Laboratorio 6

Laboratorio 6

Security Data Science

Manuel Archila - 161250

```
import scapy.all as scapy
import pandas as pd
import matplotlib.pyplot as plt
```

Preambulo

```
from scapy.all import sniff


paquetes = sniff(count=25)

print(type(paquetes))
print(len(paquetes))
print(paquetes)

print(type(paquetes[0]))

for paquete in paquetes[:5]:
    paquete.show()
```

```
<class 'scapy.plist.PacketList'>
25
<Sniffed: TCP:10 UDP:12 ICMP:0 Other:3>
<class 'scapy.layers.l2.Ether'>
###[ Ethernet ]###
  dst       = 2c:fd:b3:91:9d:25
  src       = c4:03:a8:2e:bd:64
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 150
     id        = 38502
     flags     = DF
     frag      = 0
```

```
      ttl       = 128
      proto     = tcp
      chksum    = 0x0
      src       = 192.168.68.108
      dst       = 192.168.68.109
      \options   \
###[ TCP ]###
         sport     = 64019
         dport     = 8009
         seq       = 2847727646
         ack       = 624561081
         dataofs   = 5
         reserved  = 0
         flags     = PA
         window    = 513
         chksum    = 0xab3
         urgptr    = 0
         options   = []
###[ Raw ]###
            load      =
"\x17\x03\x03\x00i\\xf7\x06\\xc34W\\x98\rc5'HO6\\xfbQ\x1c;\\xd2P\x01\\x85E\\x87î\n(\\xc8,\\xcc\\x
e6\\xf9\\xfb\\xf8\\xb4\x14
\\xa0\\xdb\x1a9\\xe9L\\x90\\x9d\\xde\\xed9\x1fW\x0cd\\xf0n\\x8d\\xf0M\\xfd\\xef\\xad\\xf2=P\x08:\
\xa7*z\\x99\\xa3@Q]\\x817\\x9e\\xc8\x1dF龑
\\xf4\\xb8\\xc5\\xc0Q\\xbb\n\x16\\xa8\\x99\x17\x0b\\xd5\x00\\xd4\x13\\xa1\x17\x04\\xbd\\xe5"

###[ Ethernet ]###
  dst       = c4:03:a8:2e:bd:64
  src       = 2c:fd:b3:91:9d:25
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 150
     id        = 27042
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xc695
     src       = 192.168.68.109
     dst       = 192.168.68.108
     \options   \
###[ TCP ]###
        sport     = 8009
        dport     = 64019
        seq       = 624561081
        ack       = 2847727756
        dataofs   = 5
        reserved  = 0
```

```
        flags     = PA
        window    = 762
        chksum    = 0xf782
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      =
'\x17\x03\x03\x00i\\xceJ\\x8c\\xc1\\x8c\\xa2\x19\\xab\\xffZ7\x04\x0e$\\xab\\xd2\\xd7i\\x93\x1dh\r
W`\\x96\\xec\\xd31o\x1aш\\xba\\xba\x0eM\\xa9\\x9f\\x8f\x08\\xf0\\xbb-
\\xd37\\xb2\\xc6\\xd9)F\\x8d\\xcb\\xdbX\\xb2\\xd4k\\xf4V\x11.\x06\\x8fp\\xd2\\xf4\\xfc6\\xf6\\xf8
\\xf7\\xaa8Rk\\xa5\x0bV\\xf5\x05Ef\\xecpV\x0e0\\xff\x1f\x0b\\xb7\\xbc\\xc2\x01\x15\\xdf\x08\\xe9S
\\xe5\\xff\\x86\\xff^K'

###[ Ethernet ]###
  dst       = 2c:fd:b3:91:9d:25
  src       = c4:03:a8:2e:bd:64
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 38503
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x0
     src       = 192.168.68.108
     dst       = 192.168.68.109
     \options   \
###[ TCP ]###
        sport     = 64019
        dport     = 8009
        seq       = 2847727756
        ack       = 624561191
        dataofs   = 5
        reserved  = 0
        flags     = A
        window    = 512
        chksum    = 0xa45
        urgptr    = 0
        options   = ''

###[ Ethernet ]###
  dst       = 01:00:5e:00:00:fb
  src       = 62:d4:40:04:7c:e6
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
```

```
      tos       = 0x0
      len       = 154
      id        = 30335
      flags     = DF
      frag      = 0
      ttl       = 255
      proto     = udp
      chksum    = 0x1eac
      src       = 192.168.68.131
      dst       = 224.0.0.251
      \options   \
###[ UDP ]###
         sport     = 5353
         dport     = 5353
         len       = 134
         chksum    = 0x54af
###[ DNS ]###
            id        = 2
            qr        = 0
            opcode    = QUERY
            aa        = 0
            tc        = 0
            rd        = 0
            ra        = 0
            z         = 0
            ad        = 0
            cd        = 0
            rcode     = ok
            qdcount   = 4
            ancount   = 0
            nscount   = 0
            arcount   = 0
            \qd        \
             |###[ DNS Question Record ]###
             |  qname     =
'_%9E5E7C8F47989526C9BCD95D24084F6F0B27C5ED._sub._googlecast._tcp.local.'
             |  qtype     = PTR
             |  qclass    = IN
             |###[ DNS Question Record ]###
             |  qname     = '_CFE7FEDA._sub._googlecast._tcp.local.'
             |  qtype     = PTR
             |  qclass    = IN
             |###[ DNS Question Record ]###
             |  qname     = '_CC32E753._sub._googlecast._tcp.local.'
             |  qtype     = PTR
             |  qclass    = IN
             |###[ DNS Question Record ]###
             |  qname     = '_googlecast._tcp.local.'
             |  qtype     = PTR
             |  qclass    = IN
            an        = None
```

```
            ns        = None
            ar        = None

###[ Ethernet ]###
  dst       = 20:1f:3b:8e:9e:15
  src       = c4:03:a8:2e:bd:64
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 150
     id        = 3352
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x0
     src       = 192.168.68.108
     dst       = 192.168.68.110
     \options   \
###[ TCP ]###
        sport     = 64017
        dport     = 8009
        seq       = 2055298832
        ack       = 1334774286
        dataofs   = 5
        reserved  = 0
        flags     = PA
        window    = 512
        chksum    = 0xab4
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      =
'\x17\x03\x03\x00iAN\\xee\\xc2DX\x05b\\xcfqZ\\xe5\\x8b(\\xa6C*\\xbb\\xccr\x04\\xacC{\\xf8\\xa4\\x
c2A\\xbb\\xa5|Z\\xe0F\\xf0CGb\\xa64\\x96\\xb1\\x92\\xcc\\xfd\x19@\\xab\x1f\\xf8;\x02\x06\\xa0\x15
\\xda,\\xb9n]U)pHP[\x18g\\xc91&\x03vp\\xe3E\x02\\x80\\xa8un\x07\\x8a\\xeeJm\\xb1\x1c\\xe8+\\xac\\
x9a\x10*\\x86^08}T\\xa9\\xeaj\\xe3'
```

