# Assignment 4 : Network Traffic Analysis

2024JCS2040

Used Libraries:

- **pandas**: For working with tabular data (dataframes).
- **glob**: To find all the CSV files in a directory (wildcard search).
- **Counter**:
- **numpy**: Used for mathematical operations ( for variance).

**1) Basic Network Traffic Statistics**

The logs contain network flow details, including traffic between different IPs, packet statistics, protocol information, and timestamps.

```
Total Network Flows: 171380
Top 5 Protocols: protocolName
tcp_ip     122298
udp_ip      48453
icmp_ip       623
igmp            4
ip              2
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.5.122    30976
192.168.2.107    11669
192.168.2.113    11650
192.168.4.121    11191
192.168.1.104    10545
192.168.2.109     9069
192.168.2.106     8945
192.168.2.112     8430
192.168.3.115     7569
192.168.1.103     6999
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
198.164.30.2      25581
192.168.5.122     18296
203.73.24.75       8602
67.220.214.50      8482
208.116.9.82       3785
4.71.173.89        3581
74.63.11.179       3061
67.15.184.7        2951
67.205.110.248     2371
192.168.2.255      1587
Name: count, dtype: int64
Average Packet Size: 752.2405202549941
```

For dataset1

1) The total number of flows (i.e rows in the dataset) = 171380
2) Using column ‘protocolName’ -> calculated most frequent Top 5 used protocols.
3) Using source ip’s and destination ip’s found top 10 IP addresses that appear most as source and destination.

So these are heavy communicators.

4) the average packet size = Total Bytes / total number of packets
where : total packets= sum of total source bytes and total destination bytes
total number of packets = sum of total source packets and destination packets.

5) To find most common source-destination pair, we group together the source and destination and find which pair appears maximum times.

Here : (192.168.5.122, 198.164.30.2)

Appeared 25581 times.

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(25581))
Consistently Communicating IPs: source          destination
131.202.240.209  192.168.5.122       798
192.168.1.101    192.168.5.122       933
                 204.10.69.6        1332
192.168.1.104    67.220.214.50      6542
192.168.1.105    208.116.9.82       2246
192.168.2.106    192.168.5.122       862
                 67.192.26.89       1354
                 74.63.11.179       3051
192.168.2.107    203.73.24.75       2604
192.168.2.108    192.168.5.122       933
192.168.2.109    62.140.213.243     1105
                 72.32.186.241      1263
192.168.2.111    67.220.214.50      1821
192.168.2.112    192.168.5.122      1097
192.168.2.113    192.168.5.122      1049
                 4.71.173.89        3581
                 67.205.110.248     2371
192.168.3.115    192.168.5.122       814
                 203.73.24.75       3221
                 208.116.9.82       1037
192.168.3.116    72.11.132.253       932
192.168.4.121    192.168.5.122      3502
                 67.15.184.7        1095
192.168.5.122    198.164.30.2      25581
dtype: int64
mean_traffic:  118.84882108183079
Traffic Spikes Detected: time_bin
2010-06-14 04:44:00     915
2010-06-14 05:01:00    1303
2010-06-14 08:03:00     938
2010-06-14 08:28:00     904
2010-06-14 10:26:00     913
2010-06-14 11:09:00     979
2010-06-14 17:19:00    1600
dtype: int64
Packet Size Variance: 2571002941992.944
Source Packet Size Variance: 3403479782566.779
Destination Packet Size Variance: 1738092212097.9087
```

6) Here I printed most frequent communication pairs that communicated more than 797 times. I kept this 797 threshold just to analyse.
7) To detect irregular spikes in traffic volume over time, I used this formula
   if flow count > mean + (threshold * standard deviation)
   where mean => Average flows per minute
   Standard deviation => How much traffic varies from minute to minute. High std = more fluctuation.
   threshold => Multiplier for how far from the mean to consider a spike
   low threshold- detects even small spikes
   high threshold- detects only very big spikes

   for example : flow count > mean + (5* std)
   it means we are considering it as a spike if it is outside the normal range- more than 5 std deviations above mean
   this method is more reliable and adaptive (i.e.; Only true outliers are considered spikes and Works on different scales of traffic)

   Dataset 2

