In [56]:

```
import pandas as pd
import pandas.util.testing as tm
import seaborn as sns
import matplotlib.pyplot as plt
import ipaddress
import numpy as np
from scipy import stats
from scipy.stats import chi2_contingency
from datetime import datetime, timedelta
import math
import missingno as msno
plt.style.use('ggplot')
import warnings
warnings.filterwarnings('ignore')
```

In [43]:

```
df = pd.read_csv('Cybersecurity_attacks.csv')
df.shape
```

Out[43]:

```
(178031, 11)
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Attack category', 'Attack subcategory', 'Protocol', 'Source IP',
       'Source Port', 'Destination IP', 'Destination Port', 'Attack Nam
e',
       'Attack Reference', '.', 'Time'],
      dtype='object')
```

In [6]:

```
df.head(4)
```

Out[6]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port
0	Reconnaissance	HTTP	tcp	175.45.176.0	13284	149.171.126.16	80
1	Exploits	Unix 'r' Service	udp	175.45.176.3	21223	149.171.126.18	32780
2	Exploits	Browser	tcp	175.45.176.2	23357	149.171.126.16	80
3	Exploits	Miscellaneous Batch	tcp	175.45.176.2	13792	149.171.126.16	5555

In [8]:

```
df[['Start time','Last time']] = df['Time'].str.split('-',expand=True)
df.head()
```

Out[8]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port
0	Reconnaissance	HTTP	tcp	175.45.176.0	13284	149.171.126.16	80
1	Exploits	Unix 'r' Service	udp	175.45.176.3	21223	149.171.126.18	32780
2	Exploits	Browser	tcp	175.45.176.2	23357	149.171.126.16	80
3	Exploits	Miscellaneous Batch	tcp	175.45.176.2	13792	149.171.126.16	5555
4	Exploits	Cisco IOS	tcp	175.45.176.2	26939	149.171.126.10	80

In [8]:

```
df = pd.read_csv('Cybersecurity_attacks.csv')
df.columns
```

Out[8]:

```
Index(['Attack category', 'Attack subcategory', 'Protocol', 'Source IP',
       'Source Port', 'Destination IP', 'Destination Port', 'Attack Nam
e',
       'Attack Reference', '.', 'Time'],
      dtype='object')
```

In [10]:

```
df['.'].unique()
```

Out[10]:

```
array(['.'], dtype=object)
```

In [11]:

```
df = df.drop(['.', 'Time'],axis=1)# Drop columns and make a copy in memory of the object
df.head()
```

Out[11]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port
0	Reconnaissance	HTTP	tcp	175.45.176.0	13284	149.171.126.16	80
1	Exploits	Unix 'r' Service	udp	175.45.176.3	21223	149.171.126.18	32780
2	Exploits	Browser	tcp	175.45.176.2	23357	149.171.126.16	80
3	Exploits	Miscellaneous Batch	tcp	175.45.176.2	13792	149.171.126.16	5555
4	Exploits	Cisco IOS	tcp	175.45.176.2	26939	149.171.126.10	80

In [12]:

```
df.shape
```

Out[12]:

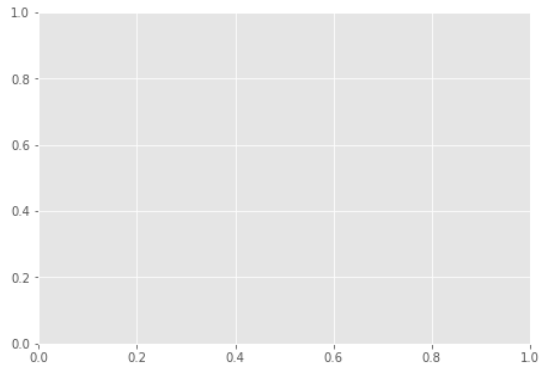
```
(178031, 11)
```

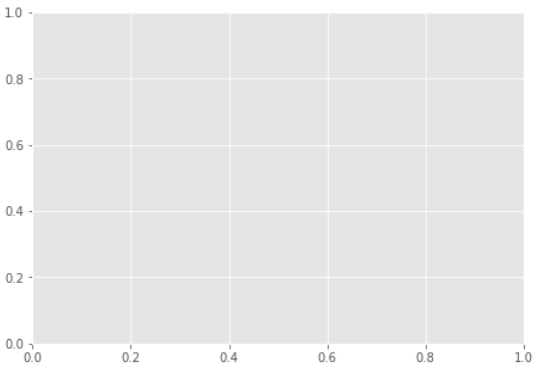
In [46]:

```
figure, (ax1, ax2) = plt.subplots(1, 2, figsize=(16,5))
msno.matrix(df, ax=ax1, sparkline=False, color=(0.1, 0.25, 0.35))
msno.bar(df, ax=ax2, color=(0.25, 0.7, 0.25))
plt.show()
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-46-8c36fab60f35> in <module>
      1 figure, (ax1, ax2) = plt.subplots(1, 2, figsize=(16,5))
----> 2 msno.matrix(df, ax=ax1, sparkline=False, color=(0.1, 0.25, 0.35))
      3 msno.bar(df, ax=ax2, color=(0.25, 0.7, 0.25))
      4 plt.show()

TypeError: matrix() got an unexpected keyword argument 'ax'
```





In [11]:

```
df.isnull().sum()
```

Out[11]:

```
Attack category           0
Attack subcategory     4192
Protocol                  0
Source IP                 0
Source Port               0
Destination IP            0
Destination Port          0
Attack Name               0
Attack Reference      51745
.                         0
Time                      0
dtype: int64
```

In [12]:

```
df["Attack subcategory"] = df["Attack subcategory"].fillna("Not Registered")
```

In [13]:

```
df.isnull().sum()
```

Out[13]:

```
Attack category           0
Attack subcategory        0
Protocol                  0
Source IP                 0
Source Port               0
Destination IP            0
Destination Port          0
Attack Name               0
Attack Reference      51745
.                         0
Time                      0
dtype: int64
```