1. Descargue e archivo analisis_paquetes.pcap y asignelo a una variable.

```
import scapy.all as scapy


archivo_pcap = 'analisis_paquetes.pcap'

paquetes = scapy.rdpcap(archivo_pcap)

datos_paquetes = []
```

```
        for paquete in paquetes:
            if paquete.haslayer(scapy.IP):
                src_ip = paquete["IP"].src
                dst_ip = paquete["IP"].dst
                src_port = paquete["IP"].sport
                dst_port = paquete["IP"].dport
                payload = len(paquete.payload)
                timestamp = float(paquete.time)
                payload_cont = paquete.payload

                datos_paquetes.append([src_ip, dst_ip, src_port, dst_port, payload, timestamp])
```

2, 3. Convierta la variable a un DataFrame y muestre el contenido de las primeras 5 filas del dataset.

```
        df_paquetes = pd.DataFrame(datos_paquetes, columns=['Src Address', 'Dst Address', 'Src P

        df_paquetes.head()
```

	Src Address	Dst Address	Src Port	Dst Port	Payload Size	Timestamp
0	10.1.10.53	84.54.22.33	53	53	961	1.532199e+09
1	84.54.22.33	10.1.10.53	53	53	84	1.532199e+09
2	10.1.10.53	84.54.22.33	53	53	975	1.532199e+09
3	84.54.22.33	10.1.10.53	53	53	84	1.532199e+09
4	10.1.10.53	84.54.22.33	53	53	1012	1.532199e+09

5.a y 5.b Muestre todas las IP origen y muestre todas las IP destino

```
        print("IPs Origen:")
        print(df_paquetes['Src Address'].unique())

        print("IPs Destino:")
        print(df_paquetes['Dst Address'].unique())
```

```
IPs Origen:
['10.1.10.53' '84.54.22.33' '75.75.75.75']
IPs Destino:
['84.54.22.33' '10.1.10.53' '75.75.75.75']
```

5.c ¿Cuál es la IP origen más frecuente?

- ¿A qué IP destino se comunica con más frecuencia?
- ¿A que puerto destino se comunica? ¿Cuál es el propósito de este puerto?
- ¿Desde que puertos origen se comunica?

