**Prediction of upcoming attacks based on the request proposed under Social-Engineering Attacks**

Data WareHousing and Mining Project Report

Bachelor of Technology

in

**Department of**

**Electronics and Computer Engineering**

By

# K. GNANENDRA (180050001)

**K. PRADEEP (180060010)**

under the supervision of **Dr. M. Ramesh sir Associate Professor**



November,2020

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **S.No** | **Contents** | **Page No** |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
| 3 | Implementation & Methodology | 5-34 |
| 4 | Result | 35 |
| 5 | Conclusion | 36 |
| 6 | References | 36 |
|  |  |  |

**Abstract**

In this Data ware-housing and mining project, the main aim of the present project is to predict the attacks in specific domain in the areas of social-engineering.

In corporations, the main variation of this threat is called spear phishing. This is because cybercriminals collect extremely specific and objective information through social engineering, in addition to targeting precisely certain organizations. In this way, the attacks are smaller, but much more powerful and invasive. Honeypots can be another great source of security information on cyber threats, attackers, their tools, as well as threat actor tactics, techniques, and procedures (TTPs) along with additional valuable information related to the attack.

Social engineering attacks are a major threat to organizations and individuals as digitization and connectivity through the internet increase. This study aims to review scholarly research analyzing the topic of social engineering and further chart the evolution of the threat. The review identifies methods of such attacks on various platforms and devices and discusses motivations behind social engineering attacks.

Finally, the paper analyzes the nature and impact of social engineering attacks and presents a taxonomy on socially engineered attacks by analyzing their anatomy

Our purpose is to classify the attacks and make sure the secure world here after from social engineering-attacks

3

Done By:

180050001 : K. GNANENDRA

180060010 : K. PRADEEP

# Introduction

Semantic social engineering attacks target the user-computer interface in order to deceive a user into performing an action that will breach a system’s information security. On any system, the user interface is always vulnerable to abuse by authorised users, with or without their knowledge. Traditional deception-based attacks, such as phishing emails, spoofed websites and drive-by downloads, have shifted to new and emerging platforms in social media, cloud applications and near field communications.

Today, organizations are greatly dependent on information systems. This reliance has led to vulnerability to information security threats that put data and people at risk.

Furthermore, social engineering fraud has been increasing with advancements in technology. Social engineering is defined in several studies as manipulating and persuading people to disclose sensitive information through online networks or by granting access to restricted areas or systems

In computer security, it is usually computer systems, networks, applications and data that are monitored to be able to detect and mitigate threats. Researchers have also attempted to monitor and profile unauthorised users or witting insiders performing unauthorised actions. However, semantic social engineering attacks target authorised users and lure them into performing an authorised (albeit compromising) action. Recent research in this area has focused on demographic attributes and psychological indicators as methods for predicting user susceptibility. So here is the way we are going to make it ease for the end-user friendly suggestion to any system administrator.

# Methodology

# **DATA COLLECTION**

Data is beeing gathered from the Citizen Lab Malware Indicators Threat intelligence gathered the information and we have taken from it. We can also take the Malware prone attack datasets compromised using the social engineering attack.

# **DATA PRE-PROCESSING**

# ( Loading, Dependent and in-dependent Features)

# Filling Missing data

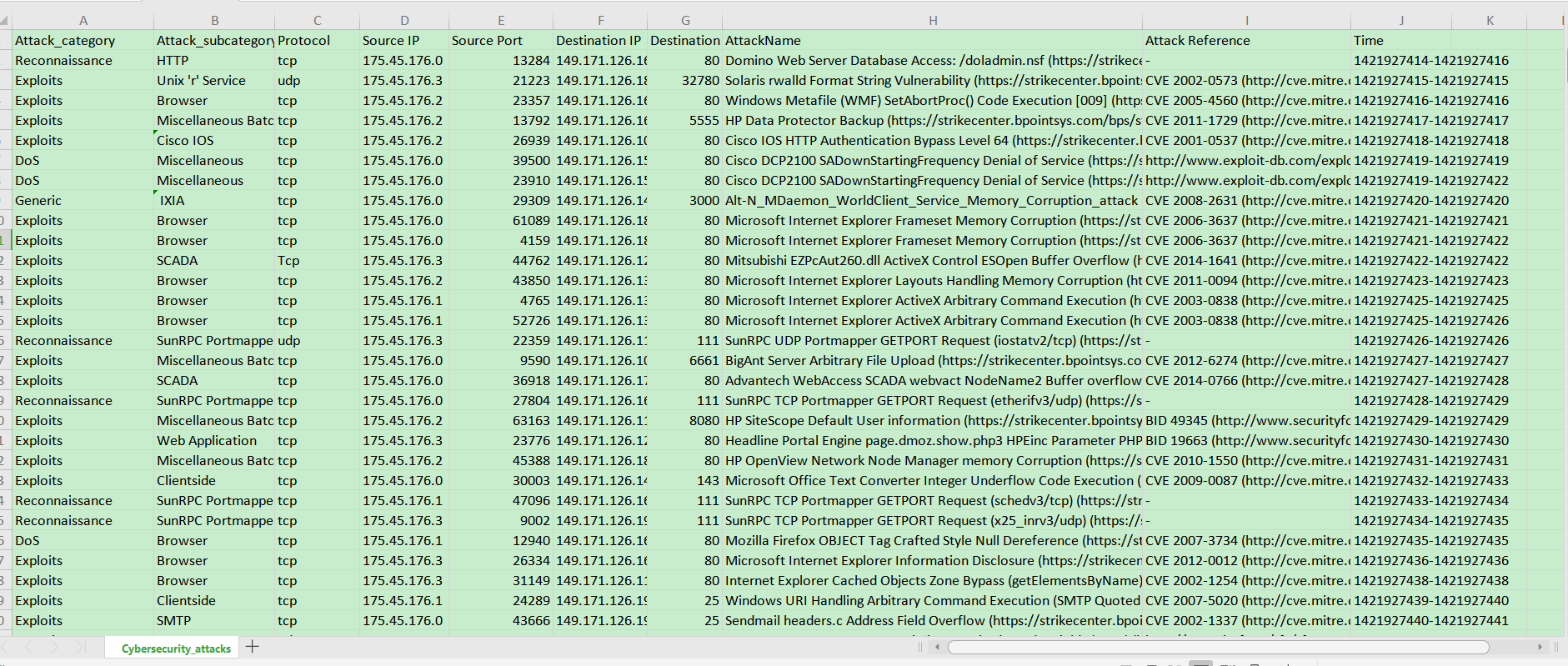
# Scaling Data Transformation

# Encoding for categorical Values

# Finally Complete Data-Visualization needed for *DATA-SCIENTIST* to implement the Machine learning algorithms like the Supervised and unsupervised here we have directly used Chi-squared test

# Implementation

Dataset collected from the threat Intelligence



**Problem statement** : The aim of the present project is to predict the next attacks in specific domain in the areas of social-engineering using Python

In [4]:

*# 1) Most targeted Destination IP Address # 2) Most Logical Ports attacked*

*# 3) Most Frequently/common type of Attack*

*# 4) Different time of the day , (odd , hours, day or night) # 5) Find the Pattern*

|  |  |  |
| --- | --- | --- |
| In [5]: | | **import pandas as pd**  **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**  plt.style.use('ggplot')  **import warnings** |
|  |  |  |
| In | [6]: | warnings.filterwarnings('ignore') |
|  |  |  |
| In | [7]: | df = pd.read\_csv('Cybersecurity\_attacks.csv') df.shape |