In [14]:

```
df[pd.isnull(df).any(axis=1)].shape
```

Out[14]:

```
(51745, 11)
```

In [15]:

```
df[df.duplicated()].shape
```

Out[15]:

```
(6, 11)
```

In [16]:

```
print('Dimensions before dropping duplicated rows: ' + str(df.shape))
df = df.drop(df[df.duplicated()].index)
print('Dimensions after dropping duplicated rows: ' + str(df.shape))
```

```
Dimensions before dropping duplicated rows: (178031, 11)
Dimensions after dropping duplicated rows: (178025, 11)
```

In [17]:

```
df[df.duplicated()]
```

Out[17]:

Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port	Attack Name	Atta Referer

In [ ]:

```
#port range 0 to 65535
```

In [18]:

```
invalid_SP = (df['Source Port'] < 0) | (df['Source Port'] > 65535)
invalid_DP = (df['Destination Port'] < 0) | (df['Destination Port'] > 65535)
df[invalid_SP | invalid_DP]
```

Out[18]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destina
174347	Generic	IXIA	udp	175.45.176.1	67520	149.171.126.18	
174348	Exploits	Browser	tcp	175.45.176.3	78573	149.171.126.18	
174349	Reconnaissance	HTTP	tcp	175.45.176.1	71804	149.171.126.10	
174350	DoS	Ethernet	pnni	175.45.176.3	0	149.171.126.19	
174351	Fuzzers	OSPF	trunk-1	175.45.176.0	73338	149.171.126.13	
...	...	...	...	...	...	...	
178026	Generic	IXIA	udp	175.45.176.0	72349	149.171.126.12	
178027	Exploits	Browser	sep	175.45.176.3	67647	149.171.126.18	
178028	Exploits	Office Document	tcp	175.45.176.0	78359	149.171.126.13	
178029	Exploits	Browser	tcp	175.45.176.2	68488	149.171.126.19	
178030	Reconnaissance	ICMP	unas	175.45.176.3	77929	149.171.126.19	

3684 rows × 11 columns

In [19]:

```
df = df[~(invalid_SP | invalid_DP)].reset_index(drop=True)
```

In [20]:

```
df.shape
```

Out[20]:

```
(174341, 11)
```

In [21]:

```
print('Total number of different protocols:', len(df['Protocol'].unique()))
print('Total number of different Attack categories:', len(df['Attack category'].unique()
df['Protocol'].unique()[:15]
```

```
Total number of different protocols: 131
Total number of different Attack categories: 14
```

Out[21]:

```
array(['tcp', 'udp', 'Tcp', 'UDP', 'ospf', 'sctp', 'sep', 'mobile',
       'sun-nd', 'swipe', 'pim', 'ggp', 'ip', 'ipnip', 'st2'],
      dtype=object)
```

In [22]:

```
df['Attack category'].unique()
```

Out[22]:

```
array(['Reconnaissance', 'Exploits', 'DoS', 'Generic', 'Shellcode',
       ' Fuzzers', 'Worms', 'Backdoors', 'Analysis', ' Fuzzers ',
       ' Reconnaissance ', 'Backdoor', ' Shellcode ', 'Reconnaissance '],
      dtype=object)
```

In [23]:

```
df['Protocol'] = df['Protocol'].str.upper().str.strip()
df['Attack category'] = df['Attack category'].str.upper().str.strip()
df['Attack category'] = df['Attack category'].str.strip().replace('BACKDOORS','BACKDOOR'

df
```

Out[23]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	De
0	RECONNAISSANCE	HTTP	TCP	175.45.176.0	13284	149.171.126.16	
1	EXPLOITS	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	
2	EXPLOITS	Browser	TCP	175.45.176.2	23357	149.171.126.16	
3	EXPLOITS	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	
4	EXPLOITS	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	
...	...	...	...	...	...	...	
174336	DOS	IGMP	TCP	175.45.176.0	33654	149.171.126.12	
174337	FUZZERS	SMB	TCP	175.45.176.3	36468	149.171.126.15	
174338	RECONNAISSANCE	SunRPC Portmapper (TCP) UDP Service	TCP	175.45.176.2	64395	149.171.126.18	
174339	GENERIC	IXIA	UDP	175.45.176.0	47439	149.171.126.10	
174340	EXPLOITS	Office Document	TCP	175.45.176.0	17293	149.171.126.17	

174341 rows × 11 columns

In [24]:

```
print('Total number of different protocols:', len(df['Protocol'].unique()))
print('Total number of different Attack categories:', len(df['Attack category'].unique()
```

```
Total number of different protocols: 129
Total number of different Attack categories: 9
```

In [25]:

```
df[pd.isnull(df['Attack Reference'])].shape
```

Out[25]:

```
(50638, 11)
```

In [26]:

```
print(df[pd.isnull(df['Attack Reference'])]['Attack category'].value_counts())
```

```
FUZZERS           29649
RECONNAISSANCE    18149
ANALYSIS           1617
SHELLCODE           747
GENERIC             341
BACKDOOR             66
DOS                  53
WORMS                11
EXPLOITS              5
Name: Attack category, dtype: int64
```

In [27]:

```
print(df['Attack category'].value_counts())
```

```
EXPLOITS          68211
FUZZERS           33638
DOS               24582
RECONNAISSANCE    20136
GENERIC           19860
BACKDOOR           4353
ANALYSIS           1881
SHELLCODE          1511
WORMS               169
Name: Attack category, dtype: int64
```

In [28]:

```
#Percentage of missing values in 'Attack Reference' per Attack Category
((df[pd.isnull(df['Attack Reference'])]['Attack category'].value_counts()/df['Attack cat
```

