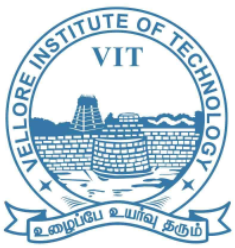


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ANALYSIS OF CYBER ATTACKS

Project Report submitted as part of the course
Information Visualization (CSE3044)

School Of Computer Science and Engineering VIT Chennai

Winter Semester 2020-2021

TITLE

Analysis and Visualization of Cyber Security Attacks Dataset

Abstract

Information theft is the most expensive and fastest growing segment of cybercrime. Largely driven by the increasing exposure of identity information to the web via cloud services. But it is not the only target. Industrial controls that manage power grids and other infrastructure can be disrupted or destroyed. And identity theft isn't the only goal, cyberattacks may aim to compromise data integrity (destroy or change data) to breed distrust in an organization or government.

Cybercriminals are becoming more sophisticated, changing what they target, how they affect organizations and their methods of attack for different security systems.

Social engineering remains the easiest form of cyber-attack with ransomware, phishing, and spyware being the easiest form of entry. Third-party and fourth-party vendors who process your data and have poor cybersecurity practices are another common attack vector, making vendor risk management and third-party risk management all the more important.

Data breaches can involve financial information like credit card numbers or bank account details, protected health information (PHI), personally identifiable information (PII), trade secrets, intellectual property and other targets of industrial espionage. Other terms for data breaches include unintentional information

disclosure, data leak, cloud leak, information leakage or a data spill.

Other factors driving the growth in cybercrime include:

- The distributed nature of the Internet
- The ability for cybercriminals to attack targets outside their jurisdiction making policing extremely difficult
- Increasing profitability and ease of commerce on the dark web
- The proliferation of mobile devices and the Internet of Things.

Introduction

The use of cyber security can help prevent cyber-attacks, data breaches, and identity theft and can aid in risk management.

Cyber security is the application of technologies, processes and controls to protect systems, networks, programs, devices and data from cyberattacks. It aims to reduce the risk of cyberattacks and protect against the unauthorized exploitation of systems, networks and technologies.

We all know that Cybersecurity attack is any form of malicious activity that targets IT systems, or the people using them, to gain unauthorized access to the systems and the data or information they contain. In most cases, the cyber-attackers are criminals looking to exploit the attack for financial gain.

For analysing the cyber security attacks, we have collected related dataset which have Attack Category, Source IP, Time of attack etc. By considering these domains we analyse the

data in order to predict the pattern of the attack which may occur in future.

By analysing the dataset of cyber security attacks with respect to type of attack, source and destination IP address we will be finding

- The most targeted destination IP address
- Logical ports attacked
- Most common type of attack
- Number of attacks per hour w.r.t type of attack

Finally, we will find the frequency and pattern of an attack and predict the attacks which have great possibilities of occurrence in future. We will analyse the data using different statistical methods like univariate and multivariate analysis, correlation and regression analysis etc. in order to predict the data with the best accuracy.

From the final output of our analysis, we will be providing a report regarding the predicted pattern to the victims(companies) which had been affected previously by any type of an attack or the companies which are trying to improve their security in order to avoid those attacks.

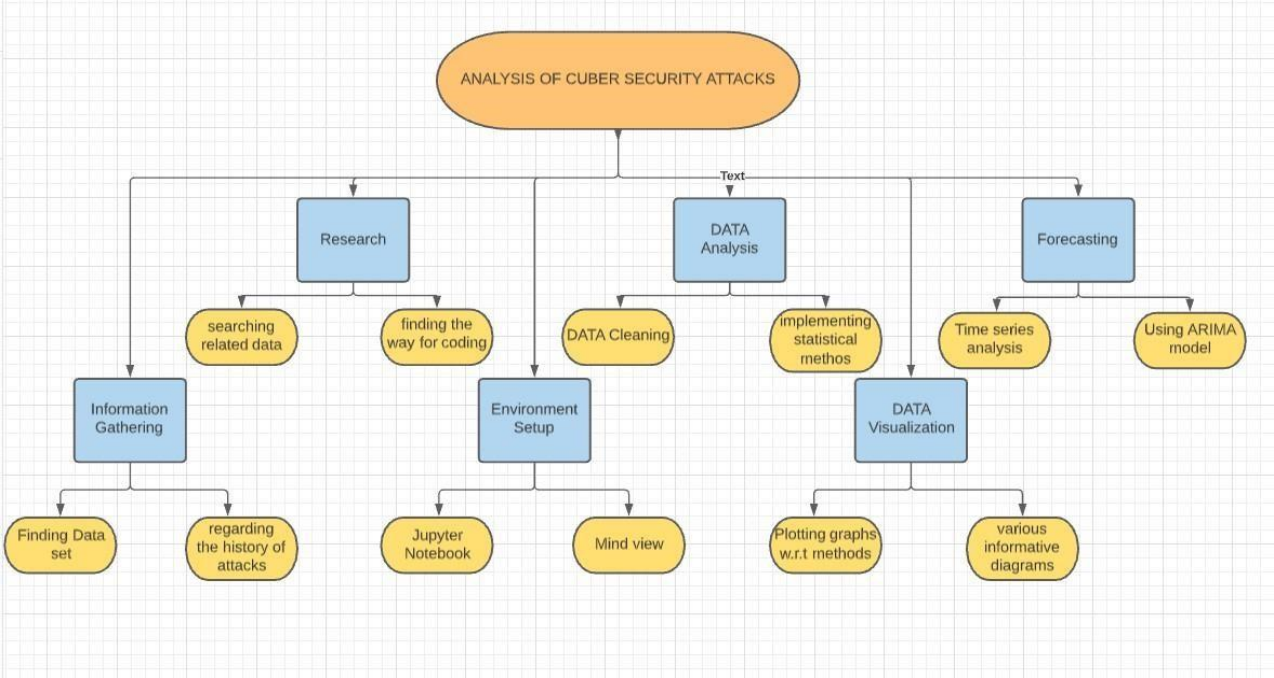
Our analysis is also very helpful for our government (Central or state) to protect the confidential data from any type of cyberattack by knowing the pattern of attack and preventing it before the attack takes place.

