Análisis IPs

Garcia Justo

1 Análisis de IPs reportadas como atacantes por SSH

Los ataques por SSH (Secure Shell) son intentos maliciosos de comprometer un sistema a través del protocolo SSH.

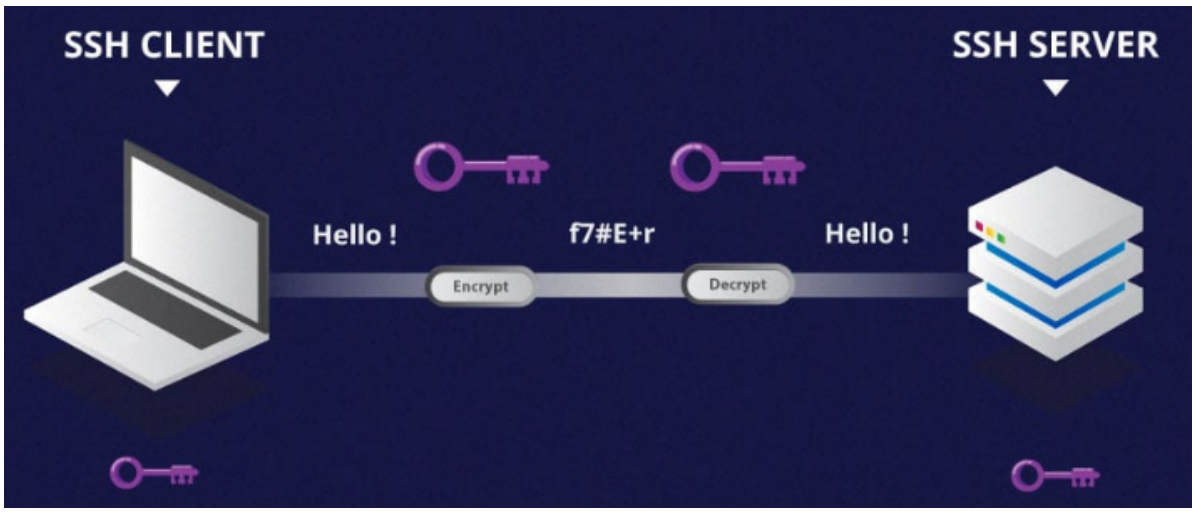


Figure 1: Protocolo SSH

SSH es muy utilizado porque permite establecer una conexión segura y cifrada entre un cliente y un servidor. Pero, establecida la conexión, puede ser también aprovechado por los atacantes.

Dado que es información proporcionada, no se sabe la naturaleza del ataque, pero entre los más comunes estan los de fuerza bruta.

1.1 Extracción de IPs

Para la realización de este análisis la cátedra proporcionó una lista de IPs que fueron reportadas por conexiones SSH y ataques DDoS. En este caso extraeré las IPs de SSH.

```
patron = "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
!grep -Eo "$patron" data/SSH.txt > data/IPsSSH.txt
```

```
with open("data/IPsSSH.txt") as ips:
    ipSSH = ips.read()

ipSSH = ipSSH.replace("\n", " ").split()
ipSSH
```

```
['54.144.244.57',
 '188.166.216.223',
 '220.94.228.162',
 '218.92.0.99',
 '116.193.159.2',
 '109.117.92.13',
 '167.99.112.43',
 '89.248.163.219',
 '143.198.204.177',
 '61.177.173.45',
 '8.222.204.225',
 '220.135.119.188']
```

1.2 Importamos la implementación de las peticiones a la API

```
!cp ../app/modulos/abuseIPDB.py modulos/abuseIPDB.py
```

```
#Importo los modulos necesarios
from modulos.abuseIPDB import AbuseIPDB
import pandas as pd
```

```
#Construyo el objeto
apiAbuse = AbuseIPDB()
```

```
import os
import seaborn as sns
```

```
#Declaro las keys de la info que devuelve mi implementación de requests
keys = ['esPublica', 'estaEnWhitelist', 'scoreAbuso', 'pais', 'codigoPais', 'isp', 'tipoDe

diccDf = {'ip' : []}

if os.path.isfile("data/ipSSH.csv"):

    df = pd.read_csv("data/ipSSH.csv")

else:
    for ip in ipSSH:
        diccDf['ip'].append(ip)
        info = apiAbuse.getInfo(ip)
        for key in keys:
            if key in diccDf:
                diccDf[key].append(info[key])
            else:
                diccDf[key] = [info[key]]

        df = pd.DataFrame(data=diccDf)
```

```
df
```

	ip	esPublica	estaEnWhitelist	scoreAbuso	pais	codigoPais	isp
0	54.144.244.57	True	False	55	NaN	US	Amazon Data Services
1	188.166.216.223	True	False	100	NaN	SG	DigitalOcean LLC
2	220.94.228.162	True	False	100	NaN	KR	KT Corporation
3	218.92.0.99	True	False	100	NaN	CN	ChinaNet Jiangsu Provi
4	116.193.159.2	True	False	100	NaN	HK	Pacswitch Globe Teleco
5	109.117.92.13	True	False	100	NaN	IT	Vodafone Italia S.p.A.
6	167.99.112.43	True	False	100	NaN	US	DigitalOcean LLC
7	89.248.163.219	True	False	100	NaN	NL	FiberXpress BV
8	143.198.204.177	True	False	100	NaN	SG	DigitalOcean LLC
9	61.177.173.45	True	False	100	NaN	CN	ChinaNet Jiangsu Provi
10	8.222.204.225	True	False	100	NaN	SG	Alibaba.com Singapore
11	220.135.119.188	True	False	100	NaN	TW	Chunghwa Telecom Co.

```
unameds = [i for i in df.columns if 'Unnamed' in i]
for i in unameds:
    df.drop(i, axis=1, inplace=True)
```

```
df.to_csv("data/ipSSH.csv", index=False)
```

1.3 Índices de abuso

```
recuento = df["scoreAbuso"].value_counts().to_dict()

pd.DataFrame(data={"Score": list(recuento.keys()), "Reportes": list(recuento.values())})
```

Table 2: Indice de abuso y las veces que se repite

	Score	Reportes
0	100	11
1	55	1

Como se puede apreciar en la tabla de arriba, han sido múltiples veces reportadas por distintos usuarios a lo largo del mundo. Por lo tanto, tenemos la certeza de que son IPs que han sido utilizadas con fines malintencionados.

1.4 Análisis de procedencia

```
import pycountry
```

```
df['pais'] = df['codigoPais'].apply(lambda codigo: pycountry.countries.get(alpha_2=codigo)
dfgdp = df.copy()
dfgdp['codigoPais'] = df['pais'].apply(lambda nombre: pycountry.countries.search_fuzzy(nom
```

```
import geopandas as gpd
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np

mapa = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

> **Note**
>
> Todas estas librerias utilizan convenciones, por lo cual es importante checkear que esten presentes todos los paises que queremos plotear

```
print(np.unique(dfgdp["codigoPais"].loc[~dfgdp["codigoPais"].isin(mapa["iso_a3"])]))
```

```
['HKG' 'SGP']
```

Pude notar que tanto Hong Kong, como Singapur no estan representadas en el mapa mundi por ser ciudades. Por ello, debo cargarlas desde otro dataset

```
paisesMarcados = mapa[mapa['iso_a3'].isin(dfgdp["codigoPais"])]

fig, ax = plt.subplots(figsize=(15, 10))

mapa.plot(ax=ax, edgecolor='grey', color='lightgrey')
paisesMarcados.plot(ax=ax, edgecolor='black', color='red')

ciudades = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))

singapur = ciudades[ciudades['name'] == 'Singapore']
hongkong = ciudades[ciudades['name'] == 'Hong Kong']

singapur.plot(ax=ax, edgecolor='black', color='blue')
hongkong.plot(ax=ax, edgecolor='black', color='blue')

plt.show()
```

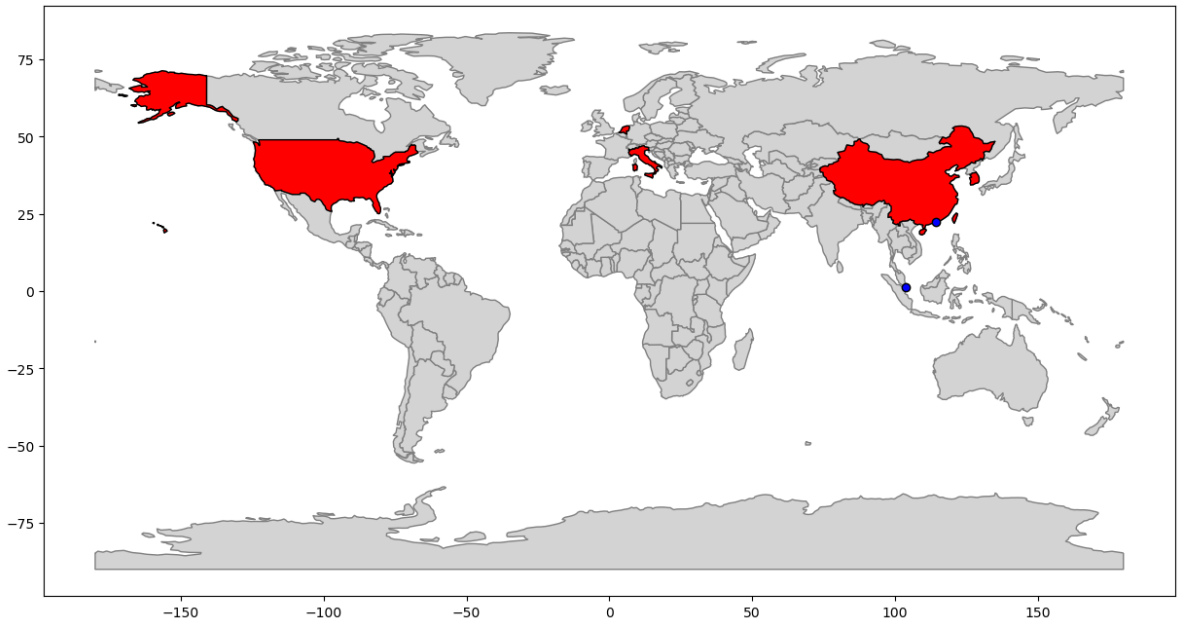


Figure 2: Mapa con los lugares del que proceden las IPs

```
counts = dfgdp["codigoPais"].value_counts().rename_axis('pais').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')

fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["pais"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```

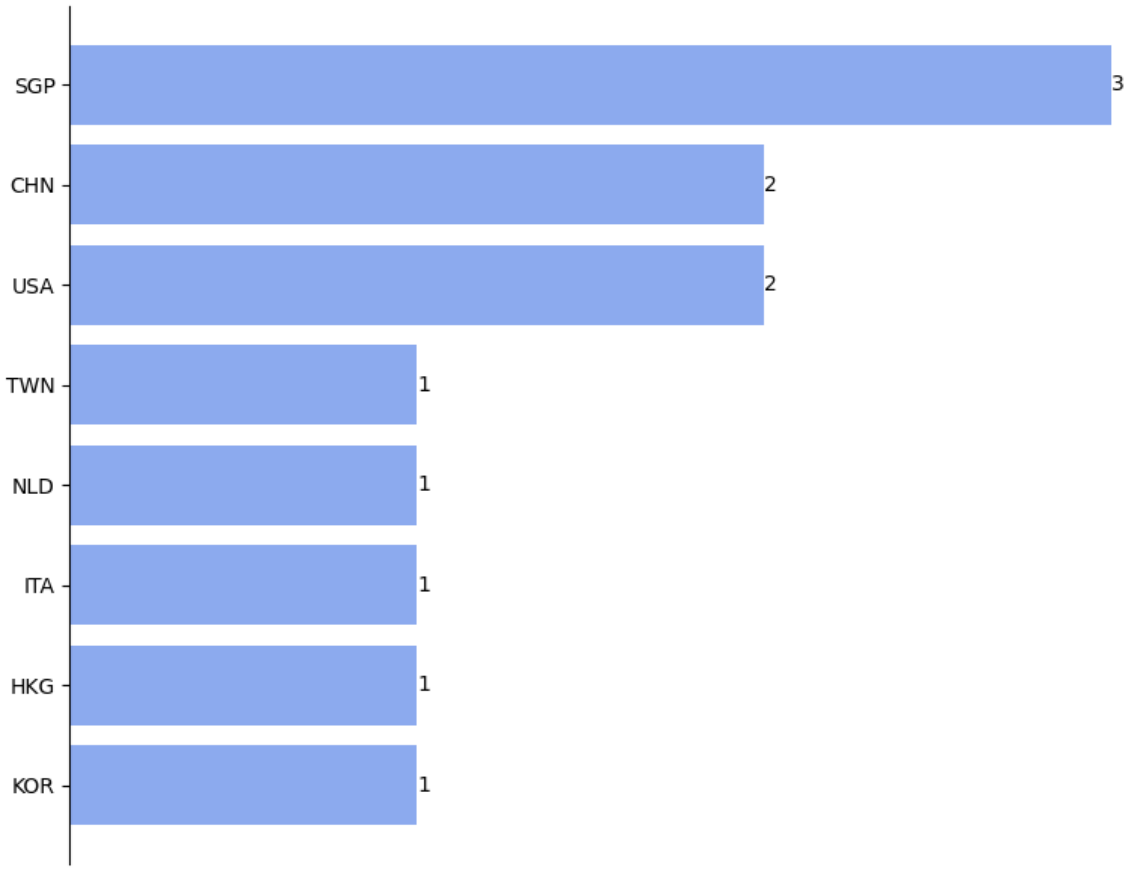


Figure 3: Recuento del número de reportes

```
recuento = dfgdp["codigoPais"].value_counts().to_dict()

dfPlot = pd.DataFrame(data={"codigoPais": list(recuento.keys()), "Reportes": list(recuento
dfPlot

fig = px.scatter_geo(dfPlot, locations="codigoPais", color="codigoPais", size="Reportes",
                     projection="equirectangular")
fig.write_image("data/plotlySSH.png")
```

```
from PIL import Image

image = np.asarray(Image.open('data/plotlySSH.png'))
plt.imshow(image)
```

```
plt.grid(False)
plt.axis(False)
plt.show()
```

