Random Forest

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
v = CountVectorizer(ngram_range=(1,1))
x = v.fit_transform(data['Processed Text'])
model=LogisticRegression()
#forest = RandomForestClassifier(n_estimators = 300)

model.fit(x, data['Vulnerability Type'])
```

```
#we are not getting the absolute value
feature_importance=pd.DataFrame({'feature':v.get_feature_names(),'feature_importance':model.coef_[0]})
feature_importance.sort_values('feature_importance',ascending=False).head(20)
```

```
from sklearn.feature_extraction.text import CountVectorizer
matrix = CountVectorizer(max_features=100)
X = matrix.fit_transform(data['Processed Text']).toarray()
y=df.iloc[:, 1]
```

```
X.shape,y.shape
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data['Processed Text'], data['Vulnerability Type'], train_size=0.8, stratify=data['Vu
```

```
y_train.value_counts(normalize=True)
```

```
Code Execute                 0.244811
Denial of Service            0.159795
Cross Site Scripting (XSS)   0.130793
Overflow                     0.128519
Gain Information             0.071936
SQL Injection                0.058857
Bypass                       0.048905
Memory Corruption            0.038385
Directory Traversal          0.031845
gain priviledge              0.031845
CSRF                         0.021609
Security Vulnerabilities     0.019903
File Inclusion               0.011942
Http Response Splitting      0.000853
Name: Vulnerability Type, dtype: float64
```

```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
v1= CountVectorizer()
x = v.fit_transform(X_train)
```

```
#forest = RandomForestClassifier()

#forest.fit(x, y_train)
```

```
#we are not getting the absolute value
#feature_importance=pd.DataFrame({'feature':v.get_feature_names(),'feature_importance':model.coef_[0]})
#feature_importance.sort_values('feature_importance',ascending=False).head(20)
```

```
"""import json

with open('dict.json') as json_file:
    data = json.load(json_file)"""
```

```
'import json\n\nwith open('dict.json') as json_file:\n    data = json.load(json_fil
e)'
```

import dataset

```
import pandas as pd
```

```
import nltk
import re
import string
string.punctuation
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords

def tokenization(text):
    tokens = text.split(' ')
    return tokens

def remove_punctuation(text):
    punctuationfree=[]
    for t in text:
        punctuationfree.append(''.join(i for i in t if i not in string.punctuation))
    return punctuationfree


nltk.download('stopwords')
stopwords=stopwords.words('english')

def remove_stopwords(text):
    output= [i for i in text if i not in stopwords]
    return output


porter_stemmer = PorterStemmer()

def stemming(text):
    stem_text = [porter_stemmer.stem(word) for word in text]
    return stem_text
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

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

```
import pandas as pd

df = pd.read_excel(r'Data.xlsx')
```

```
df['Processed Text']=df['Summary Text'].str.replace('\d+', '')
df['Processed Text']= df['Processed Text'].apply(lambda x: x.lower())
df['Processed Text']=df['Processed Text'].apply(lambda x: tokenization(x))
df['Processed Text']=df['Processed Text'].apply(lambda x: remove_punctuation(x))
df['Processed Text']=df['Processed Text'].apply(lambda x: remove_stopwords(x))

df['Processed Text']=df['Processed Text'].apply(lambda x: ' '.join(x))
```

```
df.head(5)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
0	CVE-2022-43766	Denial of Service	Apache IoTDB version 0.12.2 to 0.12.6, 0.13.0 ...	apache iotdb version vulnerable denial ser...
1	CVE-2022-43365	Denial of Service	IP-COM EW9 V15.11.0.14(9732) was discovered to...	ipcom ew v discovered contain buffer overflow ...
2	CVE-2022-43035	Denial of Service	An issue was discovered in Bento4 v1.6.0-639	issue discovered bento v heapbufferoverflow ap

```
df.shape
```

```
(175850, 4)
```

```
df.sample(10)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
80041	CVE-2017-9120	Overflow	PHP 7.x through 7.1.5 allows remote attackers ...	php x allows remote attackers cause denial se...
26860	CVE-2002-1850	Denial of Service	mod_cgi in Apache 2.0.39 and 2.0.40 allows loc...	modcgi apache allows local users possibly re...
126374	CVE-	Cross Site Scripting	Cross-site scripting (XSS)	crosssite scripting xss

```
data=df
```

memory corruption

WORDCLOUD

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
```

```
print(set(data['Vulnerability Type']))
```

```
{'SQL Injection', 'gain priviledge', 'Http Response Splitting', 'Code Execute', 'Overflow', 'Memory Corruption', 'Gain Information'
```

Code Execution

```
CoE=' '.join(list(data['Processed Text'][data['Vulnerability Type'].str.contains('Code Execute')]))
```

```
WC_CoE = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(CoE)
```

```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(WC_CoE)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Denial of Service

```
DoS=' '.join(list(data['Processed Text'][data['Vulnerability Type'].str.contains('Denial of Service')]))
```

```
WC_DoS = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(DoS)
```

```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(WC_DoS)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

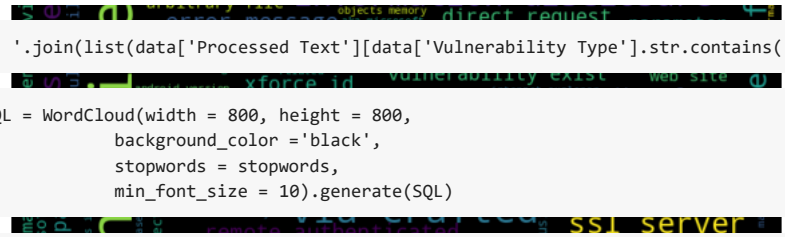
Gain Information

```
GI=' '.join(list(data['Processed Text'][data['Vulnerability Type'].str.contains('Gain Information')]))
```

```
WC_GI = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(GI)
```

```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(WC_GI)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

SQL Injection



```
SQL=' '.join(list(data['Processed Text'][data['Vulnerability Type'].str.contains('SQL Injection')]))
```



```
WC_SQL = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(SQL)
```



```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(WC_SQL)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Feature Importances

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
v = CountVectorizer(ngram_range=(1,1))
x = v.fit_transform(data['Processed Text'])
```

```
model=LogisticRegression()
#forest = RandomForestClassifier(n_estimators = 300)

model.fit(x, data['Vulnerability Type'])
```

```
▾ LogisticRegression
LogisticRegression()
```