We also included Time series analysis

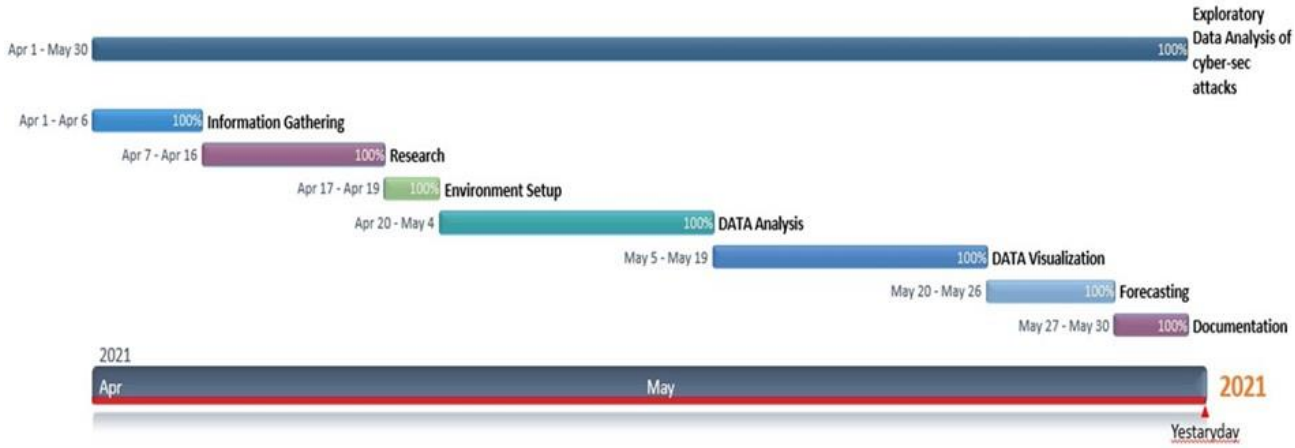
- In the field for machine learning and data science, most of the real-life problems are based upon the prediction of future such as stock market prediction, future sales prediction and so on. Time series problem is basically the

prediction of such problems using various machine learning tools. Time series problem is tackled efficiently when first it is analysed properly (Time Series Analysis) and according to that observation suitable algorithm is used (Time Series Forecasting). Using Time Series Analysis, we will be predicting the Future attacks using ARIMA Model

Architecture (Work Flow)



Gantt Chart



1. Finding Appropriate DATASET

Finding the dataset to initiate the process of research and analysis

2. Research

Gathering more information to enhance our analysis

3. Environment setup

Setting up the environment for analysis

Analysis and Visualization – Jupyter Notebook (PYTHON)

4.DATA Analysis

Implementing data analytical techniques

5. DATA Visualization

Visualizing the data after every action of data analysis

6. FORECASTING

Forecasting attacks for succeeding year

DESCRIPTION OF MODULES

We have divided the project into four modules:

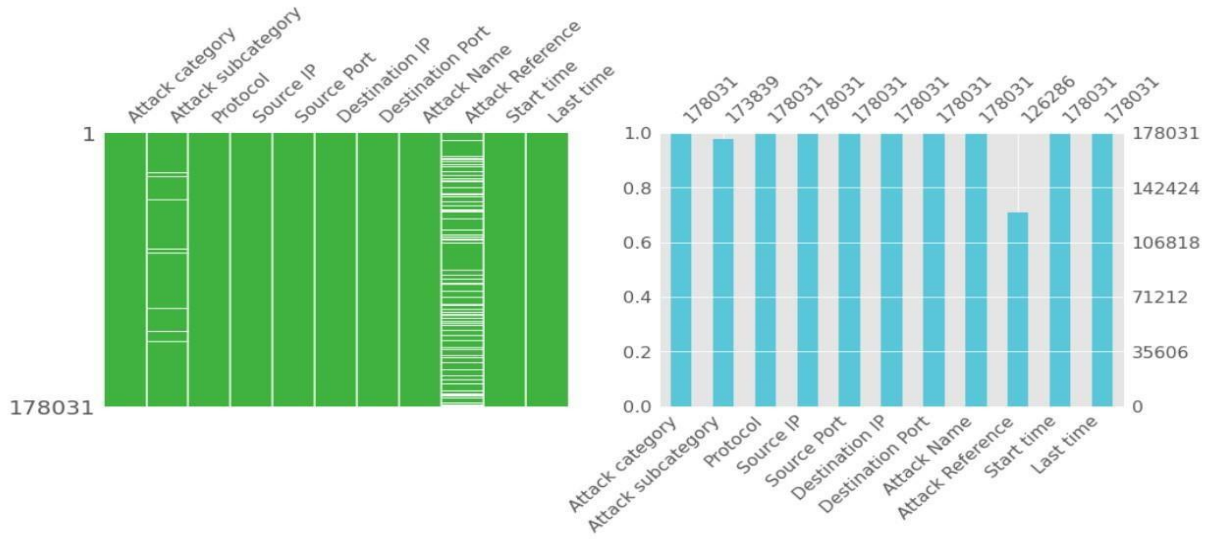
1.DATA cleaning and Missing Value Analysis
2.Correlation calculation
3.Statistical and Informative Graphs
4.Time Series Analysis (Forecasting)

1.DATA CLEANING:

1. Python libraries and packages were imported
2. Extracted Data set of cybersecurity attacks Dropping duplicated content in the data

- After Dropping the unwanted data, it will go through missing value analysis in which finding the missing values in different Categories by plotting two types of graphs i.e., matrix and bar graph.

```
figure, (ax1, ax2) = plt.subplots(1, 2, figsize=(16,5))
msno.matrix(df, ax=ax1, sparkline=False, color=(0.25, 0.70, 0.25))
msno.bar(df, ax=ax2, color=(0.35, 0.78, 0.85))
plt.show()
```



- From the above graph we can notice that the two domains attack sub-categeory and attack reference have missing values remaining domains have no missing values • As we have to predict exact patten of an attack we did'nt use any imputation methods rather we assign the missing values as "NotRegistered".
- After filling the missing values as Not-Registered, the attack subcategeory will have no more missing values
- Now, for Attack Reference We Drop Duplicated Rows
- And removed invalid Source Ports and Destination Ports as the valid range is from 0 to 65535 i.e less than 0 and greater that 65535 are considered as invalid source and destination ports

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

```
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
Start time                0
Last time                 0
dtype: int64
```

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

```
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
Start time                0
Last time                 0
dtype: int64
```

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

```
(6, 11)
```

```
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)
```

- These are Percentages of Missing values of a particular attack
  Reference w.r.t Attack Category

```
# 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().sor
```

```
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
```

- And it is assigned with Nan (Not Available now)
- Merging Data Sets
- After assigning the missing values the data is cleaned in order to merge with another Dataset i.e TCP Ports.csv

- Which have complete Ports Data and Description Using merge function in pandas

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

- By Dropping the Duplicated Column “port” the Data set is Clean for Further Analysis

2.CORRELATION ANALYSIS:

Performing correlation:

We will be using two methods for correlation calculation:
•Pearson's correlation: evaluates the linear relationships between two variables. If the value is close to 0, there is a week or nonexistent linear relationship between the variables.
•Spearman's correlation: evaluates the monotonic relationships between two variables. If the value is close to 0, there is a weak or non-existent monotonic relationship between the variables.