Out[28]:

```
RECONNAISSANCE    90.132102
FUZZERS           88.141388
ANALYSIS          85.964912
SHELLCODE         49.437459
WORMS              6.508876
GENERIC            1.717019
BACKDOOR           1.516196
DOS                0.215605
EXPLOITS           0.007330
Name: Attack category, dtype: float64
```

In [30]:

```
tcp_ports = pd.read_csv('TCP-ports.csv')
tcp_ports['Service'] = tcp_ports['Service'].str.upper()
tcp_ports.head()
```

Out[30]:

	Port	Service	Description
0	0	NaN	Reserved
1	1	TCPMUX	TCP Port Service Multiplexer
2	2	COMPRESSNET	Management Utility
3	3	COMPRESSNET	Compression Process
4	5	RJE	Remote Job Entry

In [31]:

```
print('Dimensions before merging dataframes: ' ,(df.shape))

newdf = pd.merge(df, tcp_ports[['Port','Service']], left_on='Destination Port', right_or
newdf = newdf.rename(columns={'Service':'Destination Port Service'})

print('Dimensions after merging dataframes: ' + str(newdf.shape))
```

```
Dimensions before merging dataframes:  (174341, 11)
Dimensions after merging dataframes: (174341, 13)
```

In [32]:

```
newdf = newdf.drop(columns=['Port'])
newdf.head()
```

Out[32]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destinat I
0	RECONNAISSANCE	HTTP	TCP	175.45.176.0	13284	149.171.126.16	
1	EXPLOITS	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	32
2	EXPLOITS	Browser	TCP	175.45.176.2	23357	149.171.126.16	
3	EXPLOITS	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	5
4	EXPLOITS	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	

In [33]:

```
newdf['Attack category'].unique()
```

Out[33]:

```
array(['RECONNAISSANCE', 'EXPLOITS', 'DOS', 'GENERIC', 'SHELLCODE',
       'FUZZERS', 'WORMS', 'BACKDOOR', 'ANALYSIS'], dtype=object)
```

In [34]:

```
newdf['Attack category'].value_counts()
```

Out[34]:

```
EXPLOITS          68211
FUZZERS           33638
DOS               24582
RECONNAISSANCE    20136
GENERIC           19860
BACKDOOR           4353
ANALYSIS           1881
SHELLCODE          1511
WORMS               169
Name: Attack category, dtype: int64
```

In [35]:

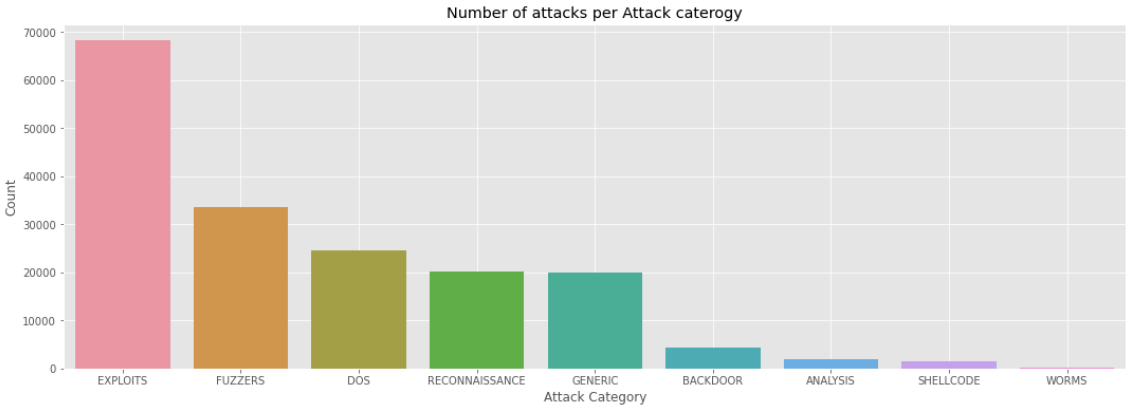
```
newdf['Attack category'].value_counts()*100/newdf['Attack category'].value_counts().sum(
```

Out[35]:

```
EXPLOITS          39.125048
FUZZERS           19.294371
DOS               14.099954
RECONNAISSANCE    11.549779
GENERIC           11.391468
BACKDOOR           2.496831
ANALYSIS           1.078920
SHELLCODE          0.866692
WORMS              0.096936
Name: Attack category, dtype: float64
```

In [36]:

```
plt.figure(figsize=(18,6))
sns.barplot(x=newdf['Attack category'].value_counts().index,y=newdf['Attack category'].v
plt.xlabel('Attack Category')
plt.ylabel('Count')
plt.title('Number of attacks per Attack caterogy')
plt.grid(True)
```



In [37]:

```
pd.DataFrame(newdf['Attack category'].value_counts())[:]
```

Out[37]:

	Attack category
EXPLOITS	68211
FUZZERS	33638
DOS	24582
RECONNAISSANCE	20136
GENERIC	19860
BACKDOOR	4353
ANALYSIS	1881
SHELLCODE	1511
WORMS	169

In [38]:

```
a=pd.DataFrame(newdf['Attack category'].value_counts())[:6]
```

In [95]:

```
a.plot(kind='pie', subplots=True, figsize=(7, 7))
plt.title('Top five attacks')
plt.legend(loc='left')
plt.show()
```