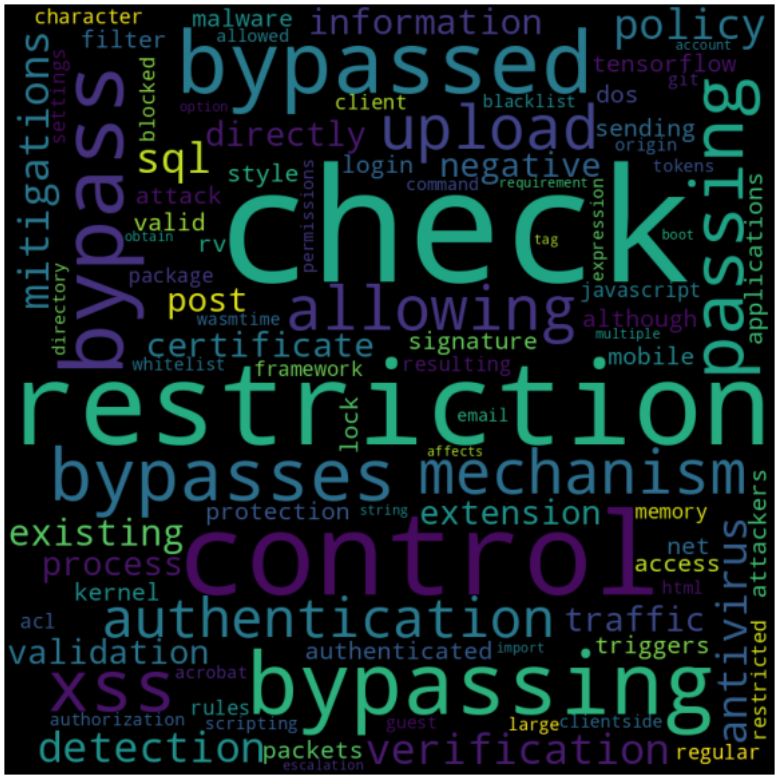
```
#we are not getting the absolute value
feature_importance=pd.DataFrame({'feature':v.get_feature_names_out(),'feature_importance':model.coef_[0]})
feature_importance.sort_values('feature_importance',ascending=False).head(20)
```

	feature	feature_importance
15676	bypass	5.060708
15678	bypassed	3.580581
15681	bypassing	3.558767
15679	bypasses	2.890906
88753	passing	1.187195
18854	check	0.871880
137126	xss	0.640448
19024	checks	0.605755
5996	allowing	0.601696
125872	upload	0.546278
73730	mechanism	0.536310
10446	authentication	0.509995
128665	verification	0.493612
25808	controls	0.491435
102016	restriction	0.479124
32067	detection	0.448312
75604	mitigations	0.433879
93135	policy	0.426112
113843	sql	0.419165
102017	restrictions	0.419123

```
points=list(feature_importance.sort_values('feature_importance',ascending=False)[:100].feature)
```

```
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in points:

    # typecaste each val to string
    val = str(val)
    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)
```

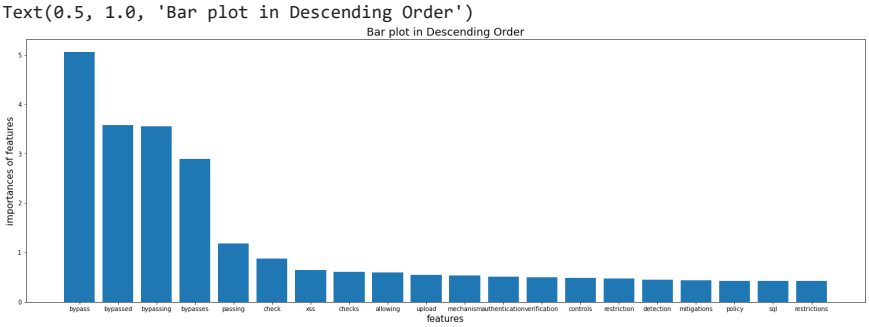
```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
df_sorted= feature_importance.sort_values('feature_importance',ascending=False)[:20]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('feature', 'feature_importance',data=df_sorted)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```



Bi_grams

```
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
```

```
ngrams = c_vec.fit_transform(data['Processed Text'])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

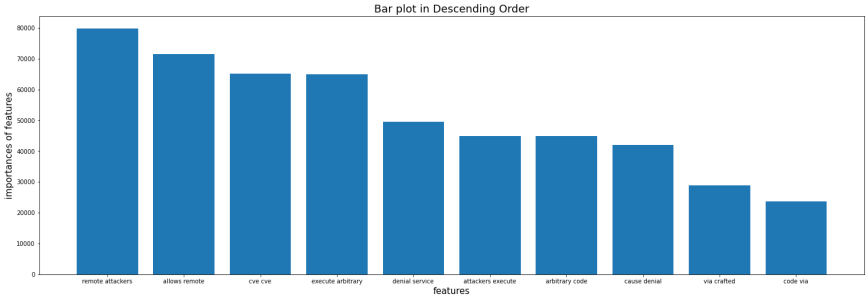
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                       term   rank
482694     remote attackers  79852
26845         allows remote  71456
135553              cve cve  65048
194041    execute arbitrary  64987
147509       denial service  49546
46520     attackers execute  44967
37290        arbitrary code  44863
77498          cause denial  42027
626262          via crafted  28895
97134              code via  23748
349073    memory corruption  22548
66597       buffer overflow  22347
46426       attackers cause  21006
547878        sql injection  18073
132065  crosssite scripting  17829
25598          allow remote  16548
101582         commands via  15550
510390        scripting xss  15324
96066        code execution  13935
37750         arbitrary sql  13838
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```



```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['remote_attackers', 'allows_remote', 'cve_cve', 'execute_arbitrary', 'denial_service', 'attackers_execute', 'arbitrary_code', 'cau
```

```
print(bi_string[:20])
```

```
remote_attackers all
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```

code execute

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
```

```
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Code Execute')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

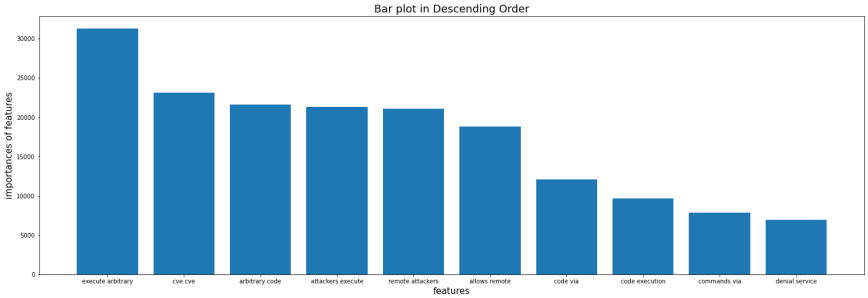
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                          term   rank
70206         execute arbitrary  31309
49700                   cve cve  23111
13174            arbitrary code  21596
16936         attackers execute  21267
178874         remote attackers  21086
9374              allows remote  18786
35041                  code via  12089
34280            code execution   9643
37166              commands via   7865
54026            denial service   6907
24019           buffer overflow   6494
230610             via crafted   6342
27536              cause denial   6330
201722            sql injection   6325
13362             arbitrary sql   6226
201702             sql commands   6169
127535        memory corruption   5729
104042  injection vulnerability   5499
178889              remote code   5410
8948               allow remote   4633
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```



