CS 433 Assignment 1

Ayush Modi-21110039
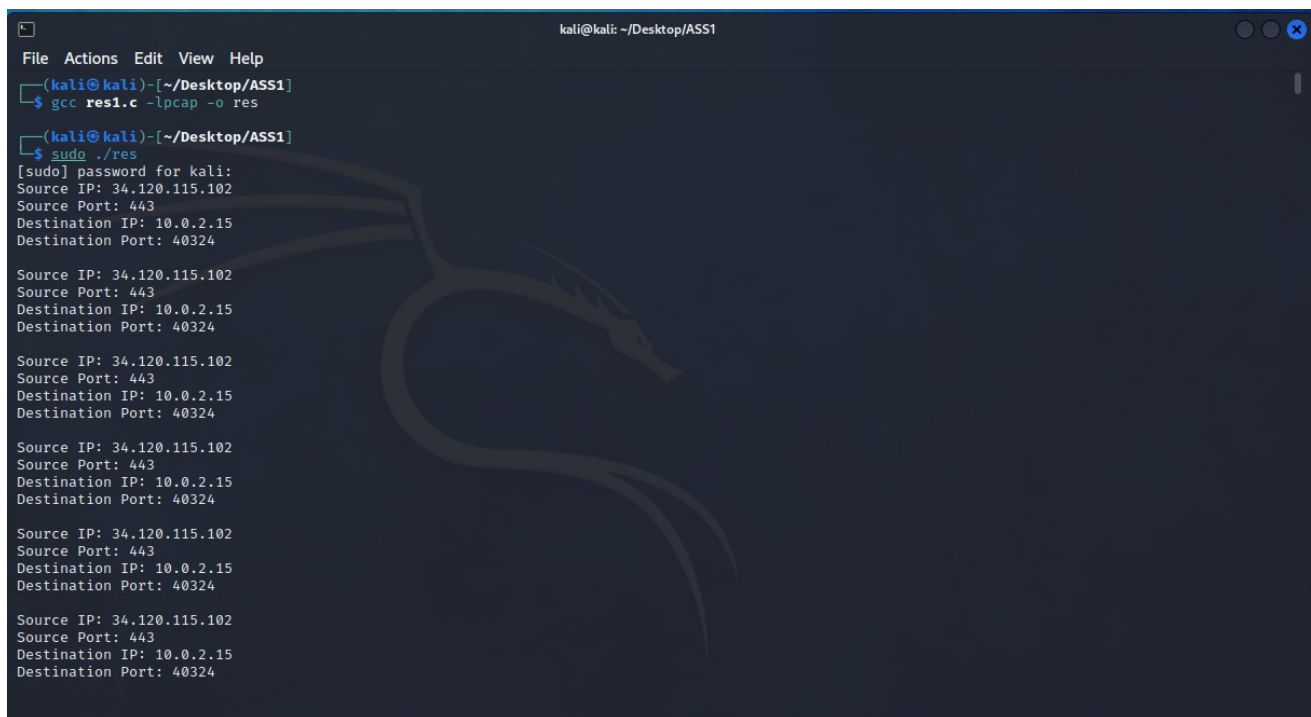Anish Karnik-21110098

GitHub Link : https://github.com/anish-karnik/CS433CN_Assignment_01

Part 1:

a. First, we start by including statements for C libraries and headers that are necessary for networking and socket programming. The function called "process_packet" is responsible for extracting and displaying information from a captured network packet. This includes details like the source and destination IP addresses, as port numbers. To achieve this the function interprets the data of the packet using pointers and converts the addresses and port numbers into a format that can be easily understood by humans before printing them to the console.

Next we proceed to create a socket by utilizing the socket function. Following that we implement a loop which continuously listens for packets. Once we exit this loop the raw socket is closed effectively terminating the program as it returns 0.

Terminal output showing the packet(four tuple) going through the network interface.