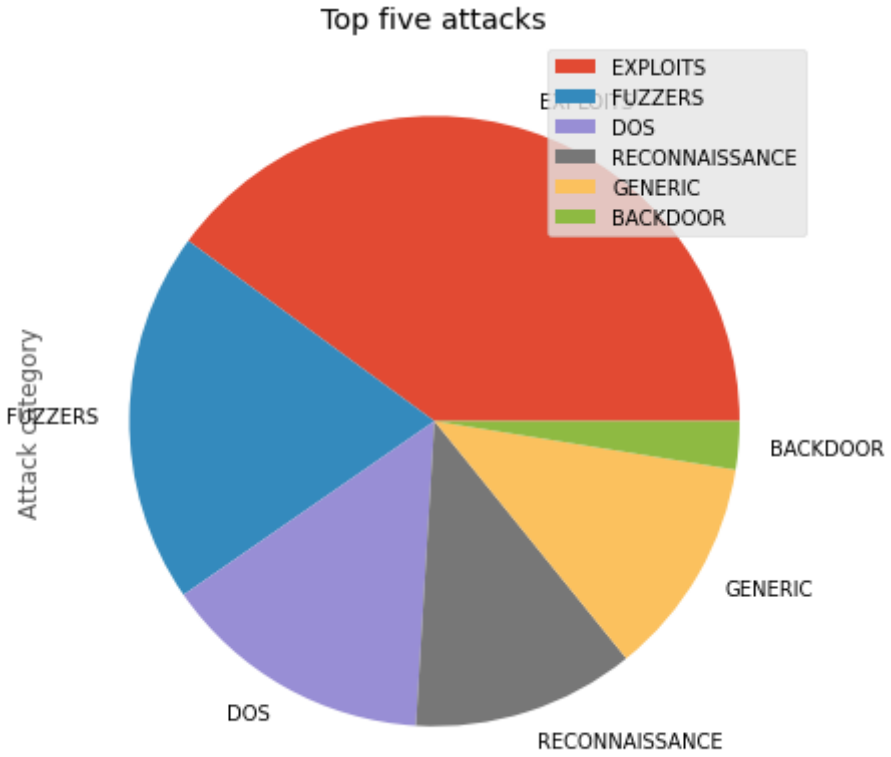
```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-95-5297792e1a53> in <module>
      1 a.plot(kind='pie', subplots=True, figsize=(7, 7))
      2 plt.title('Top five attacks')
----> 3 plt.legend(loc='left')
      4 plt.show()

C:\Users\POOJA SRi\anaconda3\lib\site-packages\matplotlib\pyplot.py in le
gend(*args, **kwargs)
   2736 @_copy_docstring_and_deprecators(Axes.legend)
   2737 def legend(*args, **kwargs):
-> 2738     return gca().legend(*args, **kwargs)
   2739
   2740

C:\Users\POOJA SRi\anaconda3\lib\site-packages\matplotlib\axes\_axes.py i
n legend(self, *args, **kwargs)
    415         if len(extra_args):
    416             raise TypeError('legend only accepts two non-keyword
arguments')
--> 417         self.legend_ = mlegend.Legend(self, handles, labels, **kw
args)
    418         self.legend_._remove_method = self._remove_legend
    419         return self.legend_

C:\Users\POOJA SRi\anaconda3\lib\site-packages\matplotlib\legend.py in __
init__(self, parent, handles, labels, loc, numpoints, markerscale, marker
first, scatterpoints, scatteryoffsets, prop, fontsize, labelcolor, border
pad, labelspacing, handlelength, handleheight, handletextpad, borderaxesp
ad, columnspacing, ncol, mode, fancybox, shadow, title, title_fontsize, f
ramealpha, edgecolor, facecolor, bbox_to_anchor, bbox_transform, frameon,
handler_map)
    453         if isinstance(loc, str):
    454             if loc not in self.codes:
--> 455                 raise ValueError(
    456                     "Unrecognized location {!r}. Valid locations
are\n\t{}\n"
    457                     .format(loc, '\n\t'.join(self.codes)))

ValueError: Unrecognized location 'left'. Valid locations are
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
```



In [1]:

```
newdf['Start time']
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-efcad954d76a> in <module>
----> 1 newdf['Start time']

NameError: name 'newdf' is not defined
```

In [ ]:

```
newdf['Start time'] = pd.to_datetime(newdf['Start time'], unit='s')
newdf['Last time'] = pd.to_datetime(newdf['Last time'], unit='s')
newdf['Duration'] = ((newdf['Last time'] - newdf['Start time']).dt.seconds).astype(int)
```

In [63]:

```
newdf[:5]
```

Out[63]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destinat I
0	RECONNAISSANCE	HTTP	TCP	175.45.176.0	13284	149.171.126.16	
1	EXPLOITS	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	32
2	EXPLOITS	Browser	TCP	175.45.176.2	23357	149.171.126.16	
3	EXPLOITS	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	5
4	EXPLOITS	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	

In [ ]:

```
newdf['Start time'].astype(str).str.split(' ').str[0].unique()
```

In [65]:

```
newdf.describe()
```

Out[65]:

	Source Port	Destination Port
count	174341.000000	174341.000000
mean	15391.130382	1304.599423
std	21707.824000	7466.035607
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	31862.000000	80.000000
max	65535.000000	65535.000000

In [66]:

```
statistic, pvalue = stats.ttest_ind( newdf['Source Port'], newdf['Destination Port'], ec
print('p-value in T-test: ' + str(pvalue))
```

```
p-value in T-test: 0.0
```

In [67]:

```
newdf.corr(method='pearson')
```

Out[67]:

	Source Port	Destination Port
Source Port	1.000000	0.137155
Destination Port	0.137155	1.000000

In [68]:

```
newdf.corr(method='spearman')
```

Out[68]:

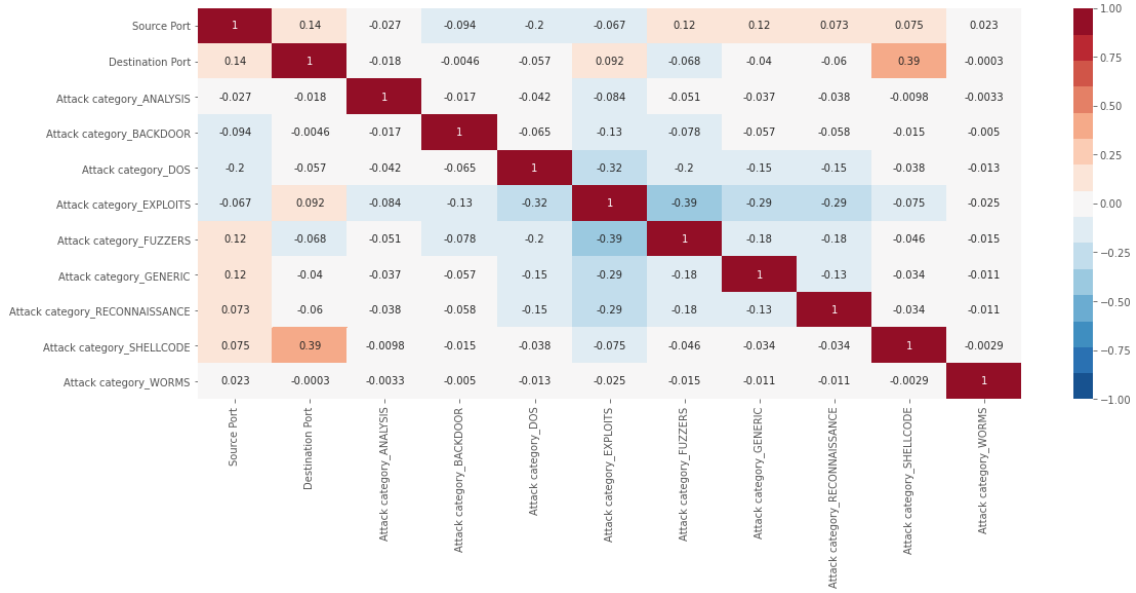
	Source Port	Destination Port
Source Port	1.000000	0.885328
Destination Port	0.885328	1.000000

In [69]:

```
df_dummies = pd.get_dummies(newdf, columns=['Attack category'])
```

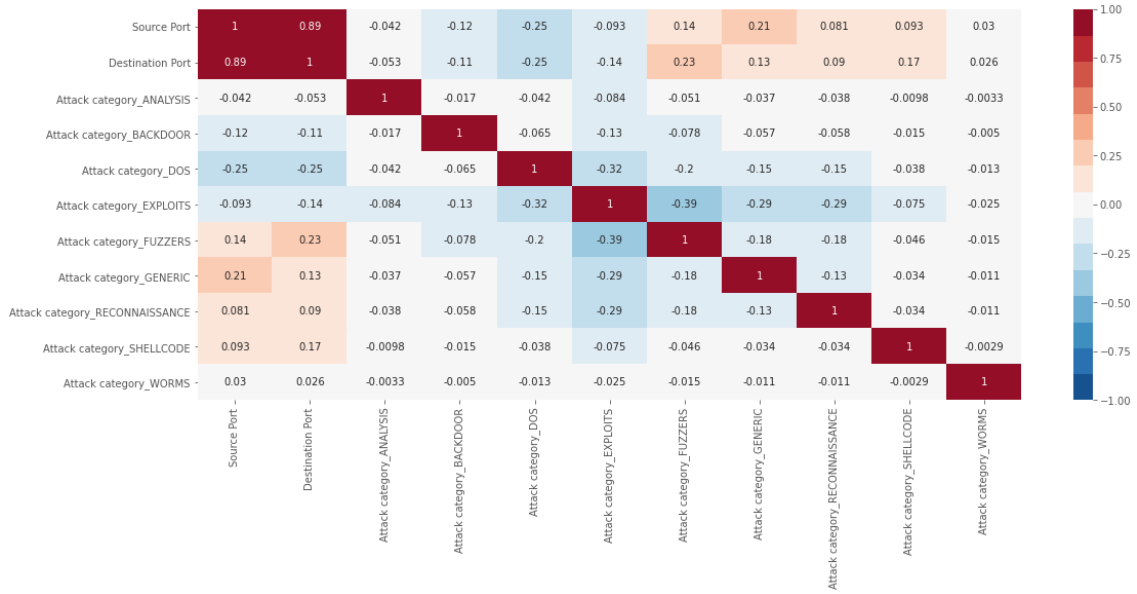
In [70]:

```
plt.figure(figsize=(18,7))
sns.heatmap(df_dummies.corr(method='pearson'),
            annot=True, vmin=-1.0, vmax=1.0, cmap=sns.color_palette("RdBu_r", 15))
plt.show()
```



In [71]:

```
plt.figure(figsize=(18,7))
sns.heatmap(df_dummies.corr(method='spearman'),
            annot=True, vmin=-1.0, vmax=1.0, cmap=sns.color_palette("RdBu_r", 15))
plt.show()
```



In [ ]:

```
g = sns.pairplot(newdf)
g.fig.set_size_inches(11,7)
plt.show()
```

In [78]:

```
newdf['Destination IP'].value_counts()[:5]
```

Out[78]:

```
149.171.126.17    43199
149.171.126.10    24002
149.171.126.19    21619
149.171.126.13    20464
149.171.126.18    13301
Name: Destination IP, dtype: int64
```

In [ ]:

```
plt.figure(figsize=(18,7))
sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start time'], y=newd
plt.xlim(left=newdf['Start time'].min()-timedelta(days=1),right=newdf['Start time'].max(
plt.grid(True)
plt.show()
```

In [ ]:

```
plt.figure(figsize=(18,7))
sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start time'], y=newd
plt.xlim(left=newdf['Start time'].min(),right=datetime.strptime('15-01-23', '%y-%m-%d'))
plt.grid(True)
plt.show()
```

Duration vs Destination ports

In [ ]:

```
plt.figure(figsize=(18,7))
sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start time'], y=newd
plt.xlim(left=datetime.strptime('15-02-18', '%y-%m-%d'),right=newdf['Start time'].max())
plt.grid(True)
plt.show()
```

