Visualizing Cyber Attacks

In [1]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns

import matplotlib.pyplot as plt
import plotly.graph_objects as go
import matplotlib.ticker as ticker
import plotly.figure_factory as ff

from fbprophet import Prophet
from pandas_datareader import data
from scipy import stats
from sklearn.metrics import mean_absolute_error
from statsmodels.tsa.seasonal import seasonal_decompose

from sklearn.preprocessing import MinMaxScaler

from matplotlib.ticker import FixedFormatter, FixedLocator

import warnings
warnings.filterwarnings('ignore')
```

```
/opt/conda/lib/python3.7/site-packages/geopandas/_compat.py:115: UserWarning: The Sha
pely GEOS version (3.9.1-CAPI-1.14.2) is incompatible with the GEOS version PyGEOS wa
s compiled with (3.10.1-CAPI-1.16.0). Conversions between both will be slow.
  shapely_geos_version, geos_capi_version_string
```

In [2]:
```
df = pd.read_csv("../input/aws-honeypot-attack-data/AWS_Honeypot_marx-geo.csv")
```

In [3]:
```
month_list = ['Jan.', 'Feb.', 'Mar.', 'Apr.', 'May', 'June', 'July', 'Aug.', 'Sept.',
attack_date = {"year": [], "month": [], "day": [], "time": [], "hour": []}

for d_date in df['datetime']:
    d_date = d_date.split('/')
    mon = d_date[0]
    day = d_date[1]

    year_time = d_date[2].split(' ')
    year = year_time[0]
    time = year_time[1]

    attack_date["year"].append(year)
    attack_date["month"].append(month_list[int(mon) + 1])
    attack_date["day"].append(day)
    attack_date["time"].append(time)
    attack_date["hour"].append(f"{time.split(':')[0]}h")