b. This code should print both the total number of observed flows and the 4-tuple details for each flow when Ctrl+C is pressed.

```
┌──(kali㊀kali)-[~/Desktop/ASS1]
└─$ sudo ./1b
^C
Total number of flows observed: 1033
Flow 1: 0.69.63.102:32768 → 0.0.0.0:21076
Flow 2: 192.168.122.1:60788 → 239.255.255.250:1900
Flow 3: 192.168.122.197:37860 → 192.168.122.1:53
Flow 4: 192.168.122.1:53 → 192.168.122.197:37860
Flow 5: 192.168.122.197:59308 → 23.32.177.81:443
Flow 6: 23.32.177.81:443 → 192.168.122.197:59308
Flow 7: 53.2.10.0:1 → 2.2.255.255:2048
Flow 8: 192.168.122.197:34119 → 192.168.122.1:53
Flow 9: 192.168.122.197:34081 → 192.168.122.1:53
Flow 10: 192.168.122.197:35615 → 192.168.122.1:53
Flow 11: 192.168.122.1:53 → 192.168.122.197:34119
Flow 12: 192.168.122.1:53 → 192.168.122.197:34081
Flow 13: 192.168.122.1:53 → 192.168.122.197:35615
Flow 14: 192.168.122.197:51406 → 192.168.122.1:53
Flow 15: 192.168.122.197:51410 → 192.168.122.1:53
Flow 16: 192.168.122.1:53 → 192.168.122.197:51406
Flow 17: 192.168.122.1:53 → 192.168.122.197:51410
Flow 18: 192.168.122.197:46302 → 44.215.135.131:443
Flow 19: 192.168.122.197:33934 → 108.158.228.47:443
Flow 20: 192.168.122.197:33946 → 108.158.228.47:443
Flow 21: 192.168.122.197:33954 → 108.158.228.47:443
```

