


```
data_360 = pd.read_json(path_or_buf='拓展数据/360威胁情报数据.json', orient=None, typ='frame', dtype=True,
convert_axes=True, convert_dates=True, keep_default_dates=True,
numpy=False, precise_float=False, date_unit=None, encoding=None,
lines=False, chunksize=None, compression='infer')
data_360
```

	113.226.193.120	112.224.74.70	117.136.38.161	111.148.4.240	112.42.249.207	
botnet_info	[]	[]	{'latest_botnet_time': '2018-09-29', 'botnet_...	[]	[]	
geo	{'latitude': '38.91459', 'city': '大连', 'provin...	{'latitude': '36.094406', 'city': '青岛', 'provi...	{'latitude': '39.904989', 'city': '北京', 'provi...	{'latitude': '23.125178', 'city': '广州', 'provi...	{'latitude': '40.124296', 'city': '丹东', 'provi...	{'lat...
geo_detail	{'province': '', 'city': '', 'district': '', '...	{'province': '', 'city': '', 'district': '', '...	{'province': '', 'city': '', 'district': '', '...	{'province': '', 'city': '', 'district': '', '...	{'province': '', 'city': '', 'district': '', '...	{'pro...
malicious_info	{'latest_brute_force_time': '', 'scanner_confli...	{'latest_brute_force_time': '', 'scanner_confli...	{'latest_brute_force_time': '', 'scanner_confli...	{'latest_brute_force_time': '', 'scanner_confli...	{'latest_brute_force_time': '', 'is_web_attack...	{'latest...
normal_info	{'is_proxy': False, 'latest_proxy_time': '', '...	{'is_proxy': False, 'latest_proxy_time': '', '...	{'is_proxy': False, 'latest_proxy_time': '', '...	{'is_proxy': False, 'latest_proxy_time': '', '...	{'is_proxy': False, 'latest_proxy_time': '', '...	{'latest...
summary	{'ip': '113.226.193.120', 'is_botnet': False, ...}	{'ip': '112.224.74.70', 'is_botnet': False, 'w...	{'ip': '117.136.38.161', 'is_botnet': True, 'w...	{'ip': '111.148.4.240', 'is_botnet': False, 'w...	{'ip': '112.42.249.207', 'is_botnet': False, '...	{'ip': 'i...

6 rows × 18870 columns

2. **数据分析**：威胁情报中包含了DDOS,SCANNER,SPAM,BRUTE,WEB_ATTACKER五种攻击以及是否为僵尸主机的字段，这里我们认为DDOS和僵尸网络是关联的，所以可以把这两个字段划分为一个。接下来，首先提取出上述的5种攻击字段对应的IP，由于数据里有is_botnet字段，因此我们新建了一个BOTNET标签。

```
ip_list = data_360.keys()
mal_list = dict()
for i in ip_list:
    mal_list[i] = data_360[i].summary['malicious_label']
    if not data_360[i].summary['malicious_label']:
        # 检查是否为僵尸主机
        if data_360[i].summary['is_botnet']:
            mal_list[i] = ['BOTNET']
```

mal_list

```
{'113.226.193.120': ['DDOS'],
'112.224.74.70': ['DDOS'],
'117.136.38.161': ['BOTNET'],
'111.148.4.240': ['SPAM'],
'112.42.249.207': [],
'113.102.120.240': [],
'110.86.177.49': [],
'112.224.74.79': ['DDOS'],
'113.226.193.129': ['DDOS', 'SCANNER'],
'106.38.241.113': ['DDOS'],
'116.3.202.48': ['WEB_ATTACKER'],
'106.38.241.115': ['DDOS'],
'113.226.70.243': ['DDOS'],
'106.38.241.117': ['DDOS'],
'112.9.96.140': [],
'113.226.116.81': ['DDOS'],
'112.224.74.72': ['DDOS'],
'112.255.82.223': ['DDOS'],
'113.227.121.75': [],
'116.3.223.226': ['DDOS']}
```

3. **筛选出僵尸主机**，并将数据转换为pandas的dataframe格式。

```
botnet_list = dict()
for i in mal_list:
    try:
        botnet_list[i] = list(data_360[i].botnet_info[0].values())
    except:
        continue
```

```
botnet_list
'112.66.126.189': ['2018-09-26', '僵尸网络', 'Nitol'],
'114.232.201.216': ['2018-07-18', '', 'Minerd'],
'112.42.33.25': ['2018-08-12', '', 'Delf'],
'113.234.30.44': ['2018-10-11', '', 'Sality'],
'116.249.186.189': ['2018-08-22', '', 'Gh0st'],
'112.195.59.151': ['2018-10-22', '黑市工具', 'MinerdPool'],
'112.42.29.23': ['2018-09-20', '', 'Delf'],
'112.195.59.154': ['2018-11-24', '黑市工具', 'MinerdPool'],
'112.245.181.154': ['2018-10-15', '', 'Minerd'],
'115.219.78.33': ['2019-01-23', '黑市工具', 'Minerd'],
'106.114.192.27': ['2018-09-01', '', 'Minerd'],
'113.226.152.117': ['2018-09-22', '', 'Trojan'],
'113.226.198.167': ['2018-10-17', '', 'Reconyc'],
'113.234.146.21': ['2018-05-26', '', 'Cogyeka'],
'114.106.135.87': ['2019-01-23', '黑市工具', 'Minerd'],
'112.245.181.159': ['2018-10-18', '', 'Flyboy'],
'112.252.108.126': ['2018-12-13', '', 'Generic Trojan'],
'112.255.102.96': ['2017-07-01', '', 'Trojan'],
'114.230.150.32': ['2019-01-06', '黑市工具', 'Minerd'],
'113.226.161.159': ['2018-07-11', '', 'Blackgear_18_7_18 APT'],
```

```
botnet.head()
```

	ip	time	type	famaily
0	117.136.38.161	2018-09-29		Trojan
1	112.224.74.79	2018-08-30		Avzhan
2	113.226.70.243	2018-04-15		Minerd
3	113.226.116.81	2017-06-22		NetReaper
4	112.224.74.72	2018-11-02		MsraMiner

4. 根据famaily字段，可以粗略得出152个僵尸网络簇。以Trojan为例：

```
botnet[botnet['famaily'] == 'Trojan'].head()
```

	ip	time	type	famaily
0	117.136.38.161	2018-09-29		Trojan
7	111.182.102.150	2018-09-26		Trojan
24	113.226.152.117	2018-09-22		Trojan
30	112.255.102.96	2017-07-01		Trojan
39	112.224.74.218	2018-10-28		Trojan

5. **攻击者刻画**：现在已经获取了一些僵尸网络IP，但还没有挖掘出攻击者。考虑到攻击者获取僵尸主机的一个常见手段是利用电子邮件进行钓鱼，所以这里我们选择了**SPAM**字段来进行对攻击者的筛查（注：通过对数据的分析，我们发现SPAM和BOTNET数据没有重叠的部分，但DDOS和SPAM有重叠，而这类重叠的IP很有可能是实际活跃的攻击者）。经过这一步骤，我们得到了581个SPAM主机IP。

```
spam_list = dict()
for i in mal_list:
    if 'SPAM' in data_360[i].summary['malicious_label']:
        # 需要考虑whitelist, 即流量出口度量
        try:
            whitelist = data_360[i].summary['whitelist']
        except:
            whitelist = None
        if data_360[i].summary['is_botnet']: # 将botnet筛选出去
            continue
        spam_list[i] = ['SPAM', whitelist]
```

```
spam = pd.DataFrame(spam_list).T
spam = spam.reset_index()
spam.columns = ['ip', 'type', 'whitelist']
```

```
spam.head()
```

	ip	type	whitelist
0	111.148.4.240	SPAM	3
1	104.234.223.14	SPAM	None
2	113.23.212.191	SPAM	1
3	113.171.23.47	SPAM	1
4	109.87.30.25	SPAM	1

```
print('spam主机个数为:', spam['ip'].size)
```

```
spam主机个数为: 581
```

6. **SPAM主机IP-绑定域名信息映射**：载入domain_info数据，查找SPAM主机对应的域名绑定信息。删除没有domain绑定信息的IP，最终得到了34个有domain数据的SPAM主机。

```
In [18]: domain_info = pd.read_csv('拓展数据/domain_info.csv', header=None)
```

```
In [19]: domain_info.columns = ['ip', 'time', 'domain', 'whois']
```

```
In [20]: domain_info.head()
```

```
Out [20]:
```

	ip	time	domain	whois
0	123.168.89.127	2018.10.30	[]	[]
1	112.255.196.93	2018.11.07	[]	[]
2	182.40.122.24	2018.11.04	[]	[]
3	144.52.47.82	2018.10.31	[]	[]
4	122.241.2.89	2018.11.06	[]	[]

删除没有domain绑定数据的ip

```
In [21]: domain_info = domain_info[domain_info['domain']!= '[]']
```

```
In [22]: set(spam['ip']) < set(domain_info['ip'])
```

```
Out [22]: False
```

```
In [23]: count = 0
sd_list = []
for i in spam['ip']:
    if i in domain_info['ip'].tolist():
        sd_list.append(i)
        count += 1
count
```

```
Out [23]: 34
```

7. **局域网聚类**: 将得到的34个SPAM攻击者根据IP地址进行进一步划分, 我们假设处于同一局域网的多个IP隶属于同一个攻击者。由于数量少, 这里我们直接手动进行了划分 (自动划分的代码见Web攻击溯源部分), 得到的结果如下: (共26类)

1. 103.210.237.168
2. 103.213.250.5
3. 103.38.43.51 , 103.38.43.74
4. 103.54.60.154
5. 104.131.178.223
6. 104.234.223.14
7. 106.12.113.92
8. 106.12.116.32 , 106.12.116.33
9. 106.12.126.87
10. 106.12.127.201
11. 106.12.148.210
12. 107.151.103.229 , 107.151.103.239
13. 109.225.42.191
14. 109.61.253.177
15. 110.36.221.182
16. 112.124.39.146
17. 113.197.36.67
18. 114.112.101.89
19. 114.33.201.83
20. 115.124.30.115 , 115.124.30.3
21. 116.240.251.240
22. 117.27.143.36
23. 117.60.178.159 , 117.60.178.6
24. 117.68.175.207
25. 117.80.78.47 , 117.80.78.6
26. 117.82.100.174 , 117.82.100.178 , 117.82.100.224

8. **spam_domain数据构造**：以下构造的spam_domain数据原计划和第4步得到的152个僵尸网络簇进行整理合并，将僵尸网络和实际操控者联系起来（见流程图）。但分析日志数据时间开销较大，所以未能完成。

```
In [24]: spam_domain = dict()
for i in sd_list:
    domain = domain_info[domain_info['ip']==i]['domain'].tolist()
    whois = domain_info[domain_info['ip']==i]['whois'].tolist()
    spam_domain[i] = [domain, whois]
```

```
In [25]: spam_domain = pd.DataFrame(spam_domain).T
spam_domain = spam_domain.reset_index()
spam_domain.columns = ['ip', 'domain', 'whois']
```

```
In [26]: spam_domain.head()
```

Out [26]:

	ip	domain	whois
0	104.234.223.14	[[{"ip": "104.234.223.14;", "domain": "mc.pcg...}	[[{"domainWhois": [{"status": ["clientTransfer...
1	106.12.113.92	[[{"ip": "106.12.113.92;", "domain": "annuotec...	[[{"domainWhois": [{"status": ["serverTransfer...
2	114.33.201.83	[[{"ip": "114.33.201.83;", "domain": "114-33-2...	[[{"domainWhois": [{"status": ["ok"], "city": ...}
3	115.124.30.115	[[{"ip": "115.124.30.115;", "domain": "out30-1...	[[{"domainWhois": [{"status": ["clientDeletePr...
4	110.36.221.182	[[{"ip": "110.36.221.182;", "domain": "wimaxus...	[[{"domainWhois": [{"status": ["registrar-lock...

2.2 Web攻击溯源

1. 数据载入和分析处理同上。
2. 筛选出web_attacker

```
In [100]: web_attackers = list()
          for i in ip_list:
              if 'WEB_ATTACKER' in data_360[i].summary['malicious_label']:
                  web_attackers.append(i)

In [195]: print('web攻击主机的数量为%d' % len(np.unique(web_attackers)))

web攻击主机的数量为1145
```

3. 局域网IP聚类：因为IP数量较多，局域网IP的聚集写了简单的循环自动执行。

```
In [150]: web_attackers = sorted(web_attackers)

In [185]: web_cate = collections.defaultdict(list)
          for i in range(len(web_attackers)):
              if web_attackers[i] in sum(list(web_cate.values()), []):
                  continue
              ip = [web_attackers[i]]
              for j in range(i+1, len(web_attackers)):
                  if web_attackers[i].split('.')[3] == web_attackers[j].split('.')[3]: # 属于同一局域网
                      ip.append(web_attackers[j])

              web_cate[i] = ip
```

```
In [189]: web_cate

Out[189]: defaultdict(list,
                        {0: ['103.1.209.172'],
                         1: ['103.115.42.43'],
                         2: ['103.193.174.220'],
                         3: ['103.200.125.50'],
                         4: ['103.204.177.143'],
                         5: ['103.204.179.26'],
                         6: ['103.210.237.168', '103.210.237.55'],
                         8: ['103.211.69.58'],
                         9: ['103.212.35.69'],
                         10: ['103.213.251.69'],
                         11: ['103.213.96.156'],
                         12: ['103.214.171.214'],
                         13: ['103.215.80.170'],
                         14: ['103.218.240.182'],
                         15: ['103.223.122.13'],
                         16: ['103.224.83.111', '103.224.83.37'],
                         18: ['103.228.170.104',
                              '103.228.170.74',
                              '103.228.170.79']})
```

```
In [192]: print('Web攻击者数量为: %d' % len(web_cate))

Web攻击者数量为: 837
```

同样地，暂时认为这837个keys分别对应837个攻击者

2.3 虚假IP溯源

暂未完成

2.4 IP-恶意样本映射

通过IP-恶意样本的映射，进一步对攻击者进行分析，考虑同一个攻击者或攻击团伙可能会投放相似的恶意样本。暂未完成，部分过程展示如下：

1. 载入部分终端可疑样本以及可疑样本md5与文件名映射

部分终端可疑样本载入：

```
In [199]: terminal = pd.read_csv('拓展数据/部分终端可疑样本.csv', header=None)
```

```
In [200]: terminal.columns = ['ip', 'md5', 'mid', 'time']
```

```
In [201]: terminal.head()
```

```
Out[201]:
```

	ip	md5	mid	time
0	101.20.12.195	06e4e6631d50eeba230f46569712057f	14a60bcdce03410a4009c924b07ab5dd	1.540803e+12
1	101.20.12.195	071277cc2e3df41eeea8013e2ab58d5a	0046940b7ad58dcc78271059ea9d8f5	1.540796e+12
2	101.20.12.195	071277cc2e3df41eeea8013e2ab58d5a	0046940b7ad58dcc78271059ea9d8f5	1.540796e+12
3	101.20.12.195	07b2228a4868ff949d6eee81298fc050	c7af0a4d9dce04f8e81f65e8975651ab	1.540780e+12
4	101.20.12.195	07b2228a4868ff949d6eee81298fc050	c7af0a4d9dce04f8e81f65e8975651ab	1.540780e+12

文件有点大，先做一下内存优化

由于后面要用到md5的映射，而category类型在修改时需要先添加值到容器，比较麻烦，所以这一步先不做md5的内存优化。

```
In [202]: pre = terminal.memory_usage().sum()
terminal['ip'] = terminal['ip'].astype('category')
# terminal['md5'] = terminal['md5'].astype('category')
terminal['mid'] = terminal['mid'].astype('category')
# terminal['time'] = terminal['time'].astype('category')
after = terminal.memory_usage().sum()
print('内存优化比例为: %f' % ((pre-after)/pre))
```

内存优化比例为: 0.171396

可疑样本md5与文件名映射载入：


```
In [31]: f = open('拓展数据/可疑样本md5与文件名映射.json')
```

```
In [32]: for i in f:  
         filename = json.loads(i)
```

```
In [33]: filename
```

```
Out [33]: {'5aa27e04e4a4be01146ff3e7bd79b74e': ['驱动精灵_v9.61.309.1412.exe'],  
          '4a4e470c77e953464bb21628d18a6b8e': ['pror.p2fms.client.exe'],  
          'ad947de2d8935ccff4132c88590d8405': ['eshop5transfer.exe'],  
          'f2a7883bbd196f73ef4228a8e14a2d93': ['cdpmain.exe'],  
          'db0eb4955f5be8c8eb831746067d0f12': ['tagspaces.exe'],  
          '572c2026348f48753fe2ab700e04807d': ['gsddz.exe'],  
          '91ae0aeda7a02a16c382278e3b4604d9': ['龙虎妖杀v1.exe'],  
          '6327f13d21baf2558020768228c7832c': ['彩仙阁计划快彩版.exe'],  
          'da7f9f212a8c9760c1d504dd0ef79f9d': ['上古之战[公益]v.3.0.exe'],  
          '102e81536d0850eba6f6bfac00486952': ['小度音箱_极速安装.exe'],  
          '5bd14cca87ef2233d3d82950224b0eff': ['umins000.exe'],  
          'b5fb808283a73fab4d0e67d30488741b': ['ib_14994.bat'],  
          'ib_a100d.bat',  
          'ib_ad2aa.bat',  
          'ib_7aa6d.bat',  
          'ib_b9895.bat'],  
          '145ac7d44087ae63f1c0f44f3f5160d3': ['player.exe'],  
          'f3089758b4663738cd15fa636c753f2b': ['真能传奇[云].exe'],  
          '62c84a315747b52b09147437de9e9727': ['edbug.exe'],
```

2. 将可疑IP投放的样本md5与文件名映射对应，并根据mid字段进行去重

```
In [70]: terminal['md5'] = terminal['md5'].map(filename)
```

```
In [71]: terminal
```

Out [71]:

	ip	md5	mid	time
0	101.20.12.195	[minisite.exe]	14a60bcdce03410a4009c924b07ab5dd	1.540803e+12
1	101.20.12.195	[iexplore.exe]	0046940b7ad58dcc78271059ea9d8f5	1.540796e+12
2	101.20.12.195	[iexplore.exe]	0046940b7ad58dcc78271059ea9d8f5	1.540796e+12
3	101.20.12.195	[wncore.exe]	c7af0a4d9dce04fbe81f65e8975651ab	1.540780e+12
4	101.20.12.195	[wncore.exe]	c7af0a4d9dce04fbe81f65e8975651ab	1.540780e+12
5	101.20.12.195	[bwfastinvoice.exe]	ae3d13400cd535a1d500c393fe29f29e	1.540818e+12
6	101.20.12.195	[bwfastinvoice.exe]	ae3d13400cd535a1d500c393fe29f29e	1.540818e+12
7	101.20.12.195	[bwfastinvoice.exe]	ae3d13400cd535a1d500c393fe29f29e	1.540818e+12
8	101.20.12.195	[bwfastinvoice.exe]	ae3d13400cd535a1d500c393fe29f29e	1.540820e+12
9	101.20.12.195	[mainshellapp.exe]	10701029f2281a354cd98875a4727604	1.540783e+12

```
In [72]: terminal.rename(columns={'md5':'malware'}, inplace=True)
```

根据mid进行去重

```
In [73]: terminal.drop_duplicates('mid', inplace=True)
```

```
In [74]: terminal
```

Out [74]:

	ip	malware	mid	time
0	101.20.12.195	[minisite.exe]	14a60bcdce03410a4009c924b07ab5dd	1.540803e+12
1	101.20.12.195	[iexplore.exe]	0046940b7ad58dcc78271059ea9d8f5	1.540796e+12
3	101.20.12.195	[wncore.exe]	c7af0a4d9dce04fbe81f65e8975651ab	1.540780e+12
5	101.20.12.195	[bwfastinvoice.exe]	ae3d13400cd535a1d500c393fe29f29e	1.540818e+12
9	101.20.12.195	[mainshellapp.exe]	10701029f2281a354cd98875a4727604	1.540783e+12
13	101.20.12.195	[mainshellapp.exe]	d092b6d2d8d8786f934c5893a2fec947	1.540742e+12
15	101.20.12.195	[mainshellapp.exe]	ffc6a82b65295e33d11c02ea9a224f63	1.540774e+12
27	101.20.12.195	[kzreport.exe]	a5faf76d2bf67336da9b0ed914a6dc83	1.540785e+12
28	101.20.12.195	[wncore.exe]	88fbd2a2f684673a44f21dc25348cbb4	1.540776e+12
29	101.20.12.195	[wncore.exe]	9af5a5e5c683a6e46db144456e9a1f04	1.540802e+12

3. 构造IP-投放样本字典

```
In [88]: ip_malware = dict()
for i in terminal['ip'].unique():
    try:
        ip_malware[i] = sum(terminal[terminal['ip']==i]['malware'].tolist(), [])
    except:
        continue
```

```
In [193]: ip_malware
```

```
Out[193]: {'101.20.12.195': ['minisite.exe',
                             'iexplore.exe',
                             'wncore.exe',
                             'bwfastinvoice.exe',
                             'mainshellapp.exe',
                             'mainshellapp.exe',
                             'mainshellapp.exe',
                             'kzreport.exe',
                             'wncore.exe',
                             'wncore.exe',
                             'hnupdate.exe',
                             'hnupdate.exe',
                             'crashrpt32.exe',
                             'wnpicfg.exe',
                             'wnpicconfig.exe',
                             'crashrpt.exe',
                             'hnreport.exe',
                             'jisuptowdata.exe',
                             'updatecheck.exe',
                             'pduttv.exe',
```

3 源代码

见.py文件