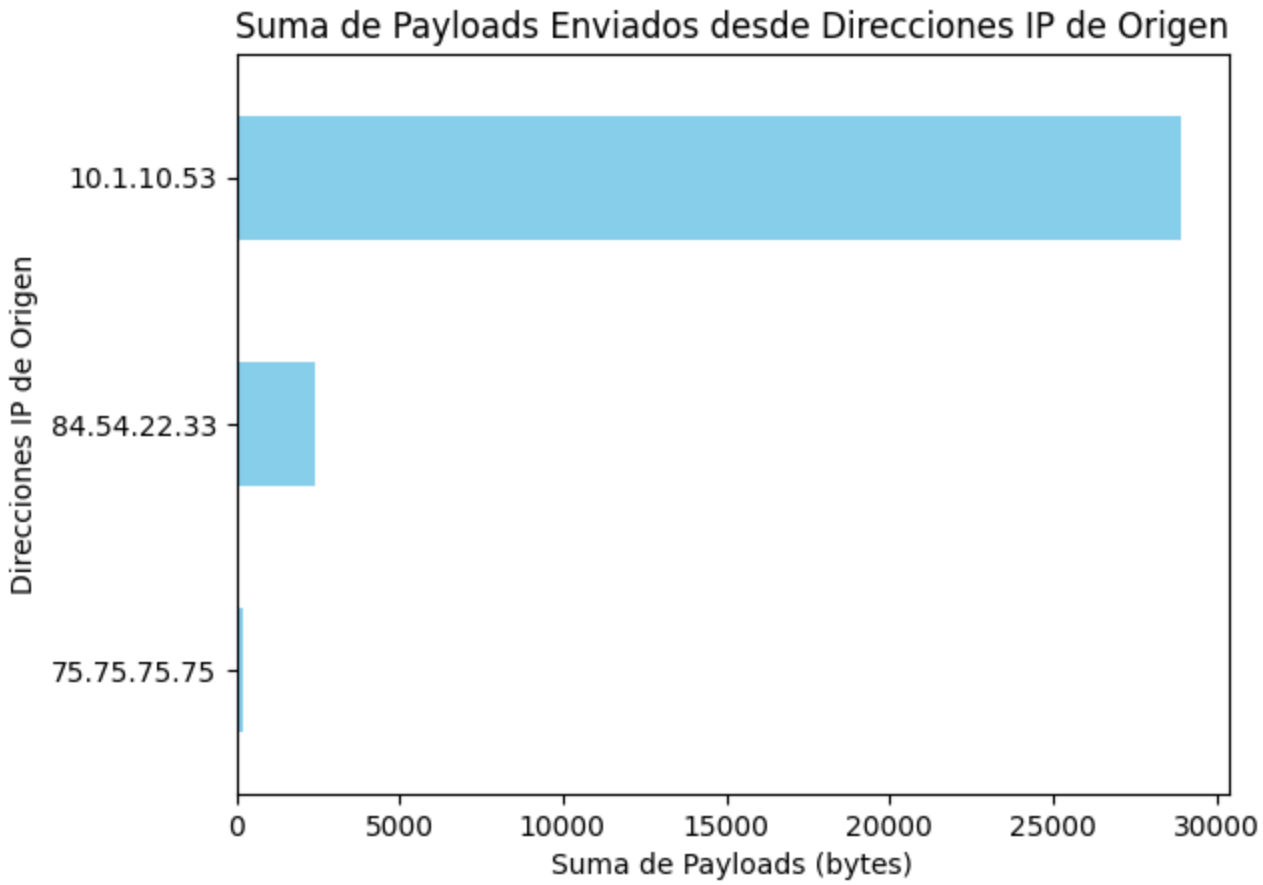
```
ip_origen_frecuente = df_paquetes['Src Address'].mode()[0]
print(f"IP origen más frecuente: {ip_origen_frecuente}")

datos_ip_frecuente = df_paquetes[df_paquetes['Src Address'] == ip_origen_frecuente]

ip_destino_frecuente = datos_ip_frecuente['Dst Address'].mode()[0]
print(f"IP destino más frecuente para la IP origen {ip_origen_frecuente}: {ip_destino_fr

puerto_destino_frecuente = datos_ip_frecuente['Dst Port'].mode()[0]
print(f"Puerto destino más frecuente para la IP origen {ip_origen_frecuente}: {puerto_de

puertos_origen = datos_ip_frecuente['Src Port'].unique()
print(f"Puertos origen desde los que se comunica la IP {ip_origen_frecuente}: {puertos_o
```

```
IP origen más frecuente: 10.1.10.53
IP destino más frecuente para la IP origen 10.1.10.53: 84.54.22.33
Puerto destino más frecuente para la IP origen 10.1.10.53: 53
Puertos origen desde los que se comunica la IP 10.1.10.53: [   53 15812 23903]
```

6.a. Genere una gráfica de barras 2D horizontales, en el eje Y las IPs origen, y en el eje X lasuma de los payloads (bytes) enviados desde dichas direcciones.