Reverse DNS Lookup:

```
┌──(kali㊀kali)-[~/Desktop/ASS1]
└─$ nslookup 52.16.32.113
113.32.16.52.in-addr.arpa     name = ec2-52-16-32-113.eu-west-1.compute.amazonaws.com.

Authoritative answers can be found from:
.       nameserver = k.root-servers.net.
.       nameserver = j.root-servers.net.
.       nameserver = i.root-servers.net.
.       nameserver = a.root-servers.net.
.       nameserver = g.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = e.root-servers.net.
.       nameserver = b.root-servers.net.
.       nameserver = m.root-servers.net.
.       nameserver = c.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = d.root-servers.net.
g.root-servers.net      internet address = 192.112.36.4
g.root-servers.net      has AAAA address 2001:500:12::d0d
j.root-servers.net      internet address = 192.58.128.30
j.root-servers.net      has AAAA address 2001:503:c27::2:30
e.root-servers.net      internet address = 192.203.230.10
e.root-servers.net      has AAAA address 2001:500:a8::e
l.root-servers.net      internet address = 199.7.83.42
l.root-servers.net      has AAAA address 2001:500:9f::42
d.root-servers.net      internet address = 199.7.91.13
```

```
┌──(kali㊀kali)-[~/Desktop/ASS1]
└─$ nslookup 202.12.27.33
33.27.12.202.in-addr.arpa     name = M.ROOT-SERVERS.NET.

Authoritative answers can be found from:
.       nameserver = e.ROOT-SERVERS.NET.
.       nameserver = g.ROOT-SERVERS.NET.
.       nameserver = b.ROOT-SERVERS.NET.
.       nameserver = c.ROOT-SERVERS.NET.
.       nameserver = j.ROOT-SERVERS.NET.
.       nameserver = M.ROOT-SERVERS.NET.
.       nameserver = i.ROOT-SERVERS.NET.
.       nameserver = d.ROOT-SERVERS.NET.
.       nameserver = k.ROOT-SERVERS.NET.
.       nameserver = h.ROOT-SERVERS.NET.
.       nameserver = f.ROOT-SERVERS.NET.
.       nameserver = a.ROOT-SERVERS.NET.
.       nameserver = l.ROOT-SERVERS.NET.
g.ROOT-SERVERS.NET      internet address = 192.112.36.4
```

```
┌──(kali㊀kali)-[~/Desktop/ASS1]
└─$ nslookup 202.12.27.33
33.27.12.202.in-addr.arpa     name = M.ROOT-SERVERS.NET.

Authoritative answers can be found from:
.       nameserver = e.ROOT-SERVERS.NET.
.       nameserver = g.ROOT-SERVERS.NET.
.       nameserver = b.ROOT-SERVERS.NET.
.       nameserver = c.ROOT-SERVERS.NET.
.       nameserver = j.ROOT-SERVERS.NET.
.       nameserver = M.ROOT-SERVERS.NET.
.       nameserver = i.ROOT-SERVERS.NET.
.       nameserver = d.ROOT-SERVERS.NET.
.       nameserver = k.ROOT-SERVERS.NET.
.       nameserver = h.ROOT-SERVERS.NET.
.       nameserver = f.ROOT-SERVERS.NET.
.       nameserver = a.ROOT-SERVERS.NET.
.       nameserver = l.ROOT-SERVERS.NET.
g.ROOT-SERVERS.NET      internet address = 192.112.36.4
g.ROOT-SERVERS.NET      has AAAA address 2001:500:12::d0d
j.ROOT-SERVERS.NET      internet address = 192.58.128.30
j.ROOT-SERVERS.NET      has AAAA address 2001:503:c27::2:30
e.ROOT-SERVERS.NET      internet address = 192.203.230.10
e.ROOT-SERVERS.NET      has AAAA address 2001:500:a8::e
l.ROOT-SERVERS.NET      internet address = 199.7.83.42
l.ROOT-SERVERS.NET      has AAAA address 2001:500:9f::42
d.ROOT-SERVERS.NET      internet address = 199.7.91.13
d.ROOT-SERVERS.NET      has AAAA address 2001:500:2d::d
a.ROOT-SERVERS.NET      internet address = 198.41.0.4
```

```
┌──(kali㊀kali)-[~/Desktop/ASS1]
└─$ nslookup 192.112.36.4
4.36.112.192.in-addr.arpa     name = G.ROOT-SERVERS.NET.

Authoritative answers can be found from:
.       nameserver = m.ROOT-SERVERS.NET.
.       nameserver = b.ROOT-SERVERS.NET.
.       nameserver = h.ROOT-SERVERS.NET.
.       nameserver = c.ROOT-SERVERS.NET.
.       nameserver = a.ROOT-SERVERS.NET.
.       nameserver = f.ROOT-SERVERS.NET.
.       nameserver = k.ROOT-SERVERS.NET.
.       nameserver = j.ROOT-SERVERS.NET.
.       nameserver = e.ROOT-SERVERS.NET.
.       nameserver = G.ROOT-SERVERS.NET.
.       nameserver = l.ROOT-SERVERS.NET.
.       nameserver = i.ROOT-SERVERS.NET.
```

```
G.ROOT-SERVERS.NET      has AAAA address 2001:500:12::d0d
j.ROOT-SERVERS.NET      internet address = 192.58.128.30
j.ROOT-SERVERS.NET      has AAAA address 2001:503:c27::2:30
e.ROOT-SERVERS.NET      internet address = 192.203.230.10
e.ROOT-SERVERS.NET      has AAAA address 2001:500:a8::e
l.ROOT-SERVERS.NET      internet address = 199.7.83.42
l.ROOT-SERVERS.NET      has AAAA address 2001:500:9f::42
d.ROOT-SERVERS.NET      internet address = 199.7.91.13
d.ROOT-SERVERS.NET      has AAAA address 2001:500:2d::d
a.ROOT-SERVERS.NET      internet address = 198.41.0.4

┌──(kali㉿kali)-[~/Desktop/ASS1]
└─$ nslookup 198.41.0.4

4.0.41.198.in-addr.arpa name = a.root-servers.net.

Authoritative answers can be found from:
.       nameserver = c.root-servers.net.
.       nameserver = b.root-servers.net.
.       nameserver = a.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = g.root-servers.net.
.       nameserver = d.root-servers.net.
.       nameserver = k.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = e.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = m.root-servers.net.
.       nameserver = i.root-servers.net.
.       nameserver = j.root-servers.net.
g.root-servers.net      internet address = 192.112.36.4
g.root-servers.net      has AAAA address 2001:500:12::d0d
j.root-servers.net      internet address = 192.58.128.30
j.root-servers.net      has AAAA address 2001:503:c27::2:30
e.root-servers.net      internet address = 192.203.230.10
e.root-servers.net      has AAAA address 2001:500:a8::e
l.root-servers.net      internet address = 199.7.83.42
l.root-servers.net      has AAAA address 2001:500:9f::42
d.root-servers.net      internet address = 199.7.91.13
d.root-servers.net      has AAAA address 2001:500:2d::d
a.root-servers.net      internet address = 198.41.0.4

┌──(kali㉿kali)-[~/Desktop/ASS1]
└─$ nslookup 199.7.91.13

13.91.7.199.in-addr.arpa        name = d.root-servers.net.
```

```
.       nameserver = i.root-servers.net.
.       nameserver = j.root-servers.net.
g.root-servers.net      internet address = 192.112.36.4
g.root-servers.net      has AAAA address 2001:500:12::d0d
j.root-servers.net      internet address = 192.58.128.30
j.root-servers.net      has AAAA address 2001:503:c27::2:30
e.root-servers.net      internet address = 192.203.230.10
e.root-servers.net      has AAAA address 2001:500:a8::e
l.root-servers.net      internet address = 199.7.83.42
l.root-servers.net      has AAAA address 2001:500:9f::42
d.root-servers.net      internet address = 199.7.91.13
d.root-servers.net      has AAAA address 2001:500:2d::d
a.root-servers.net      internet address = 198.41.0.4

┌──(kali㉿kali)-[~/Desktop/ASS1]
└─$ nslookup 199.7.91.13

13.91.7.199.in-addr.arpa        name = d.root-servers.net.

Authoritative answers can be found from:
.       nameserver = i.root-servers.net.
.       nameserver = k.root-servers.net.
.       nameserver = a.root-servers.net.
.       nameserver = c.root-servers.net.
.       nameserver = m.root-servers.net.
.       nameserver = j.root-servers.net.
.       nameserver = b.root-servers.net.
.       nameserver = g.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = d.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = e.root-servers.net.
g.root-servers.net      internet address = 192.112.36.4
g.root-servers.net      has AAAA address 2001:500:12::d0d
j.root-servers.net      internet address = 192.58.128.30
j.root-servers.net      has AAAA address 2001:503:c27::2:30
e.root-servers.net      internet address = 192.203.230.10
e.root-servers.net      has AAAA address 2001:500:a8::e
l.root-servers.net      internet address = 199.7.83.42
l.root-servers.net      has AAAA address 2001:500:9f::42
d.root-servers.net      internet address = 199.7.91.13
d.root-servers.net      has AAAA address 2001:500:2d::d
a.root-servers.net      internet address = 198.41.0.4
```

