1st Nov'22

```
import requests
from bs4 import BeautifulSoup as bs
import re
```

```
#run the program to extract cve details from the website www.cvedetails.com

def geturl(typ,n):
  return dic[typ][0]+str(n)+dic[typ][1]

def looper(typ,a,b,c):
  summaryText=[]
  cvearray=[]
  tot=dic[typ][2]
  arr=[i for i in range(1,tot+1)]
  def cvedetails(url,cvearray,summaryText):
    r = requests.get(url)
    soup = bs(r.content)
    ID=[]
    summ=[]
    summarySoup=soup.find_all('td',class_="cvesummarylong",text=True)
    #print(len(summarySoup))
    for line in summarySoup:
      processed=line.text.replace('\n','').replace('\t','')
      if processed:
        summ.append(processed)
      else:
        summ.append('Nan')
    table = soup.find('table',attrs={'class','searchresults'})
    if not table:
      return
    for a in table.find_all('a',href=True):
      m = re.search("CVE-\d{4}-\d{4,7}",a['href'])
      if m:
        ID.append(m.group(0))
    if len(summ)!=50:
      return
    cvearray+=ID
    summaryText+=summ

  for i in arr:
    cvedetails(geturl(typ,i),cvearray,summaryText)
  vulType=[typ for _ in range(len(cvearray))]
  a+=cvearray
  b+=summaryText
  c+=vulType
```

```
# all the links related to the particular vulnerability type is stored and data from each link will be extracted
dic={}
dic['Denial of Service']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdo
dic['Code Execute']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0&o
dic['Overflow']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0&opec=
dic['Memory Corruption']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdo
dic['SQL Injection']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0&
dic['Cross Site Scripting (XSS)']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hase
dic['Directory Traversal']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&op
dic['Http Response Splitting']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=
dic['Bypass']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0&opec=0&
dic['Gain Information']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos
dic['gain priviledge']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=
dic['CSRF']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0&opec=0&op
dic['File Inclusion']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp=0&opdos=0
dic['Security Vulnerabilities']=['https://www.cvedetails.com/vulnerability-list.php?vendor_id=0&product_id=0&version_id=0&page=','&hasexp
```

```
# looper function here is used to extract all the data from given link at cveID,Vul_Type and SummText stores the CVE ID,Vulnerability Typ
cveID=[]
Vul_Type=[]
SummText=[]
for item in dic.keys():
  looper(item,cveID,SummText,Vul_Type)
```

```
# saving data in a excel file
import json
```

```
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile
data = {'CVE ID Number': cveID,'Vulnerability Type':Vul_Type, 'Summary Text': SummText}
df = pd.DataFrame(data,columns=['CVE ID Number','Vulnerability Type','Summary Text'])
writer = ExcelWriter('Data.xlsx')
df.to_excel(writer,'CVE Details',index=False)
writer.save()
```

```
from google.colab import files
files.download('Data.xlsx')
# Data.xlsx file contains 'CVE No.','CVE description' and 'Vulnerability Type'. Dataset count is 190K
```

8th Nov'22

```
import pandas as pd

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

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

```
print(df.columns)
```

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

```
df.shape
```

```
df['Vulnerability Type'].v  alue_counts()['Denial of Service']
```

```
df['Vulnerability Type'].value_counts()['Code Execute']
```

```
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
```

DoS

```
# visulisation on word count for Denial of Service
DoS= df[df['Vulnerability Type'].str.contains('Denial of Service')]
```