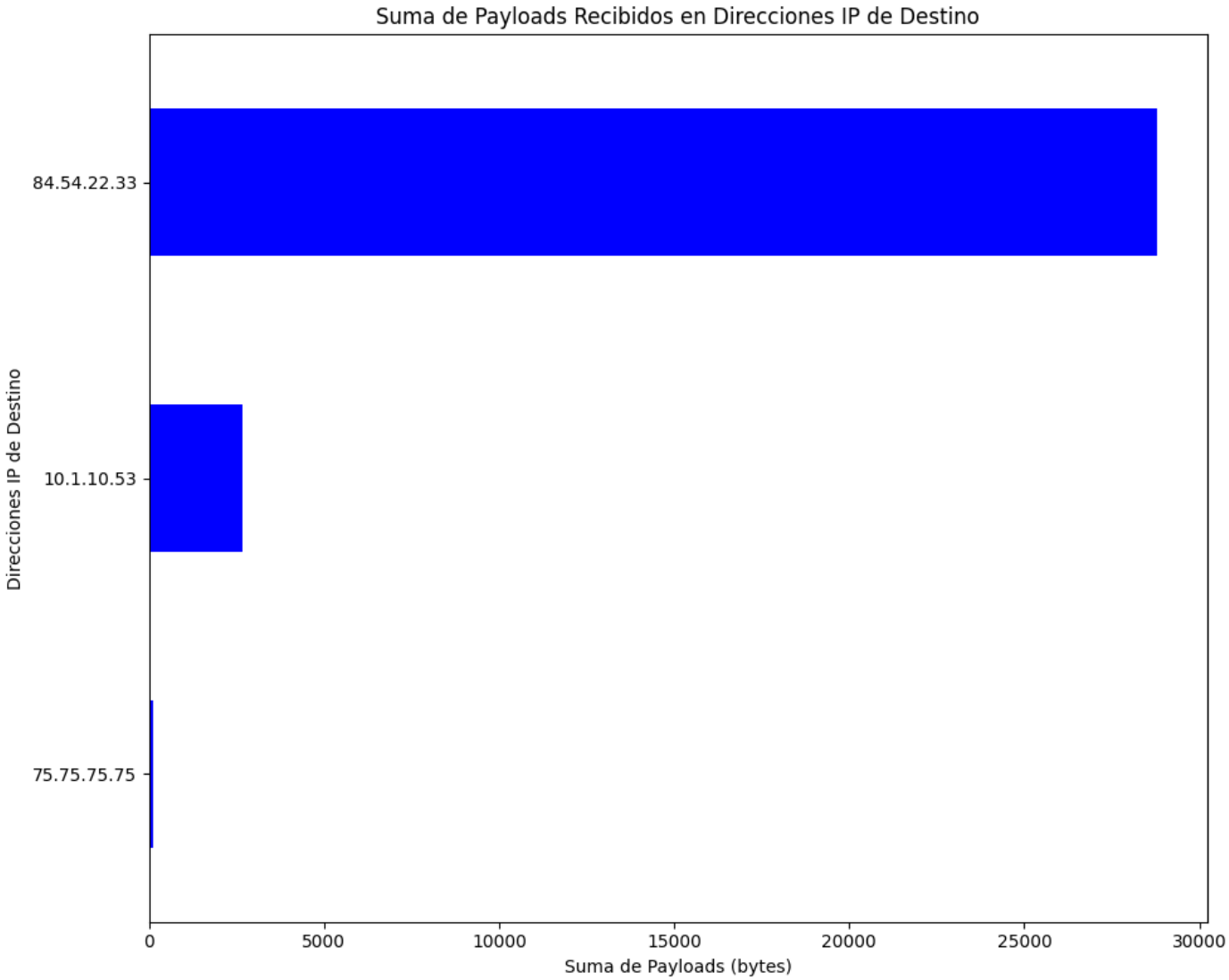
```
sum_payloads = df_paquetes.groupby('Src Address')['Payload Size'].sum()
```

```
sum_payloads.sort_values().plot(kind='barh', color='skyblue')
plt.xlabel('Suma de Payloads (bytes)')
plt.ylabel('Direcciones IP de Origen')
plt.title('Suma de Payloads Enviados desde Direcciones IP de Origen')
plt.show()
```