In [ ]:

```
plt.figure(figsize=(18,7))
sns.scatterplot(x='Start time', y='Destination Port', hue='Attack category',
                data=newdf[(newdf['Destination IP']=='149.171.126.17')&(newdf['Destinati
                s=65)
plt.xlim(left=datetime.strptime('15-02-18 00:00:00', '%y-%m-%d %H:%M:%S'),
         right=datetime.strptime('15-02-18 13:00:00', '%y-%m-%d %H:%M:%S'))
plt.grid(True)
plt.show()
```

plt.figure(figsize=(18,7)) sns.scatterplot(x='Destination Port', y='Duration', hue='Attack category', data=newdf[newdf['Destination IP']=='149.171.126.17']) plt.grid(True) plt.show()

In [ ]:

```
plt.figure(figsize=(18,7))
sns.violinplot(x='Attack category', y='Duration', data=newdf)
plt.grid(True)
plt.show()
```

In [92]:

```
def heatmap_graph(df, xlabel, ylabel, title):
    plt.figure(figsize=(18,8))
    ax = sns.heatmap(df)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    plt.show()
```

In [ ]:

```
newdf["Start time"][1].hour
```

In [ ]:

```
df_pivot = newdf.copy()
df_pivot['hour'] = df_pivot.apply(lambda row: '0'*(2-len(str(row['Start time'].hour)))+s
```

In [ ]:

```
df_pivot[:5]
```

In [ ]:

```
df_p1 = pd.pivot_table(df_pivot,values='Attack Name', index=['hour'], columns=['Attack c
df_p1
```

In [ ]:

```
heatmap_graph(df = df_p1, xlabel = 'Attack category', ylabel = 'Hour', title = 'Number c
```

In [ ]:

```
heatmap_graph(df = df_p1/df_p1.sum(), xlabel = 'Attack category', ylabel = 'Hour', title
```

In [ ]:

```
df_p2 = pd.pivot_table(df_pivot, values='Attack Name', index=['hour'], columns=['Destina
heatmap_graph(df = df_p2/df_p2.sum(), xlabel = 'Destination IP', ylabel = 'Hour', title
```

In [ ]:

```
df_p3 = pd.pivot_table(df_pivot, values='Attack Name', index=['Destination IP'], columns
heatmap_graph(df = df_p3/df_p3.sum(), xlabel = 'Attack category', ylabel = 'Destination
```

In [ ]:

```
for attack in list(newdf['Attack category'].unique()):
    df_attack = newdf[newdf['Attack category'] == attack].copy()
    statistic, pvalue = stats.ttest_ind(df_attack['Source Port'], df_attack['Destinatior
    print('p-value in T-test for ' + attack + ' attack: ' + str(pvalue))
```

In [80]:

```
df_crosstab = pd.crosstab(newdf['Attack category'], newdf['Destination Port'])
df_crosstab
```

Out[80]:

Destination Port	0	10	21	22	23	25	31	42	53	67	...	65455	65460
Attack category													
ANALYSIS	1442	0	0	0	0	6	0	0	0	0	...	0	0
BACKDOOR	4000	0	7	0	0	0	7	0	0	0	...	0	0
DOS	20825	4	75	0	13	425	0	0	154	33	...	0	0
EXPLOITS	40143	0	2198	14	135	4412	0	21	209	98	...	2	2
FUZZERS	13355	0	758	0	0	0	0	0	0	0	...	0	0
GENERIC	2612	0	26	6	0	427	0	0	13438	54	...	0	0
RECONNAISSANCE	8324	0	0	0	7	7	0	0	41	0	...	0	0
SHELLCODE	0	0	0	0	0	0	0	0	0	0	...	0	0
WORMS	0	0	0	0	0	0	0	0	0	0	...	0	0

9 rows × 3182 columns

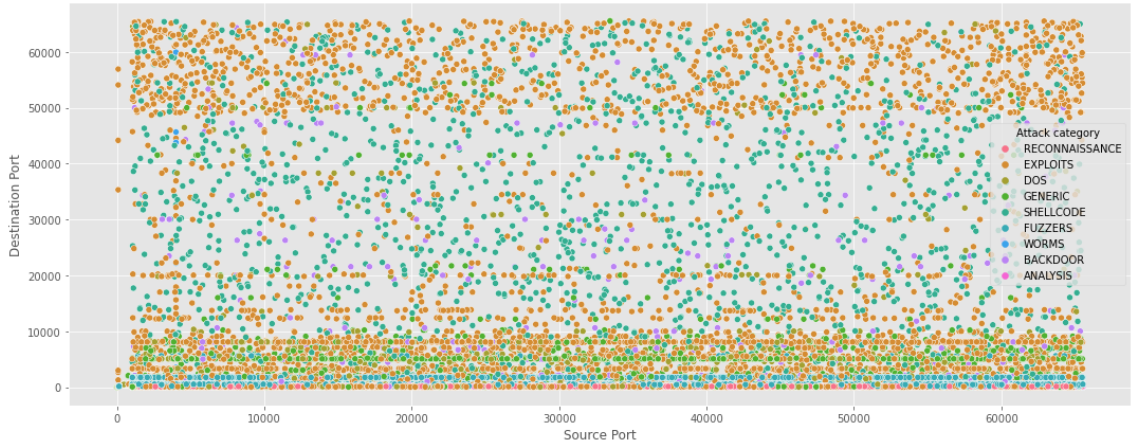
In [81]:

```
chi2, p_value, dof, expected = chi2_contingency(df_crosstab)
print("p-value of Chi-square test for Attack category vs. Destination Port =", p_value)
```

```
p-value of Chi-square test for Attack category vs. Destination Port = 0.0
```