Out[7]: (178031, 10)

In [8]:

df.columns

Out[8]: Index(['Attack\_category', 'Attack\_subcategory', 'Protocol', 'Source IP', 'Source Port', 'Destination IP', 'Destination Port', 'AttackName', 'Attack Reference', 'Time'],

dtype='object')

In [9]:

df.head(4)

Out[9]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | **Port** |  | **P** |
| **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 | 327 |
| **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 | 55 |

**Destination IP Destinati**



In [10]:

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

Out[10]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | **Port** |  | **P** |
| **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 | 327 |
| **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 | 55 |
| **4** | Exploits | Cisco IOS | tcp | 175.45.176.2 | 26939 | 149.171.126.10 |  |

**Destination IP Destinati**



In [11]:

df.columns

Out[11]: Index(['Attack\_category', 'Attack\_subcategory', 'Protocol', 'Source IP', 'Source Port', 'Destination IP', 'Destination Port', 'AttackName', 'Attack Reference', 'Time', 'Start time', 'Last time'],

dtype='object')

In [12]:

df.shape

Out[12]: (178031, 12)

In [13]:

df.isnull().sum()

Out[13]: Attack\_category 0

Attack\_subcategory 4192

Protocol 0

Source IP 0

Source Port 0

Destination IP 0

Destination Port 0

AttackName 0

Attack Reference 51745

Time 0

Start time 0

Last time 0

dtype: int64

In [14]:

df["Attack\_subcategory"] = df["Attack\_subcategory"].fillna("Not Registered")

In [15]:

df.isnull().sum()

Out[15]: Attack\_category 0

Attack\_subcategory 0

Protocol 0

Source IP 0

Source Port 0

|  |  |
| --- | --- |
| Destination IP | 0 |
| Destination Port | 0 |
| AttackName | 0 |
| Attack Reference | 51745 |
| Time | 0 |
| Start time | 0 |
| Last time | 0 |
| dtype: int64 |  |

In [16]:

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

Out[16]: (51745, 12)

In [17]:

df[df.duplicated()].shape

Out[17]: (6, 12)

In [18]:

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: (178031,12)

Dimensions after dropping duplicated rows: (178025,12)

In [19]:

pqr = df[df.duplicated()] pqr

Out[19]:

**Attack\_category Attack\_subcategory Protocol Source**

**IP**

**Source**

**Port**

**Destination**

**IP**

**Destination**

**Port**

**Attac**



In [20]:

*#port range 0 to 65535*

In [21]:

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[21]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | **Port** |  |
| **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 |

**Destination IP De**

3684 rows × 12 columns



In [22]:

df = df[~(invalid\_SP | invalid\_DP)].reset\_index(drop=**True**) df

Out[22]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | **Port** |  |
| **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 |

**Destination IP De**

174341 rows × 12 columns



In [23]:

df.shape

Out[23]: (174341, 12)

In [24]:

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[24]: array(['tcp', 'udp', 'Tcp', 'UDP', 'ospf', 'sctp', 'sep', 'mobile',

'sun-nd', 'swipe', 'pim', 'ggp', 'ip', 'ipnip', 'st2'], dtype=object)

In [25]:

df['Attack\_category'].unique()

Out[25]: array(['Reconnaissance', 'Exploits', 'DoS', 'Generic', 'Shellcode', ' Fuzzers', 'Worms', 'Backdoors', 'Analysis', ' Fuzzers ',

' Reconnaissance ', 'Backdoor', ' Shellcode ', 'Reconnaissance '], dtype=object)

In [26]:

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[26]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

**Port**

**Destination IP**

**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 × 12 columns



In [27]:

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 [28]:

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

Out[28]: (50638, 12)

In [29]:

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 [30]:

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 [31]:

*# Percentage of missing values in 'Attack Reference' per Attack Category*

((df[pd.isnull(df['Attack Reference'])]['Attack\_category'].value\_counts()/df[ 'Attack\_category'].value\_counts())\*100).dropna().sort\_values(ascending=**False**)

Out[31]: 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 [32]:

tcp\_ports = pd.read\_csv('TCP-ports.csv')

tcp\_ports['Service'] = tcp\_ports['Service'].str.upper() tcp\_ports.head()

Out[32]:

**Port Service Description**

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

In [33]:

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

newdf = pd.merge(df, tcp\_ports[['Port','Service']], left\_on='Destination Port'

, right\_on='Port', how='left')

newdf = newdf.rename(columns={'Service':'Destination Port Service'})

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

Dimensions before merging dataframes: (174341, 12)

Dimensions after merging dataframes: (174341, 14)

In [34]:

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

Out[34]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | **Port** |  |
| **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 |
|  |  |  |  |  |  |

**Destination IP Destin**

In [35]:

newdf['Attack\_category'].unique()

Out[35]: array(['RECONNAISSANCE', 'EXPLOITS', 'DOS', 'GENERIC', 'SHELLCODE', 'FUZZERS', 'WORMS', 'BACKDOOR', 'ANALYSIS'], dtype=object)

In [36]:

newdf['Attack\_category'].value\_counts()

Out[36]: EXPLOITS 68211

FUZZERS 33638

DOS 24582

RECONNAISSANCE 20136

GENERIC 19860

BACKDOOR 4353

ANALYSIS 1881

SHELLCODE 1511

WORMS 169

Name: Attack\_category, dtype: int64

In [37]:

newdf['Attack\_category'].value\_counts()\*100/newdf['Attack\_category'].value\_cou nts().sum()

Out[37]: 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

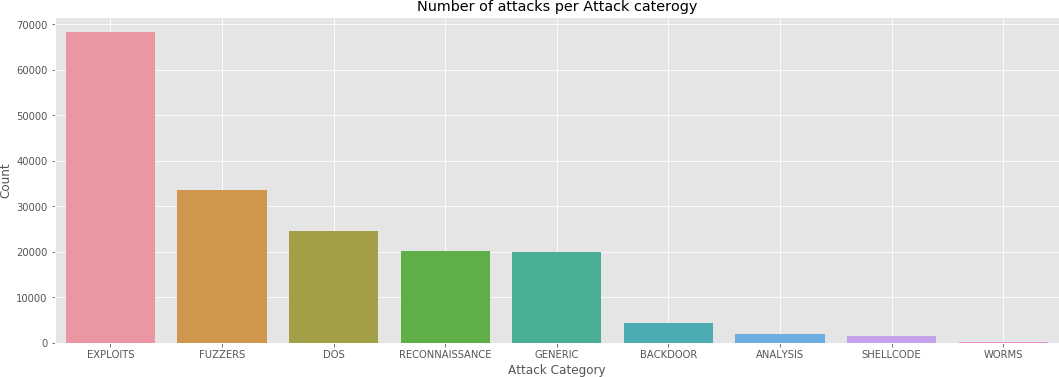
In [38]:

plt.figure(figsize=(18,6))

sns.barplot(x=newdf['Attack\_category'].value\_counts().index,y=newdf['Attack\_ca tegory'].value\_counts())

plt.xlabel('Attack Category') plt.ylabel('Count')

plt.title('Number of attacks per Attack caterogy') plt.grid(**True**)



|  |  |  |
| --- | --- | --- |
| Out[39]: |  | |
|  |  | **Attack\_category** |
|  | **EXPLOITS** | 68211 |
|  | **FUZZERS** | 33638 |
|  | **DOS** | 24582 |
|  | **RECONNAISSANCE** | 20136 |
|  | **GENERIC** | 19860 |
|  | **BACKDOOR** | 4353 |
|  | **ANALYSIS** | 1881 |
|  | **SHELLCODE** | 1511 |
|  | **WORMS** | 169 |