```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['execute_arbitrary', 'cve_cve', 'arbitrary_code', 'attackers_execute', 'remote_attackers', 'allows_remote', 'code_via', 'code_exec
```

```
print(bi_string[:20])
```

```
execute_arbitrary cv
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```

Denial of Service

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Denial of Service')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                            term   rank
37525            denial service  28628
17624              cause denial  22672
129129         remote attackers  14371
5037              allows remote  13725
9989            attackers cause  13566
34264                   cve cve  11275
168063              via crafted   7992
49804         execute arbitrary   6556
7426             arbitrary code   6460
32853                 crash via   5322
94043         memory corruption   4533
140334           service memory   4147
6857          application crash   3966
10015         attackers execute   3552
23018                code cause   3430
14983           buffer overflow   3051
163477              users cause   2635
23343                  code via   2264
39683   different vulnerability   2210
139751            service crash   2173
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```

```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

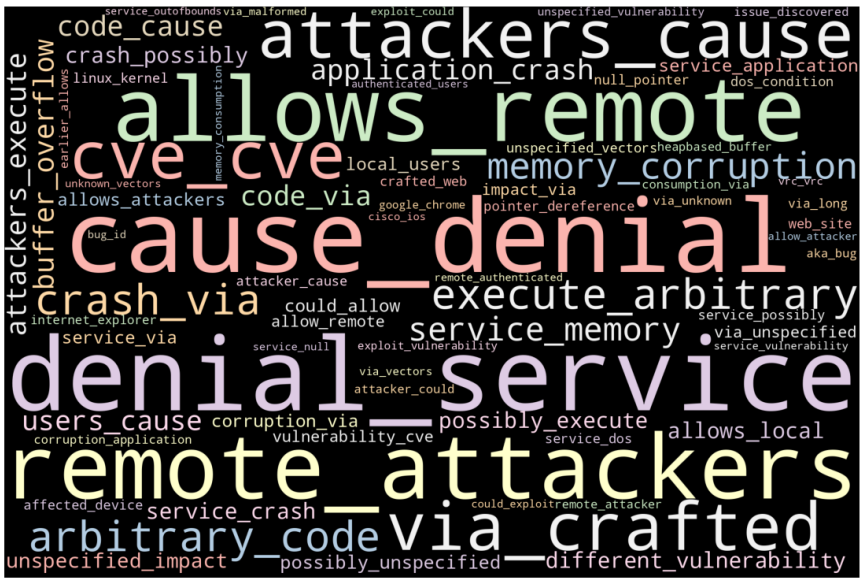
```
['denial_service', 'cause_denial', 'remote_attackers', 'allows_remote', 'attackers_cause', 'cve_cve', 'via_crafted', 'execute_arbit
```

```
print(bi_string[:20])
```

```
denial_service cause
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```



SQL Injection

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('SQL Injection')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

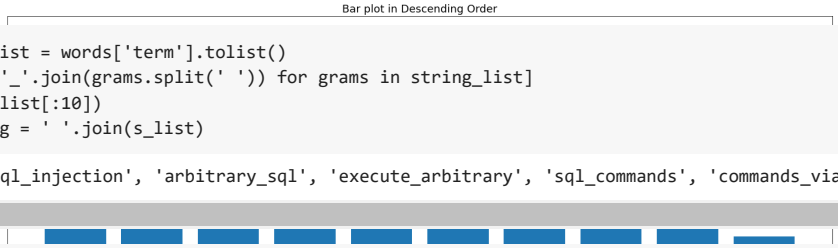
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                            term   rank
46331              sql injection  10167
3789               arbitrary sql   6414
15856          execute arbitrary   6412
46296               sql commands   6239
9017                commands via   6061
23726    injection vulnerability   5933
41459           remote attackers   5880
4559           attackers execute   5659
2920               allows remote   4984
2799                allow remote   1566
23724  injection vulnerabilities   1556
29883               multiple sql   1507
21312               id parameter   1362
53341                     via id   1200
23719              injection via    843
14068             earlier allows    706
41460       remote authenticated    652
58276     vulnerability indexphp    599
54879            via unspecified    582
49965        unspecified vectors    568
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['sql_injection', 'arbitrary_sql', 'execute_arbitrary', 'sql_commands', 'commands_via', 'injection_vulnerability', 'remote_attacker
```

```
print(bi_string[:20])
```

```
sql_injection arbitr
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```

gain priviledge

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('gain priviledge')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