1. Pearson Correlation:

```
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()
```


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

	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

2. Spearman Correlation

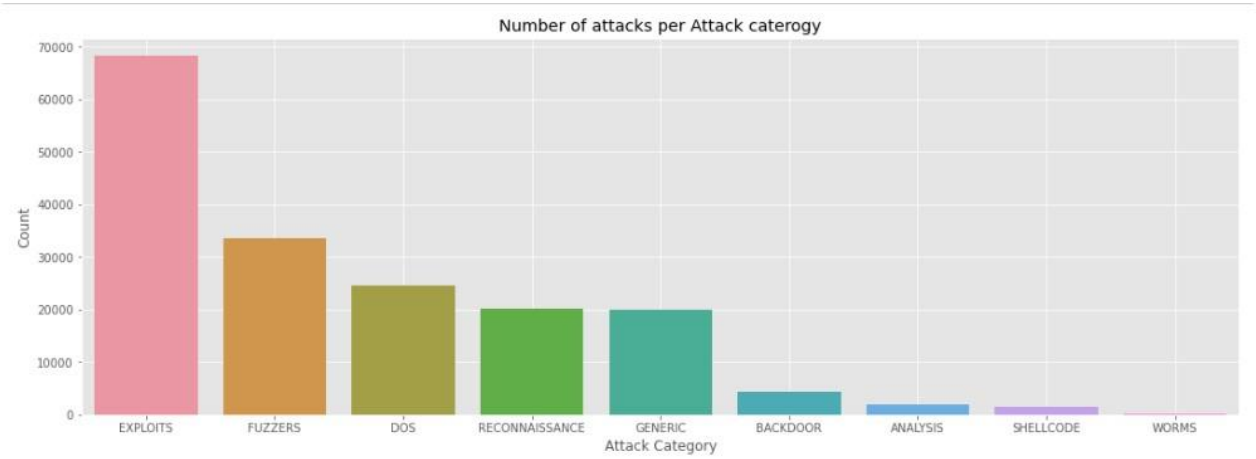
```
plt.figure(figsize=(18,7))
sns.heatmap(df_dummies.corr(method='spearman'),
            annot=True, vmin=-1.0, vmax=1.0, cmap=sns.color_palette("RdBu_r", 15))
plt.show()
```

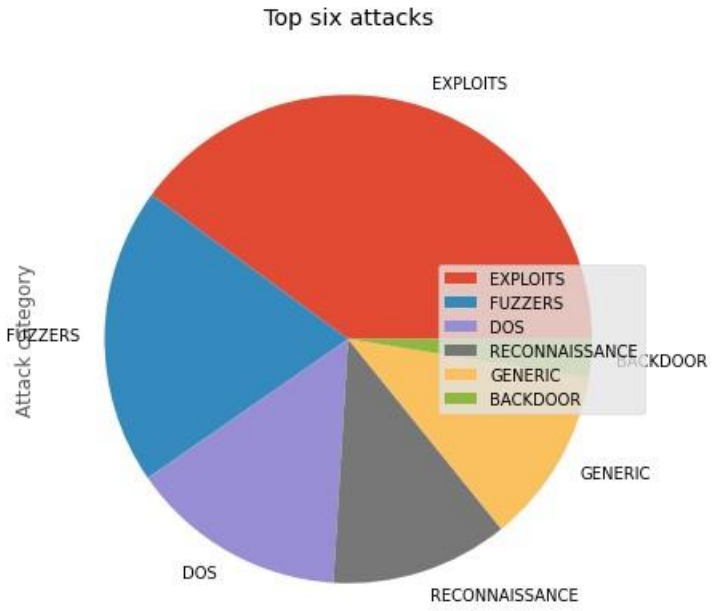

```
newdf.corr(method='spearman')
```

	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

3. Statistical and Informative Graphs

Number of Attacks per Attack Category





Analysing Attacks with DATE AND TIME

egory	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port	Attack Name	Attack Reference	Start time	Last time	Destination Port Service
ANCE	HTTP	TCP	175.45.176.0	13284	149.171.126.16	80	Domino Web Server Database Access: /doladmin.n...	-	1421927414	1421927416	HTTP
LOITS	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	32780	Solaris rwalld Format String Vulnerability (ht...	CVE 2002-0573 (http://cve.mitre.org/cgi-bin/cv...	1421927415	1421927415	NaN
LOITS	Browser	TCP	175.45.176.2	23357	149.171.126.16	80	Windows Metafile (WMF) SetAbortProc() Code Exe...	CVE 2005-4560 (http://cve.mitre.org/cgi-bin/cv...	1421927416	1421927416	HTTP
LOITS	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	5555	HP Data Protector Backup (https://strikecenter...	CVE 2011-1729 (http://cve.mitre.org/cgi-bin/cv...	1421927417	1421927417	PERSONAL-AGENT
LOITS	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	80	Cisco IOS HTTP Authentication Bypass Level 64 ...	CVE 2001-0537 (http://cve.mitre.org/cgi-bin/cv...	1421927418	1421927418	HTTP

```
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)
```