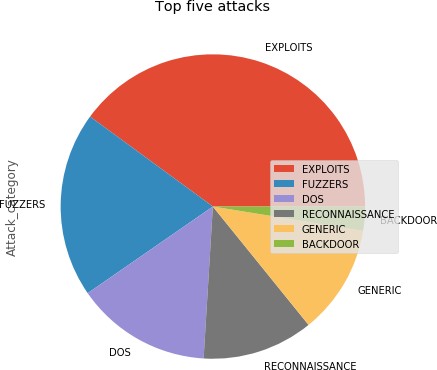
In [40]:

a=pd.DataFrame(newdf['Attack\_category'].value\_counts())[:6]

In [41]:

a.plot(kind='pie', subplots=**True**, figsize=(7, 7)) plt.title('Top five attacks')

plt.legend(loc='right') plt.show()



# NOW TO ANALYSE Attacks WITH DATE AND TIME

In [42]:

newdf['Start time']

|  |  |  |
| --- | --- | --- |
| Out[42]: | 0 | 1421927414 |
|  | 1 | 1421927415 |
|  | 2 | 1421927416 |
|  | 3 | 1421927417 |
|  | 4 | 1421927418 |
|  |  | ... |
|  | 174336 | 1424262066 |
|  | 174337 | 1424262067 |
|  | 174338 | 1424262067 |
|  | 174339 | 1424262068 |
|  | 174340 | 1424262068 |

Name: Start time, Length: 174341, dtype: object

In [43]:

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).as type(int)

In [44]:

newdf[:5]

Out[44]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | **Port** |  |
| **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 |

**Destination IP Destin**



In [45]:

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

Out[45]: array(['2015-01-22', '2015-02-18'], dtype=object)

CASE: we can take as => we are going to execute from now on is based on information related to two days, Thursday - January 22nd/2015, and on Wednesday - February 18th/2015.

In [46]:

newdf.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Out[46]: |  | | | |
|  |  | **Source Port** | **Destination Port** | **Duration** |
|  | **count** | 174341.000000 | 174341.000000 | 174341.000000 |
|  | **mean** | 15391.130382 | 1304.599423 | 2.341572 |
|  | **std** | 21707.824000 | 7466.035607 | 9.309381 |
|  | **min** | 0.000000 | 0.000000 | 0.000000 |
|  | **25%** | 0.000000 | 0.000000 | 0.000000 |
|  | **50%** | 0.000000 | 0.000000 | 0.000000 |
|  | **75%** | 31862.000000 | 80.000000 | 1.000000 |
|  | **max** | 65535.000000 | 65535.000000 | 60.000000 |

Mean and 75% percentile is very different for SOurcePOrt and Destination Port is very different. However minimum and maximum is same. Here comes the Hypothesis testing.

In [47]:

statistic, pvalue = stats.ttest\_ind( newdf['Source Port'], newdf['Destination Port'], equal\_var=**False**)

print('Finally p-value in T-test is: ' + str(pvalue))

Finally p-value in T-test is: 0.0

In [48]:

newdf.corr(method='pearson')

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Out[48]: |  | | | |
|  |  | **Source Port** | **Destination Port** | **Duration** |
|  | **Source Port** | 1.000000 | 0.137155 | -0.078024 |
|  | **Destination Port** | 0.137155 | 1.000000 | -0.026770 |
|  | **Duration** | -0.078024 | -0.026770 | 1.000000 |

In [49]:

newdf.corr(method='spearman')

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Out[49]: |  | | | |
|  |  | **Source Port** | **Destination Port** | **Duration** |
|  | **Source Port** | 1.000000 | 0.885328 | 0.361013 |
|  | **Destination Port** | 0.885328 | 1.000000 | 0.346909 |
|  | **Duration** | 0.361013 | 0.346909 | 1.000000 |

In [50]:

df\_dummies = pd.get\_dummies(newdf, columns=['Attack\_category'])

In [51]:

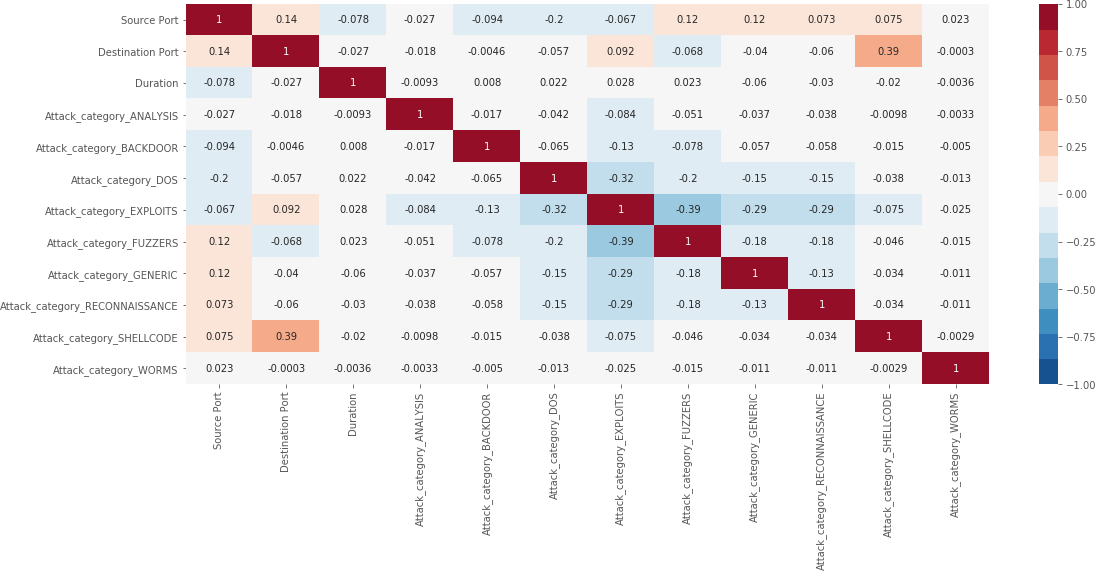
plt.figure(figsize=(18,7))

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 [52]:

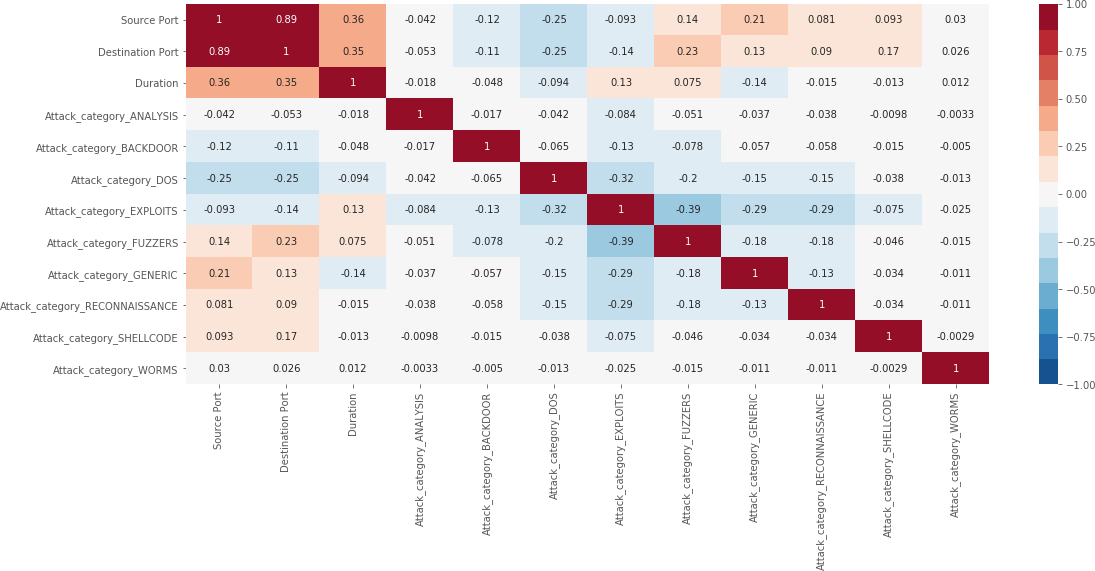
plt.figure(figsize=(18,7))

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 [53]:

g = sns.pairplot(newdf)

g.fig.set\_size\_inches(11,7) plt.show()



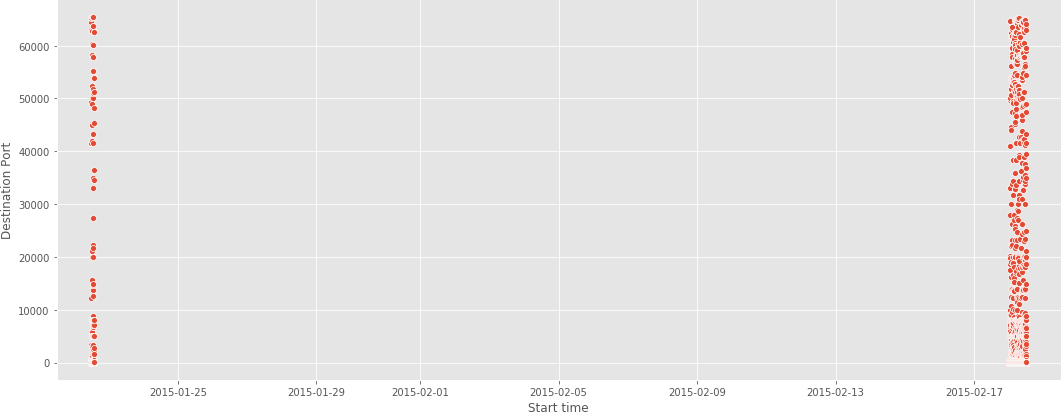
|  |  |
| --- | --- |
|  |  |
| In [54]: newdf['Destination | IP'].value\_counts()[:5] |
| Out[54]: 149.171.126.17 | 43199 |
| 149.171.126.10 | 24002 |
| 149.171.126.19 | 21619 |
| 149.171.126.13 | 20464 |
| 149.171.126.18 | 13301 |
| Name: Destination | IP, dtype: int64 |

In [55]:

plt.figure(figsize=(18,7))

sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start tim e'], y=newdf[newdf['Destination IP']=='149.171.126.17']['Destination Port']) plt.xlim(left=newdf['Start time'].min()-timedelta(days=1),right=newdf['Start t ime'].max()+timedelta(days=1))

plt.grid(**True**) plt.show()

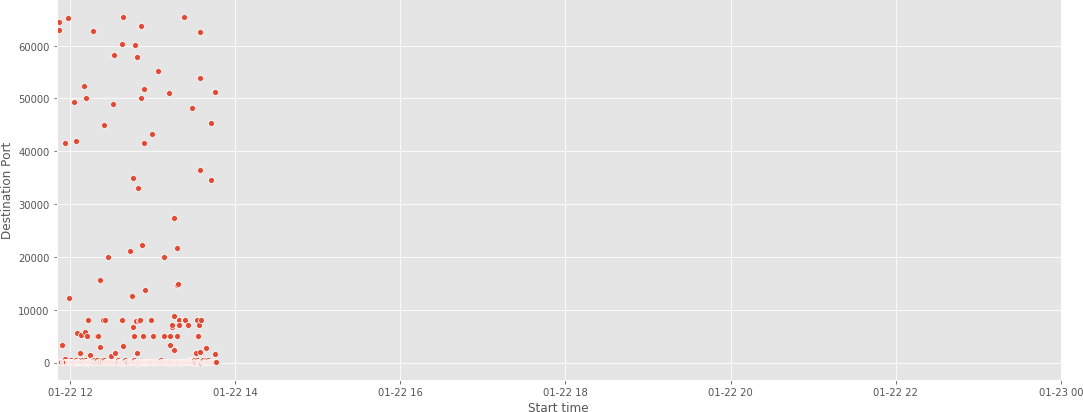


In [56]:

plt.figure(figsize=(18,7))

sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start tim e'], y=newdf[newdf['Destination IP']=='149.171.126.17']['Destination Port']) plt.xlim(left=newdf['Start time'].min(),right=datetime.strptime('15-01-23', '% y-%m-**%d**'))

plt.grid(**True**) plt.show()

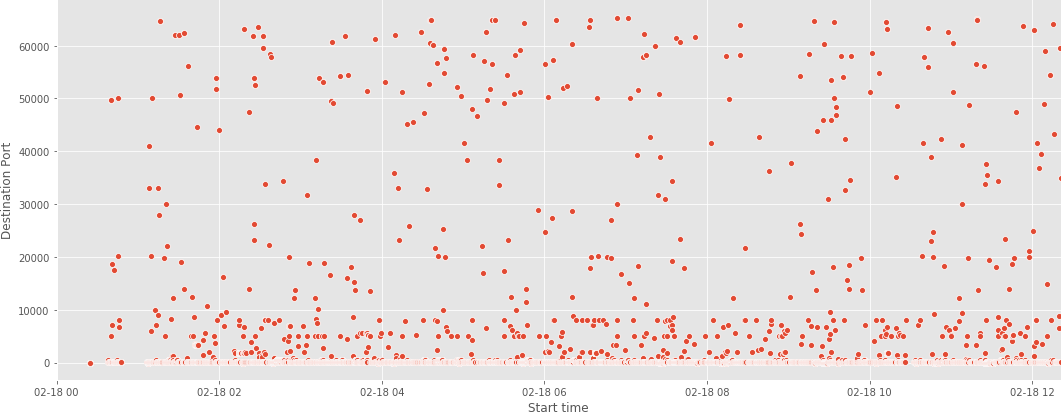


In [57]:

plt.figure(figsize=(18,7))

sns.scatterplot(x=newdf[newdf['Destination IP']=='149.171.126.17']['Start tim e'], y=newdf[newdf['Destination IP']=='149.171.126.17']['Destination Port']) plt.xlim(left=datetime.strptime('15-02-18', '%y-%m-**%d**'),right=newdf['Start tim e'].max())

plt.grid(**True**) plt.show()



In [58]:

plt.figure(figsize=(18,7))