df["year"] = attack_date["year"]
df["month"] = attack_date["month"]
df["day"] = attack_date["day"]
df["time"] = attack_date["time"]
df["hour"] = attack_date["hour"]
```

In [4]:
```
df.head(20)
```

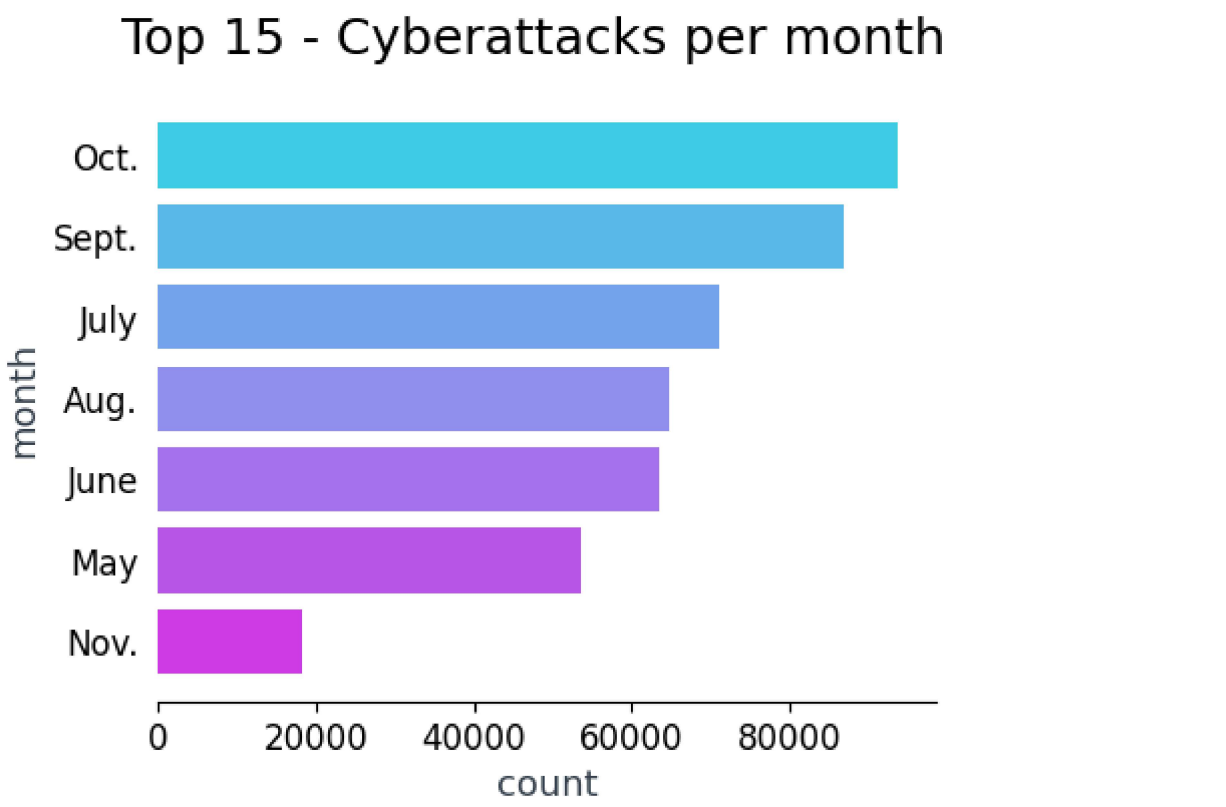
Out[4]:

	datetime	host	src	proto	type	spt	dpt	srcstr	cc	country	...
0	3/3/13 21:53	groucho-oregon	1032051418	TCP	NaN	6000.0	1433.0	61.131.218.218	CN	China	...
1	3/3/13 21:57	groucho-oregon	1347834426	UDP	NaN	5270.0	5060.0	80.86.82.58	DE	Germany	...
2	3/3/13 21:58	groucho-oregon	2947856490	TCP	NaN	2489.0	1080.0	175.180.184.106	TW	Taiwan	...
3	3/3/13 21:58	groucho-us-east	841842716	UDP	NaN	43235.0	1900.0	50.45.128.28	US	United States	...
4	3/3/13 21:58	groucho-singapore	3587648279	TCP	NaN	56577.0	80.0	213.215.43.23	FR	France	...
5	3/3/13 21:58	groucho-tokyo	3323217250	TCP	NaN	32628.0	2323.0	198.20.69.98	US	United States	...
6	3/3/13 21:59	groucho-oregon	3730416887	TCP	NaN	6000.0	1433.0	222.89.164.247	CN	China	...
7	3/3/13 22:07	groucho-singapore	3738622573	TCP	NaN	6000.0	3306.0	222.214.218.109	CN	China	...
8	3/3/13 22:12	groucho-oregon	3683919430	TCP	NaN	6000.0	1433.0	219.148.38.70	CN	China	...
9	3/3/13 22:14	groucho-singapore	1007884304	TCP	NaN	6000.0	1433.0	60.19.24.16	CN	China	...
10	3/3/13 22:14	groucho-tokyo	3639889826	TCP	NaN	6000.0	1433.0	216.244.79.162	US	United States	...
11	3/3/13 22:20	groucho-oregon	1965603898	TCP	NaN	9907.0	1433.0	117.40.188.58	CN	China	...
12	3/3/13 22:20	groucho-sa	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
13	3/3/13 22:20	zeppo-norcal	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
14	3/3/13 22:19	groucho-norcal	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
15	3/3/13 22:20	groucho-us-east	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
16	3/3/13 22:21	groucho-eu	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
17	3/3/13 22:21	groucho-sydney	3672981807	TCP	NaN	33367.0	22.0	218.237.65.47	KR	South Korea	...
18	3/3/13 22:23	groucho-oregon	2382398543	TCP	NaN	6000.0	1433.0	142.0.132.79	US	United States	...
19	3/3/13 22:26	groucho-tokyo	1965603898	TCP	NaN	26180.0	1433.0	117.40.188.58	CN	China	...

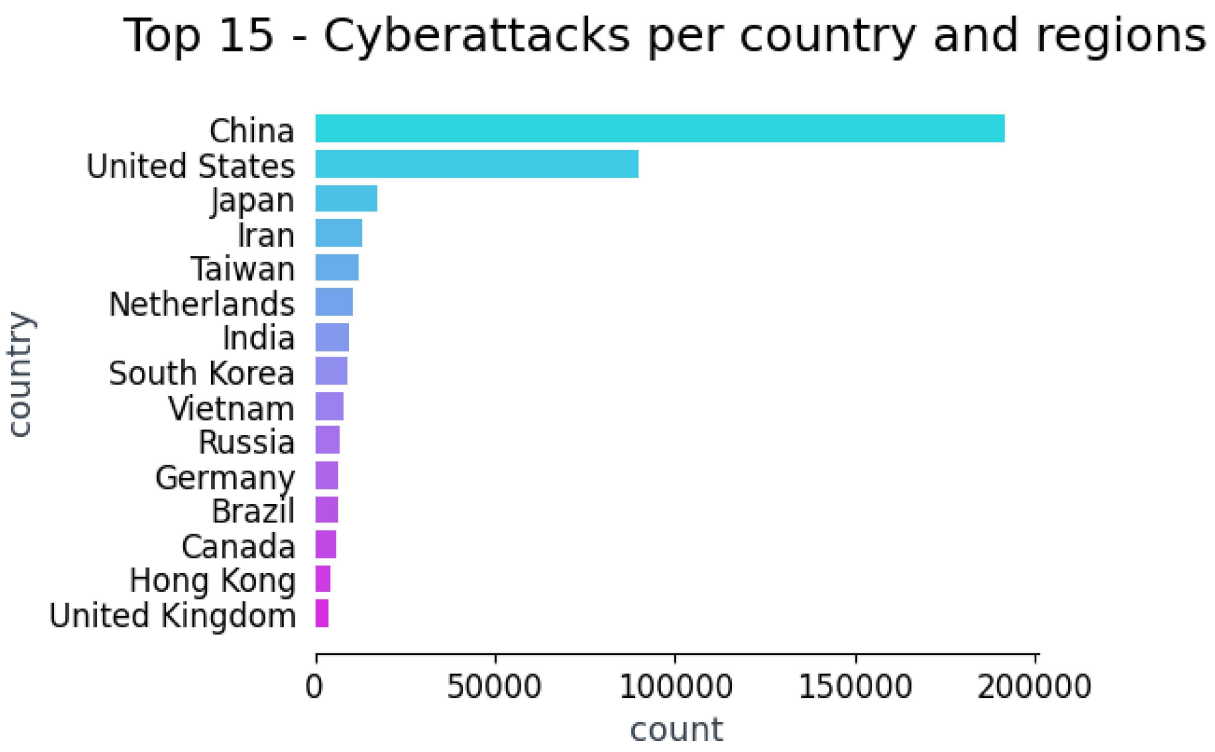
20 rows × 21 columns

In [5]:
```
def bar_plot_data(df: dict, field: str, subtitle: str, figsize=(5, 4), top_filter=15):
    fig, ax1 = plt.subplots(figsize=figsize, dpi=100)

    for spline in ['top', 'right', 'left']:
        ax1.spines[spline].set_visible(False)

    df_filter = df[field].value_counts().rename_axis(field).reset_index(name='counts')
    if top_filter:
        df_filter = df_filter.head(top_filter)
    sns.barplot(data=df_filter, palette='cool', x='counts', y=field)
    ax1.tick_params(axis='both', which='both', labelsize=12, bottom=True, left=False)
    ax1.set_xlabel(f'count', fontsize=13, color = '#333F4B')
    ax1.set_ylabel(f'{field}', fontsize=13, color = '#333F4B')

    plt.plot(color="white", lw=3)
    fig.suptitle(subtitle, fontsize=18)
    plt.show()
```

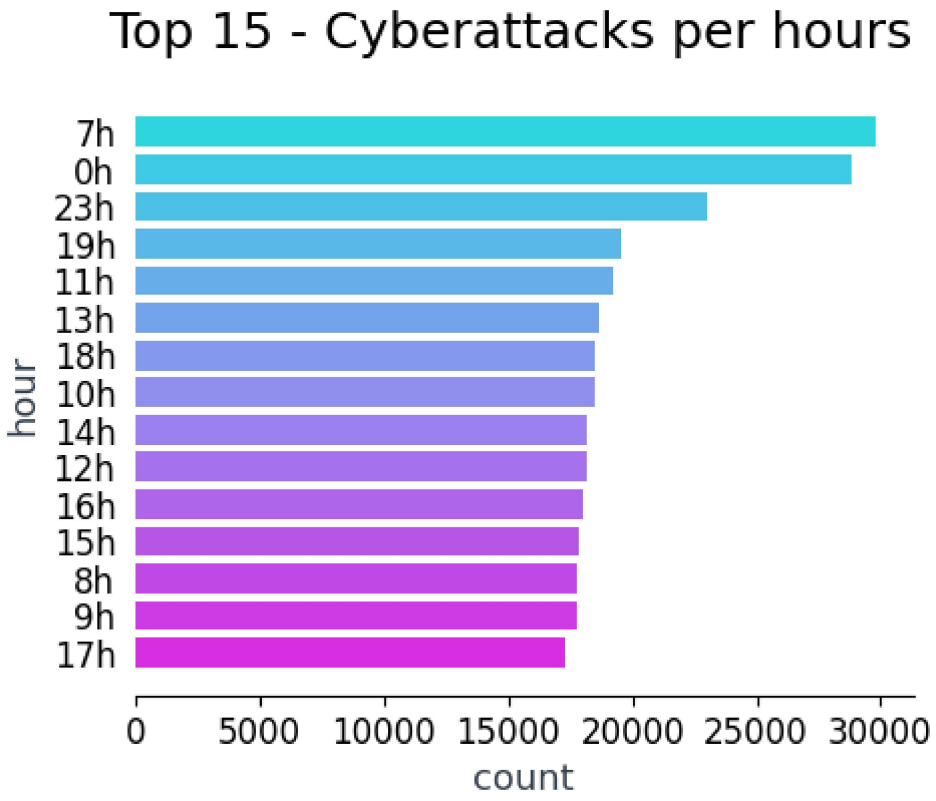
In [6]:
```
bar_plot_data(df, 'month', 'Top 15 - Cyberattacks per month')
```



In [21]:
```
bar_plot_data(df, 'country', 'Top 15 - Cyberattacks per country and regions')
```



In [8]:
```
bar_plot_data(df, 'hour', 'Top 15 - Cyberattacks per hours')
```



In [9]:
```
df['srcstr'].value_counts().rename_axis('ipaddress').reset_index(name='counts')
```

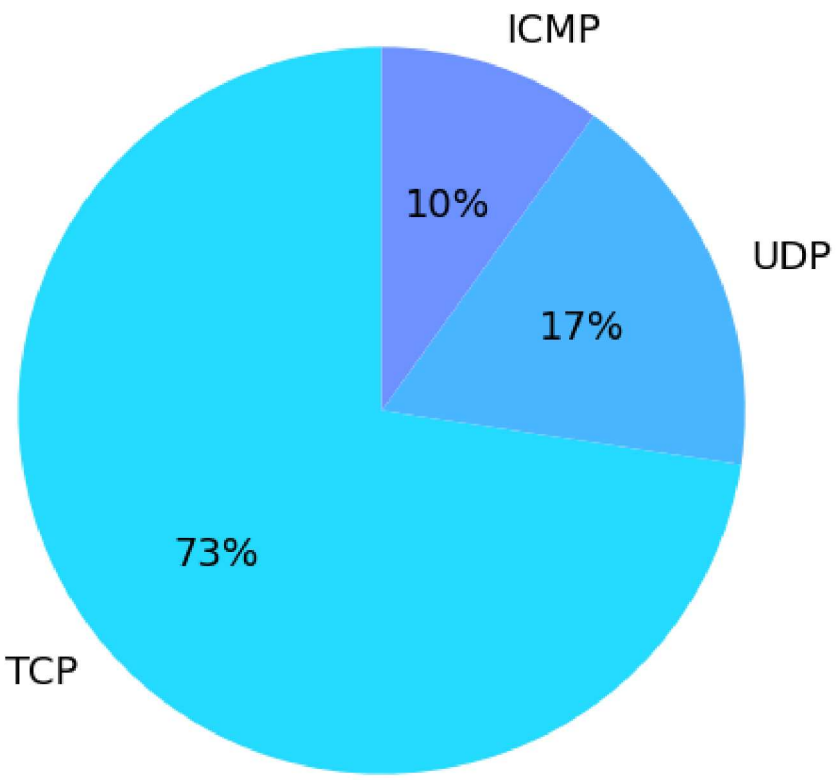
Out[9]:

	ipaddress	counts
0	175.146.199.252	18472
1	2.186.189.218	11116
2	203.178.148.19	7086
3	128.9.168.98	7046
4	129.82.138.44	6772
...	...	...
69597	113.161.81.118	1
69598	116.252.182.193	1
69599	113.81.65.41	1
69600	122.144.11.167	1
69601	175.201.224.104	1

69602 rows × 2 columns

In [10]:
```
def get_pie_plot_count(df: dict, field: str, subtitle: str):
    fig, ax1 = plt.subplots(figsize=(10, 6), dpi=100)
    colors = sns.color_palette('cool')
    df_filter = df[field].value_counts().rename_axis(field).reset_index(name='counts')
    ax1.tick_params(axis='both', which='both', labelsize=12, bottom=True, left=False)
    plt.pie(df_filter['counts'], colors=colors, labels=df_filter[field], autopct = '%
            startangle=90,  textprops={'fontsize': 14})