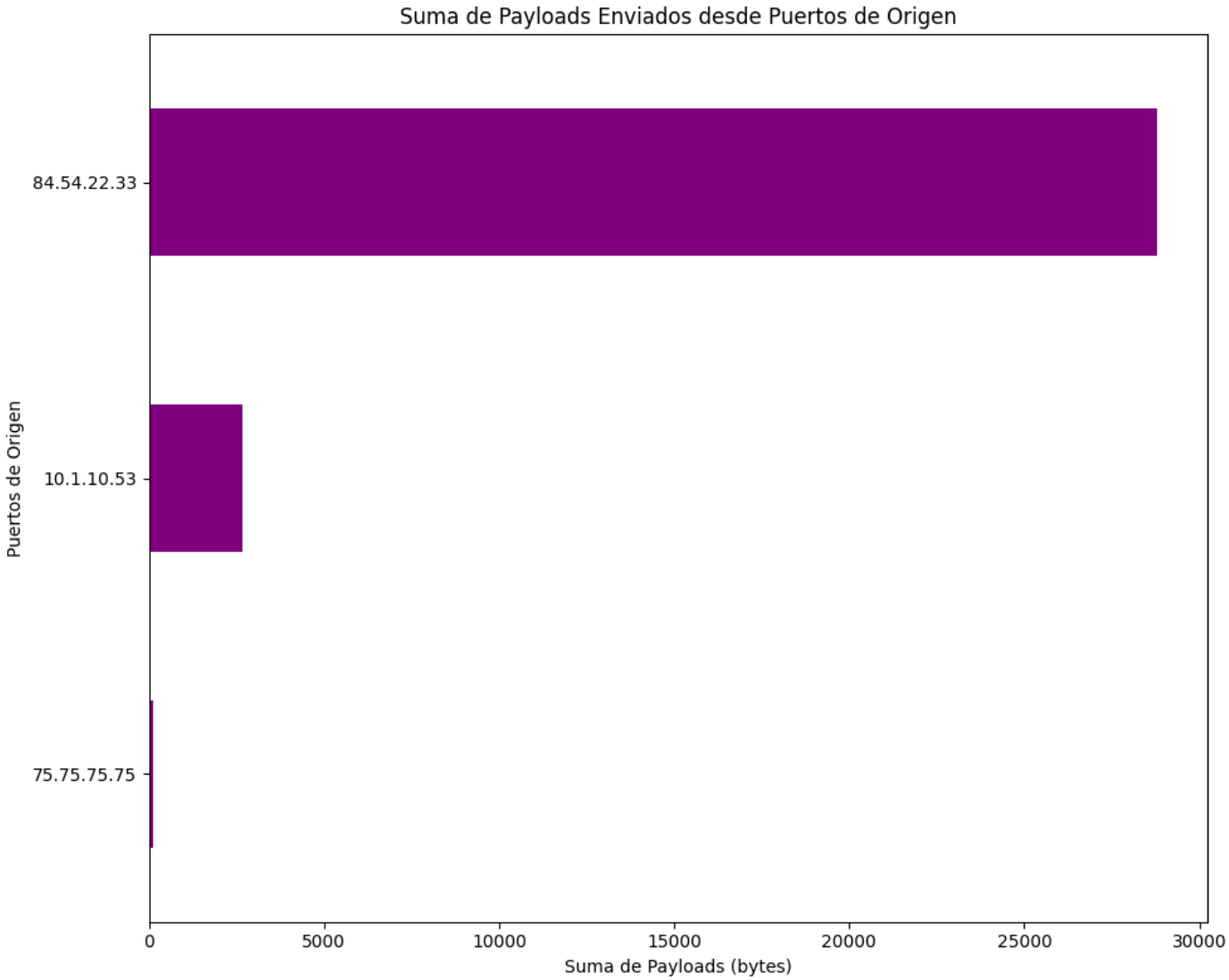
6.b. Genere una gráfica de barras 2D horizontales, en el eje Y las IP destino, y en el eje Xla suma de los payloads (bytes) recibidos en dichas direcciones.

```
payloads_por_destino = df_paquetes.groupby('Dst Address')['Payload Size'].sum()

payloads_ordenados = payloads_por_destino.sort_values()

# Crear gráfica de barras horizontal
plt.figure(figsize=(10, 8))
payloads_ordenados.plot(kind='barh', color='blue')
plt.xlabel('Suma de Payloads (bytes)')
plt.ylabel('Direcciones IP de Destino')
plt.title('Suma de Payloads Recibidos en Direcciones IP de Destino')
plt.tight_layout()
plt.show()
```



