.144.126.118”. Sum of my connection ports will lead you to a person**



Source port = 121

Destination port = 60116

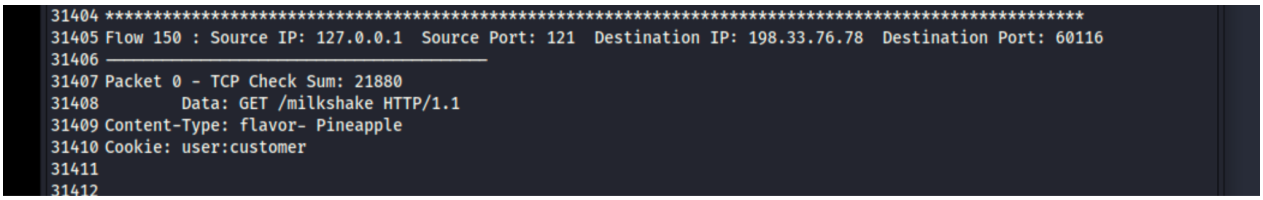
Sum of ports = 60237



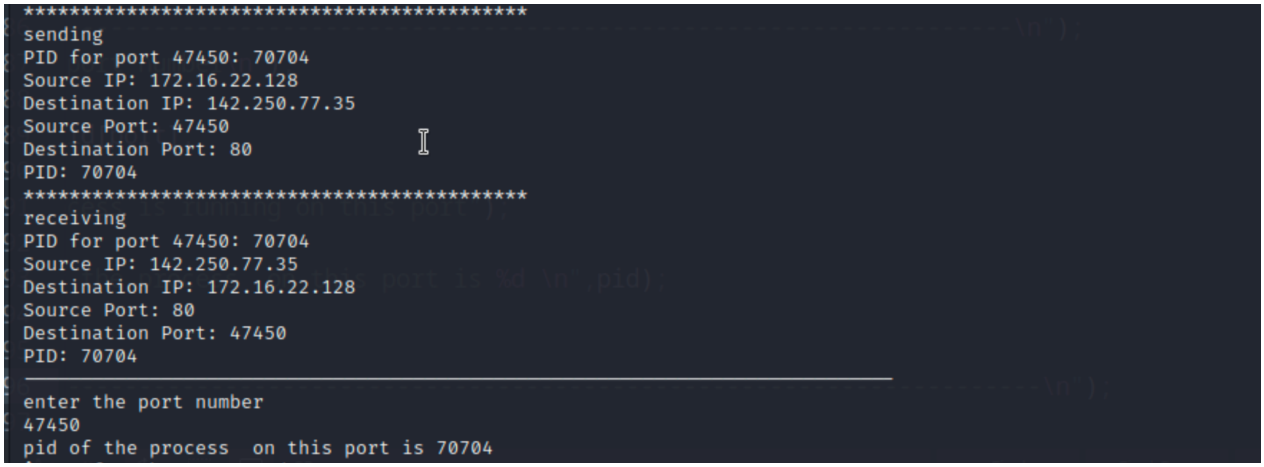
Name = John Keats

5. **I come from localhost, I requested a milkshake. Find my flavour.**

flavor- Pineapple



Part III: Link captured packets to the corresponding process : (20 points)



Brief Explanation of code:

This C code is designed to capture and analyze network packets using raw sockets. It collects information about incoming and\ outgoing TCP packets, associates them with processes running on specific ports, and records the packet details to a file.

Brief explanation of the code:

1. Header Files: The code includes various header files for socket programming, packet handling, and data structures.

2. Global Variables:

- BufferSize: Defines the maximum size for the packet buffer.
- IP: A pointer to store the local IP address.
- packetsCaptured: Keeps track of the number of packets captured.
- portToPid: An array used to map port numbers to process IDs.
- struct packet: A custom structure to store information about each captured packet, including source and destination IP, port numbers, TCP checksum, packet data, length, and associated PID.
- BufferFlow: An array of struct packet to store captured packets and their details.

3. getipaddress() Function:

- Retrieves the local IP address by running the `ifconfig` command.
- Stores the IP address in the `IP` variable.

4. getpidofport(port) Function*:

- Obtains the PID of the process listening on a specific port by executing a system

command using `netstat`.
- Returns the PID if found or -1 if not found.

5. process_packet(unsigned char buffer, int size) Function :
- Processes a captured packet.
- Extracts the IP header and TCP header from the packet.
- Determines if the packet is a TCP packet (protocol 6).
- Retrieves the source and destination IP addresses, ports, and TCP checksum.
- Stores the packet data and associated PID in the `BufferFlow` array.

6. signalhandler() Function :
- Intended to be a signal handler (currently commented out).
- Writes captured packet information to a file, grouped by source and destination IP and port combinations.