    plt.plot(color="white", lw=3)
    fig.suptitle(subtitle, fontsize=18)
    plt.show()
```

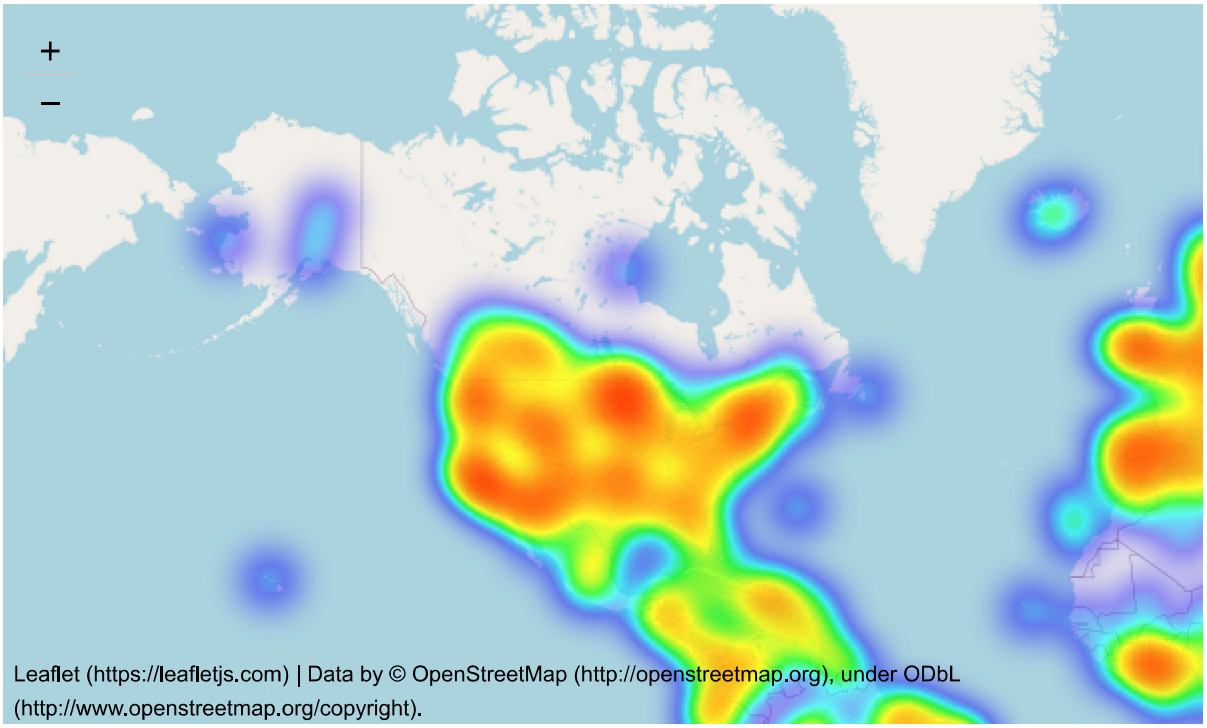
In [11]:
```
get_pie_plot_count(df, 'proto', 'Cyberattacks per communication protocol')
```



In [12]:
```
attack= df[['latitude','longitude']]
attack.latitude.fillna(0, inplace = True)
attack.longitude.fillna(0, inplace = True)

World =folium.Map(location=[0,0],zoom_start=2)
HeatMap(data=attack, radius=16).add_to(World)

print('Top cyberattacks by country')
World
```

```
Top cyberattacks by country
```

Out[12]:

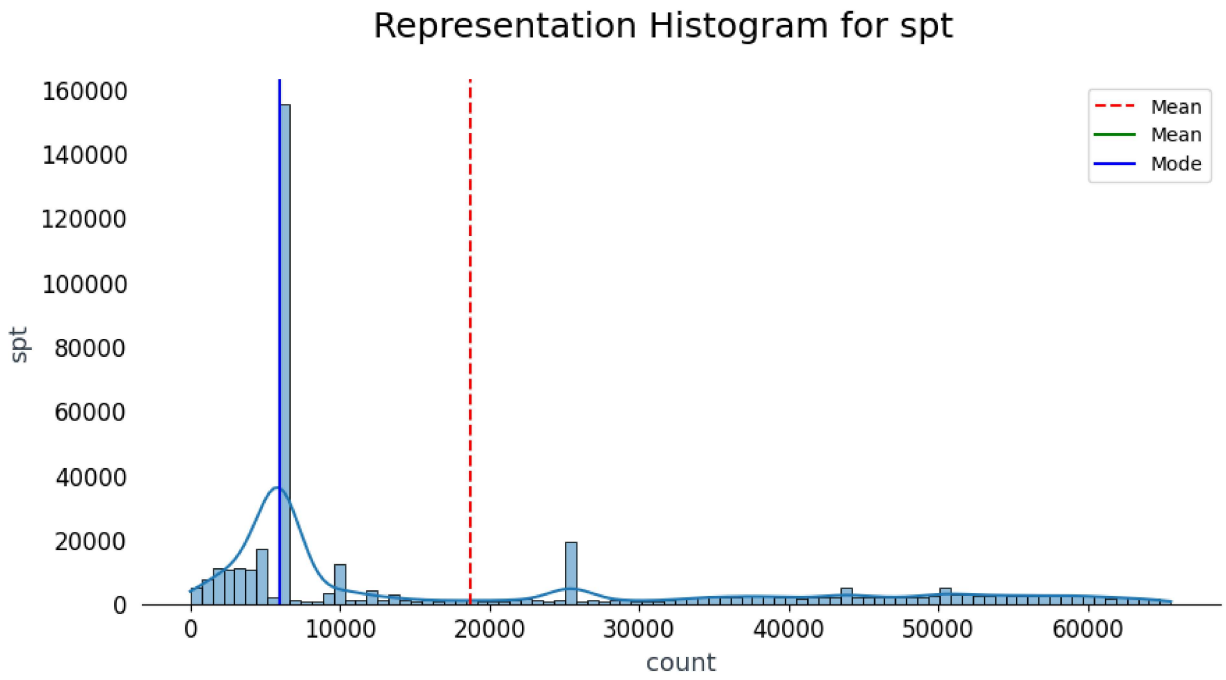


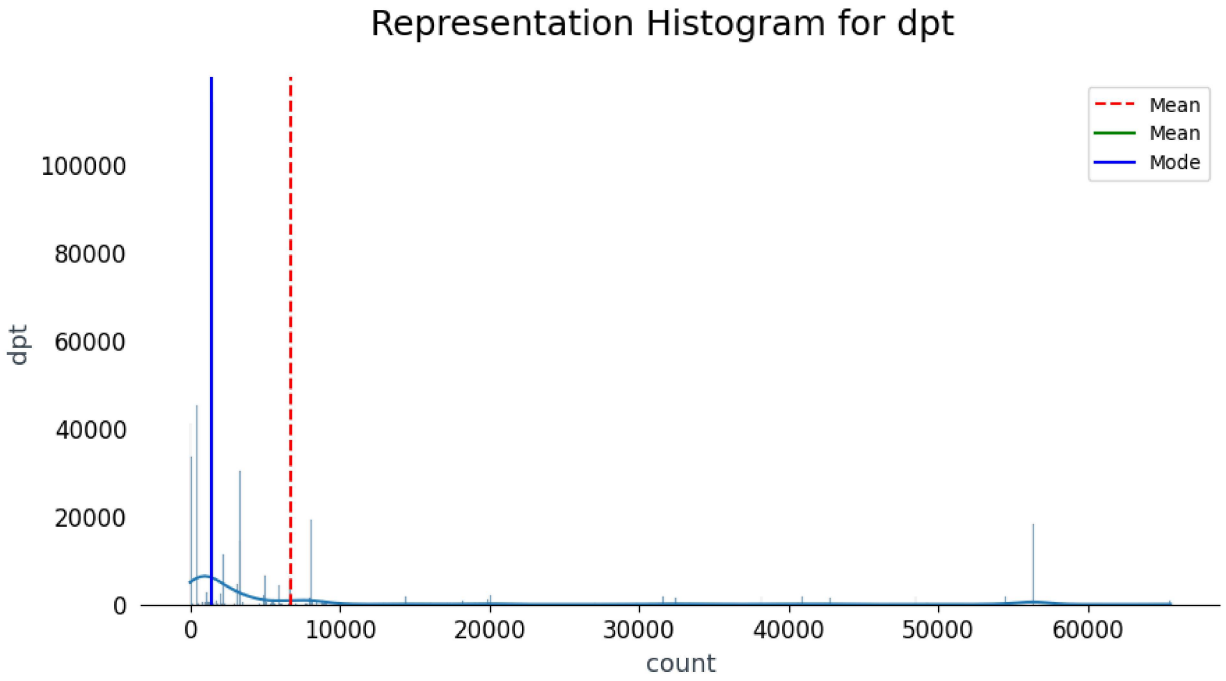
In [13]:

```
def get_histplot_central_tendency(df: dict, fields: list):
    colors = sns.color_palette('cool')
    for field in fields:
        f, (ax1) = plt.subplots(figsize=(10, 5), dpi=100)
        v_dist_1 = df[field].values

        for spline in ['top', 'right', 'left']:
            ax1.spines[spline].set_visible(False)

        sns.histplot(v_dist_1, ax=ax1, palette=colors, kde=True)

        mean=df[field].mean()
        median=df[field].median()
        mode=df[field].mode().values[0]

        ax1.axvline(mean, color='r', linestyle='--', label="Mean")
        ax1.axvline(median, color='g', linestyle='-', label="Mean")
        ax1.axvline(mode, color='b', linestyle='-', label="Mode")
        ax1.tick_params(axis='both', which='both', labelsize=12, bottom=True, left=Fal
        ax1.set_xlabel(f'count', fontsize=13, color = '#333F4B')
        ax1.set_ylabel(f'{field}', fontsize=13, color = '#333F4B')
        ax1.legend()
        plt.grid(False)
        plt.plot(color="white", lw=3)
        f.suptitle(f"Representation Histogram for {field}", fontsize=18)
```

In [14]:

```
get_histplot_central_tendency(df, ['spt', 'dpt'])
```



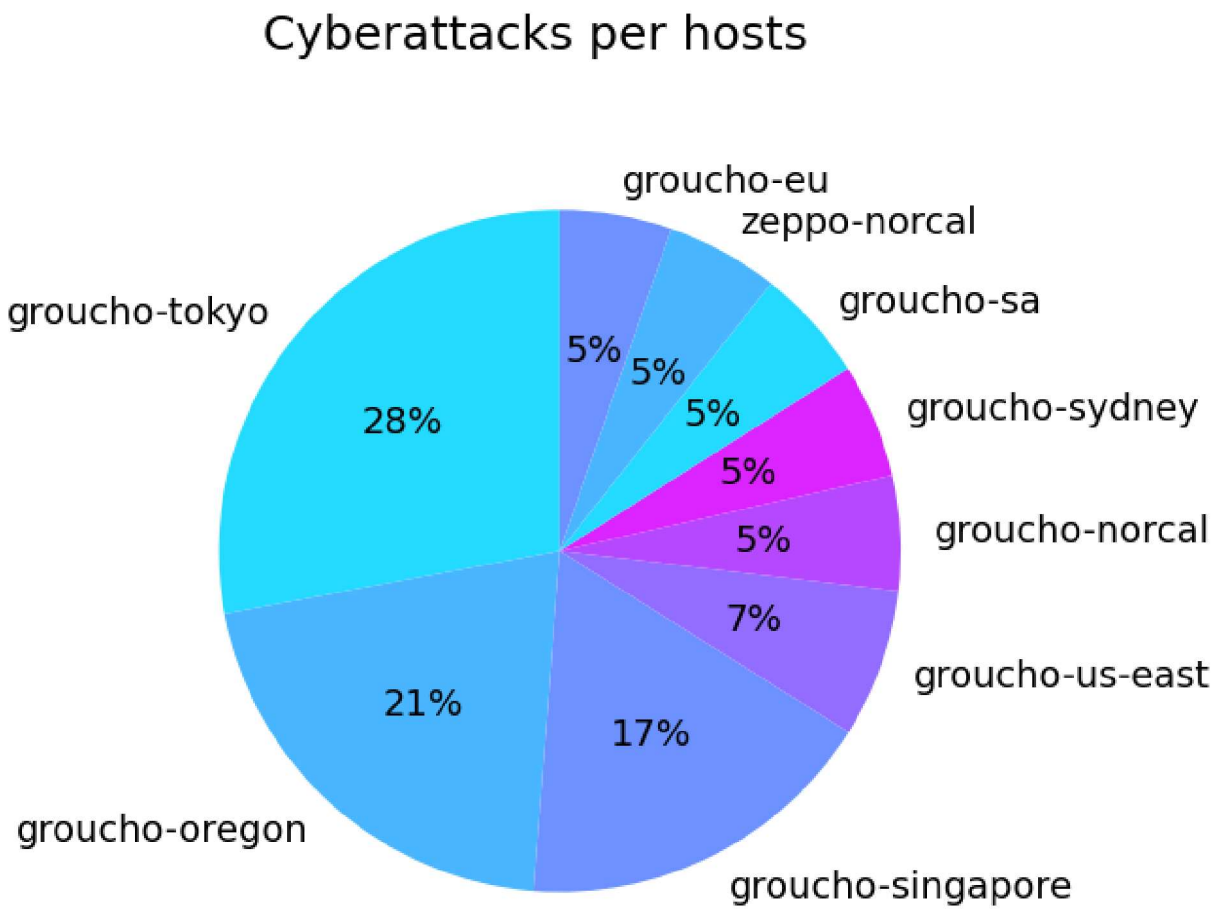


In [15]:
```
df['type'].value_counts().rename_axis('id_type_attack').reset_index(name='counts')
```

Out[15]:

	id_type_attack	counts
0	8.0	38597
1	3.0	4251
2	11.0	1156
3	0.0	536
4	13.0	142
5	5.0	127
6	12.0	2

In [16]:
```