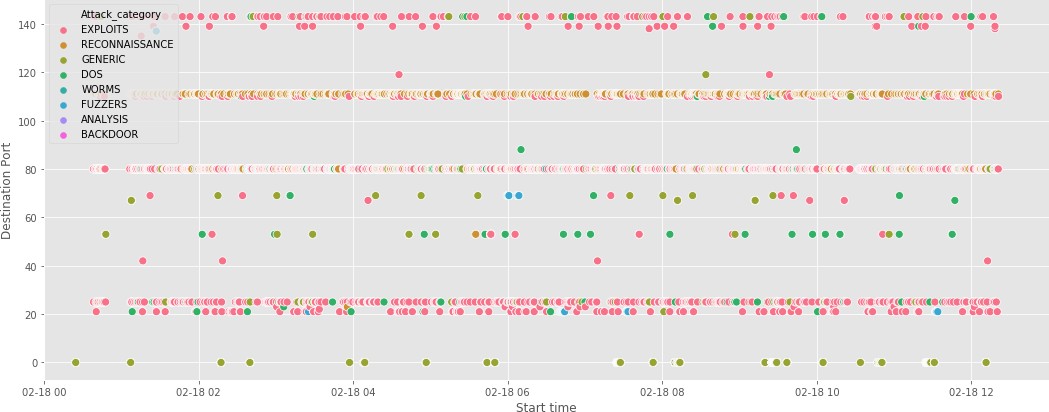
sns.scatterplot(x='Start time', y='Destination Port', hue='Attack\_category',

data=newdf[(newdf['Destination IP']=='149.171.126.17')&(newdf[ 'Destination Port']<=150)],

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()



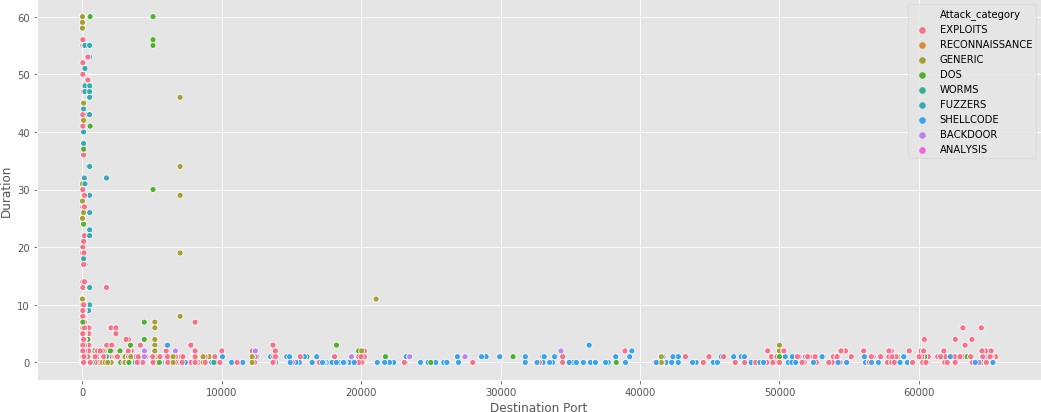
# Duration vs Destination Ports

In [61]:

plt.figure(figsize=(18,7))

sns.scatterplot(x='Destination Port', y='Duration', hue='Attack\_category', dat a=newdf[newdf['Destination IP']=='149.171.126.17'])

plt.grid(**True**) plt.show()

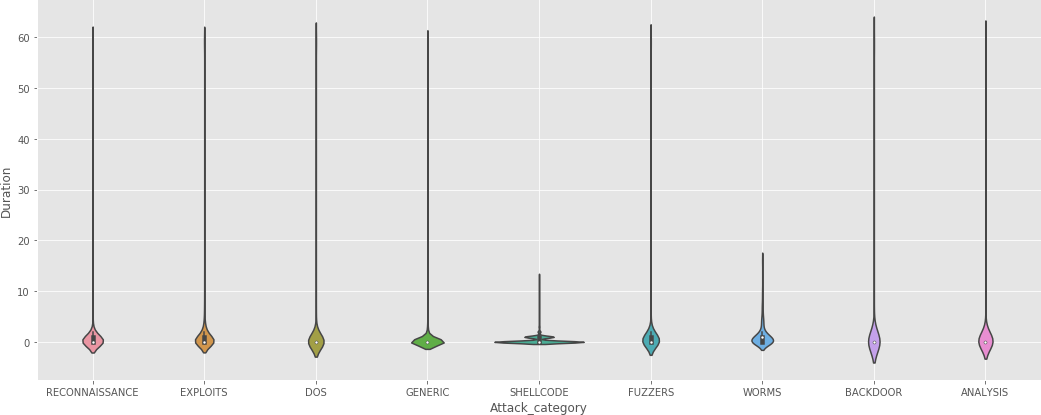


In [62]:

plt.figure(figsize=(18,7))

sns.violinplot(x='Attack\_category', y='Duration', data=newdf) plt.grid(**True**)

plt.show()



In [63]:

**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 [64]:

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

Out[64]: 11

In [65]:

df\_pivot = newdf.copy()

df\_pivot['hour'] = df\_pivot.apply(**lambda** row: '0'\*(2-len(str(row['Start time']

.hour)))+str(row['Start time'].hour)+':00:00', axis=1)

In [66]:

df\_pivot[:5]

Out[66]:

**Attack\_category Attack\_subcategory Protocol Source IP Source**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | **Port** |  |
| **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 |

**Destination IP Destin**



In [67]:

df\_p1 = pd.pivot\_table(df\_pivot,values='AttackName', index=['hour'], columns=[ 'Attack\_category'], aggfunc='count')

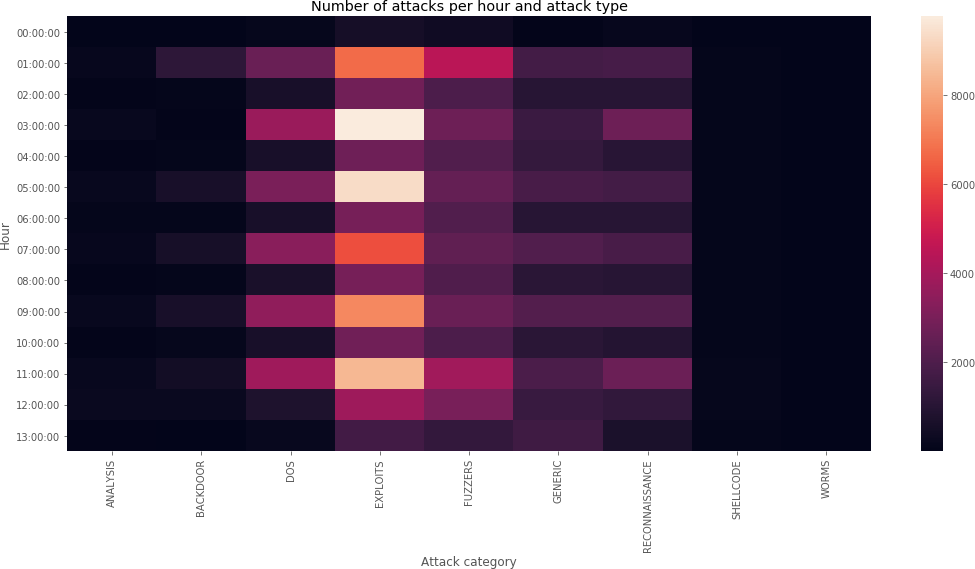
df\_p1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[67]: |  | | | | | | | |
|  | **Attack\_category** | **ANALYSIS** | **BACKDOOR** | **DOS** | **EXPLOITS** | **FUZZERS** | **GENERIC** | **RECONNAISSA** |
|  | **hour** |  |  |  |  |  |  |  |
|  | **00:00:00** | 3 | 16 | 127 | 543 | 391 | 60 |  |
|  | **01:00:00** | 186 | 1148 | 2640 | 6716 | 4477 | 1748 |  |
|  | **02:00:00** | 71 | 100 | 630 | 2861 | 1983 | 1031 |  |
|  | **03:00:00** | 226 | 60 | 3755 | 9759 | 2743 | 1513 |  |
|  | **04:00:00** | 64 | 87 | 617 | 2776 | 2090 | 1349 |  |
|  | **05:00:00** | 198 | 645 | 3038 | 9368 | 2536 | 1834 |  |
|  | **06:00:00** | 84 | 90 | 637 | 2968 | 2065 | 994 |  |
|  | **07:00:00** | 179 | 578 | 3390 | 6151 | 2413 | 2076 |  |
|  | **08:00:00** | 73 | 111 | 664 | 2938 | 2048 | 1081 |  |
|  | **09:00:00** | 199 | 635 | 3545 | 7325 | 2667 | 2108 |  |
|  | **10:00:00** | 79 | 121 | 643 | 2794 | 1981 | 1081 |  |
|  | **11:00:00** | 203 | 470 | 3890 | 8461 | 3923 | 1920 |  |
|  | **12:00:00** | 257 | 266 | 778 | 3845 | 3005 | 1460 |  |
|  | **13:00:00** | 59 | 26 | 228 | 1706 | 1316 | 1605 |  |