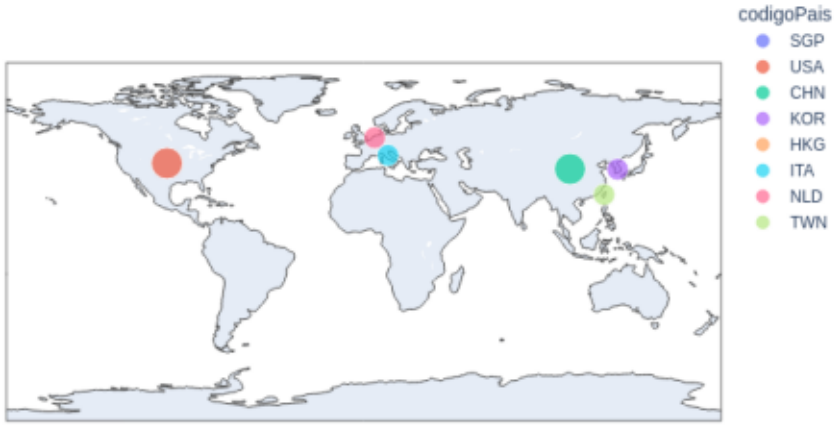


Figure 4: Scatterplot con tamaño en función de número de reportes

```
recuento = df["pais"].value_counts().to_dict()

pd.DataFrame(data={"Pais": list(recuento.keys()), "Reportes": list(recuento.values())})
```

	Pais	Reportes
0	Singapore	3
1	United States	2
2	China	2
3	Korea, Republic of	1
4	Hong Kong	1
5	Italy	1
6	Netherlands	1
7	Taiwan, Province of China	1

1.5 Análisis de frecuencia

Para así poder de tratar de identificar cierto patron asociado a la hora de ataque.

1.5.1 Extracción de información

```
patron = "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
!grep -Eo "$patron" data/SSH.txt > data/IPsSSH.txt

patron = "[a-z]{3}\/[0-9]{2}\/[0-9]{4}"
!grep -Eo "$patron" data/SSH.txt >> data/IPsSSH.txt

patron = "[0-9]{2}\:[0-9]{2}\:[0-9]{2}"
!grep -Eo "$patron" data/SSH.txt >> data/IPsSSH.txt
```

```
with open("data/IPsSSH.txt") as ips:
    data = ips.read()
    data = data.replace("\n", " ").split()
```

```
for i in range(int(len(data)/3)):
    data[i] = data[i] + " " + data[int(len(data)/3)+i] + " " + data[(int(len(data)/3))*2+i

data = data[:int(len(data)/3)]
```

```
from datetime import datetime, time
diccInfo = {
    "IP": [],
    "Fecha": [],
    "Hora": []
}
eventos = []

for i in data:
    diccInfo["IP"].append(i.split()[0])
    diccInfo["Fecha"].append(i.split()[1])
    mes = 5
    dia = int(i.split()[1].split(sep="/")[1])
    año = int(i.split()[1].split(sep="/")[2])
    h = int(i.split()[2].split(sep=":")[0])
```

```
    m = int(i.split()[2].split(sep=":")[1])
    s = int(i.split()[2].split(sep=":")[2])
    diccInfo["Hora"].append(time(hour=int(h), minute=int(m), second=int(s)))
    #diccInfo["Hora"].append(i.split()[2])

    eventos.append((i.split()[0],datetime(year=año, month=mes, day=dia, hour=h, minute=m))
```

```
dfHora = pd.DataFrame(data=diccInfo)
```

```
dfHora["Pais"] = None
for index, row in dfHora.iterrows():
    ip = row["IP"]
    row["Pais"] = df[df['ip'] == ip].iloc[0]['pais']
```

```
fig, ax = plt.subplots()

fecha = [evento[1] for evento in eventos]
etiquetas = [evento[0] for evento in eventos]
ax.eventplot(fecha, lineoffsets=0.1, linelengths=0.1, color='r')
ax.set_ylabel(None)
ax.set_yticklabels([])
ax.set_xlim(datetime(2023, 5, 23, 0, 0), datetime(2023, 5, 23, 23, 59))
fig.autofmt_xdate()
```

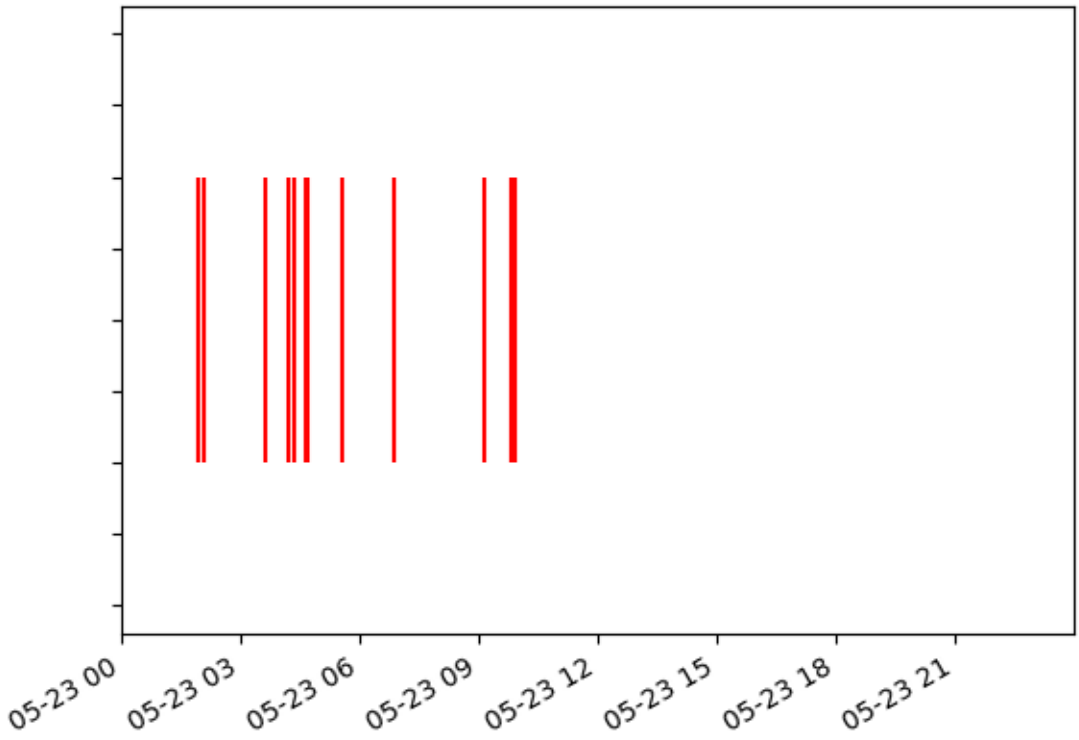


Figure 5: Visualización de eventos de SSH

1.6 Análisis de ISPs

```
recuento = df["isp"].value_counts().to_dict()

recuento = pd.DataFrame(data={"ISP": list(recuento.keys()), "Reportes": list(recuento.valu
recuento
```