get_pie_plot_count(df, 'host', 'Cyberattacks per hosts')
```



In [17]:
```
df_filter = df[(df.host == "groucho-oregon") & (df.month == "May") & (df.hour == "20h'
df_filter
```

Out[17]:

	datetime	host	src	proto	type	spt	dpt	srcstr	cc	country	.
1437	3/4/13 20:00	groucho-oregon	2036715561	TCP	NaN	27984.0	22.0	121.101.208.41	CN	China	
1441	3/4/13 20:05	groucho-oregon	3736745024	TCP	NaN	6000.0	1433.0	222.186.52.64	CN	China	
1442	3/4/13 20:06	groucho-oregon	3658020275	TCP	NaN	6000.0	1433.0	218.8.245.179	CN	China	
1444	3/4/13 20:08	groucho-oregon	3630406730	TCP	NaN	6000.0	1433.0	216.99.156.74	US	United States	
1445	3/4/13 20:09	groucho-oregon	1019141837	TCP	NaN	6000.0	1433.0	60.190.222.205	CN	China	
...	...	...	...	...	...	...	...	...	...	...	
53246	3/31/13 20:46	groucho-oregon	1995742862	TCP	NaN	6000.0	1433.0	118.244.158.142	CN	China	
53248	3/31/13 20:49	groucho-oregon	3736734735	TCP	NaN	6000.0	1433.0	222.186.12.15	CN	China	
53249	3/31/13 20:49	groucho-oregon	3689613428	TCP	NaN	6000.0	1433.0	219.235.8.116	CN	China	
53258	3/31/13 20:52	groucho-oregon	3683824005	TCP	NaN	6000.0	1433.0	219.146.177.133	CN	China	
53265	3/31/13 20:57	groucho-oregon	1995742035	TCP	NaN	6000.0	3306.0	118.244.155.83	CN	China	

442 rows × 21 columns

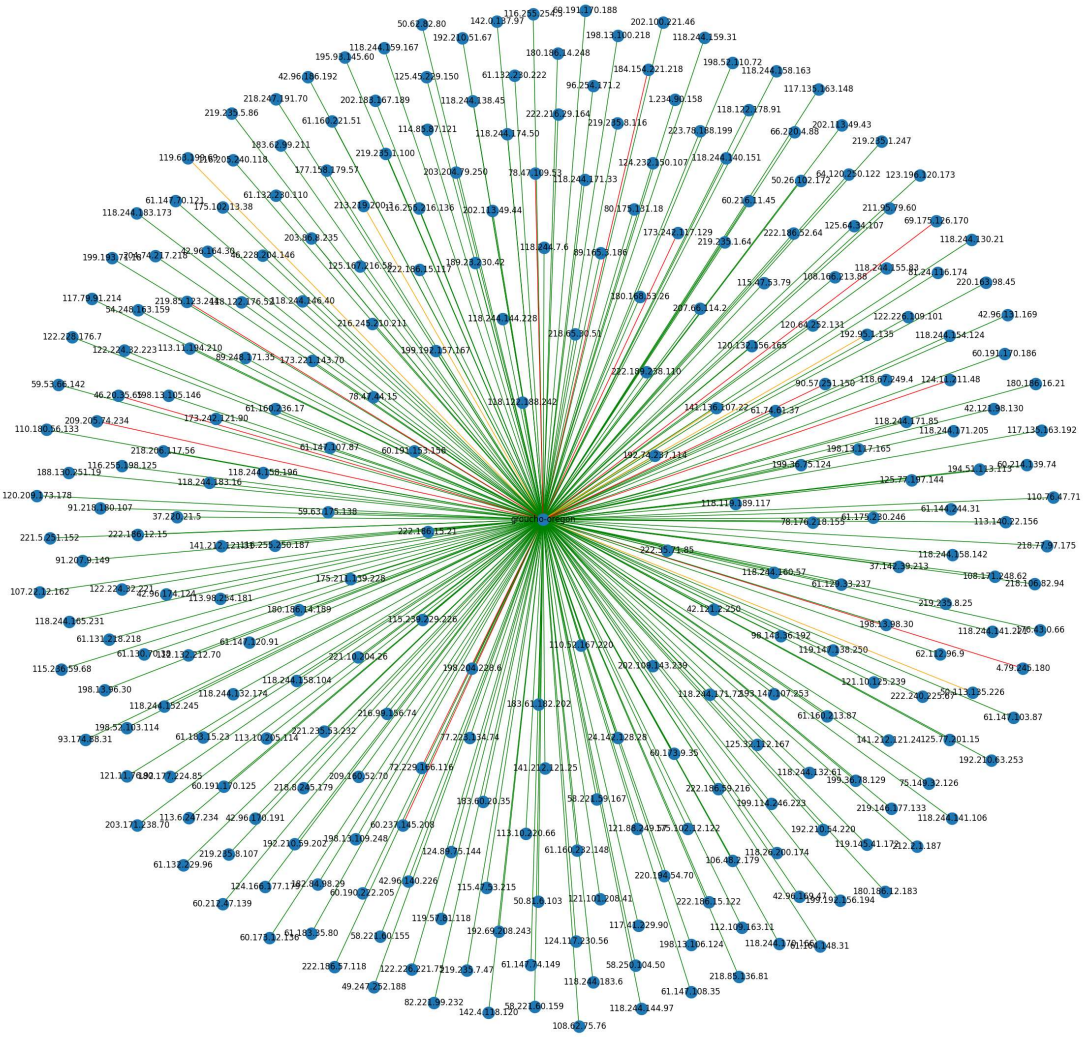
In [18]:
```
df_filter['proto'].value_counts()
```

Out[18]:
```
TCP     392
UDP      35
ICMP     15
Name: proto, dtype: int64
```

In [19]:
```
import networkx as nx