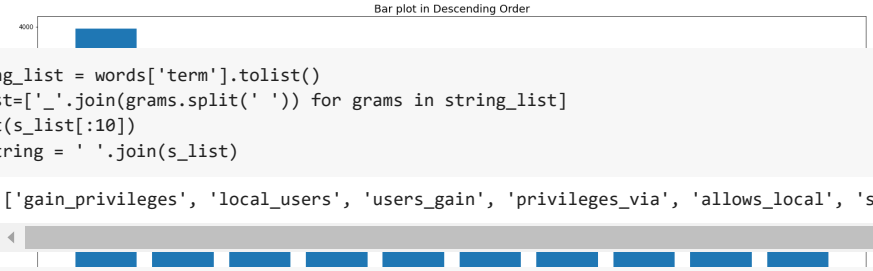
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                        term  rank
17774      gain privileges  3973
24551          local users  2968
45898           users gain  2814
34065       privileges via  2788
2241          allows local  2628
41091           sp windows  1160
47353          via crafted   997
4011        attackers gain   784
49681       windows server   778
2261         allows remote   733
9723   crafted application   677
41075                sp sp   623
43979         trojan horse   619
36355     remote attackers   577
47654           via trojan   553
38470          search path   536
39201            server sp   521
2130           allow local   511
37421      root privileges   510
11437       denial service   485
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['gain_privileges', 'local_users', 'users_gain', 'privileges_via', 'allows_local', 'sp_windows', 'via_crafted', 'attackers_gain', '
```

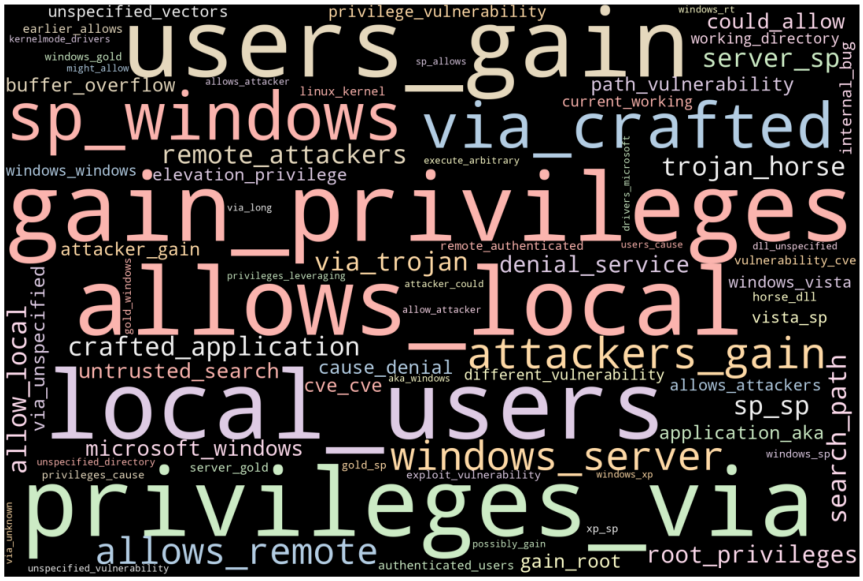


```
print(bi_string[:20])
```

```
gain_privileges loca
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```




```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Http Response Splitting

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Http Response Splitting')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

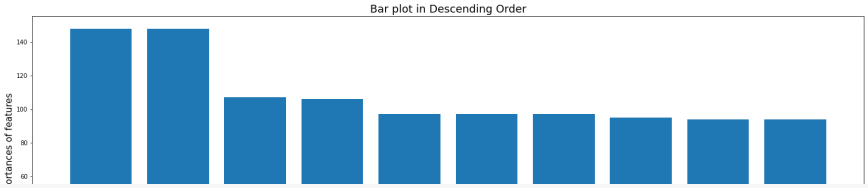
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                          term  rank
711             http response   148
1261       response splitting   148
1450        splitting attacks   107
706              http headers   106
324              conduct http    97
100             arbitrary http    97
778           inject arbitrary    97
393             crlf injection    95
68               allows remote    94
1227         remote attackers    94
163                attacks via    93
673            headers conduct    90
793   injection vulnerability    86
141          attackers inject    85
1711           via unspecified    33
1560      unspecified vectors    30
396       crosssite scripting    23
1682               via crafted    22
1824          vulnerable http    16
212            cache poisoning    14
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```


plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['http_response', 'response_splitting', 'splitting_attacks', 'http_headers', 'conduct_http', 'arbitrary_http', 'inject_arbitrary',
```

```
print(bi_string[:20])
```

```
http_response respon
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```

```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Memory Corruption

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Memory Corruption')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

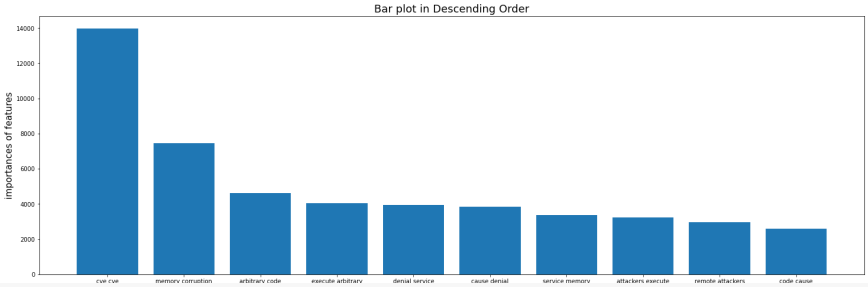
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                           term   rank
5512                    cve cve  13988
13907         memory corruption   7441
1274             arbitrary code   4617
7692          execute arbitrary   4046
5989             denial service   3939
2765               cause denial   3844
20404            service memory   3366
1668          attackers execute   3245
18977          remote attackers   2957
3493                 code cause   2594
870               allows remote   2558
24092               via crafted   2462
4721             corruption via   2076
4725   corruption vulnerability   1994
6280    different vulnerability   1860
24561         vulnerability cve   1721
11422         internet explorer   1445
5224                crafted web   1339
25003                  web site   1216
1166          application crash   1149
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['cve_cve', 'memory_corruption', 'arbitrary_code', 'execute_arbitrary', 'denial_service', 'cause_denial', 'service_memory', 'attack
```

```
print(bi_string[:20])
```

```
cve_cve memory_corru
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Gain Information


```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Gain Information')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```