	ISP	Reportes
0	DigitalOcean LLC	3
1	ChinaNet Jiangsu Province Network	2
2	Amazon Data Services NoVa	1
3	KT Corporation	1
4	Pacswitch Globe Telecom Limited	1
5	Vodafone Italia S.p.A.	1
6	FiberXpress BV	1
7	Alibaba.com Singapore E-Commerce Private Limited	1

	ISP	Reportes
8	Chunghwa Telecom Co. Ltd.	1

```
counts = df["isp"].value_counts().rename_axis('isp').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')

fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["isp"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```

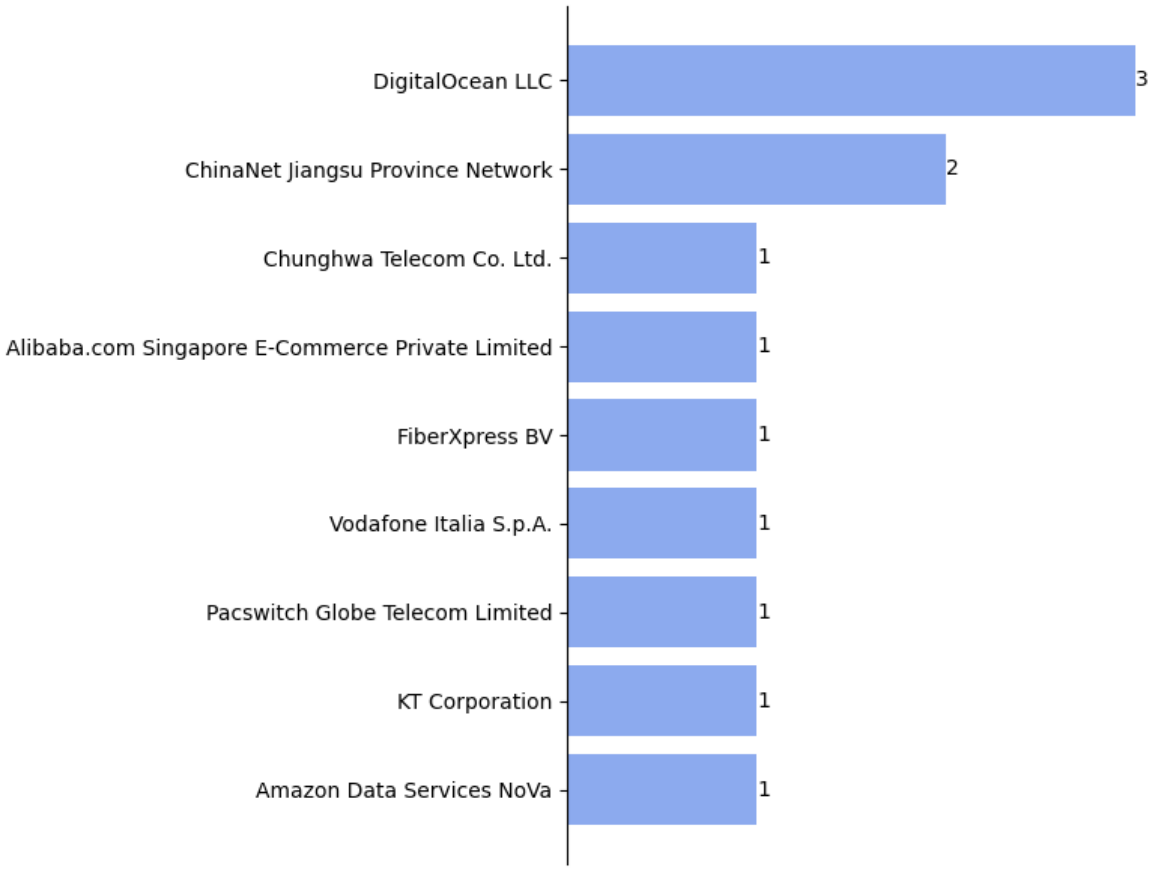


Figure 6: Recuento del número de veces que fue reportado un ISP

1.7 Análisis de uso

```
recuento = df["tipoDeUso"].value_counts().to_dict()

recuento = pd.DataFrame(data={"uso": list(recuento.keys()), "Reportes": list(recuento.valu
recuento
```

	uso	Reportes
0	Data Center/Web Hosting/Transit	8
1	Fixed Line ISP	1

```
counts = df["tipoDeUso"].value_counts().rename_axis('uso').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')

fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["uso"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```