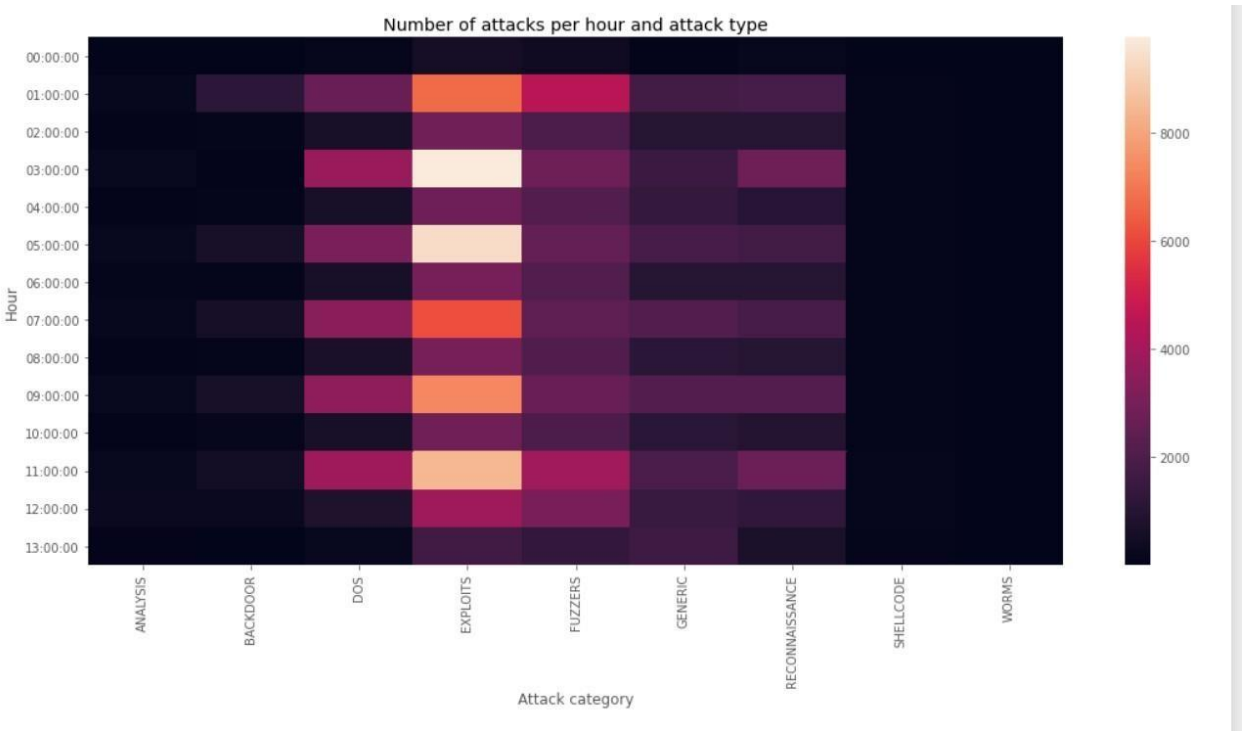
```
newdf[:5]
```

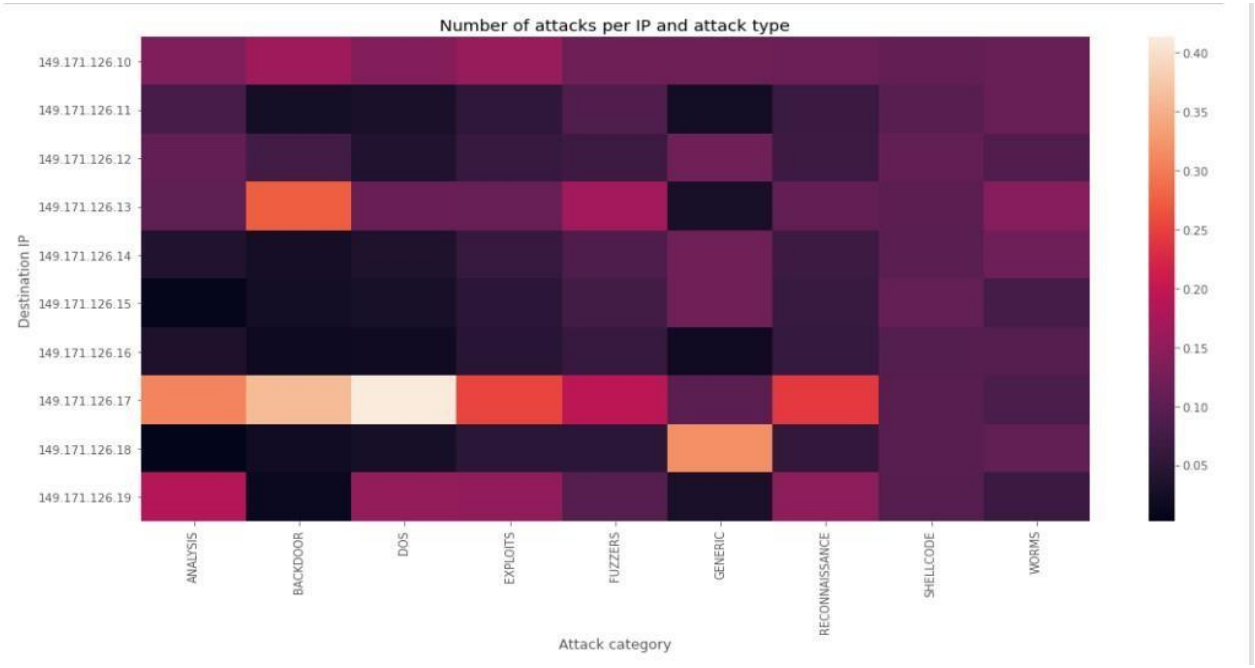
	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port	Attack Name	Attack Reference	Start time	Last time
0	RECONNAISSANCE	HTTP	TCP	175.45.176.0	13284	149.171.126.16	80	Domino Web Server Database Access: /doladmin.n...	-	2015-01-22 11:50:14	2015-01-22 11:50:16
1	EXPLOITS	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	32780	Solaris rwalld Format String Vulnerability (ht...	CVE 2002-0573 (http://cve.mitre.org/cgi-bin/cv...	2015-01-22 11:50:15	2015-01-22 11:50:15
2	EXPLOITS	Browser	TCP	175.45.176.2	23357	149.171.126.16	80	Windows Metafile (WMF) SetAbortProc() Code Exe...	CVE 2005-4560 (http://cve.mitre.org/cgi-bin/cv...	2015-01-22 11:50:16	2015-01-22 11:50:16
3	EXPLOITS	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	5555	HP Data Protector Backup (https://strikecenter...	CVE 2011-1729 (http://cve.mitre.org/cgi-bin/cv...	2015-01-22 11:50:17	2015-01-22 11:50:17
4	EXPLOITS	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	80	Cisco IOS HTTP Authentication Bypass Level 64 ...	CVE 2001-0537 (http://cve.mitre.org/cgi-bin/cv...	2015-01-22 11:50:18	2015-01-22 11:50:18

```
: newdf.describe()
```