```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

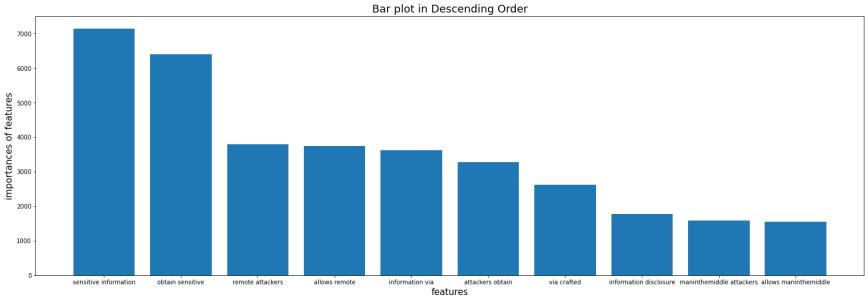
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                          term  rank
82070     sensitive information  7152
62433          obtain sensitive  6401
76052          remote attackers  3799
5370              allows remote  3750
46294           information via  3618
8804           attackers obtain  3274
98993               via crafted  2619
45630    information disclosure  1778
55789  maninthemiddle attackers  1585
5345      allows maninthemiddle  1558
8838             attackers spoof  1522
83081             servers obtain  1521
22168       crafted certificate  1498
86760             spoof servers  1490
83041            servers allows  1489
98158       verify certificates  1483
87058               ssl servers  1475
14402          certificates ssl  1471
6825        application android  1368
5927              android verify  1350
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

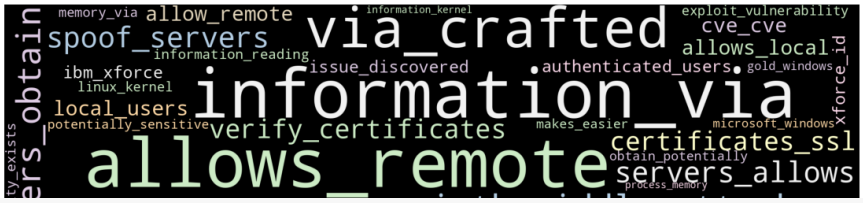
```
['sensitive_information', 'obtain_sensitive', 'remote_attackers', 'allows_remote', 'information_via', 'attackers_obtain', 'via_craf
```

```
print(bi_string[:20])
```

```
sensitive_informatio
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

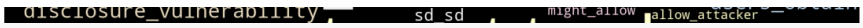
```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```




```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Cross Site Scripting (XSS)




```
data['Processed Text'][data['Vulnerability Type'].str.contains('Cross Site Scripting (XSS)')]
```

```
Series([], Name: Processed Text, dtype: object)
```



```
data['Processed Text'][data['Vulnerability Type']=='Cross Site Scripting (XSS)']
```

```
Series([], Name: Processed Text, dtype: object)
```

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type']=='Cross Site Scripting (XSS)'])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

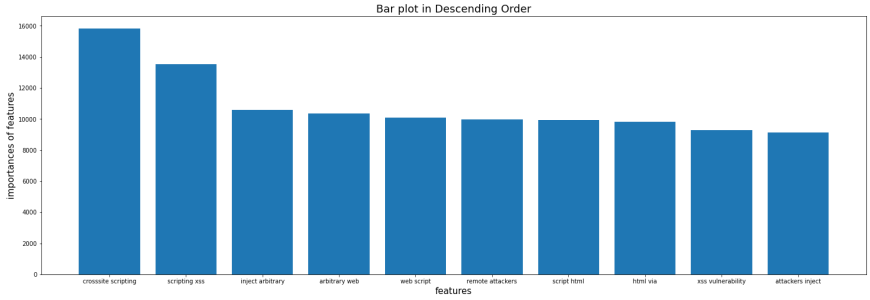
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                          term   rank
25754       crosssite scripting  15841
99411             scripting xss  13546
54890          inject arbitrary  10576
8595              arbitrary web  10339
135186               web script  10082
93809          remote attackers   9985
98988               script html   9920
50860                  html via   9805
140230        xss vulnerability   9268
10307          attackers inject   9125
6343              allows remote   8184
6039               allow remote   3157
140228      xss vulnerabilities   3058
70449        multiple crosssite   2959
125587          via unspecified   2079
115953      unspecified vectors   1875
99402   scripting vulnerability   1744
140209                  xss via   1571
93811      remote authenticated   1355
25708                cross site   1314
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['crosssite_scripting', 'scripting_xss', 'inject_arbitrary', 'arbitrary_web', 'web_script', 'remote_attackers', 'script_html', 'htm
```

```
print(bi_string[:20])
```

```
crosssite_scripting
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

CSRF

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('CSRF')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

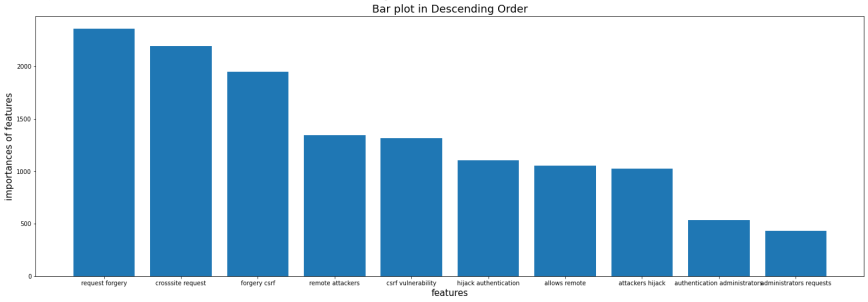
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                              term  rank
22351               request forgery  2361
7197               crosssite request  2196
11701                   forgery csrf  1948
22042              remote attackers  1347
7581              csrf vulnerability  1318
12955          hijack authentication  1106
2057                   allows remote  1052
3263                attackers hijack  1023
3535   authentication administrators   534
1321         administrators requests   434
6816                     could allow   402
1923                     allow remote   402
7580            csrf vulnerabilities   377
17505             multiple crosssite   361
31979               wordpress plugin   344
20203               plugin wordpress   325
2752                 arbitrary users   286
28343                 users requests   261
3538        authentication arbitrary   261
7313                     csrf attack   248
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['request_forgery', 'crosssite_request', 'forgery_csrf', 'remote_attackers', 'csrf_vulnerability', 'hijack_authentication', 'allows
```

```
print(bi_string[:20])
```

```
request_forgery cros
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Security Vulnerabilities

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Security Vulnerabilities')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