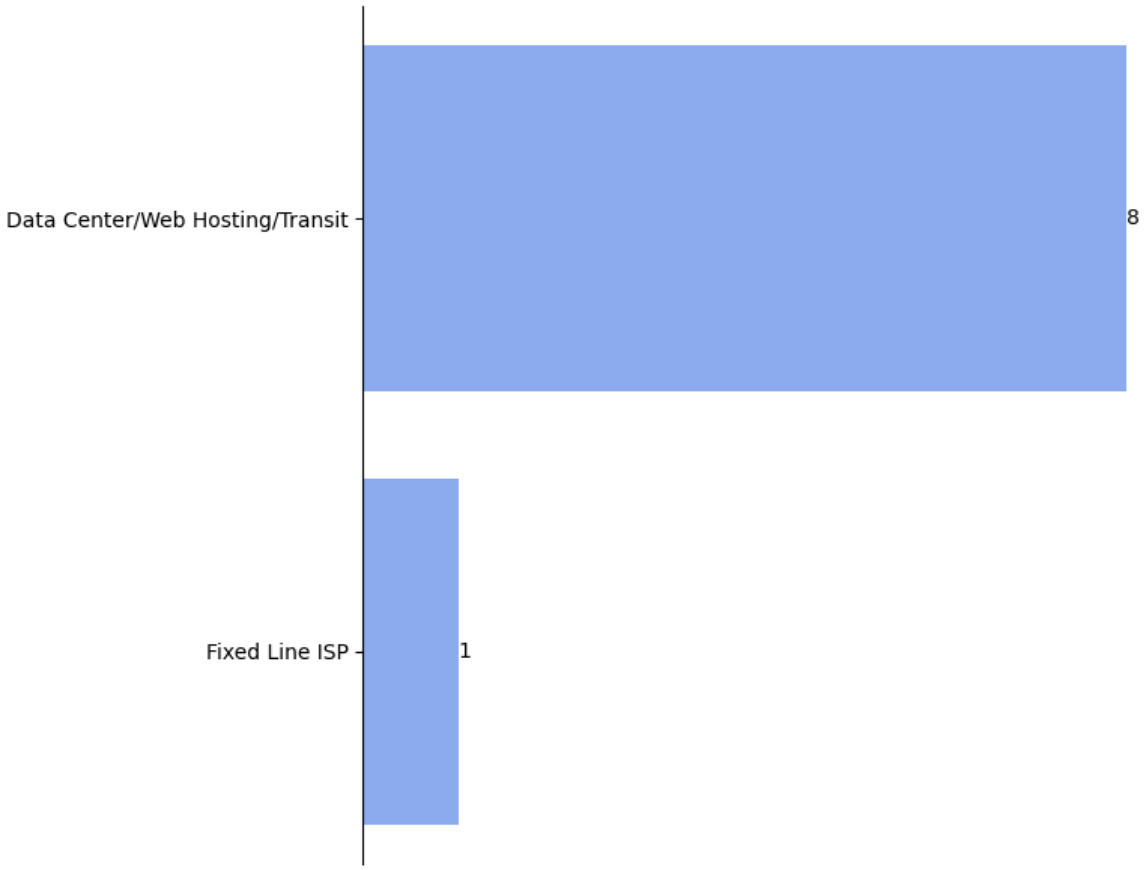


Figure 7: Recuento de los usos que se les da a las IPs

2 Análisis de IPs reportadas por ataque DDoS

2.1 Extracción de información

```
patron = "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
!grep -Eo "$patron" data/DDOS.txt > data/IPsDDoS.txt

patron = "[a-z]{3}\/[0-9]{2}\/[0-9]{4}"
!grep -Eo "$patron" data/DDOS.txt >> data/IPsDDoS.txt

patron = "[0-9]{2}\:[0-9]{2}\:[0-9]{2}"
!grep -Eo "$patron" data/DDOS.txt >> data/IPsDDoS.txt
```

```
with open("data/IPsDDoS.txt") as ips:
    data = ips.read()
    data = data.replace("\n", " ").split()
```

```
ips = []
dia = []
hora = []
for i in range(int(len(data)/3)):
    ips.append(data[i])
    dia.append(data[int(len(data)/3)+i])
    hora.append(data[(int(len(data)/3))*2+i])
    data[i] = data[i] + " " + data[int(len(data)/3)+i] + " " + data[(int(len(data)/3))*2+i
```

```
data = data[:int(len(data)/3)]
```

```
data = pd.DataFrame(data={'ip': ips,
                          'dia': dia,
                          'hora': hora})
```

```
apiAbuse = AbuseIPDB()
```

```
from IPython.display import clear_output
#Declaro las keys de la info que devuelve mi implementación de requests
keys = ['esPublica', 'estaEnWhitelist', 'scoreAbuso', 'pais', 'codigoPais', 'isp', 'tipoDe

diccDf = {'ip' : []}

if os.path.isfile("data/ipDDOS.csv"):

    df = pd.read_csv("data/ipDDOS.csv")

else:
    for ip, i in zip(ips, range(len(ips))):
        clear_output()
        print(f"{i}/{len(ips)}")
        diccDf['ip'].append(ip)
        info = apiAbuse.getInfo(ip)
        for key in keys:
            if key in diccDf:
```

```
                diccDf[key].append(info[key])
            else:
                diccDf[key] = [info[key]]

    df = pd.DataFrame(data=diccDf)


df["hora"] = data["hora"]
df["dia"] = data["dia"]
```

```
df
```

	ip	esPublica	estaEnWhitelist	scoreAbuso	pais	codigoPais	isp
0	45.204.126.117	True	NaN	0	NaN	HK	Intercontinental Interne
1	59.153.100.70	True	NaN	0	NaN	BD	Dot Internet
2	1.32.249.141	True	NaN	0	NaN	HK	CTG Server Ltd.
3	103.116.15.134	True	NaN	0	NaN	TW	Shine Telecom Co. Ltd.
4	43.225.58.180	True	NaN	0	NaN	HK	Dragon Spirit Investmer
...	...	...	...	...	...	...	...
373	43.225.58.251	True	NaN	0	NaN	HK	Dragon Spirit Investmer
374	43.225.58.88	True	NaN	0	NaN	HK	Dragon Spirit Investmer
375	43.225.58.182	True	NaN	0	NaN	HK	Dragon Spirit Investmer
376	103.119.129.64	True	NaN	0	NaN	HK	Suniway Group Limited
377	43.225.58.19	True	NaN	0	NaN	HK	Dragon Spirit Investmer

```
unameds = [i for i in df.columns if 'Unnamed' in i]
for i in unameds:
    df.drop(i, axis=1, inplace=True)

df.to_csv("data/ipDDOS.csv", index=False)
```

2.2 Índices de abuso

```
recuento = df["scoreAbuso"].value_counts().to_dict()

pd.DataFrame(data={"Pais": list(recuento.keys()), "Reportes": list(recuento.values())})
```

Table 7: Indice de abuso y las veces que se repite

	Pais	Reportes
0	0	375
1	2	2
2	10	1

2.3 Análisis de procedencia

```
import pycountry
```

```
df['pais'] = df['codigoPais'].apply(lambda codigo: pycountry.countries.get(alpha_2=codigo)
dfgdp = df.copy()
dfgdp['codigoPais'] = df['pais'].apply(lambda nombre: pycountry.countries.search_fuzzy(nom
```

```
import geopandas as gpd
import matplotlib.pyplot as plt

mapa = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

```
print(np.unique(dfgdp["codigoPais"].loc[~dfgdp["codigoPais"].isin(mapa["iso_a3"])]))
```

```
['HKG']
```

```
paisesMarcados = mapa[mapa['iso_a3'].isin(dfgdp["codigoPais"])]

fig, ax = plt.subplots(figsize=(15, 10))

mapa.plot(ax=ax, edgecolor='grey', color='lightgrey')
paisesMarcados.plot(ax=ax, edgecolor='black', color='red')

ciudades = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))

hongkong = ciudades[ciudades['name'] == 'Hong Kong']

hongkong.plot(ax=ax, edgecolor='black', color='blue')

plt.show()
```

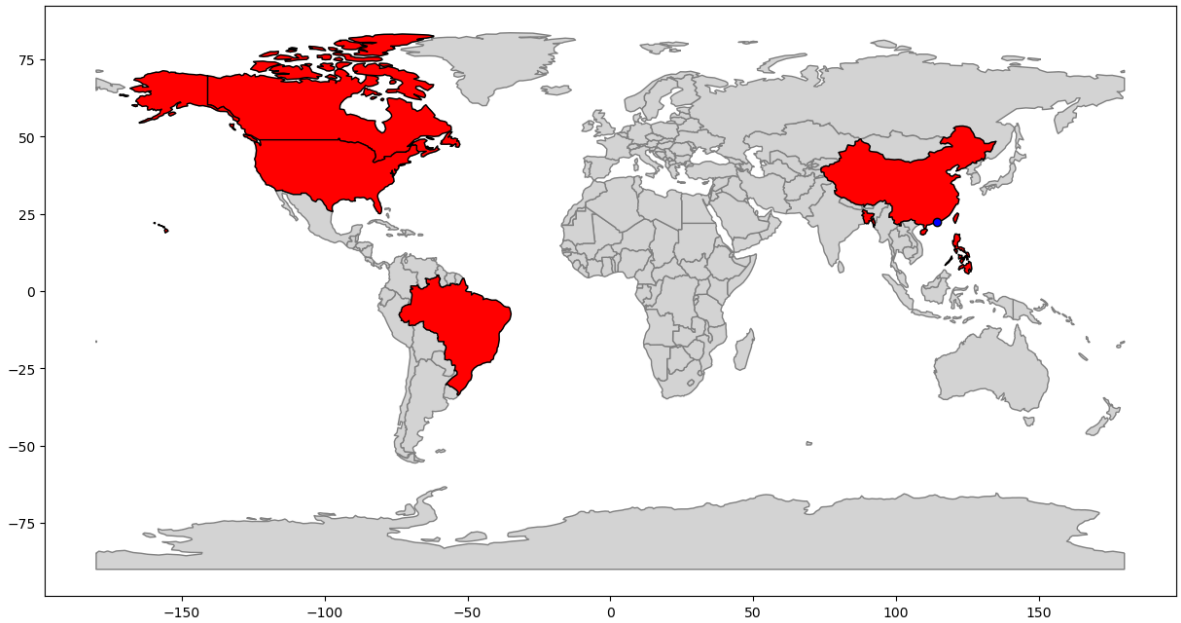


Figure 8: Mapa con los lugares del que proceden las IPs