7. main() Function:
- Calls `getipaddress()` to retrieve the local IP address.
- Initializes the `portToPid` array.
- Sets up a raw socket to capture network packets.
- Enters an infinite loop to continuously capture and process packets.
- Captures packets for 30 seconds and then prompts the user to input a port number.
- Retrieves the PID associated with the specified port and displays it.
- Sleeps for 10 seconds before continuing the loop.

It's important to note that this code has some limitations and may require superuser privileges to run due to the use of raw sockets and system commands. Additionally, error handling and validation are limited, and the signal handling functionality is currently disabled. The code assumes IPv4 packets and may not handle IPv6 packets correctly.

Part IV: Network Tools (10 points)

a. List at-least 5 different network protocols that we have not discussed so far in the classroom and describe in 1-2 sentences the operation/usage of protocol and its layer of operation and indicate the associated RFC number if any (5 points).

(i) The 5 different network protocols that we are not discussed in the class while capturing the trace of the network packets are QUIC, TLSV1.2, ARP, OCSP, NTP.

QUIC (Quick UDP Internet Connections):

Operation/Usage: QUIC is a transport layer protocol that operates over UDP. It is designed to provide secure, low-latency, and multiplexed transport for web traffic.It combines elements of TCP and TLS, reducing latency and improving performance for web applications.

Layer of Operation: QUIC operates at the transport layer (Layer 4) of the OSI model.

RFC Number:RFC 9000

TLSv1.2 (Transport Layer Security version 1.2):

Operation/Usage: TLSv1.2 is a cryptographic protocol that operates at the transport layer and provides secure communication over a network, typically used for securing web traffic. It provides authentication, data integrity, and confidentiality by encrypting data between two communicating parties.

Layer of Operation: TLS 1.2 operates at the transport layer (Layer 4) of the OSI model.

RFC Number: RFC 5246

ARP (Address Resolution Protocol):

Operation/Usage: ARP is used for mapping an IP address to a physical (MAC) address within a local network. It helps devices on the same subnet communicate by resolving the target device's hardware address.

Layer of Operation: ARP operates at the link layer (Layer 2) of the OSI model.

RFC Number: RFC 826

OCSP (Online Certificate Status Protocol):

Operation/Usage: OCSP is used for checking the validity of digital certificates in a Public Key Infrastructure (PKI). It enables clients to query a Certificate Authority (CA) to determine if a specific certificate is still valid, rather than relying on potentially outdated Certificate Revocation Lists (CRLs).

Layer of Operation: OCSP operates at the application layer (Layer 7) of the OSI model.

RFC Number: RFC 6960.

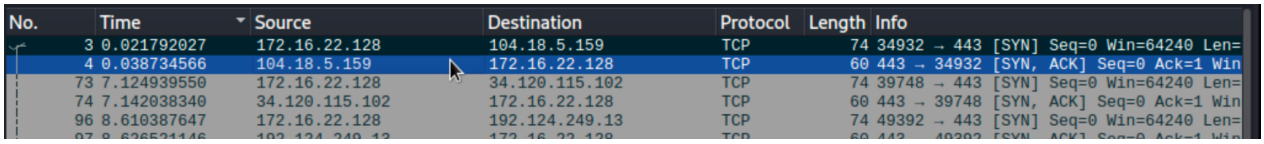
NTP (Network Time Protocol):

Operation/Usage: NTP is a protocol used for synchronizing the clocks of devices on a computer network. It ensures that all devices maintain accurate and consistent time, which is crucial for various network operations, including security, logging, and coordination of events.

Layer of Operation: NTP operates at the application layer (Layer 7) of the OSI model. It uses UDP (User Datagram Protocol) as the transport protocol.

RFC Number: The primary RFC for NTP is RFC 5905.

b. Identify any one connection and try to estimate the RTT of that connection



No.	Time	Source	Destination	Protocol	Length	Info
3	0.021792027	172.16.22.128	104.18.5.159	TCP	74	34932 → 443 [SYN] Seq=0 Win=64240 Len=
4	0.038734566	104.18.5.159	172.16.22.128	TCP	60	443 → 34932 [SYN, ACK] Seq=0 Ack=1 Win
73	7.124939550	172.16.22.128	34.120.115.102	TCP	74	39748 → 443 [SYN] Seq=0 Win=64240 Len=
74	7.142038340	34.120.115.102	172.16.22.128	TCP	60	443 → 39748 [SYN, ACK] Seq=0 Ack=1 Win
96	8.610387647	172.16.22.128	192.124.249.13	TCP	74	49392 → 443 [SYN] Seq=0 Win=64240 Len=

The client send packet to server at a timestamp of 0.021790027 sec

The server sends the acknowledgement at a timestamp of 0.038734566 sec

The **Round Trip Time** = 0.016942539 sec (16.9 ms)