```
Total Network Flows: 133193
Top 5 Protocols: protocolName
tcp_ip     95117
udp_ip     37966
icmp_ip       81
igmp          15
ip            13
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.5.122    24033
192.168.1.104     9455
192.168.2.109     8883
192.168.1.101     7651
192.168.2.110     7529
192.168.4.120     6583
192.168.2.113     5765
192.168.4.118     5578
192.168.3.116     5211
192.168.2.106     4994
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
192.168.5.122     22759
198.164.30.2      20951
203.73.24.75       9960
62.116.163.221     1838
192.168.2.255      1582
67.220.214.50      1562
69.84.133.138      1557
208.116.9.82       1486
91.195.240.121     1413
74.55.1.4          1179
Name: count, dtype: int64
Average Packet Size: 750.6914339469795
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(20951))
Consistently Communicating IPs: source         destination
192.168.1.101  192.168.5.122       802
192.168.1.103  192.168.5.122      1327
192.168.1.104  192.168.5.122      1671
               203.73.24.75       1071
192.168.2.106  208.116.9.82       1326
192.168.2.108  192.168.5.122       803
192.168.2.109  203.73.24.75       5678
192.168.2.110  192.168.5.122       864
               67.220.214.50      1559
               69.84.133.138      1292
192.168.2.111  192.168.5.122      1180
192.168.2.112  192.168.5.122      1114
192.168.2.113  192.168.5.122       897
192.168.3.114  203.73.24.75       1278
192.168.3.116  192.168.5.122       956
               62.116.163.221     1098
               91.195.240.121      987
192.168.3.117  192.168.5.122      1038
192.168.4.118  125.6.164.41        898
               125.6.164.43       1006
               125.6.164.51       1125
192.168.4.119  203.73.24.75        810
192.168.4.121  192.168.5.122      1090
192.168.5.122  198.164.30.2      20951
192.168.5.124  192.168.5.122      4508
80.66.211.179  192.168.5.122      2043
dtype: int64
mean_traffic:  92.62378303198888
Traffic Spikes Detected: time_bin
2010-06-12 04:11:00     689
2010-06-12 06:26:00     707
2010-06-12 06:28:00     729
2010-06-12 06:29:00     784
2010-06-12 06:30:00     782
2010-06-12 06:36:00     674
2010-06-12 07:35:00     744
2010-06-12 13:05:00    1294
2010-06-12 13:06:00     706
dtype: int64
Packet Size Variance: 434945945802.3868
Source Packet Size Variance: 4791648135.251914
Destination Packet Size Variance: 864688591500.3694
```

Dataset 3

```
Total Network Flows: 275528
Top 5 Protocols: protocolName
tcp_ip     221026
udp_ip      54076
icmp_ip       374
igmp           28
ip             24
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.2.106    38390
192.168.5.122    37698
192.168.1.105    35806
192.168.2.110    29718
192.168.3.116    18004
192.168.4.118    15198
192.168.2.112    13936
192.168.1.101    11422
192.168.4.119     8716
192.168.4.120     8510
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
198.164.30.2      32812
67.220.214.50     28962
192.168.5.122     27078
125.6.176.113     16434
203.73.24.75      16186
125.6.164.51       6932
67.111.12.102      6480
82.98.86.183       5054
192.168.2.255      3192
212.227.111.29     2708
Name: count, dtype: int64
Average Packet Size: 726.9582135018807
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(32812))
Consistently Communicating IPs: source         destination
192.168.1.101  192.168.5.122       1222
               67.111.12.102       6478
192.168.1.103  192.168.5.122       1134
               203.73.24.75        1410
192.168.1.104  192.168.5.122       1166
192.168.1.105  192.168.1.101       1188
               192.168.1.103       1198
               192.168.1.104       1204
               192.168.2.106       1636
               192.168.2.108       1642
               192.168.2.109       1602
               192.168.2.110       1624
               192.168.2.111       1652
               192.168.2.112       1828
               192.168.2.113       1630
               192.168.5.122       1856
               203.73.24.75        6020
192.168.2.106  192.168.5.122       1310
               211.120.61.129      1066
               62.140.213.243       830
               67.220.214.50      24920
192.168.2.107  192.168.5.122        884
192.168.2.108  192.168.5.122       1508
               67.15.184.7         1196
192.168.2.110  125.6.176.113      16434
               192.168.5.122        902
               203.73.24.75        2748
               209.202.252.50      1134
192.168.2.110  125.6.176.113      16434
               192.168.5.122        902
               203.73.24.75        2748
               209.202.252.50      1134
               210.188.254.34       824
               67.220.214.50       3150
192.168.2.111  125.89.196.18       1236
               192.168.5.122       1654
192.168.2.112  192.168.5.122       4004
               192.168.5.123        930
               192.168.5.124        860
192.168.3.114  203.73.24.75        1790
192.168.3.115  192.168.5.122        816
               67.20.126.72        1636
               74.220.195.150      1196
192.168.3.116  192.168.5.122       1238
               203.73.24.75        1174
               212.227.111.29      2708
               82.98.86.181        2342
               82.98.86.183        4860
192.168.3.117  192.168.5.122       1092
192.168.4.118  125.6.164.41        1700
               125.6.164.43        1396
               125.6.164.51        6932
               192.168.5.122        926
               72.11.132.253        960
192.168.4.119  192.168.5.122       1298
               203.73.24.75        1820
192.168.4.120  192.168.5.122        838
192.168.4.121  174.132.188.162     1078
               192.168.5.122       2254
192.168.5.122  198.164.30.2       32812
dtype: int64
mean_traffic:  191.07350901525658
Traffic Spikes Detected: time_bin
2010-06-13 16:37:00     3962
2010-06-13 16:42:00    12108
2010-06-13 16:58:00     3276
dtype: int64
Packet Size Variance: 289043506239.0224
Source Packet Size Variance: 6432631435.514959
Destination Packet Size Variance: 571290035706.6566
```

Dataset 4

```
Total Network Flows: 397595
Top 5 Protocols: protocolName
tcp_ip     329378
udp_ip      67658
icmp_ip       547
igmp            6
ip              6
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.5.122    50168
192.168.3.116    39758
192.168.4.118    32865
192.168.2.111    28647
192.168.1.101    22501
192.168.2.107    21471
192.168.2.112    19481
192.168.2.108    18168
192.168.2.113    17549
192.168.3.114    16542
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
198.164.30.2       44214
125.6.164.51       36360
192.168.5.122      29496
203.73.24.75       22499
202.210.143.140    13898
67.111.12.102      12806
82.98.86.183        9264
67.220.214.50       6860
204.236.225.39      4459
125.6.164.41        3845
Name: count, dtype: int64
Average Packet Size: 756.8959143165649
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(44214))
Consistently Communicating IPs: source          destination
131.202.243.90  192.168.5.122      5207
192.168.1.101   192.168.5.122      1070
                67.111.12.102     10181
192.168.1.102   67.111.12.102      1049
                67.220.214.50      5194
                                  ...
192.168.4.120   74.220.195.150     1265
192.168.4.121   192.168.5.122      3063
                210.172.144.10      855
                82.98.86.161       1233
192.168.5.122   198.164.30.2      44214
Length: 66, dtype: int64
mean_traffic:  275.53361053361056
Traffic Spikes Detected: Series([], dtype: int64)
Packet Size Variance: 238885291862.0137
Source Packet Size Variance: 1294577123.2172725
Destination Packet Size Variance: 476028487401.60095
```

Dataset 5

```
Total Network Flows: 571698
Top 5 Protocols: protocolName
tcp_ip     441563
udp_ip     124023
icmp_ip      6073
igmp           20
ip             19
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.2.107    82964
192.168.2.109    58261
192.168.5.122    56915
192.168.4.118    49861
192.168.1.101    47933
192.168.1.105    40636
192.168.4.119    39843
192.168.2.112    31307
192.168.1.102    27068
192.168.1.103    18005
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
203.73.24.75       80664
192.168.5.122      68050
198.164.30.2       49347
95.211.98.12       25095
202.210.143.140    19307
95.211.98.14       12975
82.98.86.183        9961
72.32.84.3          6112
192.168.2.107       6098
91.190.170.71       5609
Name: count, dtype: int64
Average Packet Size: 707.0657094385142
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(49347))
Consistently Communicating IPs: source         destination
192.168.1.101  115.85.145.2        1186
               123.50.56.175       4345
               192.168.5.122       1273
               198.173.85.138      1075
               203.73.24.75       24821
               208.116.9.82        1282
192.168.1.102  192.168.5.122       1089
               203.73.24.75        3245
               62.140.213.243      1830
               64.74.98.80         1028
               67.220.214.50       1581
               72.21.81.133        1358
               91.190.170.71       4622
192.168.1.103  192.168.5.122       4573
               59.106.97.232        964
192.168.1.105  192.168.5.122       5438
               203.73.24.75       17945
192.168.2.106  192.168.5.122        804
192.168.2.107  192.168.5.122       1451
               67.220.214.50       1621
192.168.2.108  192.168.5.122        959
192.168.2.109  192.168.5.122       7424
               203.73.24.75        2528
               209.202.254.14      1497
               95.154.240.242      1330
               95.211.98.12       25095
               95.211.98.14       12975
192.168.2.110  192.168.5.122       9168
192.168.2.112  192.168.5.122       1960
               212.227.111.29      1419
               213.155.64.209      4456
               82.98.86.181        4708
               82.98.86.183        9438
               85.183.249.139      1125
               91.195.240.124      1632
192.168.2.113  192.168.5.122      11901
               208.116.9.82        1971
192.168.3.115  192.168.5.122        994
               67.212.184.66       1244
               72.32.84.3          5853
192.168.3.116  203.73.24.75        2262
               67.15.255.4         1233
192.168.4.118  115.85.145.2        3174
               115.85.145.5         808
               192.168.5.122       6742
               198.173.85.138      1094
               203.73.24.75       22956
               208.111.160.6        808
               72.11.132.253       1314
192.168.4.119  192.168.5.122       1776
               202.210.143.140    19199
               202.248.89.71       1218
               203.73.24.75        1815
               219.94.168.46       1258
192.168.4.120  192.168.5.122       5892
               203.73.24.75        2183
192.168.4.121  192.168.5.122       3520
               203.73.24.75         864
192.168.5.122  198.164.30.2       49347
dtype: int64
mean_traffic:  388.9102040816326
Traffic Spikes Detected: time_bin
2010-06-15 14:45:00    2076
dtype: int64
Packet Size Variance: 583230434856.2898
Source Packet Size Variance: 1019803148224.982
Destination Packet Size Variance: 145988095006.44305
```

Dataset 6

```
Total Network Flows: 522263
Top 5 Protocols: protocolName
tcp_ip     434674
udp_ip      87070
icmp_ip       513
igmp            4
ip              2
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.2.107    88643
192.168.5.122    68477
192.168.4.121    54319
192.168.3.115    48736
192.168.1.104    32412
192.168.4.118    28209
192.168.2.110    26096
192.168.2.113    22135
192.168.1.101    20972
192.168.3.116    20440
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
125.6.164.51      62409
198.164.30.2      59504
203.73.24.75      55289
192.168.5.122     33758
209.112.44.10     21824
64.38.193.26      20014
62.140.213.243    15047
72.21.81.133       7306
195.189.244.14     5488
125.6.164.41       5424
Name: count, dtype: int64
Average Packet Size: 760.7146879144266
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(59504))
Consistently Communicating IPs: source         destination
192.168.1.101  192.168.5.122     1257
               203.73.24.75      2739
               63.236.73.147     1284
               63.236.73.220     4945
               69.28.157.210     1239
                                 ...
192.168.4.121  203.73.24.75       918
               209.112.44.10    21824
               64.74.172.200     2041
               74.86.132.183     2488
192.168.5.122  198.164.30.2     59504
Length: 65, dtype: int64
mean_traffic:  361.4276816608997
Traffic Spikes Detected: Series([], dtype: int64)
Packet Size Variance: 1989850531263.404
Source Packet Size Variance: 3077358750.8018923
Destination Packet Size Variance: 3976092082375.908
```

These are extracted from the datasets. Every dataset has kind of traffic. And there are some common communications pairs and the in every dataset average packet size is nearly 730bytes.

8) variance of packet sizes
   for dataset 1 and dataset6 has very high variance -> indicating a broad mix of packet sizes. This high variance may be because of heavy file transfers, video streaming or malicious traffic.
   Source and destination imbalance:
   In dataset 5, the source variance is 7× higher than destination → maybe upload-heavy traffic or a server sending varied responses.
   In dataset6, destination variance is huge compared to source → possibly downloads, streaming, or attacks targeting receivers.
   for datasets- 2,3 and 4 -> has lower variance indicates more uniform packet sizes.

The combined datasets traffic details:

```
Total Network Flows: 2071657
Top 5 Protocols: protocolName
tcp_ip     1644056
udp_ip      419246
icmp_ip       8211
igmp            77
ip              66
Name: count, dtype: int64
Top 10 Active Source IPs: source
192.168.5.122    268267
192.168.2.107    208379
192.168.4.118    135374
192.168.1.101    116292
192.168.4.121    105454
192.168.1.105    101359
192.168.2.109     99183
192.168.3.116     97241
192.168.2.110     90658
192.168.3.115     88915
Name: count, dtype: int64
Top 10 Active Destination IPs: destination
198.164.30.2       232409
192.168.5.122      199437
203.73.24.75       193200
125.6.164.51       106826
67.220.214.50       49298
202.210.143.140     36189
82.98.86.183        25214
95.211.98.12        25095
209.112.44.10       21824
62.140.213.243      20509
Name: count, dtype: int64
Average Packet Size: 736.9248946259129
```

```
Most Common Source-Destination Pair: (('192.168.5.122', '198.164.30.2'), np.int64(232409))
Consistently Communicating IPs: source          destination
0.0.0.0          0.0.0.0           1553
131.202.240.209  192.168.5.122     1067
131.202.240.218  192.168.5.122     2023
131.202.243.90   192.168.5.122     5221
192.168.1.101    115.85.145.2      1194
                                  ...
192.168.5.122    198.164.30.2    232409
                 209.85.51.222     3389
192.168.5.123    192.168.5.255     1357
192.168.5.124    192.168.5.122     4694
80.66.211.179    192.168.5.122     2043
Length: 249, dtype: int64
mean_traffic:  239.83063209076175
Traffic Spikes Detected: time_bin
2010-06-13 09:40:00     2042
2010-06-13 09:41:00     1814
2010-06-13 09:59:00     1738
2010-06-13 10:06:00     1806
2010-06-13 10:23:00     1760
2010-06-13 16:37:00     3962
2010-06-13 16:42:00    12108
2010-06-13 16:58:00     3276
2010-06-15 14:45:00     2076
dtype: int64
Packet Size Variance: 987536105014.1567
Source Packet Size Variance: 565170871730.293
Destination Packet Size Variance: 1409388913208.0286
```

2) **Traffic Estimation Using Sublinear Space**

Using memory-efficient probabilistic data structures to approximate counts.

### a) HyperLogLog(HLL) for estimating unique source IPs.

- Storing all IPs in a set uses too much memory. HLL estimates the count with a small, fixed amount of memory, and tolerable error.
- **HLL** uses **hash functions** and **probabilistic counting** to estimate cardinality.
- A hash function maps input items (like IPs) to a random bit string.
- The number of leading zeros in a hash gives a clue about how many unique values we've seen. With more unique items, you're more likely to see a hash with more leading 0s.

**How HLL Works**

**1**. Hash Each Input

Each element (e.g., IP) is hashed to a uniform 32-bit value.

2. Divide Hash Space

Split the hash space into m=2^p buckets/registers using the first p bits of the hash.

Each bucket records the maximum number of leading zeroes seen in that group.

3. Count Leading Zeros

For the rest of the bits (after the p bits used for bucket), count how many leading zeros there are. This value (plus 1) is stored if it's higher than the current in that register.

The below metrics are done for the all combined datasets.

```
Estimated unique IPs (HyperLogLog): 8192
Exact unique IPs: 8405
Error (%): 2.5342058298631764
```

In this using sublinear space (HLL) we got 8192 unique IP's when the p=13 and using full linear space we got 8405 unique IP's.

The error calculated as $$\text{Error Rate} = \frac{|\text{Estimated Count} - \text{Exact Count}|}{\text{Exact Count}} \times 100$$

We got 2.53% of error rate in this.

Here if we decrease the p value then the number of registers we be less then we get less number of unique IP's detection than the exact unique Ip's. So, for small dataset this is good.

if we increase the p value then the number of registers we be less then we get more number of unique IP's detection than the exact unique Ip's.

When we are finding for single dataset, then small p value (like 4 or 5) is enough to get less error rate.

The time taking for HLL and full linear space(exact method) both are linear only.

In exact method, stores all unique elements (e.g., every unique IP address) in memory using structures like Python set or dict. Space grows linearly with the number of unique elements.

In HLL method, uses a fixed-size structure; space depends on p (precision parameter), not the number of elements. Memory usage is very low: For p = 12, only 2^12 = 4096 registers are needed (~3 KB). Therefore, we sacrifice some accuracy to save memory. So, we get some error.

Therefore, In exact method the accuracy is 100%. But in Sublinear space(HLL) we get some error, so accuracy is not 100%.

Therefore for large datasets, we use sublinear space and for small datasets we use full linear space.

### b) Count-Min Sketch to approximate frequency of destination IPs.

Count-Min Sketch (CMS) is a probabilistic data structure used to estimate the frequency of elements in a data stream — without storing all elements.

It's especially useful when:

- The dataset is huge or streaming.
- We want fast, memory-efficient frequency approximations.
- We can tolerate small overestimation errors.

How CMS works:

CMS uses a 2D array of counters, indexed by multiple hash functions. Think of it like a compact grid of counters:

- Width (w): Number of columns per row (related to error).
- Depth (d): Number of rows = number of independent hash functions (related to confidence).

Update Process

1. For each item x and each hash function $h_i$:
2. Compute $h_i(x)$ to get the column index.
3. Increment CMS[i][$h_i(x)$] by the item's count.

Estimate Process

To estimate the count of x, you:

1. Hash x with all d hash functions.
2. Get the counts at those hashed indices.
3. Return the minimum of those counts.

This helps reduce the impact of hash collisions.

```
IP: 198.164.30.2, Real: 25581, Approx: 25603, Error: 0.09%
IP: 192.168.5.122, Real: 18296, Approx: 18301, Error: 0.03%
IP: 203.73.24.75, Real: 8602, Approx: 8610, Error: 0.09%
IP: 67.220.214.50, Real: 8482, Approx: 8488, Error: 0.07%
IP: 208.116.9.82, Real: 3785, Approx: 3790, Error: 0.13%
```

In this the error is very less and the top frequently contacted IP's are same in both cases.

### c) Bloom Filter to efficiently test IP existence.

A Bloom Filter is a space-efficient probabilistic data structure used to test whether an element is in a set. It can return false positives (i.e., it may say an item is in the set when it isn't).But it never returns false negatives (if it says "not present", the item is definitely not in the set).

How bloom filter works:

A Bloom Filter uses:

- A bit array of m bits, all initialized to 0.
- k independent hash functions.

Adding an Item

1. Hash the item using k different hash functions → get k indices.
2. Set the bits at those indices in the bit array to 1.

Checking Membership

1. Hash the item again using the same k hash functions.
2. Check the bits at the corresponding k indices.
3. If any of them is 0, the item is definitely not in the set.

4. If all are 1, the item is probably in the set (could be a false positive).

```
--- Membership Testing ---
Was 192.168.5.122 seen before (bloom)? True
Was 192.168.5.122 seen before (exact)? True
False Positive Rate of Bloom Filter: 0.00%
```

Therefore, Bloom filter effectively detected known IPs with zero false negatives.

```
--- Membership Testing ---
Was 192.168.5.0 seen before (bloom)? False
Was 192.168.5.0 seen before (exact)? False
False Positive Rate of Bloom Filter: 0.00%
```

Therefore, Bloom filter effectively detected that the IP is not there.

This method is sensitive to poor hash function choice.

## 3) **Advanced Anomaly Detection**

### a) **Identify anomalous behaviors based on statistical deviations and patterns.**

For anomalies detection I used two methods for keeping the threshold
1) mean + 2×std and 2) median + 2×MAD. (I used all datasets combinely)

```
Anomalies (high packet sizes using mean+std):
                source     destination
13       192.168.1.102   192.168.5.122
19       192.168.1.103   192.168.5.122
54       192.168.2.109     67.18.22.28
62       192.168.2.109   66.235.126.95
63       192.168.4.119    98.137.80.50
...                ...             ...
2071556  192.168.3.115    203.73.24.75
2071562  192.168.3.114  195.189.244.14
2071593  192.168.3.115    203.73.24.75
2071622  192.168.1.101   115.146.6.176
2071632  192.168.3.116   170.110.64.78

[124988 rows x 2 columns]
Anomalies (high packet sizes using median+mad):
                source     destination
13       192.168.1.102   192.168.5.122
16       192.168.1.102   192.168.5.122
19       192.168.1.103   192.168.5.122
23       192.168.1.102   192.168.5.122
26       192.168.4.119  65.181.169.155
...                ...             ...
2071635  192.168.3.115    203.73.24.75
2071639  192.168.4.118   68.178.178.33
2071641  192.168.4.118  207.46.141.138
2071644  192.168.2.107    125.6.164.51
2071652  192.168.2.107    125.6.164.51

[562428 rows x 2 columns]
```

```
Outlier sources by flow count:
 source
192.168.1.101    116292
192.168.1.102     52768
192.168.1.103     60189
192.168.1.104     65666
192.168.1.105    101359
192.168.2.106     71346
192.168.2.107    208379
192.168.2.108     44363
192.168.2.109     99183
192.168.2.110     90658
192.168.2.111     61491
192.168.2.112     88301
192.168.2.113     77433
192.168.3.114     55588
192.168.3.115     88915
192.168.3.116     97241
192.168.3.117     28650
192.168.4.118    135374
192.168.4.119     74958
192.168.4.120     53659
192.168.4.121    105454
192.168.5.122    268267
```

Detect anomalies if average packet size exceeds threshold.

```
Protocol distribution:
 protocolName
tcp_ip      7.935947e-01
udp_ip      2.023723e-01
icmp_ip     3.963494e-03
igmp        3.716832e-05
ip          3.185856e-05
ipv6icmp    4.827054e-07
Name: proportion, dtype: float64
```

```
Daily traffic:
 day
2010-06-11       357951
2010-06-12    316117699
2010-06-13    428332402
2010-06-14   1069942404
2010-06-15   1887100602
2010-06-16    789775661
2010-06-17    606612141
Name: totalSourceBytes, dtype: int64
```

Flow count and protocol distribution anomalies are identified.

Calculates daily traffic by aggregating source bytes sent per hour and per day for temporal comparison.

```
Hourly traffic:
 hour
0      84116806
1      37614569
2     815975481
3      55743097
4     256190070
5     180323214
6     127306077
7     114806562
8     142540702
9     180420201
10    286940425
11    259804643
12    202832671
13    141352317
14    119866696
15     99811231
16    553103433
17    148413562
18     93702791
19     83274470
20     63051800
21     86316070
22    885255241
23     79476731
```

```
Outlier IPs (high/low volume):
 source
192.168.1.101     229244162
192.168.1.102     130811393
192.168.1.103     190875178
192.168.1.104      82962706
192.168.1.105     233991932
192.168.2.106      96860665
192.168.2.107    1766723841
192.168.2.109     279555269
192.168.2.110     176514136
192.168.2.111      86681550
192.168.2.112     219642794
192.168.2.113     219011411
192.168.3.114     101839027
192.168.3.115     133416562
192.168.3.116     130994152
192.168.3.117      82619605
192.168.4.118     201191844
192.168.4.119      90075472
192.168.4.120     149211253
192.168.4.121     348236805
Name: totalSourceBytes, dtype: int64
```

IPs with extreme (high/low) byte transfers are flagged.

b) Behavioral Analysis

```
IPs with sudden traffic spikes:
 [('131.202.243.84', datetime.date(2010, 6, 14), np.int64(29938)), ('131.202.243.90', datetime.date(201
0, 6, 17), np.int64(5674675)), ('192.168.2.106', datetime.date(2010, 6, 13), np.int64(47323658)), ('192
.168.2.107', datetime.date(2010, 6, 14), np.int64(812802265)), ('192.168.2.107', datetime.date(2010, 6,
 15), np.int64(847612957)), ('192.168.2.109', datetime.date(2010, 6, 15), np.int64(206815824)), ('192.1
68.2.110', datetime.date(2010, 6, 15), np.int64(82306124)), ('192.168.2.113', datetime.date(2010, 6, 15
), np.int64(94985567)), ('192.168.3.117', datetime.date(2010, 6, 17), np.int64(43070141)), ('192.168.4.
118', datetime.date(2010, 6, 15), np.int64(102518834)), ('192.168.4.120', datetime.date(2010, 6, 15), n
p.int64(65402041)), ('192.168.5.124', datetime.date(2010, 6, 12), np.int64(485538))]
Common targets in short window:
 Index(['198.164.30.2', '192.168.5.122', '67.111.12.102', '68.178.178.97',
       '69.163.132.48', '208.116.9.82', '174.36.96.24', '125.6.164.43',
       '125.6.164.41', '174.36.30.50', '61.208.135.178', '82.98.86.181',
       '202.172.28.86', '192.168.2.255', '208.92.232.123', '208.111.161.254',
       '192.168.1.255', '72.14.204.118', '142.176.121.69', '174.120.114.27',
       '192.168.3.255', '66.232.135.20', '67.192.97.104', '72.14.204.148',
       '91.195.240.124', '192.150.18.200', '206.16.237.195', '174.129.228.1',
       '70.87.94.162'],
      dtype='object', name='destination')
```

First we check per-IP traffic over days and find sudden traffic spikes in IP if IPs with unusually high traffic on some days after prior inactivity.

c) Suspicious Communication Patterns

```
Destination IPs contacted by many sources:

destination
192.168.5.122    20
Name: source, dtype: int64
Long duration flows:
                source      destination  duration
0        192.168.5.122      224.0.0.251     840.0
1        192.168.2.111  206.217.198.186     180.0
5        192.168.4.119    142.166.14.86     180.0
63       192.168.4.119     98.137.80.50     180.0
90       192.168.1.103    192.168.5.122     180.0
...                ...              ...       ...
2070292  192.168.4.119  125.206.167.114     180.0
2070400  192.168.2.110  204.160.127.126     180.0
2070538  192.168.4.119    63.245.217.40     180.0
2070616  192.168.1.105  125.206.224.220     180.0
2070622  192.168.2.112    67.212.184.66     180.0

[56396 rows x 3 columns]
Sources using multiple protocols quickly:
 Series([], Name: protocolName, dtype: int64)
```

Finding destination IPs contacted by many different sources within the same minute.

Finding flows with duration above the 95th percentile (likely abnormal). That is the flows that lasted longer than 95% of all other flows

Finding source IPs that use more than one protocol in the same minute—could indicate scanning or tunneling activity.

## 4)Deep Threat Analysis

a) Detecting Complex Attack Patterns

**Stealthy Port Scan Detection**

Identify source IPs contacting a large number of destination ports over time, indicative of stealthy port scanning. Grouped flows by source and destination ports. Flagged IPs contacting more than 10 unique destination ports. These IPs might be attempting to map open ports slowly to avoid detection.(the dataset3 is used here)

```
Dataset loaded with 275528 flows

(a) Detecting Complex Attack Patterns
Detecting stealthy port scans...
Stealthy port scanners detected:
source
192.168.1.101      49
192.168.1.102      28
192.168.1.103      39
192.168.1.104      72
192.168.1.105    1099
192.168.2.106      25
192.168.2.107      58
192.168.2.108      56
192.168.2.109      21
192.168.2.110      53
192.168.2.111      97
192.168.2.112     775
192.168.2.113      92
192.168.3.114      96
192.168.3.115      58
192.168.3.116      29
192.168.3.117      56
192.168.4.118     105
192.168.4.119      53
192.168.4.121     545
Name: destinationPort, dtype: int64
Identifying possible slow DDoS patterns...
Potential slow DDoS targets:
destination
142.166.14.78    2
Name: count, dtype: int64
Checking for IP hopping behavior...
Destinations contacted by many different sources:
destination
192.168.5.122    35
Name: source, dtype: int64
```

Slow DDoS Detection

Detect IPs sending a large number of packets slowly to drain resources undetected. Flagged IPs with high packet counts and durations exceeding 300 seconds.

**IP Hopping Detection**

Identify cases where multiple source IPs target the same destination IP and port, suggesting IP spoofing or rotation. Grouped by destination IP and port, checked for >5 unique sources. These Ips were contacted by several distinct IPs in a short window.

b)Malicious Payload Identification

Unusual Payload Patterns

Spot non-empty payloads with irregular characters, possible malware indicators. Searched payloads for unexpected or high-entropy characters in UTF strings.

Encrypted Traffic Anomalies

Find encrypted-looking payloads where traffic wasn't expected to be encrypted. Checked Base64 payloads that appear encrypted but are over UDP or unknown applications.

```
(b) Malicious Payload Identification
Analyzing payloads for anomalies...
Flows with suspicious command patterns: 54
Checking encrypted traffic that doesn't match normal patterns...
Suspicious encrypted flows: 0
Detecting potential command-and-control communication patterns...
Repeated communication pairs:
source           destination
0.0.0.0          0.0.0.0          388
131.202.240.218  192.168.5.122    578
142.167.88.44    192.168.5.122    452
192.168.1.101    12.51.32.102      20
                 124.198.191.30    22
                                  ...
192.168.5.122    96.6.112.197      18
192.168.5.123    192.168.5.122     30
                 192.168.5.255    386
                 216.49.88.17     238
61.147.67.206    192.168.5.122     42
Length: 1364, dtype: int64
```

Malware Command & Control (C2) Pattern Matching

Find periodic low-byte communications or TCP flows with SYN-ACK flags and minimal payloads. Flagged small byte flows with known suspicious TCP flag combinations.

c)Threat Attribution and Risk Analysis

Risk Categorization

High-risk: Matches C2 + suspicious payload + stealthy behavior

Medium-risk: Only one or two patterns matched

Low-risk: Only suspicious but not conclusive

```
(c) Threat Attribution and Risk Analysis
Assigning risk categories to flows...

Threat Summary Report:
RiskLevel
High      273956
Low         1552
Medium        20
Name: count, dtype: int64

Sample High-Risk Events:
          source     destination                RiskReason
0  192.168.5.122     224.0.0.251  Repeated C2-style traffic
1  192.168.5.122     224.0.0.251  Repeated C2-style traffic
2  192.168.2.113   192.168.5.122         Stealthy port scan
3  192.168.2.113   192.168.5.122         Stealthy port scan
4  192.168.2.113  207.241.148.80         Stealthy port scan
```

- High-risk flows showed both suspicious payloads **and** command-and-control behavior.
- Medium-risk had stealthy scans or encrypted UDP traffic without other indicators.
- Low-risk had minor anomalies.

With all datasets:

```
Dataset loaded with 2071657 flows

(a) Detecting Complex Attack Patterns
Detecting stealthy port scans...
Stealthy port scanners detected:
source
192.168.1.101      235
192.168.1.102      130
192.168.1.103      202
192.168.1.104      396
192.168.1.105     1610
192.168.2.106      109
192.168.2.107    16961
192.168.2.108      207
192.168.2.109      101
192.168.2.110      275
192.168.2.111      535
192.168.2.112     1150
192.168.2.113      572
192.168.3.114      597
192.168.3.115      299
192.168.3.116      243
192.168.3.117      254
192.168.4.118      647
192.168.4.119      274
192.168.4.120      283
192.168.4.121     3668
192.168.5.122       32
Name: destinationPort, dtype: int64
```

```
Identifying possible slow DDoS patterns...
Potential slow DDoS targets:
destination
255.255.255.255    2397
192.168.2.255       805
192.168.1.255       539
192.168.3.255       536
192.168.2.109       526
Name: count, dtype: int64
Checking for IP hopping behavior...
Destinations contacted by many different sources:
destination
115.178.18.2       21
122.100.4.68       21
124.198.191.30     21
125.6.164.40       21
128.121.146.100    21
                   ..
98.137.80.31       21
98.137.80.32       21
98.137.80.33       21
98.137.80.49       21
98.137.80.50       21
Name: source, Length: 182, dtype: int64
```

```
(b) Malicious Payload Identification
Analyzing payloads for anomalies...
Flows with suspicious command patterns: 1443
Checking encrypted traffic that doesn't match normal patterns...
Suspicious encrypted flows: 0
Detecting potential command-and-control communication patterns...
Repeated communication pairs:
source           destination
0.0.0.0          0.0.0.0          1553
109.197.48.50    192.168.2.107      15
109.92.108.167   192.168.2.107      13
109.93.202.76    192.168.2.107      17
112.135.223.212  192.168.2.107      11
                                   ...
93.86.232.207    192.168.2.107      19
93.94.232.210    192.168.2.107      11
94.143.40.39     192.168.2.107      11
94.79.33.20      192.168.2.107      13
96.56.113.234    192.168.2.107      11
Length: 12249, dtype: int64
```

```
(c) Threat Attribution and Risk Analysis
Assigning risk categories to flows...

Threat Summary Report:
RiskLevel
High      2048770
Medium      22657
Low           230
Name: count, dtype: int64

Sample High-Risk Events:
          source      destination          RiskReason
0  192.168.5.122      224.0.0.251  Stealthy port scan
1  192.168.2.111  206.217.198.186  Stealthy port scan
2  192.168.4.119    192.168.5.122  Stealthy port scan
3  192.168.4.119   219.94.203.105  Stealthy port scan
4  192.168.4.119     98.137.80.50  Stealthy port scan
```