```
recuento = df["pais"].value_counts().to_dict()

pd.DataFrame(data={"Pais": list(recuento.keys()), "Reportes": list(recuento.values())})
```

	Pais	Reportes
0	Hong Kong	174
1	Philippines	118
2	China	75
3	United States	5
4	Canada	3
5	Bangladesh	1
6	Taiwan, Province of China	1
7	Brazil	1

```
counts = df["pais"].value_counts().rename_axis('pais').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')
```

```
fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["pais"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```

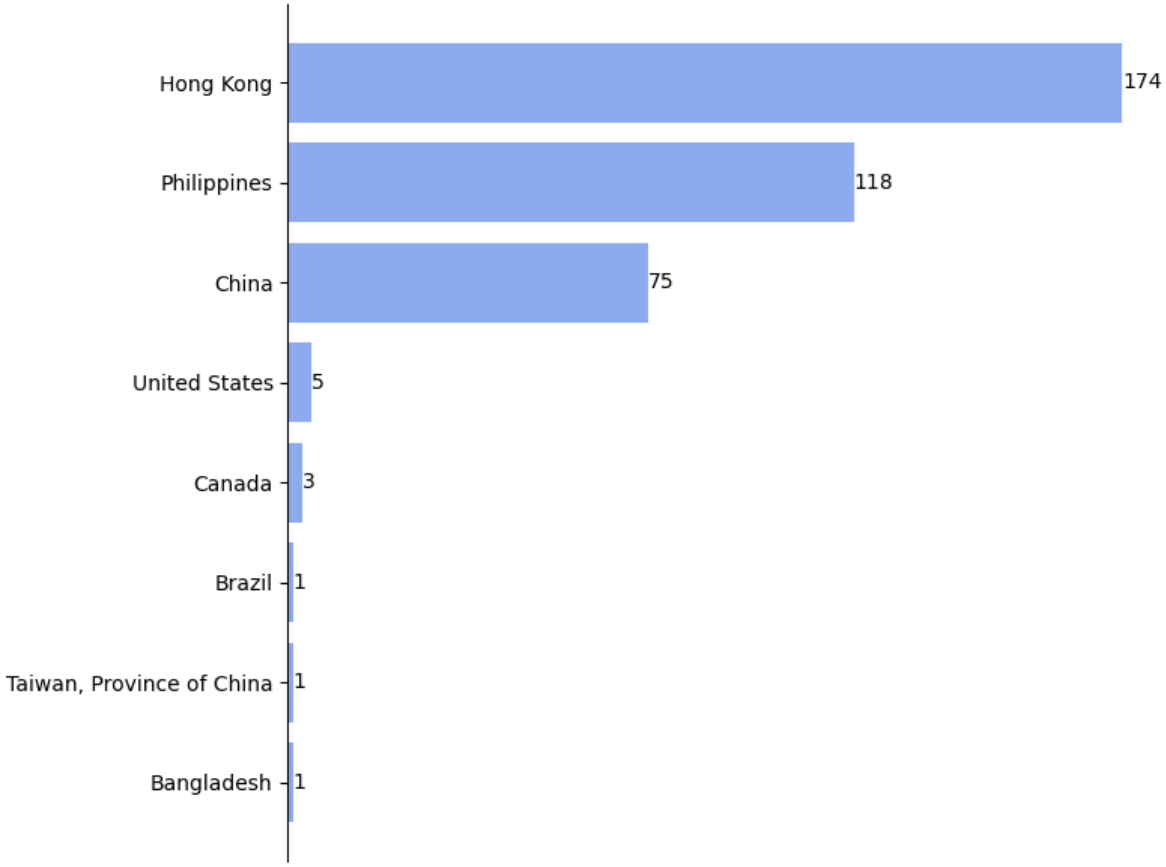


Figure 9: Número de reportes por país

```
recuento = dfgdp["codigoPais"].value_counts().to_dict()

dfPlot = pd.DataFrame(data={"codigoPais": list(recuento.keys()), "Reportes": list(recuento
dfPlot

fig = px.scatter_geo(dfPlot, locations="codigoPais", color="codigoPais", size="Reportes",
                     projection="equirectangular")
fig.write_image("data/plotlyDDoS.png")
```

```
from PIL import Image

image = np.asarray(Image.open('data/plotlyDDoS.png'))
plt.imshow(image)
plt.grid(False)
plt.axis(False)
plt.show()
```

Figure 10: Scatterplot con tamaño en función de número de reportes

2.4 Análisis de frecuencia

Para así poder de tratar de identificar cierto patron asociado a la hora de ataque.

2.4.1 Extracción de información

```
df.head(3)
```

	ip	esPublica	estaEnWhitelist	scoreAbuso	pais	codigoPais	isp
0	45.204.126.117	True	NaN	0	Hong Kong	HK	Intercontinental Int
1	59.153.100.70	True	NaN	0	Bangladesh	BD	Dot Internet
2	1.32.249.141	True	NaN	0	Hong Kong	HK	CTG Server Ltd.

```
eventosDDoS = []
for index, row in df.iterrows():
    fecha = row["dia"].split(sep="/")
    if fecha[0] != 'may':
        print(fecha[0])
    else:
        mes = 5

    dia = fecha[1]
    año = fecha[2]

    tiempo = row["hora"].split(sep=":")
    hora = tiempo[0]
    minuto = tiempo[1]
    eventosDDoS.append((row["ip"], datetime(int(año), int(mes), int(dia), int(hora), int(m
```

```
fig, ax = plt.subplots()

fecha = [evento[1] for evento in eventos]
etiquetas = [evento[0] for evento in eventos]
ax.eventplot(fecha, lineoffsets=0.1, linelengths=0.1, color='r')
ax.set_ylabel(None)
ax.set_yticklabels([])
ax.set_xlim(datetime(2023, 5, 23, 0, 0), datetime(2023, 5, 23, 23, 59))
fig.autofmt_xdate()
```