2. **Identify the application layer protocols and their versions used when visiting the following websites:**

Github.com

Netflix.com

google.com

Explain in a few lines the differences and similarities between the protocols. (2 points)

(Hint: Inspect)

The application layer protocols in **Github.com - (HTTP/2.0)**, **Netflix.com - (HTTP/2.0/1.1), google.com - (HTTP/3.0).**

Differences:

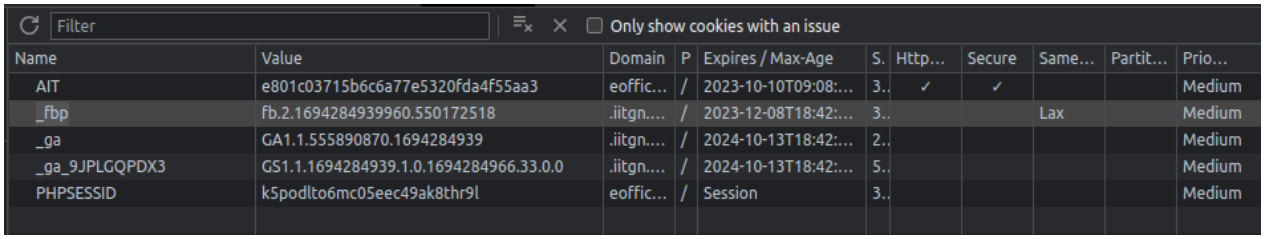
	(HTTP/1.1)	**(HTTP/2.0)**	**(HTTP/3.0)**
1. **Multiplexing:**	It uses a single connection per request, leading to head-of-line blocking issues, where one slow request can block subsequent requests.	It allows multiple requests and responses to be interleaved on a single connection, thus improving efficiency and reducing latency.	It also supports multiplexing but uses the QUIC transport protocol to further enhance multiplexing efficiency and reduce connection establishment latency.
2. **Protocol Transport:**	It uses plain text and is transported over TCP.	It also uses plain text but is transported over a single multiplexed TCP connection.	It uses QUIC, which combines transport-layer security and multiplexing into a single protocol, making it faster and more secure.
3. **Performance:**	It has performance limitations due to head-of-line blocking and inefficient handling of resources.	It improves performance significantly, especially for websites with numerous resources.	It further enhances performance with lower connection latency and improved multiplexing, making it well-suited for modern web applications.

Similarities:

1. **Functionality:** All three versions of HTTP serve the fundamental purpose of facilitating communication between clients (such as web browsers) and servers to request and transmit web resources (e.g., web pages, images, videos).
2. **Request-Response Model:** They all follow the request-response model, where a client sends a request to a server, and the server responds with the requested data.
3. **Use of TCP/IP:** They all rely on the TCP/IP protocol suite for data transmission, ensuring reliable and ordered delivery of data packets.

3. **List the cookies and identify the characteristics of the cookies setup when you visit *eoffice.iitgn.ac.in* (1 points).**

Cookies identified and characteristics of cookies .



Name	Value	Domain	P	Expires / Max-Age	S.	Http...	Secure	Same...	Partit...	Prio...
AIT	e801c03715b6c6a77e5320fda4f55aa3	eoffic...	/	2023-10-10T09:08:...	3..	✓	✓			Medium
_fbp	fb.2.1694284939960.550172518	.iitgn....	/	2023-12-08T18:42:...	3..			Lax		Medium
_ga	GA1.1.555890870.1694284939	.iitgn....	/	2024-10-13T18:42:...	2..					Medium
_ga_9JPLGQPDX3	GS1.1.1694284939.1.0.1694284966.33.0.0	.iitgn....	/	2024-10-13T18:42:...	5..					Medium
PHPSESSID	k5podlto6mc05eec49ak8thr9l	eoffic...	/	Session	3..					Medium

1. **PHPSESSID Cookie:**
   PHPSESSID cookie is used on a website to enhance user experience, improve security, and facilitate the management of user sessions. It enables the website to maintain state, track user interactions, and provide a tailored and secure online environment for employees and users.

2. **_fbp cookie:**
   "_fbp" cookie is to help advertisers and website owners track conversions that occur as a result of Facebook advertising campaigns. It allows them to measure the effectiveness of their ads and understand how users interact with their websites after clicking on a Facebook ad.

3. **_ga cookie:**
   "_ga" cookie is to distinguish unique users when they visit a website. It is used to generate statistical data about how visitors use the site, including information like the number of visitors, the pages they visit, and their interactions with the site's content.

REFERENCES:

1. https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/
2. https://www.binarytides.com/packet-sniffer-code-in-c-using-linux-sockets-bsd-part-2/