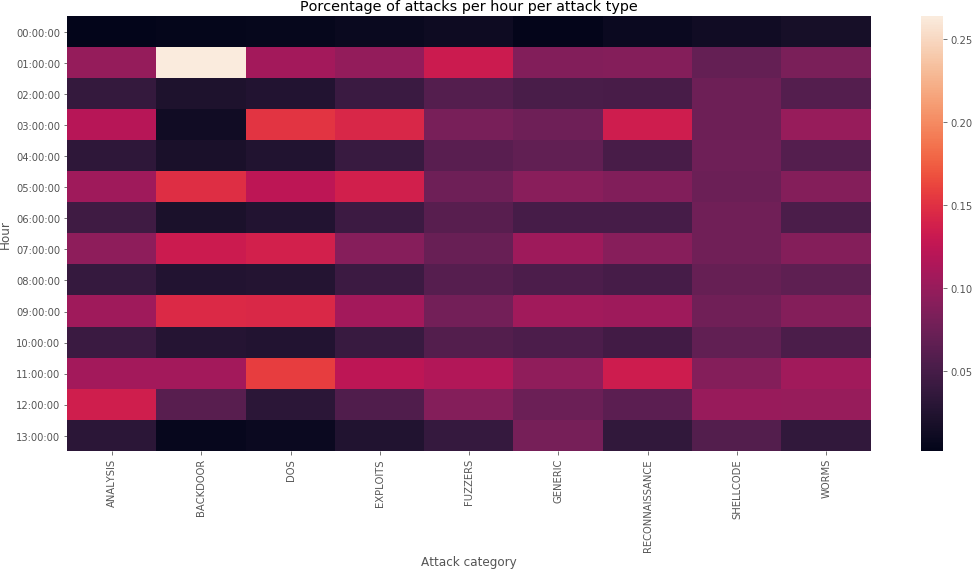
In [68]:

heatmap\_graph(df = df\_p1, xlabel = 'Attack category', ylabel = 'Hour', title = 'Number of attacks per hour and attack type')



In [69]:

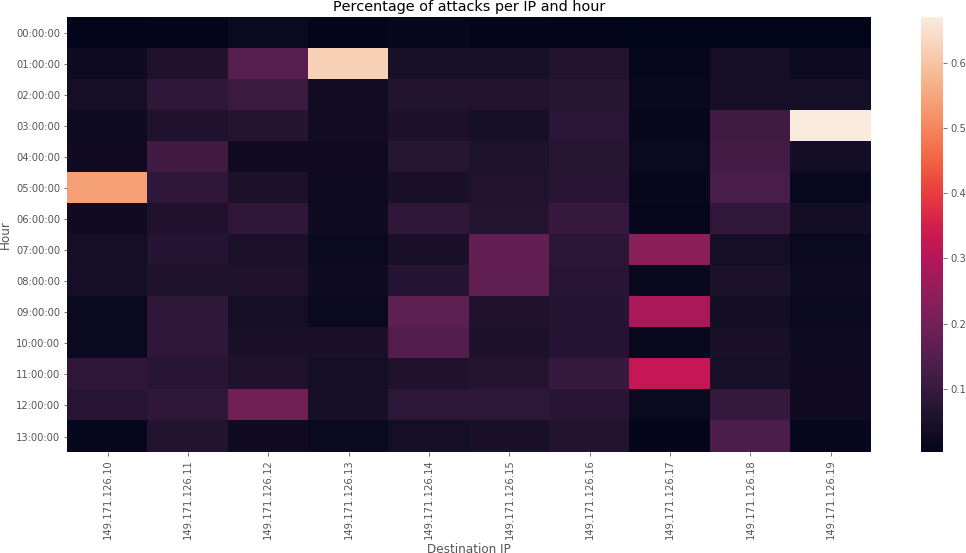
heatmap\_graph(df = df\_p1/df\_p1.sum(), xlabel = 'Attack category', ylabel = 'Ho ur', title = 'Porcentage of attacks per hour per attack type')



In [71]:

df\_p2 = pd.pivot\_table(df\_pivot, values='AttackName', index=['hour'], columns= ['Destination IP'], aggfunc='count')

heatmap\_graph(df = df\_p2/df\_p2.sum(), xlabel = 'Destination IP', ylabel = 'Hou r', title = 'Percentage of attacks per IP and hour')

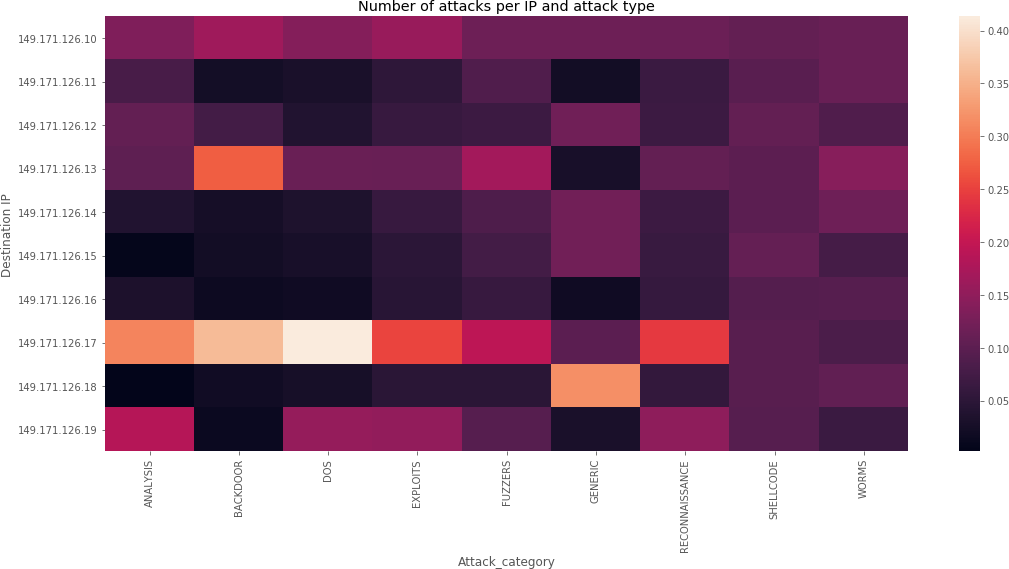


In [75]:

df\_p3 = pd.pivot\_table(df\_pivot, values='AttackName', index=['Destination IP'