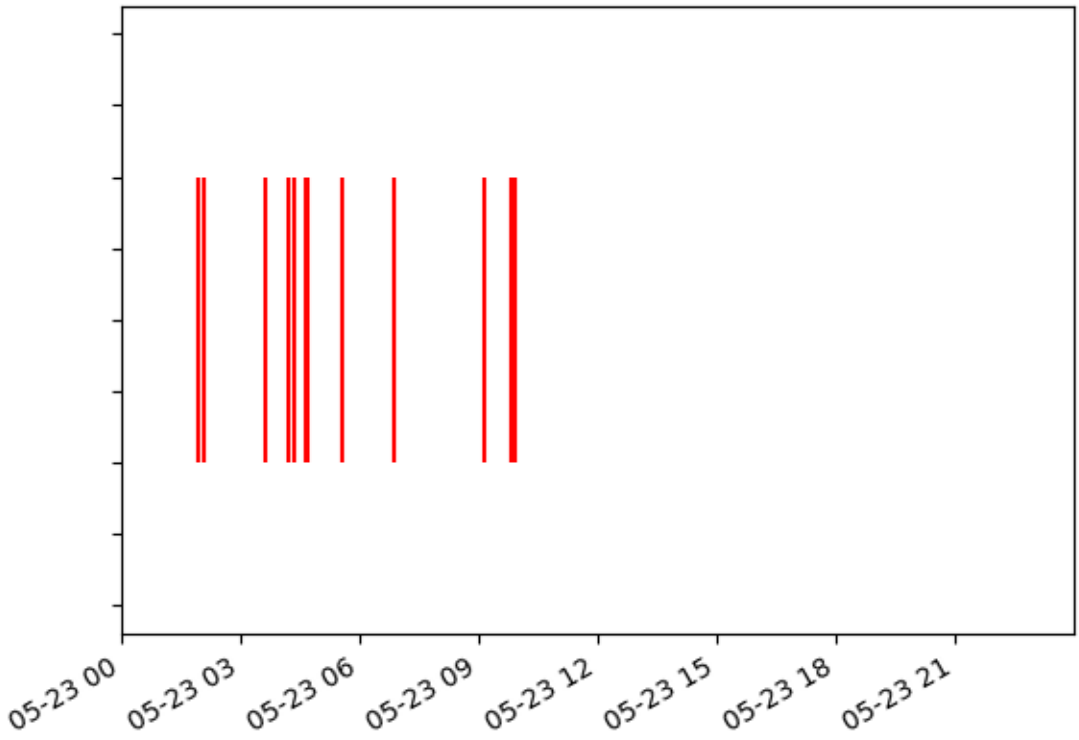


Figure 11: Visualización de eventos de DDoS

Notar que a comparación de los por SSH parecen menos siendo que son 380 ataques contra 12. Esto es porque son muy seguido.

2.5 Análisis de ISPs

```
recuento = df["isp"].value_counts().to_dict()

recuento = pd.DataFrame(data={"ISP": list(recuento.keys()), "Reportes": list(recuento.valu
recuento
```

	ISP	Reportes
0	Suniway Group Limited	75
1	Gold Experience Cloud LLC	75
2	Dragon Spirit Investments International Co. Li...	73
3	WTW Hightech Company Inc	67
4	Suniway Telecom	39

	ISP	Reportes
5	Suniway Group of Companies Inc.	36
6	Intercontinental Internet Data Corp	1
7	Bell Canada	1
8	Comcast Cable Communications LLC	1
9	Cabo Servicos de Telecomunicacoes Ltda	1
10	Shaw Communications Inc.	1
11	Delta DCCNet High Speed Internet	1
12	Nexus Bytes LLC	1
13	Dot Internet	1
14	VPSquan L.L.C.	1
15	Kurun Cloud Inc	1
16	Shine Telecom Co. Ltd.	1
17	CTG Server Ltd.	1
18	Reliablesite.net LLC	1

```
counts = df["isp"].value_counts().rename_axis('ISP').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')

fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["ISP"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```

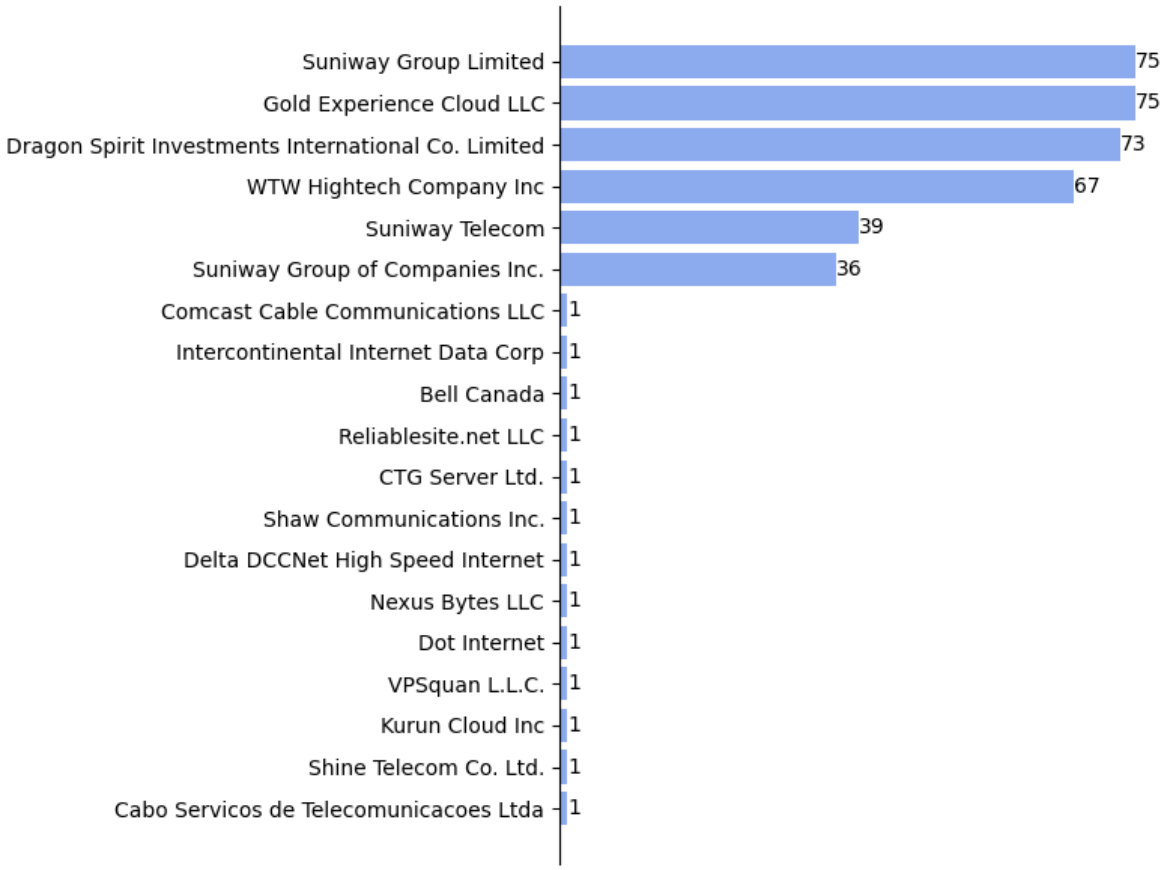


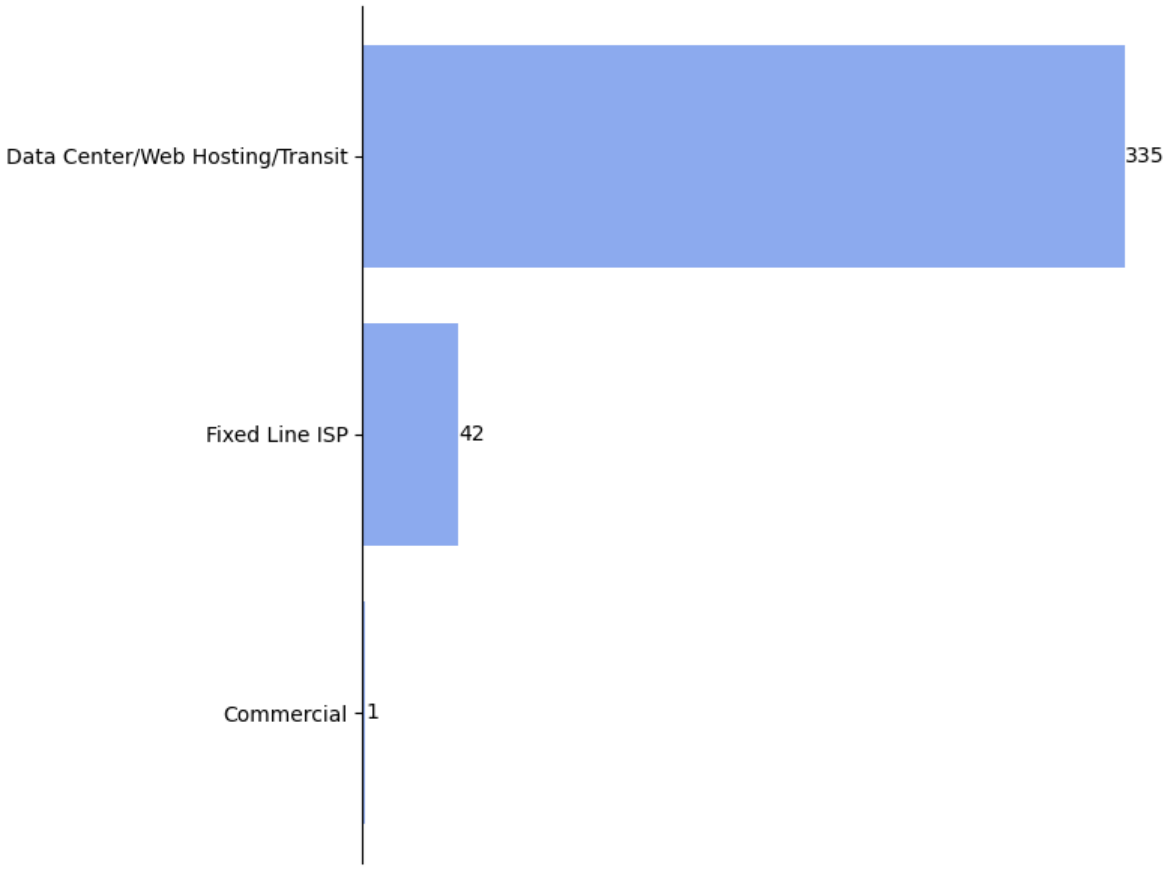
Figure 12: Número de veces que se denuncio un ISP por ataque DDoS

2.6 Análisis de uso