G = nx.Graph()
G.add_node('groucho-oregon', color='red')

ip_listx = []
edge_color = []
for ip, proto in zip(df_filter['srcstr'], df_filter['proto']):
    if ip not in ip_listx:
        if 'TCP' == proto:
            edge_color.append('green')
        if 'UDP' == proto:
            edge_color.append('red')
        if 'ICMP' == proto:
            edge_color.append('orange')
        G.add_node(ip)
        ip_listx.append(ip)
```

```
for ip in ip_listx:
    G.add_edge('groucho-oregon', ip)
fig = plt.figure(1, figsize=(25, 25), dpi=100)
print('TPC == green\nUDP == red\nICMP == orange')
nx.draw(G, with_labels=True, font_weight='normal', edge_color=edge_color, arrowsize=30
```

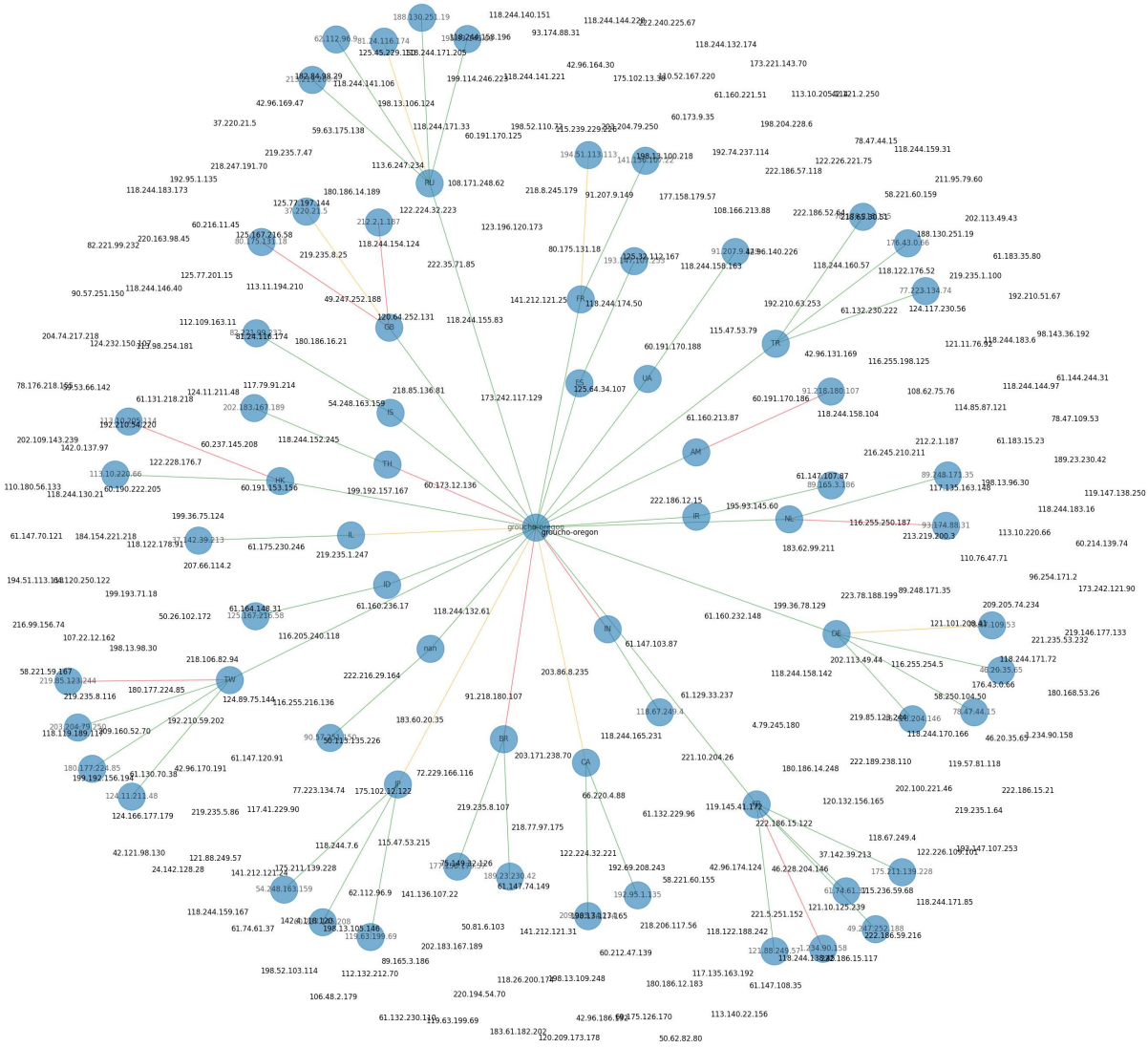
```
TPC == green
UDP == red
ICMP == orange
```



In [20]:

```
from matplotlib.pyplot import figure, text

Gx = nx.Graph()
Gx.add_node('groucho-oregon')

country_list = []
ip_list = []
edge_color = []
for country, ip, proto in zip(df_filter['cc'], df_filter['srcstr'], df_filter['proto']
    if country != 'CN' and country != 'US':
        if country not in country_list:
```

```
            Gx.add_node(country)
            country_list.append(country)

        if ip not in ip_list:
            if 'TCP' == proto:
                edge_color.append('green')
            if 'UDP' == proto:
                edge_color.append('red')
            if 'ICMP' == proto:
                edge_color.append('orange')