], columns=['Attack\_category'], aggfunc='count')

heatmap\_graph(df = df\_p3/df\_p3.sum(), xlabel = 'Attack\_category', ylabel = 'De stination IP', title = 'Number of attacks per IP and attack type')



Let's now look at this same relationship per attack category performing a pair-wise T − test:

In [77]:

**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['D estination Port'], equal\_var=**False**)

print('p-value in T-test for ' + attack + ' attack: ' + str(pvalue))

p-value in T-test for RECONNAISSANCE attack: 0.0 p-value in T-test for EXPLOITS attack: 0.0

p-value in T-test for DOS attack: 0.0

p-value in T-test for GENERIC attack: 0.0

p-value in T-test for SHELLCODE attack: 0.3205085348227197 p-value in T-test for FUZZERS attack: 0.0

p-value in T-test for WORMS attack: 4.246722648635902e-46

p-value in T-test for BACKDOOR attack: 4.8983630604388355e-17 p-value in T-test for ANALYSIS attack: 9.319524862935004e-87

As can be seen, the 𝑝-values of all but one attack category are very close to 0.0. This means that the attacks have been directed to the specific ports, except for the Shellcode attacks, whose null hypothesis cannot be rejected. For this type of attack there is a defined randomness, which means that the source and destination ports have similar averages.

To verify this statement, we will make use of a contingency table which allows

to relate the count of a certain pair of variables, similar to how we

saw the .pivot\_table()

In [79]:

df\_crosstab = pd.crosstab(newdf['Attack\_category'], newdf['Destination Port']) df\_crosstab

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[79]: |  | | | | | | | | | | | | | | |
|  | **Destination Port** | **0** | **10** | **21** | **22** | **23** | **25** | **31** | **42** | **53** | **67** | **...** | **65455** | **65460** | **654** |
|  | **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



{The attack category is independent of the destination port}

In [80]:

chi2, p\_value, dof, expected = chi2\_contingency(df\_crosstab)

print("p-value of Chi-square test for Attack category vs. Destination Port =", p\_value)

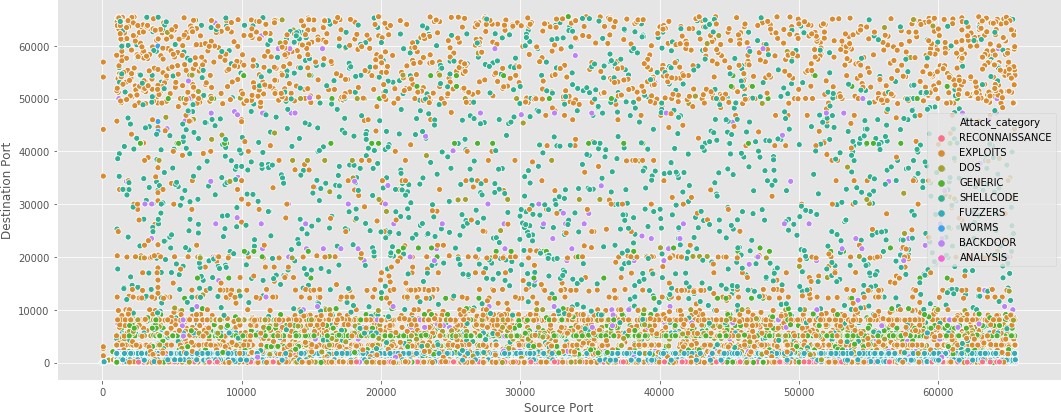
p-value of Chi-square test for Attack category vs. Destination Port = 0.0

In [82]:

plt.figure(figsize=(18,7))

sns.scatterplot(x='Source Port',y='Destination Port', hue='Attack\_category',da ta=newdf)

plt.show()

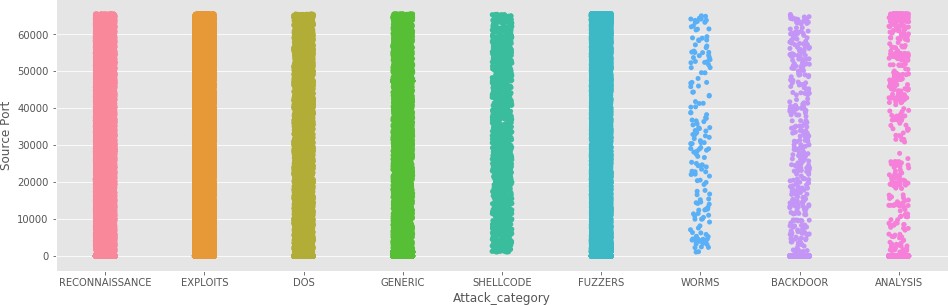


In [83]:

*# 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']

view of the distribution of destination ports by attack category and source IP:

In [87]:

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['S ource IP'] == ip], order=labels, ax=axes[int(i/2)][i%**2**])

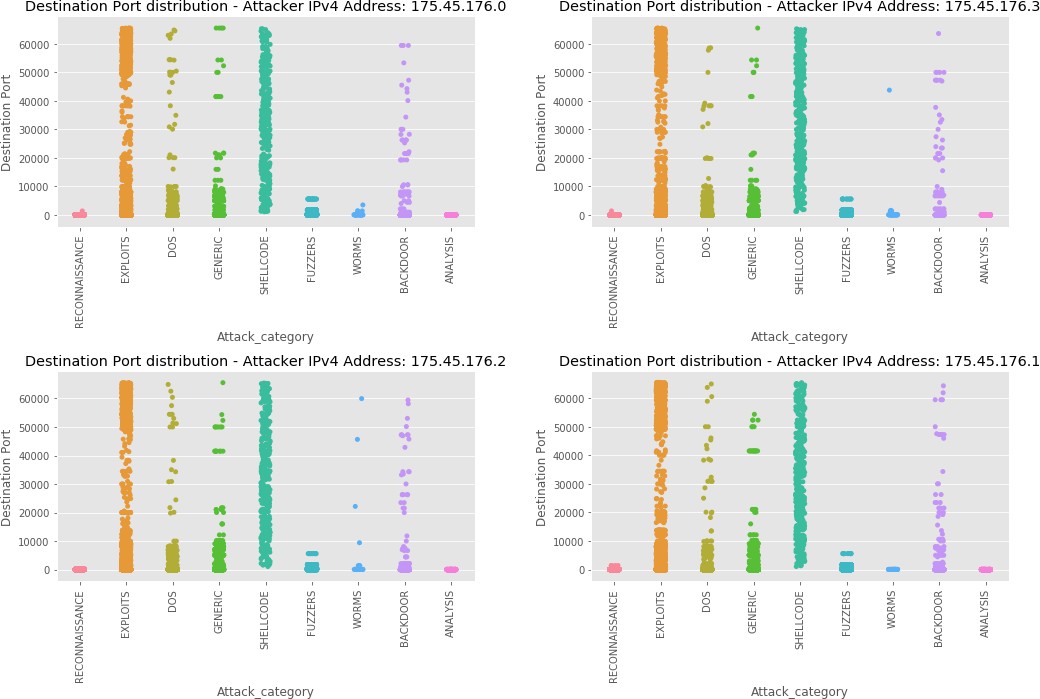
axes[int(i/2)][i%**2**].set\_xlabel('Attack\_category') axes[int(i/2)][i%**2**].set\_ylabel('Destination Port')

axes[int(i/2)][i%**2**].set\_title('Destination Port distribution - Attacker IP v4 Address: ' + ip)

axes[int(i/2)][i%**2**].set\_xticklabels(labels,rotation=90) plt.tight\_layout()

plt.show()





view of the distribution of destination ports by attack category and destination IP:

In [88]:

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

Out[88]: ['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 [89]:

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):

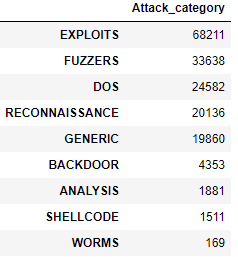
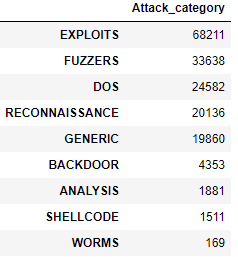
sns.stripplot(x='Attack\_category',y='Destination Port',data=newdf[newdf['D estination IP'] == ip], order=labels, ax=axes[int(i/2)][i%**2**])

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: ' + ip)

axes[int(i/2)][i%**2**].set\_xticklabels(labels,rotation=90) plt.tight\_layout()

plt.show()

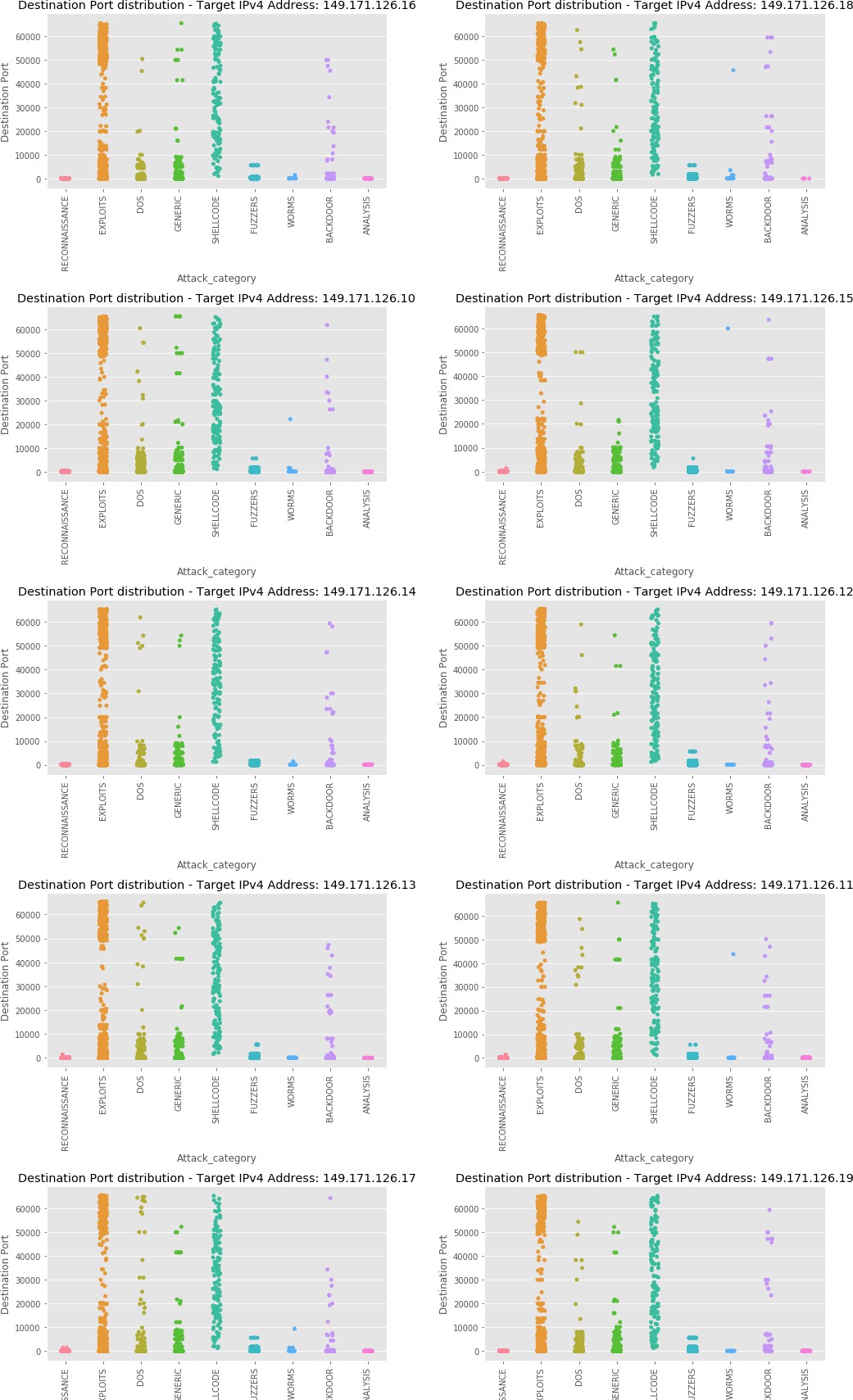


OUTPUT :

no .of attacks involved deciding our need to predict them

OUTPUT -(being Visualized to User):

Here Finally we have seperated the IP address based as a unique parameter to difference the person behind the attack or the region to specify. And therefore based on the request proposed by the Client can make us clear the type of attack because here we are considering the type of services request and the port used to get into the server session and making it vulnerable by the compromising the Server OS. Allowing us to predict the cyber-attack could occur that is only displayed per person or IP we have visualized the Attack prone to occur.



# Conclusion

Finally project suggest the most serious ways to prevent from the companies or the users to safe-guard them by this prediction advice.

An Easy Way to Test Your Cybersecurity Team: In addition to offering precise alerts and requiring low maintenance and reliability of a server can be extended giving a model of attacks occuring, honeypots can also be used to test the cybersecurity skills of your organization's employees.

Overall, the development of the **Prediction of type-Social-Engineering Attacks was successuful** by the graphical and analysis from Chi-Square Test and visuaized the Result to predict one-of-the-attack*( Expoit, Fuzzer, DOS, Reconnaissance, Generic, BackDoor attack, Shell-Code attack, Worms minor attack)* based on the above graph we are able to decide the particular IP address addressing the attacks issue from one of these.

**GITHUB\_URL:** <https://github.com/KVSSKPRADEEP/DataWareHousing-Mining_SocialEnggAttack_prediction.git>

## R eferences:

**h ttps://[www.semanticscholar.org/paper/A-Taxonomy-for-Social-Engineering-](http://www.semanticscholar.org/paper/A-Taxonomy-for-Social-Engineering-)**

**A ttacks-via-Aldawood-Skinner/9860e1b5d4713b2bf5e6ba86846714d758ebd542**

1. Orgill, G. L., Romney, G. W., Bailey, M. G. and Orgill, P. M. The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems. ACM, 2004.
2. Kumar, A., Chaudhary, M. and Kumar, N. Social engineering threats and awareness: a survey. European Journal of Advances in Engineering and Technology, 2, 11 (2015), 15-19.
3. Ivaturi, K. and Janczewski, L. A taxonomy for social engineering attacks. Centre for Information Technology, Organizations, and People, 2011.
4. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7587431>