Part 2: (1.pcap)

1. Modified the code from part 1 to find data containing "Flag". (Filename in Github – 2a.c)
   Answer: Romeo

```
Source IP: 106.106.106.106
Source Port: 106
Destination IP: 108.108.108.108
Destination Port: 116
TCP Flags: SYN
TCP Payload Data:
Hi there, this is not the Flag, skip this packet

Source IP: 109.109.109.109
Source Port: 109
Destination IP: 111.111.111.111
Destination Port: 119
TCP Flags: SYN
TCP Payload Data:
Hi there, this is not the Flag, skip this packet

Source IP: 108.99.108.99
Source Port: 991
Destination IP: 101.102.103.104
Destination Port: 1020
TCP Flags: SYN
TCP Payload Data:
Flag: Romeo

Source IP: 11.11.11.11
Source Port: 11
Destination IP: 13.13.13.13
Destination Port: 21
TCP Flags: SYN
TCP Payload Data:
Hi there, this is not the Flag, skip this packet

Source IP: 12.12.12.12
Source Port: 12
Destination IP: 14.14.14.14
```

2. To find frame containing secret (2b.c)

3. To find TCP checksum "0xf436". (2c.c)

4. Code divided into two parts: (1) Find the sum of ports having IP Address 123.134.156.178.(2d.c)
                                 (2) Find the data in the source port having this sum. (2d2.c)

```
                          kali@kali: ~/Desktop/ASS1
File  Actions  Edit  View  Help
Source Port: 1111
Destination IP: 12.34.56.78
Destination Port: 9876
TCP Flags: SYN
Sum of Ports: 10987
TCP Payload Data:


  ┌──(kali⊛kali)-[~/Desktop/ASS1]
  └─$ gcc -o 2d2 2d2.c -lpcap


  ┌──(kali⊛kali)-[~/Desktop/ASS1]
  └─$ sudo ./2d2
Source IP: 123.128.56.78
Source Port: 10987
Destination IP: 12.128.128.78
Destination Port: 443
TCP Payload Data:
The person you are looking for is Rabindranath Tagore
```

5. To find frame containing milkshake (2e.c)

```
                          kali@kali: ~/Desktop/ASS1
File  Actions  Edit  View  Help

  ┌──(kali⊛kali)-[~/Desktop/ASS1]
  └─$ gcc -o 2e 2e.c -lpcap


  ┌──(kali⊛kali)-[~/Desktop/ASS1]
  └─$ sudo ./2e
Source IP: 127.0.0.1
Source Port: 1111
Destination IP: 122.44.56.78
Destination Port: 9876
TCP Payload Data:
GET /milkshake HTTP/1.1
Cookie: user:customer
Referer: flavor- Strawberry



  ┌──(kali⊛kali)-[~/Desktop/ASS1]
  └─$ ▮
```