```
print(DoS.shape)
DoS.head(10)
```

```
(28100, 3)
```

	CVE ID Number	Vulnerability Type	Summary Text
0	CVE-2022-43766	Denial of Service	Apache IoTDB version 0.12.2 to 0.12.6, 0.13.0 ...
1	CVE-2022-43365	Denial of Service	IP-COM EW9 V15.11.0.14(9732) was discovered to...
2	CVE-2022-43035	Denial of Service	An issue was discovered in Bento4 v1.6.0-639. ...
3	CVE-2022-43033	Denial of Service	An issue was discovered in Bento4 1.6.0-639. T...
4	CVE-2022-42969	Denial of Service	The py library through 1.11.0 for Python allow...
5	CVE-2022-42300	Denial of Service	An issue was discovered in Veritas NetBackup t...
6	CVE-2022-42299	Denial of Service	An issue was discovered in Veritas NetBackup t...
7	CVE-2022-41715	Denial of Service	Programs which compile regular expressions fro...
8	CVE-2022-41665	Denial of Service	A vulnerability has been identified in SICAM P...
9	CVE-2022-41556	Denial of Service	A resource leak in gw_backend.c in lighttpd 1....

```
Preprocessing involves the following steps:
1. removing digits
2. lowering the alphabets
3. tokenize the sentence
4. removal of punctuations
5. removing Stopwords
6. Stemming
```

```
# preprocessing of data

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

```
DoS.head(10)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
0	CVE-2022-43766	Denial of Service	apache iotdb version .. to .., .. to .. are vu...	[apach, iotdb, version, , , , , vulner, denial...
1	CVE-2022-43365	Denial of Service	ip-com ew v...() was discovered to contain a b...	[ipcom, ew, v, discov, contain, buffer, overfl...
2	CVE-2022-43035	Denial of Service	an issue was discovered in bento v..-. there i...	[issu, discov, bento, v, heapbufferoverflow, a...
3	CVE-2022-43033	Denial of Service	an issue was discovered in bento ..-. there is...	[issu, discov, bento, , bad, free, compon, aph...
4	CVE-2022-42969	Denial of Service	the py library through .. for python allows re...	[py, librari, , python, allow, remot, attack, ...
5	CVE-2022-42300	Denial of Service	an issue was discovered in veritas netbackup t...	[issu, discov, verita, netbackup, , relat, ver...
6	CVE-2022-42299	Denial of Service	an issue was discovered in veritas netbackup t...	[issu, discov, verita, netbackup, , relat, ver...
7	CVE-2022-41715	Denial of Service	programs which compile regular expressions fro...	[program, compil, regular, express, untrust, s...
8	CVE-2022-41665	Denial of Service	a vulnerability has been identified in sicam p...	[vulner, identifi, sicam, p, version, , v, sic...
9	CVE-2022-41556	Denial of Service	a resource leak in gw_backend.c in lighttpd	[resourc, leak, gwbackendc, lighttpd, , , coul...

```
from collections import defaultdict

dic_DoS=defaultdict(int)
```

```
for row in DoS['Processed Text']:
    for item in row:
        if item!='':
            dic_DoS[item] += 1
```

```
dic_DoS_count = defaultdict(int)
for row in DoS['Processed Text']:
    for item in dic_DoS.keys():
        if item in row:
            dic_DoS_count[item] += 1
```

```
sorted_dt = {key: value for key, value in sorted(dic_DoS.items(), key=lambda item: item[1],reverse=True)}
```

```
print(sorted_dt)
```

```
{'servic': 30505, 'denial': 28672, 'caus': 26150, 'allow': 25153, 'attack': 25006, 'via': 20046, 'remot': 17482, 'cve': 14531, 'vul
```

```
sorted_ct = {key: value for key, value in sorted(dic_DoS_count.items(), key=lambda item: item[1],reverse=True)}
print(sorted_ct)
```

```
{'denial': 28097, 'servic': 28090, 'caus': 24613, 'allow': 24281, 'attack': 21603, 'via': 19834, 'remot': 17182, 'craft': 9948, 'vu
```

```
# word count sorted based on occurences form high to low
pd.Series(sorted_dt).to_frame()
Denial=pd.Series(sorted_dt).to_frame('[# of Occurrences, # of Summaries occurence found in]')
print(DoS.shape)
Denial[:20]
```

```
(28100, 4)
```

	[# of Occurrences, # of Summaries occurence found in]
servic	30505
denial	28672
caus	26150
allow	25153
attack	25006
via	20046
remot	17482
cve	14531
vulner	14310
craft	10106
crash	9583
x	9175
memori	7771
code	7718
execut	7530
arbitrari	7051
function	6897
possibl	5740
applic	5445
file	5423

Code Execute