:

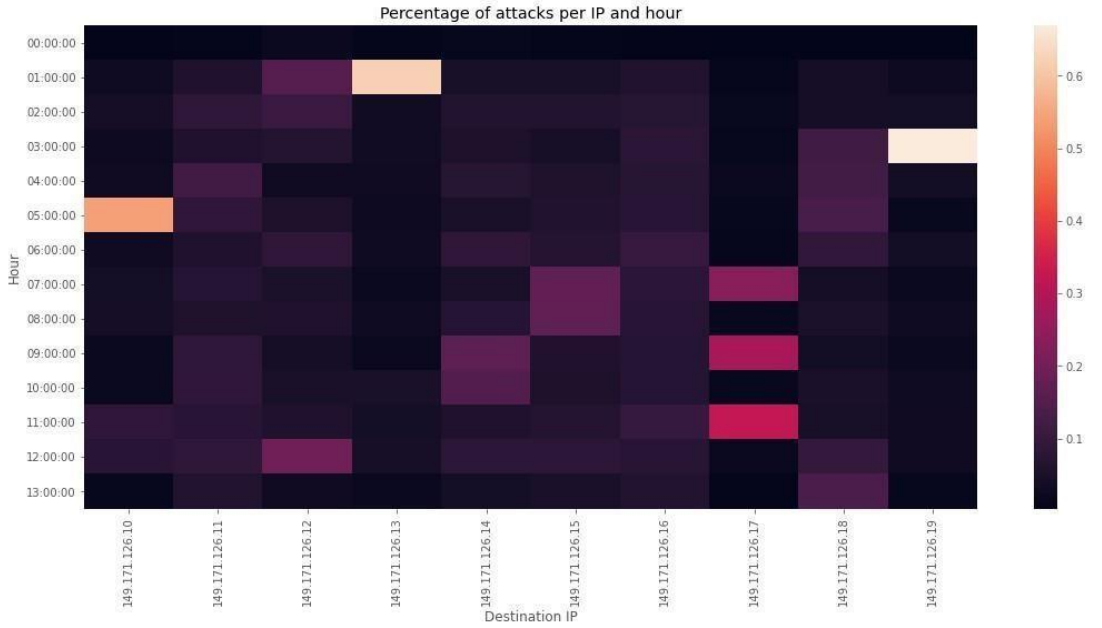
	Source Port	Destination Port	Duration
count	174341.000000	174341.000000	174341.000000
mean	15391.130382	1304.599423	2.341572
std	21707.824000	7466.035607	9.309381
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	31862.000000	80.000000	1.000000
max	65535.000000	65535.000000	60.000000



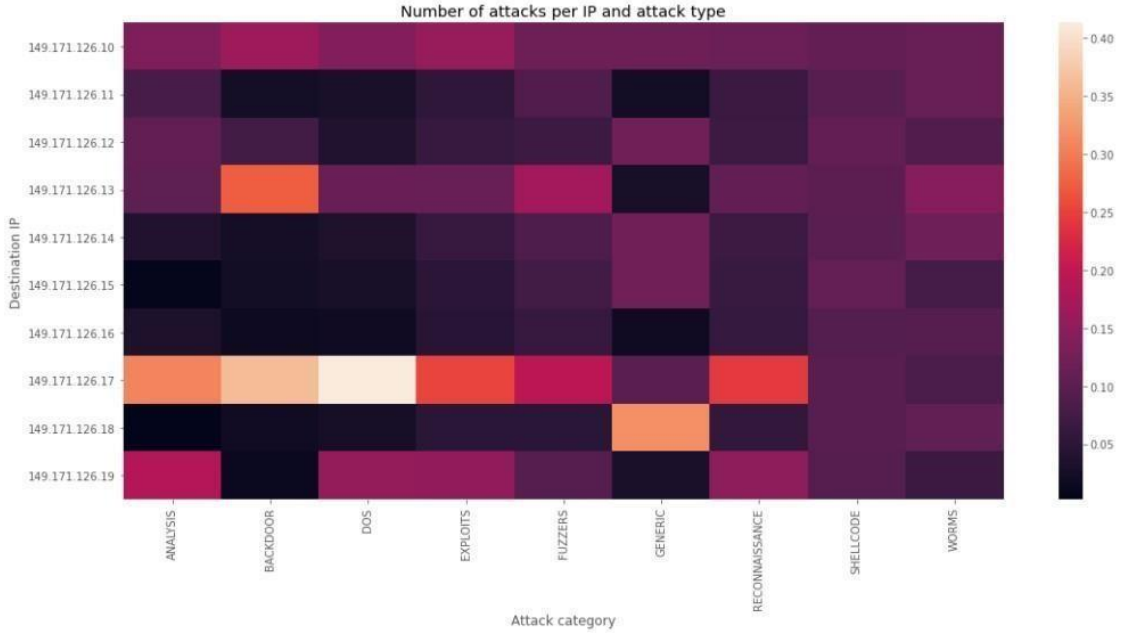

Number of attacks per IP and attack type
Destination IP
149.171.126.10
149.171.126.11
149.171.126.12
149.171.126.13
149.171.126.14
149.171.126.15
149.171.126.16
149.171.126.17
149.171.126.18
149.171.126.19
ANALYSIS
BACKDOOR
DOS
EXPLOITS
FUZZERS
GENERIC
RECONNAISSANCE
SHELLCODE
WORMS
Attack category
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05

Attack category

In [73]:
```
df_p2 = pd.pivot_table(df_pivot, values='Attack Name', index=['hour'], columns=['Destination IP'], aggfunc='count')
heatmap_graph(df = df_p2/df_p2.sum(), xlabel = 'Destination IP', ylabel = 'Hour', title = 'Percentage of attacks per IP and hour'
```



In [75]:
```
df_p3 = pd.pivot_table(df_pivot, values='Attack Name', index=['Destination IP'], columns=['Attack category'], aggfunc='count')
heatmap_graph(df = df_p3/df_p3.sum(), xlabel = 'Attack category', ylabel = 'Destination IP', title = 'Number of attacks per IP ar
```