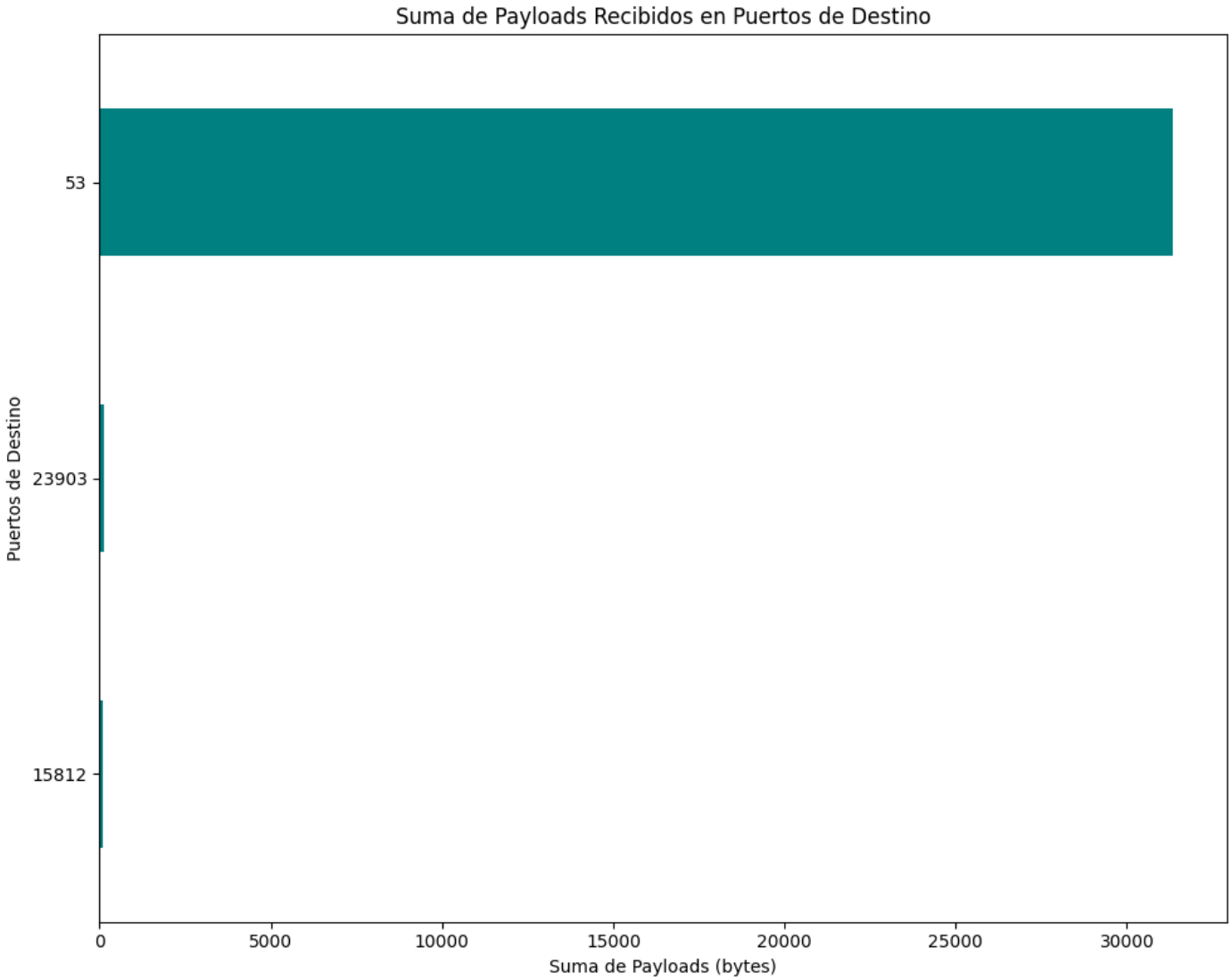
6. c. Genere una gráfica de barras 2D horizontales, en el eje Y los puertos origen, y en el eje X la suma de los payloads (bytes) enviados de dichos puertos.

```
payloads_por_puerto = df_paquetes.groupby('Src Port')['Payload Size'].sum()

plt.figure(figsize=(10, 8))
payloads_ordenados.plot(kind='barh', color='purple')
plt.xlabel('Suma de Payloads (bytes)')
plt.ylabel('Puertos de Origen')
plt.title('Suma de Payloads Enviados desde Puertos de Origen')
plt.tight_layout()
plt.show()
```