```
print ("\n\nWords : \n", words.head(20))
```

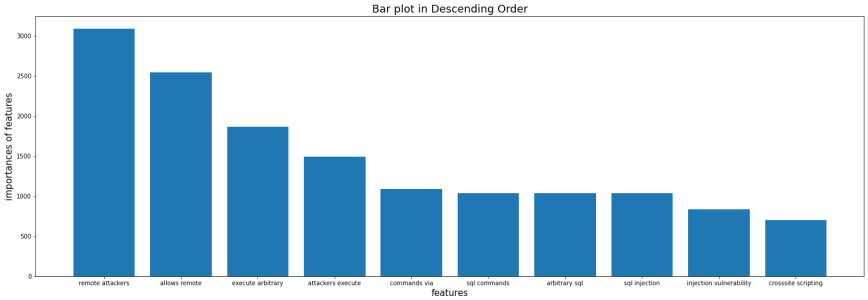
```
Words :
                          term  rank
17953         remote attackers  3094
1311             allows remote  2546
7207          execute arbitrary  1865
1896          attackers execute  1496
3994               commands via  1092
20223              sql commands  1040
1627             arbitrary sql  1040
20227            sql injection  1034
10271  injection vulnerability   836
5139       crosssite scripting   698
18863            scripting xss   697
1303              allow remote   693
10261         inject arbitrary   688
1635             arbitrary web   685
25892                web script   675
```

```
18840              script html   675
9400                  html via   674
1903          attackers inject   641
3721                  code via   510
1584            arbitrary code   477
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['remote_attackers', 'allows_remote', 'execute_arbitrary', 'attackers_execute', 'commands_via', 'sql_commands', 'arbitrary_sql', 's
```

```
print(bi_string[:20])
```

```
remote_attackers all
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Directory Traversal

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Directory Traversal')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

```
print ("\n\nWords : \n", words.head(20))
```

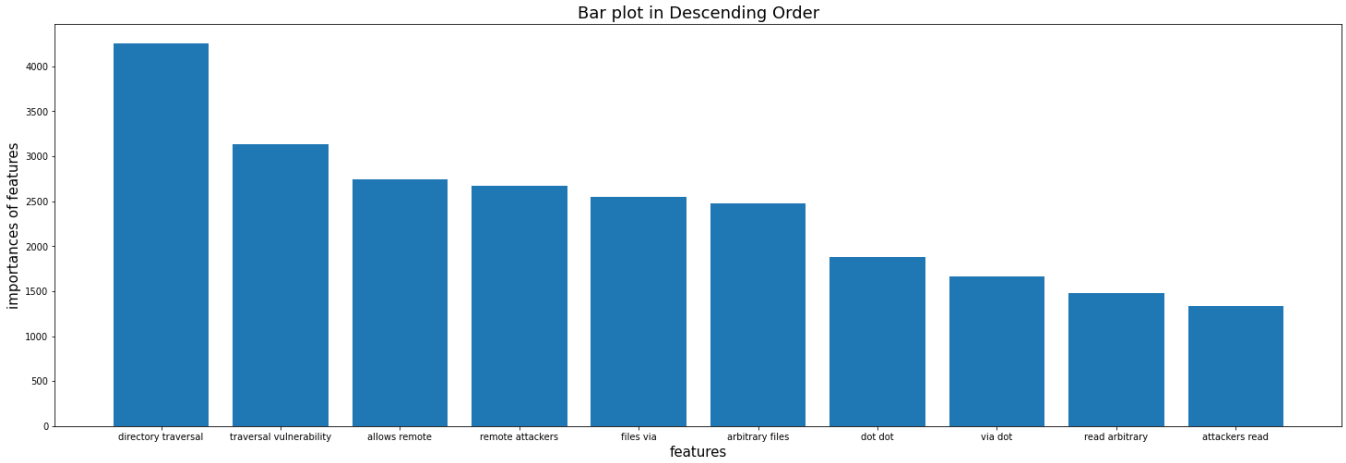
```
Words :
                          term  rank
10772       directory traversal  4258
40527   traversal vulnerability  3138
1901              allows remote  2739
32987          remote attackers  2674
15870                 files via  2544
2804            arbitrary files  2478
11560                   dot dot  1882
43732                   via dot  1666
32395            read arbitrary  1476
3686              attackers read  1335
```

```
29221             path traversal   923
13838          execute arbitrary   772
23111                local files   578
2825             arbitrary local   569
3662           attackers include   551
19549            include execute   520
1712                allow remote   463
12421             earlier allows   427
32989       remote authenticated   398
40526  traversal vulnerabilities   372
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['directory_traversal', 'traversal_vulnerability', 'allows_remote', 'remote_attackers', 'files_via', 'arbitrary_files', 'dot_dot',
```

```
print(bi_string[:20])
```

```
directory_traversal
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()
```

```
#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Bypass

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Bypass')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

```
print ("\n\nWords : \n", words.head(20))
```

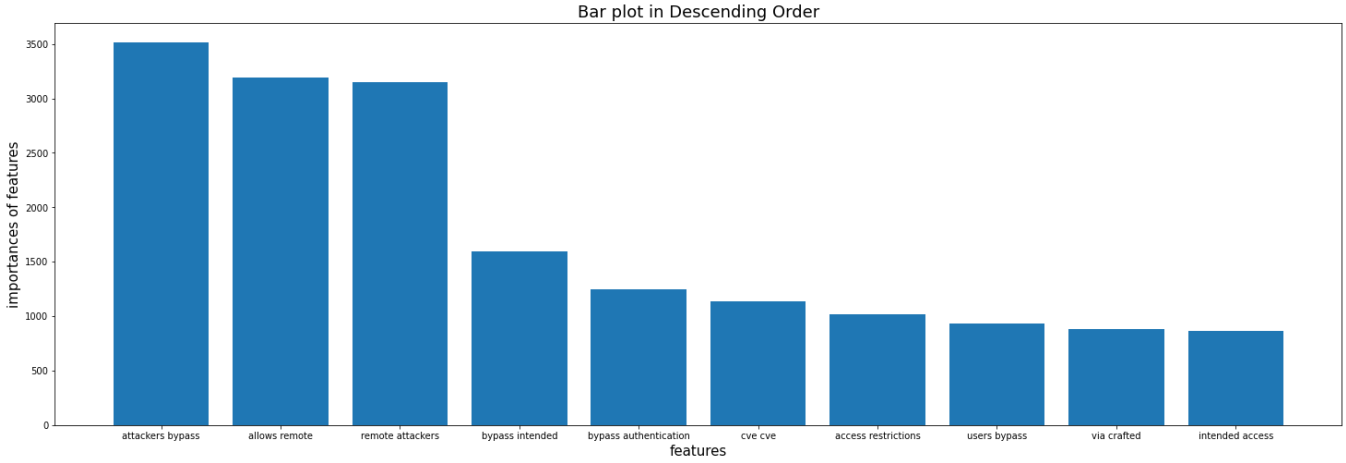
```
Words :
                        term  rank
7371        attackers bypass  3518
4000           allows remote  3196
65953       remote attackers  3150
11231        bypass intended  1598
10897  bypass authentication  1243
```