```
CoE= df[df['Vulnerability Type'].str.contains('Code Execute')]
```

```
print(CoE.shape)
CoE.head(10)
```

(43050, 3)

	CVE ID Number	Vulnerability Type	Summary Text
28100	CVE-2022-44019	Code Execute	In Total.js 4 before 0e5ace7, /api/common/ping...
28101	CVE-2022-43775	Code Execute	The HICT_Loop class in Delta Electronics DIAEn...
28102	CVE-2022-43774	Code Execute	The HandlerPageP_KID class in Delta Electronic...

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

28107	CVE-2022-43406	Code Execute	A sandbox bypass vulnerability in Jenkins Pipe...

```
CoE.head(1).T
```

	28100
CVE ID Number	CVE-2022-44019
Vulnerability Type	Code Execute
Summary Text	in total.js before eace, /api/common/ping can...
Processed Text	[totaljs, , eace, apicommonping, achieve, remo...

```
dic_CoE=defaultdict(int)
```

```
for row in CoE['Processed Text']:
        for item in row:
            if item!='':
                dic_CoE[item]+=1
```

```
dic_CoE_count = defaultdict(int)
```

```
for row in CoE['Processed Text']:
    for item in dic_CoE.keys():
        if item in row:
            dic_CoE_count[item] += 1
```

```
sorted_CoE_count = {key: value for key, value in sorted(dic_CoE_count.items(), key=lambda item: item[1],reverse=True)}
print(sorted_CoE_count)
```

```
{'arbitrary': 34550, 'execute': 33165, 'code': 32244, 'remote': 29643, 'via': 26387, 'attackers': 25283, 'allows': 24956, 'vulnerab
```

```
sorted_CoE = {key: value for key, value in sorted(dic_CoE.items(), key=lambda item: item[1],reverse=True)}
print(sorted_CoE)
```

```
{'code': 36175, 'arbitrary': 36003, 'execute': 35027, 'remote': 33461, 'vulnerability': 30904, 'cve': 29163, 'via': 26882, 'attacke
```

```
# word count for code execution
pd.Series(sorted_CoE).to_frame()
Code=pd.Series(sorted_CoE).to_frame('CoE Occurrences')
print(CoE.shape)
Code[:20]
```

(43050, 4)

	CoE Occurrences
code	36175
arbitrary	36003
execute	35027
remote	33461
vulnerability	30904
cve	29163
via	26882
attackers	25349
allows	25106

All (without stemming)

execution 11313

Use of stemming might cause some ambiguity in understanding the word itself and hard to undertand visually. But expected to give better results upon use.

```
items=set(df['Vulnerability Type'])
print(items)
```

```
{'Code Execute', 'Overflow', 'Http Response Splitting', 'Cross Site Scripting (XSS)', 'Directory Traversal', 'Gain Information', 'S
```

```
from collections import defaultdict
dic=defaultdict()
```

```
for vul in list(items):
    print(vul)
    table=df[df['Vulnerability Type'].str.contains(vul)]
    if vul == 'Cross Site Scripting (XSS)':
        table=df[df['Vulnerability Type'].str.contains('Cross Site Scripting')]
    table['Summary Text']=table['Summary Text'].str.replace('\d+', '')
    table['Summary Text']= table['Summary Text'].apply(lambda x: x.lower())
    table['Processed Text']=table['Summary Text'].apply(lambda x: tokenization(x))
    table['Processed Text']=table['Processed Text'].apply(lambda x: remove_punctuation(x))
    table['Processed Text']=table['Processed Text'].apply(lambda x: remove_stopwords(x))
    #can remove to stem words but stemming is not perfect
    #table['Processed Text']=table['Processed Text'].apply(lambda x: stemming(x))
    dic_table=defaultdict(int)
    for row in table['Processed Text']:
        for item in row:
            if item!='':
                dic_table[item] += 1
    dic_count=defaultdict(int)
    """for row in table['Processed Text']:
        for item in dic_table.keys():
            if item in row:
                dic_count[item] += 1"""
    sorted_table= {key: value for key, value in sorted(dic_table.items(), key=lambda x: x[1],reverse=True)}
    #sorted_count= {key: value for key, value in sorted(dic_count.items(), key=lambda x: x[1],reverse=True)}