```
recuento = df["tipoDeUso"].value_counts().to_dict()

recuento = pd.DataFrame(data={"uso": list(recuento.keys()), "Reportes": list(recuento.valu
recuento
```

	uso	Reportes
0	Data Center/Web Hosting/Transit	335
1	Fixed Line ISP	42
2	Commercial	1

```
counts = df["tipoDeUso"].value_counts().rename_axis('tipoDeUso').to_frame('counts')
counts.reset_index(level=0, inplace=True)

counts = counts.sort_values(by='counts')

fig, ax = plt.subplots(figsize=(8,6))

bars = plt.barh(counts["tipoDeUso"], counts['counts'], color='#8caaee')

ax.spines[['right', 'top', 'bottom']].set_visible(False)
ax.xaxis.set_visible(False)

ax.spines['left'].set_color('black')

ax.tick_params(axis='y', colors='black')
ax.bar_label(bars, color='black')
plt.tight_layout()
plt.show()
```



2.7 ¿Fue un solo ataque?

```
import netaddr

def convertirIP(ip):
    return int(netaddr.IPAddress(ip))
```

```
datetimes = [evento[1] for evento in eventosDDoS]
dicc = {"ip": df["ip"],
        "datetime": datetimes}
```

```
dfClusters = pd.DataFrame(dicc)
dfClusters["datetime"] = dfClusters["datetime"].apply(lambda x: x.utcnow().timestamp())
```

```
dfClusters["ip"] = dfClusters["ip"].apply(convertirIP)
```

```
sns.scatterplot(data=dfClusters,x="datetime", y="ip")
plt.xlabel("Tiempo")
plt.ylabel("IPs")
plt.xticks([])
plt.yticks([])
plt.show()
```

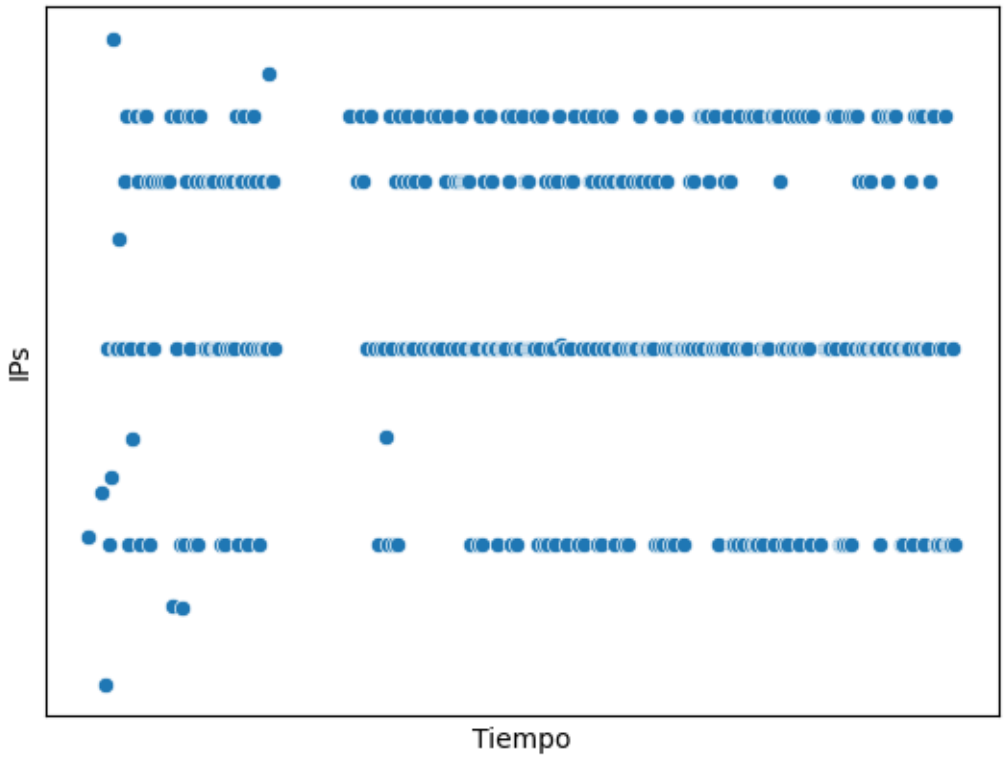


Figure 13: IPs vs Tiempo

Como se puede observar en Figure 13 pareciese ser que a pesar de que se realizaron reportes en horarios distantes, las IPs no ocupan todo el espectro. Se podría inferir que fue el mismo atacante porque uso IPs en un rango acotado