```
21524               cve cve  1134
638      access restrictions  1020
84033          users bypass   929
86867           via crafted   885
41302       intended access   865
11785   bypass vulnerability   728
19987            could allow   695
87417        via unspecified   676
7122         attacker bypass   653
68132       restrictions via   643
81926    unspecified vectors   641
65955   remote authenticated   567
46520            local users   558
63212   protection mechanism   532
7980   authentication bypass   531
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['attackers_bypass', 'allows_remote', 'remote_attackers', 'bypass_intended', 'bypass_authentication', 'cve_cve', 'access_restrictio
```

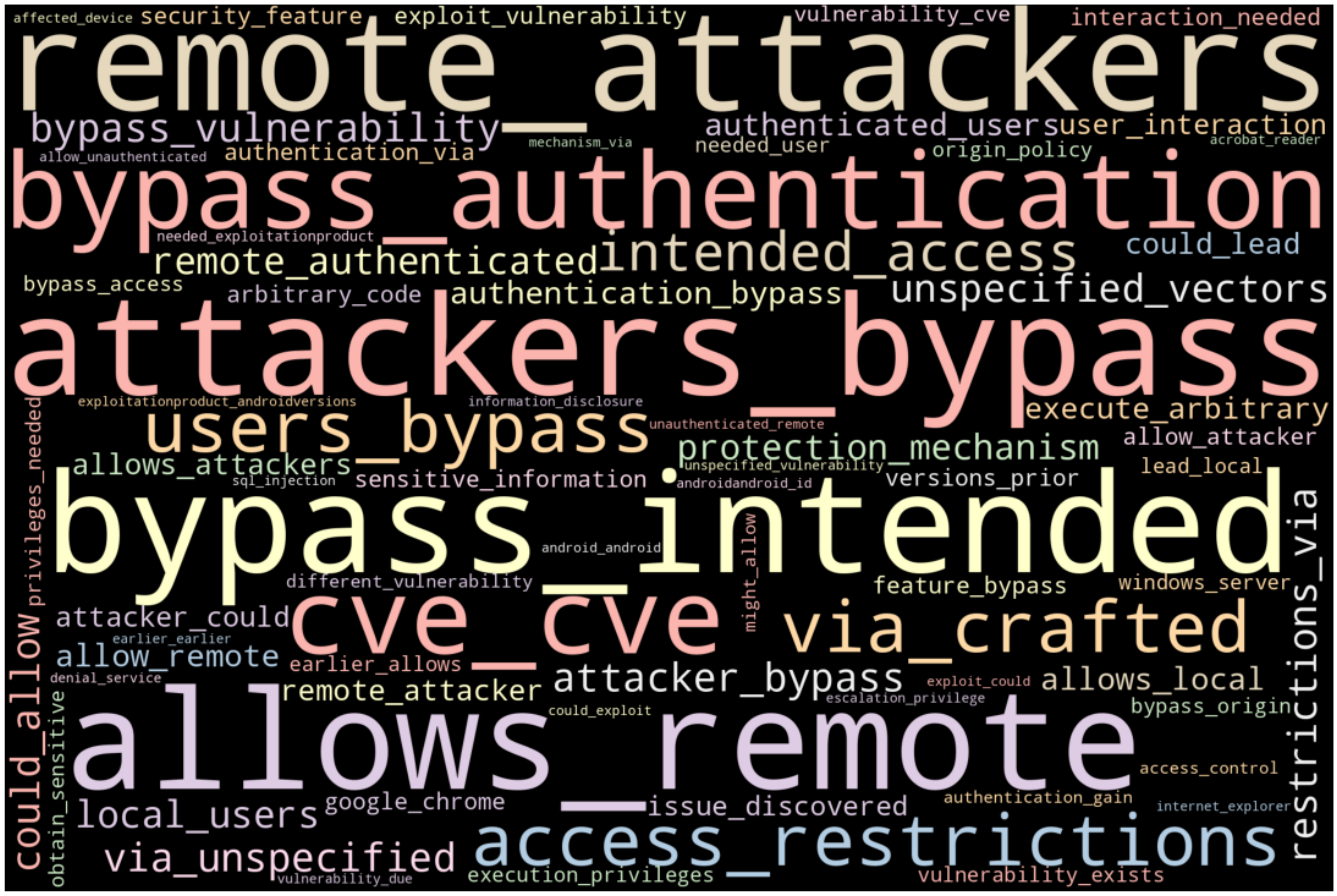
```
print(bi_string[:20])
```

```
attackers_bypass all
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
```

```
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```




```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

File Inclusion

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('File Inclusion')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

```
print ("\n\nWords : \n", words.head(20))
```

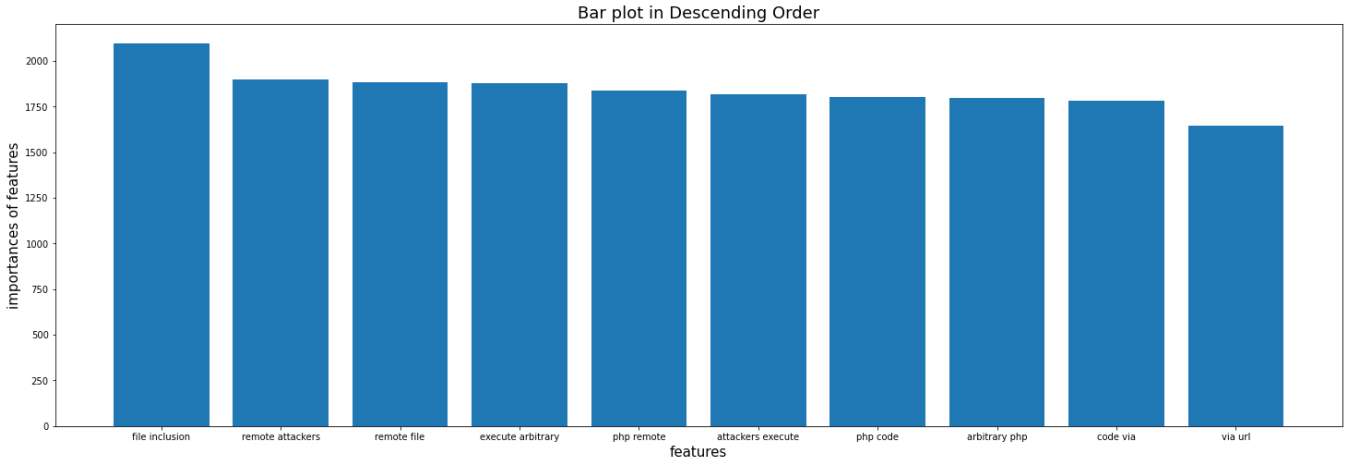
```
Words :
                          term  rank
```