4. TIME SERIES ANALYSIS:

In the field for machine learning and data science, most of the real-life problems are based upon the prediction of future which is totally oblivious to us such as stock market prediction, future sales prediction and so on.Time series problem is basically the prediction of such problems using various machine learning tools.Time series problem is tackled efficiently when first it is analyzed properly (Time Series Analysis) and according to that observation suitable algorithm is used (Time Series Forecasting).We'll study both of then later in this notebook.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('C:\\Users\\Mouli\\Desktop\\Cybersecurity.csv')
```

In [3]:
```
# A glance on the data
df.head()
```

Out[3]:

	MONTH	AVG NO OF ATTACKS
0	2015-01	648
1	2015-02	646
2	2015-03	639
3	2015-04	654
4	2015-05	630

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72 entries, 0 to 71
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   MONTH              72 non-null     object
 1   AVG NO OF ATTACKS  72 non-null     int64 
dtypes: int64(1), object(1)
memory usage: 1.2+ KB
```

Average Number of Attacks

```
In [8]: df['AVG NO OF ATTACKS'].unique()

Out[8]: array([ 648,  646,  639,  654,  630,  622,  617,  613,  661,  695,  690,
                707,  817,  839,  810,  789,  760,  724,  704,  691,  745,  803,
                780,  761,  857,  907,  873,  910,  900,  880,  867,  854,  928,
               1064, 1103, 1026, 1102, 1080, 1034, 1083, 1078, 1020,  984,  952,
               1033, 1114, 1160, 1058, 1209, 1200, 1130, 1182, 1152, 1116, 1098,
               1044, 1142, 1222, 1234, 1155, 1286, 1281, 1224, 1280, 1228, 1181,
               1156, 1124, 1205, 1260, 1188, 1212], dtype=int64)

In [36]: df = df.drop(df.index[df['AVG NO OF ATTACKS'] == ' n=74'])

In [10]: df['AVG NO OF ATTACKS'].unique()

Out[10]: array([ 648,  646,  639,  654,  630,  622,  617,  613,  661,  695,  690,
                 707,  817,  839,  810,  789,  760,  724,  704,  691,  745,  803,
                 780,  761,  857,  907,  873,  910,  900,  880,  867,  854,  928,
                1064, 1103, 1026, 1102, 1080, 1034, 1083, 1078, 1020,  984,  952,
                1033, 1114, 1160, 1058, 1209, 1200, 1130, 1182, 1152, 1116, 1098,
                1044, 1142, 1222, 1234, 1155, 1286, 1281, 1224, 1280, 1228, 1181,
                1156, 1124, 1205, 1260, 1188, 1212], dtype=int64)

In [11]: df['AVG NO OF ATTACKS'] = df['AVG NO OF ATTACKS'].astype(np.int64)

In [12]: df['MONTH'] = pd.to_datetime(df['MONTH'], format = '%Y-%m')

In [13]: df.dtypes

Out[13]: MONTH               datetime64[ns]
         AVG NO OF ATTACKS            int64
         dtype: object
```

Time Series Analysis

We all know how important data analysis is for data scientists.It gives us a brief understanding of the data and a very strange but intriguing confidence about our prediction model.Well, Time series analysis is no different.But time series problems have very special orientation when it comes to analysis.But before we move into that, let me introduce you to some jargons (Just Kidding it is pure and simple english) which are frequently used in this problem domain.

Trend: - As the name suggests trend depicts the variation in the output as time increases.It is often non-linear. Sometimes we will refer to trend as “changing direction” when it might go from an increasing trend to a decreasing trend.

Level: - It basically depicts baseline value for the time series.

Seasonal: - As its name depicts it shows the repeated pattern over time. In layman terms, it shows the seasonal variation of data over time.

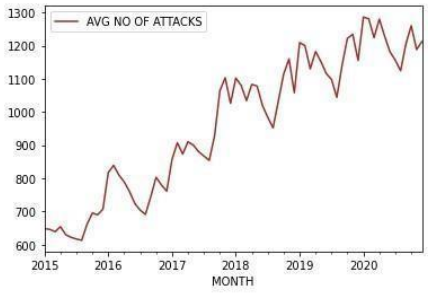
Noise: - It is basically external noises that vary the data randomly.

You can see below various graphs I plotted and what I inferred from them. which is totally oblivious to us such as stock market prediction, future sales prediction and so on.Time series problem is basically the prediction of such problems using various machine learning tools.Time series problem is tackled efficiently when first it is analyzed properly (Time Series Analysis) and according to that observation suitable algorithm is used (Time Series Forecasting).We'll study both of then later in this notebook.

Plotting monthly variation of Dataset

It gives us idea about seasonal variation of our data set

In [14]:
```
df.plot.line(x = 'MONTH', y = 'AVG NO OF ATTACKS',color='brown')
plt.show()
```



In [15]:
```
to_plot_monthly_variation = df
```

In [16]:
```
# only storing month for each index
mon = df['MONTH']
```

In [17]:
```
# decompose yyyy-mm data-type
temp= pd.DatetimeIndex(mon)
```

In [18]:
```
# assign month part of that data to ```DATE``` variable
month = pd.Series(temp.month)
```

In [19]:
```
# dropping month from to_plot_monthly_variation
to_plot_monthly_variation = to_plot_monthly_variation.drop(['MONTH'], axis = 1)
```

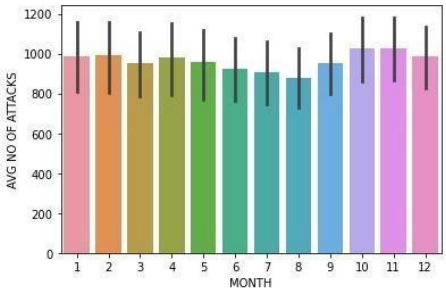
In [20]:
```
# join months so we can get month to average monthly rider mapping
to_plot_monthly_variation = to_plot_monthly_variation.join(month)
```

In [57]:
```
# A quick glance
to_plot_monthly_variation.head()
```

Out[57]:

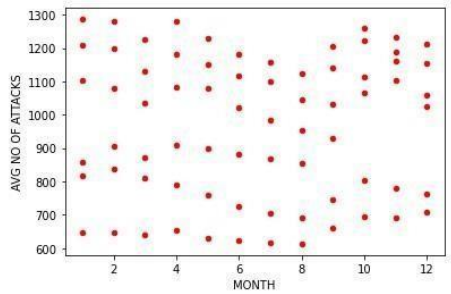
	AVG NO OF ATTACKS	MONTH
0	648	1
1	646	2
2	639	3
3	654	4
4	630	5

In [58]:
```
# Plotting bar plot for each month
sns.barplot(x = 'MONTH', y = 'AVG NO OF ATTACKS', data = to_plot_monthly_variation)
plt.show()
```



Well, this looks tough to decode. One can infer that data is too sparse for this graph to represent any pattern. Hence it cannot represent monthly variation effectively. In such a scenario we can use our traditional scatter plot to understand pattern in dataset

```
In [59]: to_plot_monthly_variation.plot.scatter(x = 'MONTH', y = 'AVG NO OF ATTACKS',color='red')
plt.show()
```




```
In [60]: attack = df[['AVG NO OF ATTACKS']]
```

We can see here the yearly variation of data in this plot. To understand this curve more effectively first look at every row from bottom to top and see each year's variation. To understand yearly variation, take a look at each column representing a month.