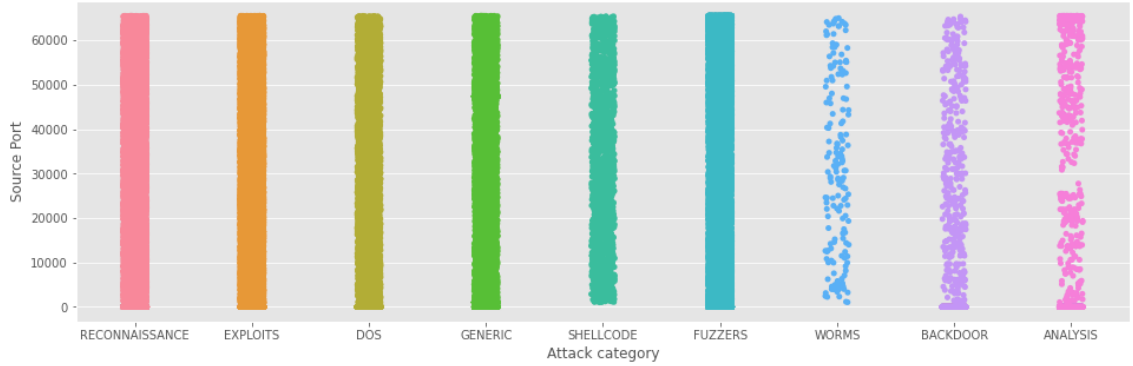
In [82]:

```
plt.figure(figsize=(18,7))
sns.scatterplot(x='Source Port',y='Destination Port', hue='Attack category',data=newdf)
plt.show()
```



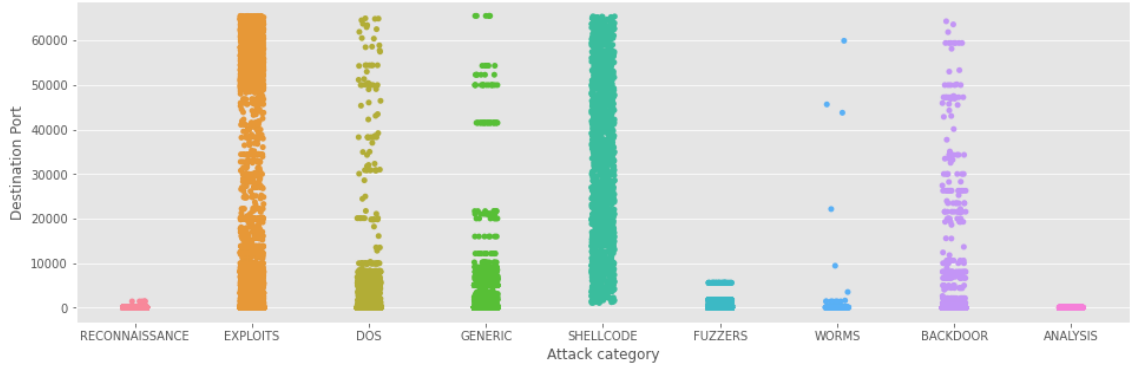
In [83]:

```
# Source ports
plt.figure(figsize=(16,5))
sns.stripplot(x='Attack category',y='Source Port',data=newdf)
plt.show()
```



In [84]:

```
# Destination ports
plt.figure(figsize=(16,5))
sns.stripplot(x='Attack category',y='Destination Port',data=newdf)
plt.show()
```



In [85]:

```
list(newdf['Source IP'].unique())
```

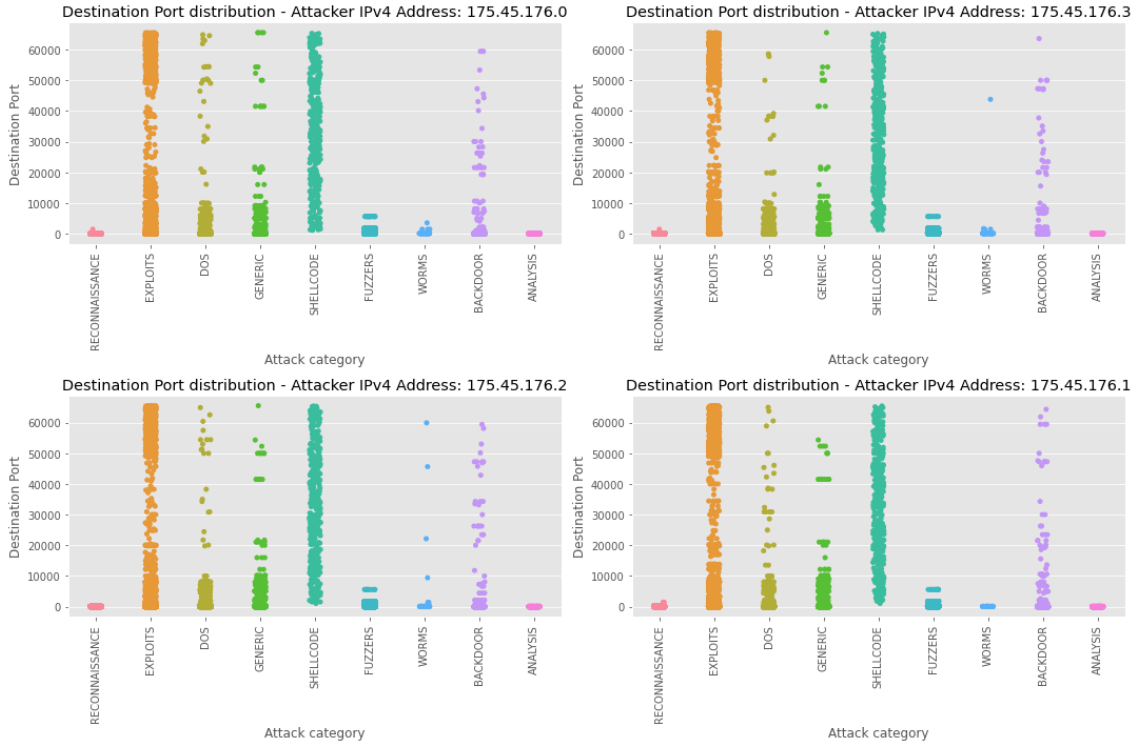
Out[85]:

```
['175.45.176.0', '175.45.176.3', '175.45.176.2', '175.45.176.1']
```