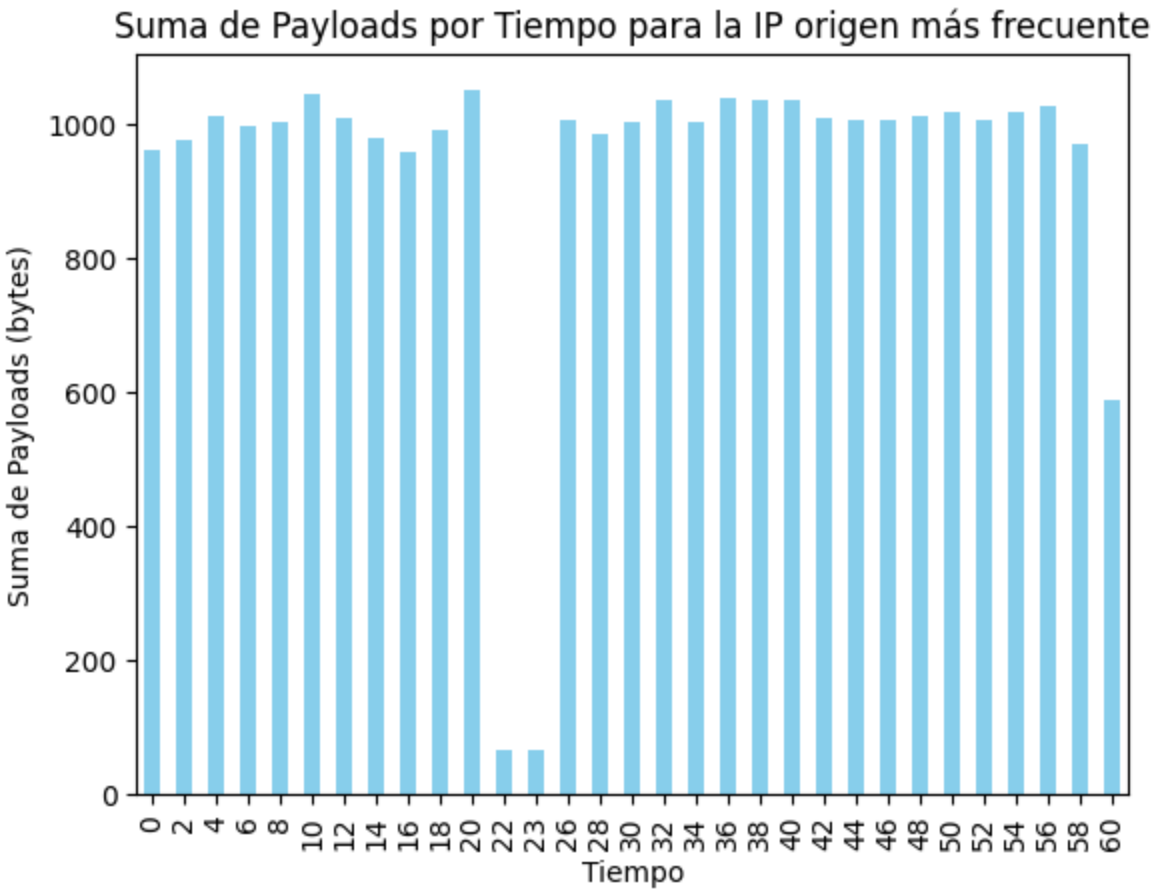
6. d. Genere una gráfica 2D de barras horizontales, en el eje Y los puertos destino, y en el eje X la suma de los payloads (bytes) recibidos en dichos puertos.

```
payloads_por_puerto_destino = df_paquetes.groupby('Dst Port')['Payload Size'].sum()

payloads_ordenados = payloads_por_puerto_destino.sort_values()

# Crear gráfica de barras horizontal
plt.figure(figsize=(10, 8))
payloads_ordenados.plot(kind='barh', color='teal')
plt.xlabel('Suma de Payloads (bytes)')
plt.ylabel('Puertos de Destino')
plt.title('Suma de Payloads Recibidos en Puertos de Destino')
plt.tight_layout()
plt.show()
```



6.e. Genere una gráfica de barras 2D verticales, en el eje Y la suma de los payload, en el eje X el tiempo, para la IP origen más frecuente.

```
payloads_por_tiempo = df_paquetes[df_paquetes['Src Address'] == ip_origen_frecuente].res
payloads_por_tiempo.plot(kind='bar', color='skyblue')
plt.ylabel('Suma de Payloads (bytes)')
plt.xlabel('Tiempo')
plt.title('Suma de Payloads por Tiempo para la IP origen más frecuente')
plt.show()
```



f. Utilizando la información de las estadísticas y la información del comportamiento del tráfico que las gráficas muestran, describa que es lo que está sucediendo. ¿Es común el comportamiento?

Al observar la grafica de tiempos se puede ver que la suma de bytes enviada a esta IP es constante, siendo valores muy cercanos al a 1GB de iformación. Ese trafico se ve en el puerto 53 de la IP lo que indica que son solicitudes entrantes al servicio DNS del servidor. Es necesario saber en que contexto esta ocurriendo todo esto para poder determina si es actividad maliciosa.

Analisis de Payload

a. Cree un nuevo DF que incluya únicamente las conexiones con la dirección IP origen más frecuente.

```
ip_origen_frecuente = df_paquetes['Src Address'].mode()[0]

df_ip_frecuente = df_paquetes[df_paquetes['Src Address'] == ip_origen_frecuente]

df_ip_frecuente
```

	Src Address	Dst Address	Src Port	Dst Port	Payload Size	Timestamp
0	10.1.10.53	84.54.22.33	53	53	961	1.532199e+09