            Gx.add_node(ip)
            Gx.add_edge(country, ip)
            ip_list.append(ip)


for country in country_list:
    Gx.add_edge('groucho-oregon', country)

options = {
    "node_color": "black",
    "node_size": 50,
    "linewidths": 0,
    "width": 0.1,
}


pos = nx.spring_layout(G)
d = dict(G.degree)
fig = plt.figure(figsize=(26, 26), dpi=100)
nx.draw(Gx, alpha=0.6, node_size=2050, with_labels=True, font_weight='normal',
        edge_color=edge_color, arrowsize=30, arrowstyle='fancy')

for node, (x, y) in pos.items():
    text(x, y, node, fontsize=12)
```


grouchy-oregon
RU
FR
GB
UA
TR
IS
TH
HK
IL
ID
IR
NL
AM
IN
DE
TW
nan
BR
CA
US
188.130.251.19
118.244.140.151
118.244.144.229
222.240.225.67
93.174.88.31
118.244.132.174
173.221.143.70
42.96.164.30
118.244.171.205
118.244.141.221
199.114.246.223
175.102.13.38
110.52.167.220
61.160.221.51
113.10.205.218
221.2.250
42.96.169.47
198.13.106.124
37.220.21.5
59.63.175.138
118.244.171.33
198.52.110.32
215.239.229.226
223.204.79.250
60.173.9.35
198.204.228.6
60.191.170.125
78.47.44.15
118.244.159.31
219.235.7.47
218.247.191.70
113.6.247.234
198.13.100.218
192.74.237.114
122.226.221.75
192.95.1.135
118.244.183.173
180.186.14.189
108.171.248.62
218.8.245.179
91.207.9.149
177.158.179.57
222.186.57.118
211.95.79.60
58.221.60.159
125.77.197.144
37.220.23.5
122.224.32.223
108.166.213.88
222.186.52.64
218.65.30.51
202.113.49.43
60.216.11.45
212.2.1.187
123.196.120.173
125.167.216.58
80.175.131.18
188.130.251.19
82.221.99.232
220.163.98.45
118.244.154.124
80.175.131.18
91.207.9.149
196.140.226
176.43.0.66
61.183.35.80
219.235.8.25
222.35.71.85
193.177.177.249
118.244.158.163
118.244.160.57
118.122.176.52
125.77.201.15
113.11.194.210
118.244.146.40
90.57.251.150
49.247.252.188
141.212.121.25
118.244.174.50
219.235.1.100
77.223.134.74
192.210.51.67
192.210.63.253
61.132.230.222
124.117.230.56
112.109.163.11
120.64.252.131
118.244.155.83
204.74.217.218
124.232.150.107
213.98.254.181
81.24.116.174
180.186.16.21
115.47.53.79
60.191.170.188
118.244.183.6
98.143.36.192
121.11.76.52
42.96.131.169
116.255.198.125
78.176.218.155
99.53.66.142
124.11.211.48
117.79.91.214
218.85.136.81
125.64.34.107
61.144.244.31
61.131.218.218
202.183.161.189
54.248.163.159
173.242.117.129
91.218.180.107
108.62.75.76
118.244.144.97
60.191.170.186
114.85.87.121
192.210.54.220
61.160.213.87
118.244.158.104
78.47.109.53
202.109.143.239
142.0.137.97
60.237.145.208
118.244.152.245
212.2.1.187
61.183.15.23
216.245.210.211
122.228.176.7
189.23.230.42
113.10.220.66
60.190.222.205
60.191.153.156
199.192.157.167
60.173.12.136
61.147.107.87
89.165.3.186
89.248.171.35
117.135.163.148
198.13.96.30
110.180.56.133
118.244.130.21
119.147.138.250
222.186.12.15
195.93.145.60
116.255.250.187
93.174.88.31
118.244.183.16
199.36.75.124
61.147.70.121
184.154.221.218
118.122.178.91
37.142.39.213
61.175.230.246
219.235.1.247
213.219.200.3
113.10.220.66
60.214.139.74
183.62.99.211
207.66.114.2
110.76.47.71
194.51.113.154
120.250.122
96.254.171.2
199.193.71.18
223.78.188.199
89.248.171.35
173.242.121.90
61.164.148.31
125.167.216.58
61.160.236.17
118.244.132.61
199.36.78.129
209.205.74.234
50.26.102.172
61.160.232.148
216.99.156.74
107.22.12.162
116.205.240.118
121.101.209.17
219.146.177.133
198.13.98.30
61.147.103.87
221.235.53.232
218.106.82.94
202.113.49.44
116.255.254.5
118.244.171.72
58.221.59.167
219.85.123.244
222.216.29.164
203.86.8.235
118.244.158.142
48.230.35.65
176.43.0.66
180.177.224.85
124.89.75.144
116.255.216.136
91.218.180.107
61.129.33.237
58.250.104.50
180.168.53.26
219.235.8.116
192.210.59.202
183.60.20.35
118.67.248.4
78.47.44.15
219.85.128.244
118.244.204.146
4.79.245.180
201.204.70.230
118.119.189.117
209.160.52.70
90.57.251.150
50.115.135.226
118.244.165.231
118.244.170.166
46.20.35.65
1.234.90.158
180.177.224.85
61.130.70.38
42.96.170.191
61.147.120.91
203.171.238.70
221.10.204.26
222.189.238.110
180.186.14.248
119.57.81.118
159.192.156.194
72.229.166.116
124.11.211.48
77.223.134.74
175.102.12.122
66.220.4.88
202.100.221.46
222.186.15.21
124.166.177.179
219.235.5.86
117.41.229.90
219.235.8.107
61.132.229.96
119.145.41.172
120.132.156.165
219.235.1.64
218.77.97.175
222.186.15.122
118.67.249.4
42.121.98.130
118.244.7.6
115.47.53.215
122.224.32.221
37.142.39.213
122.226.109.101
193.147.107.253
121.88.249.57
24.142.128.28
175.211.139.228
192.69.208.243
42.96.174.124
46.228.204.146
175.211.139.228
141.212.121.23
58.221.60.155
54.248.163.159
62.112.96.9
141.136.107.22
61.147.74.149
192.95.1.135
175.236.59.68
118.244.171.85
118.244.159.167
61.74.61.37
198.13.105.146
119.63.199.69
50.81.6.103
141.212.121.31
218.206.117.56
118.122.188.242
221.5.251.152
121.10.125.239
49.247.252.188
222.186.59.216
89.165.3.186
202.183.167.189
60.212.47.139
198.52.103.114
112.132.212.70
118.26.200.174
198.13.109.248
180.186.12.183
117.135.163.192
61.147.108.35
106.48.2.179
220.194.54.70
61.132.230.110
119.63.199.69
42.96.186.69
175.126.170
113.140.22.156
183.61.182.202
120.209.173.178
50.62.82.80