Each dot represents the years(2015-2020) month wise number of Attacks

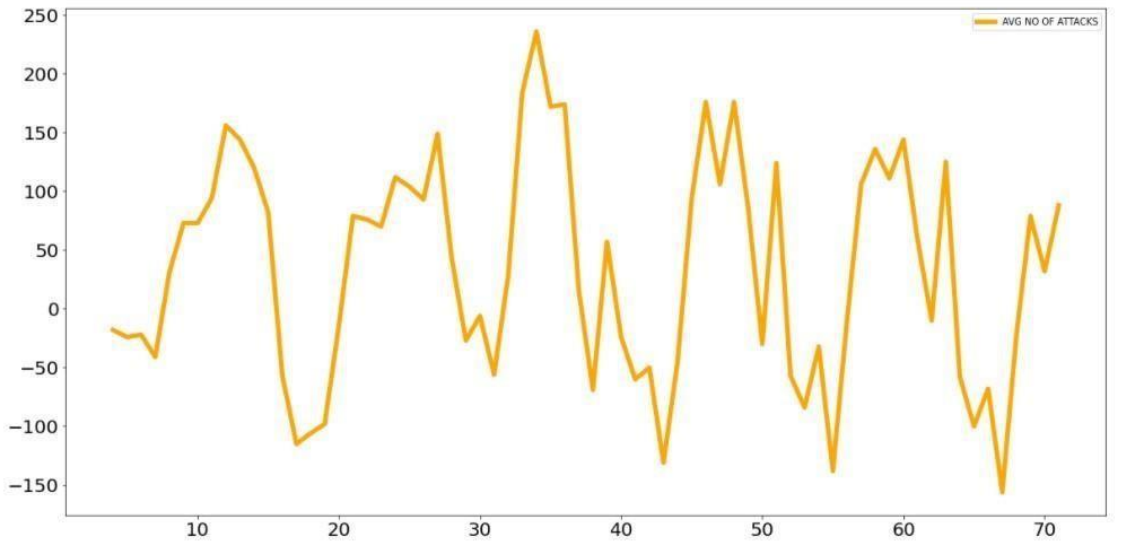
Trend Analysis:

For trend analysis, we use smoothing techniques. In statistics smoothing a data set means to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal. We implement smoothing by taking moving averages. Exponential moving average is frequently used to compute smoothed function. Here I used the rolling method which is inbuilt in pandas and frequently used for smoothing.

Two most famous season ability analysis algorithms are:-

Using 1st discrete difference of object

```
In [62]: attack.diff(periods=4).plot(figsize=(20,10), linewidth=5, fontsize=20,color='orange')
         plt.show()
```

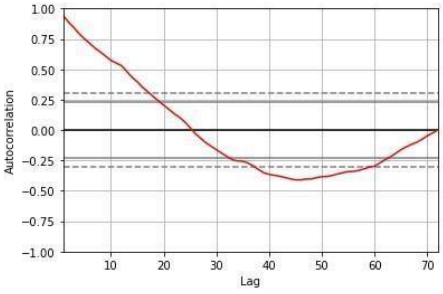


The above figure represents difference between average attack of a month and 4 months before that month i.e.

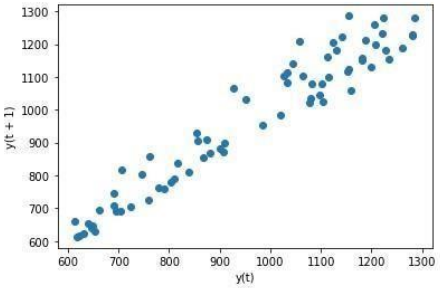
Periodicity and Autocorrelation

Auto correlation is the most famous way to understand seasonal variation till now. We can calculate the correlation for time series observations with observations with previous time steps, called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation.In this plot vertical axis is represented by the following equations:-

In [63]:
```
pd.plotting.autocorrelation_plot(df['AVG NO OF ATTACKS'],color='red')
plt.show()
```



In [64]:
```
pd.plotting.lag_plot(df['AVG NO OF ATTACKS'])
plt.show()
```



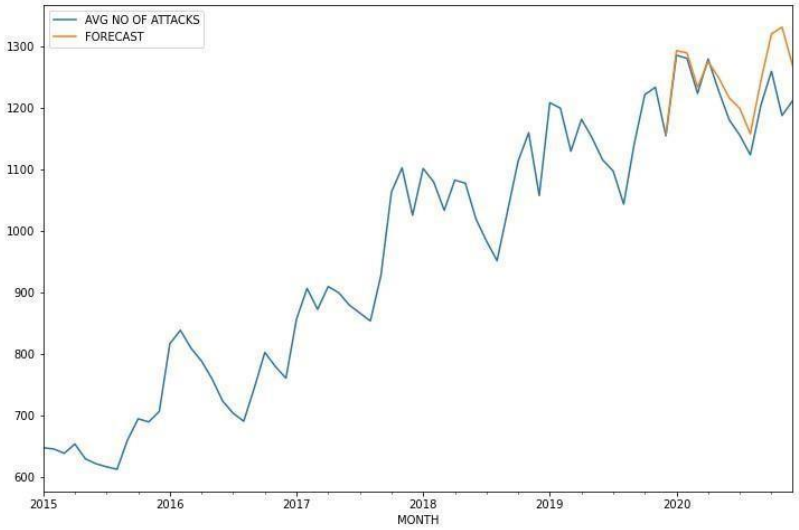
In [65]:
```
df = df.set_index('MONTH')
```

The above curve represents the relation between current time step and its previous time step