	Src Address	Dst Address	Src Port	Dst Port	Payload Size	Timestamp
2	10.1.10.53	84.54.22.33	53	53	975	1.532199e+09
4	10.1.10.53	84.54.22.33	53	53	1012	1.532199e+09
6	10.1.10.53	84.54.22.33	53	53	998	1.532199e+09
8	10.1.10.53	84.54.22.33	53	53	1003	1.532199e+09
10	10.1.10.53	84.54.22.33	53	53	1045	1.532199e+09
12	10.1.10.53	84.54.22.33	53	53	1008	1.532199e+09
14	10.1.10.53	84.54.22.33	53	53	979	1.532199e+09
16	10.1.10.53	84.54.22.33	53	53	959	1.532199e+09
18	10.1.10.53	84.54.22.33	53	53	992	1.532199e+09
20	10.1.10.53	84.54.22.33	53	53	1051	1.532199e+09
22	10.1.10.53	75.75.75.75	15812	53	65	1.532199e+09
23	10.1.10.53	75.75.75.75	23903	53	65	1.532199e+09
26	10.1.10.53	84.54.22.33	53	53	1006	1.532199e+09
28	10.1.10.53	84.54.22.33	53	53	986	1.532199e+09
30	10.1.10.53	84.54.22.33	53	53	1004	1.532199e+09
32	10.1.10.53	84.54.22.33	53	53	1037	1.532199e+09
34	10.1.10.53	84.54.22.33	53	53	1004	1.532199e+09
36	10.1.10.53	84.54.22.33	53	53	1039	1.532199e+09
38	10.1.10.53	84.54.22.33	53	53	1037	1.532199e+09
40	10.1.10.53	84.54.22.33	53	53	1035	1.532199e+09
42	10.1.10.53	84.54.22.33	53	53	1010	1.532199e+09
44	10.1.10.53	84.54.22.33	53	53	1006	1.532199e+09
46	10.1.10.53	84.54.22.33	53	53	1006	1.532199e+09
48	10.1.10.53	84.54.22.33	53	53	1013	1.532199e+09
50	10.1.10.53	84.54.22.33	53	53	1019	1.532199e+09
52	10.1.10.53	84.54.22.33	53	53	1005	1.532199e+09
54	10.1.10.53	84.54.22.33	53	53	1017	1.532199e+09
56	10.1.10.53	84.54.22.33	53	53	1027	1.532199e+09
58	10.1.10.53	84.54.22.33	53	53	969	1.532199e+09
60	10.1.10.53	84.54.22.33	53	53	588	1.532199e+09

b. Cree un nuevo DF que utilice el DF anterior con las columnas src, dst y payload y agrúpelas por dst y la suma del payload

```
df_agrupado = df_ip_frecuente.groupby('Dst Address')['Payload Size'].sum().reset_index()
```

c. Obtenga la IP destino que más ha intercambiado bytes con la IP más frecuente. Esta IP es sospechosa por la cantidad de bytes intercambiados, entre todas las direcciones.

```
ip_sospechosa = df_agrupado[df_agrupado['Payload Size'] == df_agrupado['Payload Size'].m
```

d. Cree un nuevo DF con la conversación entre la IP más frecuente y la IP sospechosa.

```
df_conversacion = df_ip_frecuente[(df_ip_frecuente['Src Address'] == ip_origen_frecuente
df_conversacion.head()
```

	Src Address	Dst Address	Src Port	Dst Port	Payload Size	Timestamp
0	10.1.10.53	84.54.22.33	53	53	961	1.532199e+09
2	10.1.10.53	84.54.22.33	53	53	975	1.532199e+09
4	10.1.10.53	84.54.22.33	53	53	1012	1.532199e+09
6	10.1.10.53	84.54.22.33	53	53	998	1.532199e+09
8	10.1.10.53	84.54.22.33	53	53	1003	1.532199e+09

e. Obtenga los payloads del DF del inciso anterior, y añada cada uno en un array.

```
array_payloads = df_conversacion['Payload Size'].to_numpy()
```

f. Muestre el contenido del array.

```
print(array_payloads)
```

```
[ 961  975 1012  998 1003 1045 1008  979  959  992 1051 1006  986 1004
 1037 1004 1039 1037 1035 1010 1006 1006 1013 1019 1005 1017 1027  969
  588]
```

g. Observe los primeros bytes del contenido, ¿encuentra algún dato que no haga sentido que se envíe al puerto destino? Describa lo que encontró

```
print("\nBytes del payload:")
for payload in array_payloads:
    print("Primeros bytes:", str(payload)[:10])
```

```
Bytes del payload:
Primeros bytes: 961
Primeros bytes: 975
Primeros bytes: 1012
Primeros bytes: 998
Primeros bytes: 1003
Primeros bytes: 1045
```

```
Primeros bytes: 1008
Primeros bytes: 979
Primeros bytes: 959
Primeros bytes: 992
Primeros bytes: 1051
Primeros bytes: 1006
Primeros bytes: 986
Primeros bytes: 1004
Primeros bytes: 1037
Primeros bytes: 1004
Primeros bytes: 1039
Primeros bytes: 1037
Primeros bytes: 1035
Primeros bytes: 1010
Primeros bytes: 1006
Primeros bytes: 1006
Primeros bytes: 1013
Primeros bytes: 1019
Primeros bytes: 1005
Primeros bytes: 1017
Primeros bytes: 1027
Primeros bytes: 969
Primeros bytes: 588
```

Al observar el contenido se puede ver que es trafico normal y no hay indicios de una actividad maliciosa. Este trafico es comun en una red y no se ve nada sospechoso en el contenido de los paquetes.