    print(sorted_table)
    print(sorted_count)
    dic[vul] = [dic_table]
```

```
Code Execute
{'code': 36175, 'arbitrary': 36003, 'execute': 35027, 'remote': 33461, 'vulnerability': 30904, 'cve': 29163, 'via': 26882, 'attacke
{'arbitrary': 34550, 'execute': 33165, 'code': 32244, 'remote': 29643, 'via': 26387, 'attackers': 25283, 'allows': 24956, 'vulnerab
Overflow
{'cve': 15924, 'overflow': 15913, 'via': 14287, 'buffer': 14204, 'code': 12622, 'allows': 12211, 'attackers': 11872, 'arbitrary': 1
{'overflow': 14563, 'via': 14113, 'buffer': 13192, 'allows': 12170, 'code': 11979, 'attackers': 11849, 'arbitrary': 11306, 'remote'
Http Response Splitting
{'http': 286, 'response': 174, 'splitting': 151, 'attacks': 134, 'vulnerability': 124, 'headers': 118, 'remote': 115, 'allows': 109
{'http': 150, 'response': 146, 'splitting': 146, 'attacks': 117, 'remote': 115, 'vulnerability': 110, 'headers': 109, 'allows': 107
Cross Site Scripting (XSS)
{'xss': 19540, 'scripting': 17224, 'crosssite': 16086, 'via': 14647, 'vulnerability': 14057, 'web': 13459, 'arbitrary': 13054, 'rem
{'xss': 18842, 'scripting': 17065, 'crosssite': 15853, 'via': 14384, 'arbitrary': 12891, 'web': 12403, 'vulnerability': 12301, 'rem
Directory Traversal
{'traversal': 5226, 'directory': 4850, 'vulnerability': 4162, 'files': 4076, 'arbitrary': 3920, 'via': 3798, 'dot': 3782, 'remote':
{'traversal': 4907, 'directory': 4288, 'files': 3711, 'via': 3654, 'arbitrary': 3650, 'allows': 3643, 'vulnerability': 3620, 'remot
Gain Information
{'information': 11733, 'sensitive': 8038, 'obtain': 7845, 'allows': 7632, 'attackers': 6456, 'via': 6330, 'remote': 5339, 'x': 4503
```

{'information': 10068, 'sensitive': 7794, 'obtain': 7737, 'allows': 7598, 'attackers': 6440, 'via': 6251, 'remote': 5195, 'vulnerab
Security Vulnerabilities
{'remote': 3514, 'via': 3427, 'attackers': 3121, 'arbitrary': 2946, 'allows': 2756, 'parameter': 2527, 'sql': 2081, 'vulnerability'
{'remote': 3327, 'via': 3310, 'attackers': 3111, 'arbitrary': 2873, 'allows': 2750, 'parameter': 2079, 'vulnerability': 1972, 'exec
Memory Corruption
{'cve': 16435, 'memory': 8709, 'corruption': 7539, 'vulnerability': 5415, 'code': 5339, 'arbitrary': 4654, 'via': 4230, 'attackers'
{'memory': 6743, 'corruption': 6738, 'code': 5209, 'arbitrary': 4615, 'via': 4223, 'attackers': 4103, 'vulnerability': 4087, 'cause
SQL Injection
{'sql': 17547, 'injection': 10252, 'via': 8485, 'parameter': 7822, 'remote': 6945, 'vulnerability': 6862, 'arbitrary': 6644, 'execu
{'sql': 10224, 'injection': 10009, 'via': 8383, 'remote': 6835, 'arbitrary': 6570, 'execute': 6480, 'parameter': 6423, 'vulnerabili
File Inclusion
{'remote': 3899, 'php': 3821, 'file': 2238, 'inclusion': 2121, 'parameter': 2076, 'arbitrary': 1951, 'via': 1945, 'code': 1941, 'at
{'inclusion': 2093, 'file': 2092, 'remote': 1976, 'arbitrary': 1927, 'via': 1921, 'attackers': 1917, 'php': 1905, 'code': 1879, 'pa
Bypass
{'bypass': 8450, 'allows': 5023, 'remote': 4518, 'attackers': 4234, 'via': 3904, 'vulnerability': 3530, 'access': 2804, 'authentica
{'bypass': 7942, 'allows': 4942, 'remote': 4381, 'attackers': 4169, 'via': 3813, 'vulnerability': 2454, 'access': 2405, 'authentica
gain priviledge
{'gain': 5507, 'privileges': 5496, 'allows': 4140, 'users': 3626, 'local': 3492, 'via': 3297, 'windows': 2949, 'vulnerability': 284
{'gain': 5439, 'privileges': 5245, 'allows': 4112, 'users': 3497, 'local': 3373, 'via': 3260, 'vulnerability': 2077, 'crafted': 114
Denial of Service
{'service': 29909, 'denial': 28661, 'cause': 25095, 'via': 20046, 'allows': 19982, 'attackers': 18002, 'remote': 17413, 'cve': 1403
{'denial': 28094, 'service': 28080, 'cause': 24171, 'allows': 19924, 'via': 19834, 'attackers': 17974, 'remote': 17136, 'crafted':
CSRF
{'csrf': 3746, 'request': 2792, 'crosssite': 2435, 'forgery': 2365, 'vulnerability': 1972, 'via': 1889, 'attackers': 1848, 'remote'
{'csrf': 3402, 'request': 2451, 'forgery': 2344, 'crosssite': 2291, 'attackers': 1839, 'vulnerability': 1715, 'remote': 1634, 'via'

```
#display(dic['Code Execute'][0])
ans=[]
for vuln in dic.keys():
    display(vuln)
    cttable=df[df['Vulnerability Type'].str.contains(vuln)]
    if vuln == 'Cross Site Scripting (XSS)':
        cttable=df[df['Vulnerability Type'].str.contains('Cross Site Scripting')]
    Count = len(cttable['Summary Text'])
    display(str(Count) + ' is the total entries')
    sorted_table= {key: value for key, value in sorted(dic[vuln][0].items(), key=lambda x: x[1],reverse=True)}
    sorted_count= {key: value for key, value in sorted(dic[vuln][1].items(), key=lambda x: x[1],reverse=True)}
    d = {
            '# of Occurrences': pd.Series(sorted_table),
            '# of Summaries word was present in': pd.Series(sorted_count),

        }
    Code = pd.DataFrame(d)
    Code = Code.sort_values(by='# of Occurrences', ascending=False)
    display(Code[:20])
    # df_styler = Code.style.set_table_attributes("style='display:inline'").set_caption('Vuln')
    ans.append(df_styler)
```

'Directory Traversal'
'5600 is the total entries'

	# of Occurrences	# of Summaries word was present in
traversal	5226	4907
directory	4850	4288
vulnerability	4162	3620
files	4076	3711
arbitrary	3920	3650
via	3798	3654
dot	3782	1848
remote	3699	3568
allows	3684	3643
attackers	2965	2939
parameter	2206	1994
read	1918	1874
file	1873	1441
path	1439	1241
attacker	1158	905
server	1013	813
local	884	845
earlier	884	723
execute	872	841
allow	855	772

'Security Vulnerabilities'
'3500 is the total entries'

	# of Occurrences	# of Summaries word was present in
remote	3514	3327
via	3427	3310
attackers	3121	3111
arbitrary	2946	2873
allows	2756	2750
parameter	2527	2079
sql	2081	1040
vulnerability	2006	1972
execute	1880	1850
commands	1127	1113
injection	1061	1060
web	862	830
crosssite	833	818
script	833	771
code	720	698
file	707	545
allow	706	694
xss	702	699
scripting	700	698
html	693	686

'SQL Injection'
'10350 is the total entries'

	# of Occurrences	# of Summaries word was present in
sql	17547	10224
injection	10252	10009
via	8485	8383

parameter	7822	6423
remote	6945	6835
vulnerability	6862	6393
arbitrary	6644	6570
execute	6584	6480
commands	6346	6277
attackers	6123	6118
allows	5692	5678
allow	1899	1819
vulnerabilities	1738	1667
id	1668	1618
multiple	1651	1639
earlier	1349	1310
indexphp	1314	1268
action	1187	1029
information	1156	894
system	1141	1071

'Denial of Service'
'28100 is the total entries'

	# of Occurrences	# of Summaries word was present in
service	29909	28080
denial	28661	28094
cause	25095	24171
via	20046	19834
allows	19982	19924
attackers	18002	17974