```
4656              file inclusion  2098
11599           remote attackers  1900
11606                remote file  1882
4388           execute arbitrary  1878
10442                php remote  1839
1237           attackers execute  1817
10388                  php code  1800
1058               arbitrary php  1799
2379                    code via  1782
15490                   via url  1647
808                allows remote  1380
6398     inclusion vulnerability  1368
781                 allow remote   549
6397   inclusion vulnerabilities   538
8639                multiple php   534
3988              earlier allows   310
9897             parameter note   209
11532    registerglobals enabled   208
13575                third party   172
7621                  local file   161
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['file_inclusion', 'remote_attackers', 'remote_file', 'execute_arbitrary', 'php_remote', 'attackers_execute', 'php_code', 'arbitrar
```

```
print(bi_string[:20])
```

```
file_inclusion remot
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
```

```
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

Overflow

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
# matrix of ngrams
ngrams = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Overflow')])
features = (c_vec.get_feature_names_out())
#print("\n\nFeatures : \n", features)
#print("\n\nX1 : \n", ngrams.toarray())
```

```
sums = ngrams.sum(axis = 0)
data1 = []
for col, term in enumerate(features):
    data1.append( (term, sums[0, col] ))
```

```
ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
words = (ranking.sort_values('rank', ascending = False))
```

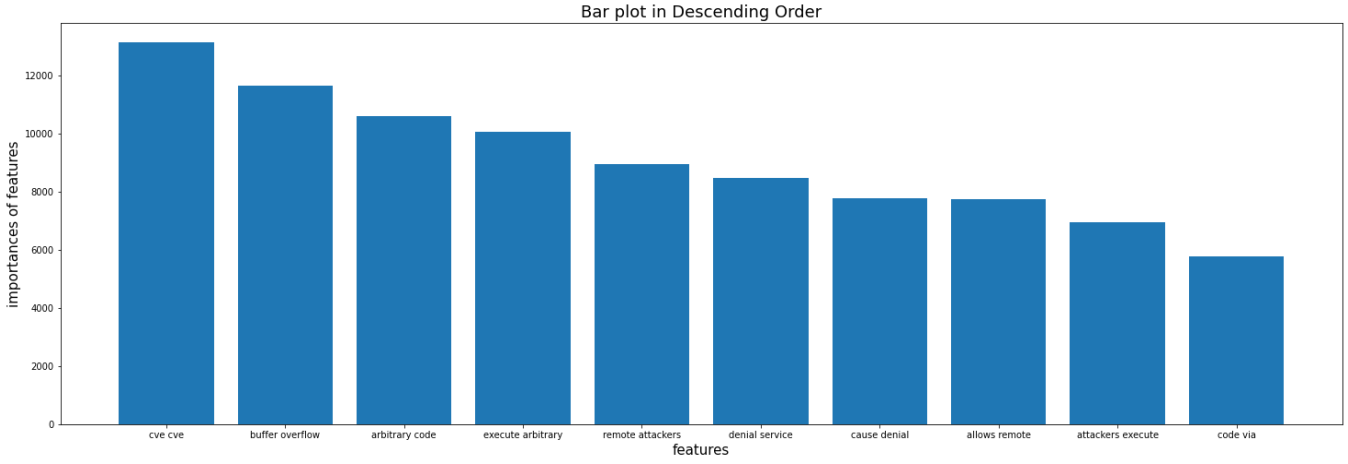
```
print ("\n\nWords : \n", words.head(20))
```

```
Words :
                     term   rank
25095             cve cve  13150
11289     buffer overflow  11651
5440       arbitrary code  10583
35771   execute arbitrary  10066
99027    remote attackers   8938
27498      denial service   8461
13451        cause denial   7783
3719        allows remote   7729
7585    attackers execute   6955
17424           code via   5769
125900       via crafted   5746
68773   memory corruption   4589
7556      attackers cause   3972
103788             sd sd   2852
111644  stackbased buffer   2698
106301     service memory   2674
17081          code cause   2582
55639    integer overflow   2571
126321           via long   2542
49586    heapbased buffer   2532
```

```
df_sor= words.sort_values('rank',ascending=False)[:10]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('term', 'rank',data=df_sor)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```




```
string_list = words['term'].tolist()
s_list=['_'.join(grams.split(' ')) for grams in string_list]
print(s_list[:10])
bi_string = ' '.join(s_list)
```

```
['cve_cve', 'buffer_overflow', 'arbitrary_code', 'execute_arbitrary', 'remote_attackers', 'denial_service', 'cause_denial', 'allows
```

```
print(bi_string[:20])
```

```
cve_cve buffer_overf
```

```
from wordcloud import WordCloud   # for the wordcloud
wordcloud = WordCloud(width = 2000, height = 1334, random_state=1, background_color='black', colormap='Pastel1', max_words = 75, collocat
```

```
# create the wordcloud
import matplotlib.pyplot as plt   # for wordclouds & charts
from matplotlib.pyplot import figure

# Define a function to plot word cloud
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(20,12), dpi=80)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.box(False)
    plt.show()
    plt.close()

#Plot
plot_cloud(wordcloud)
```


```
print(set(data['Vulnerability Type']))
```

```
{'Http Response Splitting', 'Code Execute', 'gain priviledge', 'Memory Corruption', 'Gain Information', 'CSRF', 'Denial of Service'
```

```
s='I am'# amazed'
s=s.replace('a','z')
print(s)
```

```
I zm
```

```
text = 'bat ball'

# replace 'ba' with 'ro'
replaced_text = text.replace('ba', 'ro')
print(replaced_text)

# Output: rot roll
```

```
rot roll
```