Part 3:
Github Filename: 3.c

```
                          kali@kali: ~/Desktop/ASS1
File  Actions  Edit  View  Help
Destination Port: 16390
Source IP: 8.0.69.0
Source Port: 0
Destination IP: 0.40.136.94
Destination Port: 16390
Source IP: 8.0.69.0
Source Port: 0
Destination IP: 0.40.136.95
Destination Port: 16390
Source IP: 8.0.69.0
Source Port: 16384
Destination IP: 0.40.149.183
Destination Port: 16390
Source IP: 8.0.69.0
Source Port: 0
Destination IP: 0.40.136.96
Destination Port: 16390
^CProcess ID for port 443: 32513
```

Part 4:

(1) (a)

1. ARP:
    ARP is the Address Resolution Protocol. It maps IP addresses to a physical MAC(Machine Access Control) address on a local area network.
    -   Layer of Operation: Layer 2: Data Link Layer
    -   RFC number: RFC 826

2. UDP:
    ARP is the User Datagram Protocol. It is used for real-time applications which are time-sensitive. UDP sends data packets over a network without establishing a firm connection with the destination, thereby resulting in efficient and faster data transmission.
    -   Layer of Operation: Layer 4: Transport Layer
    -   RFC number: RFC 768

3. QUIC:
    QUIC stands for Quick UDP Internet Connections. It is a very recent transport layer protocol and is made by combining UDP and other protocols. Its usage is similar to UDP and is used for quick and low-latency communication applications like gaming and video-streaming.
    -   Layer of Operation: Layer 4: Transport Layer
    -   RFC number: RFC 9000

4. TLSv1.3:
    TLS stands for Transport Layer Security. TLS version 1.3 is a security protocol and it is used for data encryption and ensures secure data communication over the internet.
    -   Layer of Operation: According to most sources, TLSv1.3 is considered to operate between the Transport Layer and the Application Layer. It operates in the Transport Layer, but may also often be considered to be a part of the Application Layer.
    -   RFC number: RFC 8446

5. TLSv1.2:
    TLSv1.2 is similar to, but is an older version of TLSv1.3. The difference between the two is that unlike TLSv1.3, TLSv1.2 does not support modern encryption and thus has slightly lower performance comparatively.
    -   Layer of Operation: Same as TLSv1.3. It operates between the Transport Layer and the Application Layer. It does operate in the Transport Layer, but may also often be considered to be a part of the Application Layer.
    -   RFC number: RFC 5246

(b) We will calculate the RTT for TCP
-   First outgoing packet sent at timestamp:
    4.668584835 seconds
-   First incoming packet from the remote server acknowledging the receipt of the packet sent above, denoted by the [SYN, ACK] packet with Ack=1 sent at timestamp:
    5.044443887 seconds

The RTT (Round Trip Time) is the difference between these two timestamps.
Therefore RTT for TCP = (5.044443887 - 4.668584835) seconds = 0.375859052 seconds

(2) The application layer protocol Https is used by all three sites. Https denotes secure connection, thus all these popular websites use Https as their application layer protocol.

(3) These are the characteristics of the cookies that we found upon visiting *eoffice.iitgn.ac.in.*
To find information about the cookies, we opened the inspect window of the website on Google Chrome and went to the Application section and under it in the section named Cookies.

There was a list of 4 cookies:

The characteristics of these cookies are as follows:

Name	Value	Domain	Path	Size	Expires
PHPSESSID	3htea3h3q4kknptk rm5u93dklo	eoffice.iitgn.ac.in	/	35	Session
AIT	e555b8032fdc6c5 eb501b84d73f1f7 12	eoffice.iitgn.ac.in	/	35	2023-10-07T21:4 0:59.912Z
_ga_L30C3Q76J7	GS1.1.169056577 9.1.1.1690565788. 51.0.0	.iitgn.ac.in	/	52	2024-08-31T17:3 6:28.555Z
_ga	GA1.1.491691878 .1690565780	.iitgn.ac.in	/	29	2024-08-31T17:3 6:19.634Z

The name column denotes the name of the cookie and the value is the value associated with a particular cookie.
The domain specifies the domain to which the cookie belongs.
The path is the path on the website in which the cookie is valid, in this case in all of them the path is '/' because we were on the homepage of the website itself.
The size of the cookie in KB is given in the size column. The Expires column denotes the expiration date and time of a cookie.

There were also three more columns: HttpOnly, Secure, and SameSite.
HttpOnly specifies whether the cookie can be accessed via client-side scripts like JavaScript, denoting its security.
Secure specifies the security associated with the cookie whether it is sent only over secure HTTPS connections.
SameSite tells how a cookie is used when dealing with cross-site requests.

Only the cookie "AIT" was checkmarked for the attributes HttpOnly and Secure, and rest all cookies had all the attributes not checked.