remote	17413	17136
cve	14033	2547
vulnerability	12347	8455
crafted	10039	9886
crash	9467	9323
x	9175	4904
memory	7771	6358
code	7698	7244
arbitrary	7051	6788
execute	6692	6486
function	5978	5544
possibly	5340	5144
unspecified	5290	5101
application	5099	4911

'Cross Site Scripting (XSS)'
'23000 is the total entries'

	# of Occurrences	# of Summaries word was present in
xss	19540	18842
scripting	17224	17065
crosssite	16086	15853
via	14647	14384
vulnerability	14057	12301
web	13459	12403
arbitrary	13054	12891
remote	12350	12218
html	11634	11251

script	11493	11108
allows	11296	11253
inject	11137	11071
attackers	10958	10925
parameter	8639	6683
allow	4818	4363
x	4082	2058
vulnerabilities	3641	3418
users	3521	3326
multiple	3494	3451
attacker	3058	2072

```
'Bypass'
'8600 is the total entries'
```

	# of Occurrences	# of Summaries word was present in
bypass	8450	7942
allows	5023	4942
remote	4518	4381
attackers	4234	4169
via	3904	3813
vulnerability	3530	2454
access	2804	2405
authentication	2705	2161
x	2520	1291
attacker	2146	1551
restrictions	2035	2011
could	1898	1304
cve	1756	555
intended	1666	1659
allow	1597	1374
security	1508	1157
users	1497	1406
user	1444	1136
file	1324	915
crafted	1311	1283

```
'gain priviledge'
'5600 is the total entries'
```

	# of Occurrences	# of Summaries word was present in
gain	5507	5439
privileges	5496	5245
allows	4140	4112
users	3626	3497
local	3492	3373
via	3297	3260
windows	2949	864
vulnerability	2843	2077
sp	2291	525
x	1648	868
server	1351	825
remote	1148	1114
crafted	1148	1141
aka	1131	1095

allow	1118	971
attacker	1104	732
attackers	1099	1091
application	1091	1010
file	1065	793
access	983	795

'CSRF'
'3800 is the total entries'

	# of Occurrences	# of Summaries word was present in
csrf	3746	3402
request	2792	2451
crosssite	2435	2291
forgery	2365	2344
vulnerability	1972	1715
via	1889	1623
attackers	1848	1839
remote	1687	1634
allows	1633	1622
authentication	1166	1135
hijack	1117	1098
plugin	997	872
requests	936	888
user	910	682
allow	886	823
users	855	759
attacker	775	601
arbitrary	774	712
wordpress	760	710
could	678	525

'Overflow'
'22600 is the total entries'

	# of Occurrences	# of Summaries word was present in
cve	15924	2430
overflow	15913	14563
via	14287	14113
buffer	14204	13192
code	12622	11979
allows	12211	12170
attackers	11872	11849
arbitrary	11473	11306
remote	11416	11032
execute	10424	10239
service	9251	8874
cause	8975	8819
denial	8468	8399
vulnerability	8220	6235
crafted	7397	7255
memory	6721	5487
x	5969	3013
function	5145	4725
corruption	4862	4286
file	4684	3992