In [66]:
```
# Applying Seasonal ARIMA model to forcast the data
mod = sm.tsa.SARIMAX(df['AVG NO OF ATTACKS'], trend='n', order=(0,1,0), seasonal_order=(1,1,1,12))
results = mod.fit()
print(results.summary())
```

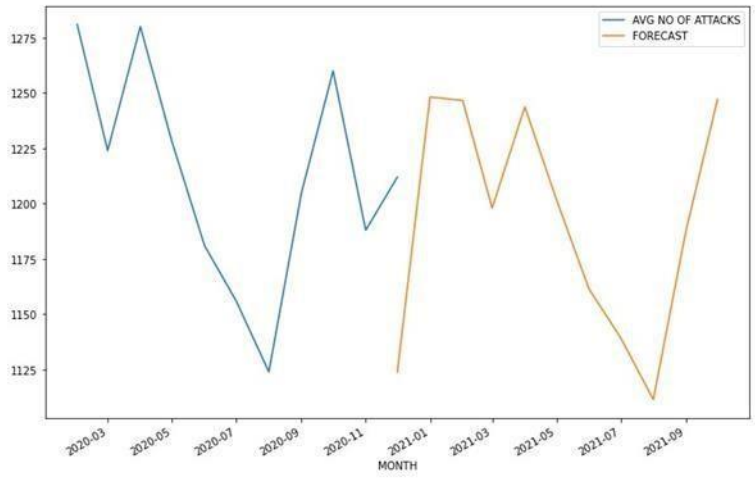
```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                   AVG NO OF ATTACKS   No. Observations:                   72
Model:             SARIMAX(0, 1, 0)x(1, 1, [1], 12)   Log Likelihood                -288.614
Date:                              Mon, 31 May 2021   AIC                            583.228
Time:                                      20:58:08   BIC                            589.461
Sample:                                  01-01-2015   HQIC                           585.661
                                       - 12-01-2020
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.S.L12       0.5610      0.490      1.144      0.253      -0.400       1.522
ma.S.L12      -0.9982     47.546     -0.021      0.983     -94.187      92.191
sigma2       871.9204   4.12e+04      0.021      0.983   -7.99e+04    8.17e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.33   Jarque-Bera (JB):                 5.48
Prob(Q):                              0.57   Prob(JB):                         0.06
Heteroskedasticity (H):               1.99   Skew:                             0.47
Prob(H) (two-sided):                  0.13   Kurtosis:                         4.16
===================================================================================
```

```
In [67]: df['FORECAST'] = results.predict(start = 59, end= 77, dynamic= True)
         df[['AVG NO OF ATTACKS', 'FORECAST']].plot(figsize=(12, 8))
         plt.show()
```




```
In [68]: def forcasting_future_months(df, no_of_months):
             df_perdict = df.reset_index()
             mon = df_perdict['MONTH']
             mon = mon + pd.DateOffset(months = no_of_months)
             future_dates = mon[-no_of_months -1:]
             df_perdict = df_perdict.set_index('MONTH')
             future = pd.DataFrame(index=future_dates, columns= df_perdict.columns)
             df_perdict = pd.concat([df_perdict, future])
             df_perdict['FORECAST'] = results.predict(start = 71, end = 82, dynamic= True)
             df_perdict[['AVG NO OF ATTACKS', 'FORECAST']].iloc[-no_of_months - 12:].plot(figsize=(12, 8))
             plt.show()
             return df_perdict[-no_of_months:]
```

```
In [69]: predicted = forcasting_future_months(df,10)
```



This is the final output in which blue coloured line indicates the previous trend of occurrence of attacks and the orange coloured line indicates the forecasted trend of average number of attacks for the next 12 months.

```
In [70]: df.tail()
Out[70]:
```

MONTH	AVG NO OF ATTACKS	FORECAST
2020-08-01	1124	1158.235265
2020-09-01	1205	1244.672526
2020-10-01	1260	1320.612436
2020-11-01	1188	1331.873071
2020-12-01	1212	1269.499220

This is to check our forecast is whether correct or not, in this we've done forecasting for existing data and we got similar values as we can observe in the screen shot.

Result and discussion:

From Module 1 we have encountered that there is unwanted data in the data set so, we have dropped the data, and for missing values we have assigned the as "Not Registered" in order to Predict with good Accuracy

In the second Module We will be using two methods for correlation calculation: •Pearson's correlation: evaluates the linear relationships between two variables. If the value is close to 0, there is a week or nonexistent linear relationship between the variables. •Spearman's correlation: evaluates the monotonic relationships between two variables. If the value is close to 0, there is a weak or nonexistent monotonic relationship between the variables

In module 3 we have plotted various types of Informative Graphs as follows Bar Graph for Number of Attacks Per attack Category, Pie Cart for top six attacks, and Heat maps for finding the pattern of attacks with respect to time and then we observe that the attacker chooses even hours more when compared to

odd hours this is the pattern which is repeatedly occurring. This is an Hourly based plots in which we can easily find the pattern of an attack with respect to hour
DOS, Exploits, Fuzzers, Generic and Reconnaissance are performed more when compared to Shell code, Worms and Back Door and then we have plotted Heat map for percentage of attacks per hour per attack type

Finally, we have found the pattern that attacker chooses even hours comparatively more than odd hours to perform an attack

In the module 4 we forecasted the average number of attacks for succeeding year by using a time series model called ARIMA. In ARIMA we have plotted the trend line of our Data set and then we performed two sesonality and auto co relation and then we smoothed the trend line for further for prediction analysis

For time series analysis we used ARIMA model for forecasting the average no of attacks for the next 12 months and we got the result for next 12 months from 1-2021 to 12-2021 with accuracy like we've checked our code by predicting the existing months then we got similar values after forecasting.

Conclusion:

We can say that the attacker is following a pattern most number of attacks are performed in the odd hours of the day, we predicted the next year attacks which is useful for Security

Professionals, and popular IT Companies which have Confidential Data to be Protected, it is also useful for the Government we can say that which is more common type of attack is Exploit, most targeted IP is 149.171.126.17, and 175.45.176.1 is the Common Source IP for most of the attacks performed, we can predict the Source IP an Destination IP of the Future and number of attacks type of attack for the succeeding years

Source Code G-Drive Link:

https://drive.google.com/drive/folders/1Cxm1kZn2wky1yM7Ays6UU1Xah9SGAod6?usp=sharing

References:

https://www.kaggle.com/iamranjann/cybersecurityattacks