In [86]:

```
ips = list(newdf['Source IP'].unique())
f, axes = plt.subplots(2, 2)
f.set_figheight(10)
f.set_figwidth(15)

labels = list(newdf['Attack category'].unique())
for i, ip in enumerate(ips):
    sns.stripplot(x='Attack category',y='Destination Port',data=newdf[newdf['Source IP']
    axes[int(i/2)][i%2].set_xlabel('Attack category')
    axes[int(i/2)][i%2].set_ylabel('Destination Port')
    axes[int(i/2)][i%2].set_title('Destination Port distribution - Attacker IPv4 Address
    axes[int(i/2)][i%2].set_xticklabels(labels,rotation=90)
plt.tight_layout()
plt.show()
```



In [87]:

```
list(newdf['Destination IP'].unique())
```

Out[87]:

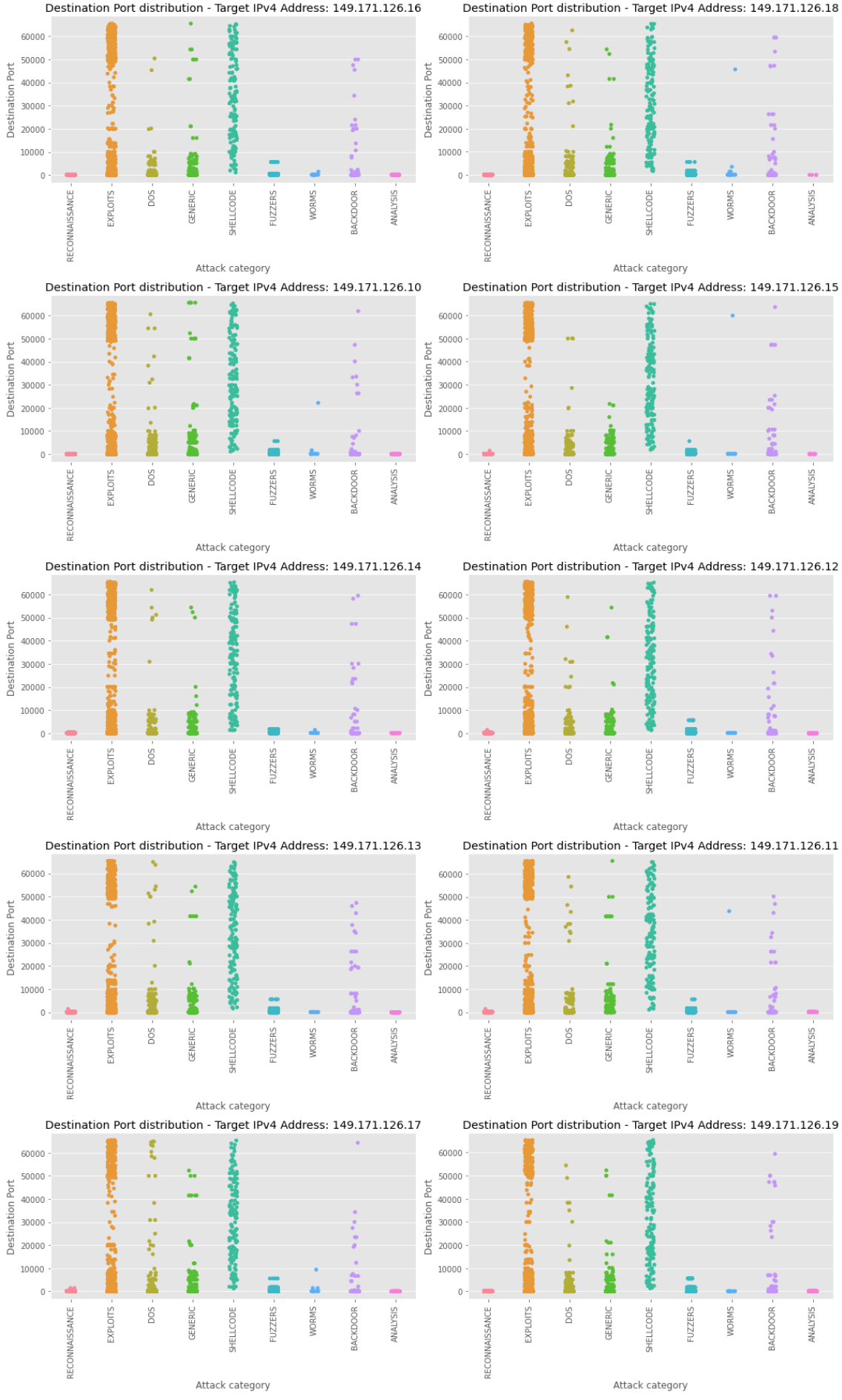
```
['149.171.126.16',
 '149.171.126.18',
 '149.171.126.10',
 '149.171.126.15',
 '149.171.126.14',
 '149.171.126.12',
 '149.171.126.13',
 '149.171.126.11',
 '149.171.126.17',
 '149.171.126.19']
```

In [88]:

```
ips = list(newdf['Destination IP'].unique())
f, axes = plt.subplots(5, 2)
f.set_figheight(25)
f.set_figwidth(15)

labels = list(newdf['Attack category'].unique())

for i, ip in enumerate(ips):
    sns.stripplot(x='Attack category',y='Destination Port',data=newdf[newdf['Destinatior
    axes[int(i/2)][i%2].set_xlabel('Attack category')
    axes[int(i/2)][i%2].set_ylabel('Destination Port')
    axes[int(i/2)][i%2].set_title('Destination Port distribution - Target IPv4 Address:
    axes[int(i/2)][i%2].set_xticklabels(labels,rotation=90)
plt.tight_layout()
plt.show()
```



These graphs show us that there is a differentiation in the way in which the attacks are performing their tasks. There is a particularization by the targets,

something that does not happen with the source devices

In [ ]: