Anomaly Detection Documentation

1 Overview

The goal is to detect scanner activity over a period. The data can help identify scanners that look anomalous by clustering similar scanners based on the past packets. Once the clusters are created, user can then choose to further analyze any sources they might find interesting

2 Dependencies

The dependencies files (for python 3.10.4) are located in the ***requirements*** folder in the code location

2.1 Pip Install

It is recommended to create a separate virtual environment. Included requirements.txt will install necessary python libraries into your environment. Use the following command:

```
pip install -r requirements.txt
```

2.2 Manual Install

You can also choose to install manually the following python libraries:

- Nltk=3.8.1
- Scipy=1.10.1
- scikit-learn=1.2.2
- numba=0.57.0
- numpy=1.24.3
- pandas=2.0.2
- matplotlib=3.7.1
- jupyter=1.0.0
- Pyyml=6.0.1

3 Description of Clustering Algorithms and Metrics

3.1 Clustering Algorithms

This code allows users to select from three different algorithms:

1. Mean Shift Clustering: Mean shift clustering algorithm works by trying to find groups of clusters based on the closest centroids (cluster centers). It requires the parameter ‘quantile’ and it is recommended to keep the quantile between 0 and .5. Larger quantile values creates less clusters and smaller quantile value creates more clusters
2. Agglomerative Clustering: Agglomerative clustering works by building a hierarchy of clusters where each node is cluster. Each node also contains cluster of its daughter node. Each source is starts as its own cluster at the bottom and the higher up we go, the more sources are merged into one cluster. It requires the following two parameters:
    a. Numbers of Clusters: Number of clusters required.

b. Linkage: How the distances are calculated from one cluster to the other. The three possible values for linkage are:
    i. Single: distances between two closest points between each cluster
    ii. Average: returns the average distance between all points in one cluster to all the points in the other clusters
    iii. Complete: distances between two farthest points between each cluster

   More information on Agglomerative clustering can be fount at: https://medium.com/@codingpilot25/hierarchical-clustering-and-linkage-explained-in-simplest-way-eef1216f30c5
3. Spectral Clustering: Spectral Clustering works by solving a graph partitioning problem where each source is treated as node in the graph. It requires one parameter, which is the number of clusters required

3.2 Clustering Metrics

The code outputs two different clustering metrics:

1. Silhouette Coefficient: This is a value between -1 and 1 based on the inter and intra cluster distances. Value closer to 1 means there is good clustering where the distances between points within one cluster are much smaller than the distances between points between different clusters. Values closer to -1 indicates bad separation and the opposite is true
2. Davies-Bouldin Index: is the inter and intra cluster distance ratio with minimum possible value of 0. Smaller values indicate a better cluster separation.

More information on clustering metrics can be found here: https://towardsdatascience.com/three-performance-evaluation-metrics-of-clustering-when-ground-truth-labels-are-not-available-ee08cb3ff4fb

4 List of Files

Below is the description of the files and the **user only needs to modify / run the 2 bolded files**, which will call all the other relevant files

- **`config.yml`** – for specifying input parameters.
- **`Timeframe Anomaly Clustering.ipynb`** – creates a `TrafficShapeAnalysis` object and creates plots of the principal component value clusters. Main file that calls all the other files
- `TrafficShapeAnalysis.py` – filters incoming packets and sorts them by source; clusters the data once it is gathered and processed.
- `MVTSRunningFeatures.py` – handles the processing for incoming packets, including calculating features, interpolating the data, and performing dimensionality reduction techniques.
- `Feature.py` – a base class for features to be calculated for each packet.
    - `FeatureDataStringCosineSimilarity.py` – calculates the similarity between consecutive packets using cosine similarity.
    - `FeatureDataStringEntropy.py` – calculates the randomness of a packet using entropy.
    - `FeatureDataStringNewInformation.py` – finds the difference between consecutive packets using KL Divergence.
    - `FeaturePacketFrequency.py` – calculates the time difference between consecutive packets.

- `FeaturePayloadLength.py` – tracks the payload length of packets.
- `FeaturePortRelativeInterest.py` – tracks which destination ports are of interest.

5 Data Description

The code takes input data in the form of CSV files which were generated with `cpcap2csv.` These files are formatted such that each line represents a packet, and each field is:

- `saddr` – the source address, in decimal.
- `daddr` – the destination address, in decimal.
- `proto` – the protocol number (6 is TCP, 17 is UDP, 1 is ICMP).
- `sport` – the source port.
- `dport` – the destination port.
- `protchksum` – the layer-4 checksum.
- `iptotlen` – the length of the packet.
- `ipid` – the IP identifier.
- `ttl` – the time-to-live.
- `ts_date` – the packet timestamp in the format: `YYYY-MM-DD HH:mm:ss.SS.`
- `ts_epoch` – the packet timestamp measured from the epoch.
- `findx` – the index of the entry in the file table in the file from which this packet was read.
- `pindx` – the index into the packet file of the packet.
- `tcp_flags` – the TCP flags (0 if not TCP).
- `tcp_seq` – the TCP sequence number (0 if not TCP).
- `tcp_ack` – the TCP acknowledgement number (0 if not TCP).
- `tcp_win` – the TCP window size (0 if not TCP).
- `tcp_off` – the offset to the TCP data segment (0 if not TCP).
- `ip_ihl` – the IP header length.

Below is an example of the first few lines of such a CSV file.

```
1509467500,2654945451,6,46336,37207,38179,40,6326,245,2022-09-01 00:00:06.750562,1662004806.750561953,0,0,2,684575050,0,1024,5,5
2892600226,2654980309,6,34247,6002,4257,44,58314,241,2022-09-01 00:00:06.750563,1662004806.750562906,0,1,2,46021741,0,1024,6,5
1509467500,2654979587,6,46336,37613,18045,40,5239,245,2022-09-01 00:00:06.751428,1662004806.751427889,0,2,2,2033598617,0,1024,5,5
1699009283,2654948141,6,8088,40000,10484,44,23883,106,2022-09-01 00:00:06.753698,1662004806.753698111,0,3,2,2893837080,0,29200,6,5
1509467500,2654978542,6,46336,36048,40443,40,50491,245,2022-09-01 00:00:06.753699,1662004806.753699064,0,4,2,3833432069,0,1024,5,5
2892600217,2654948703,6,50013,9443,19901,44,51520,241,2022-09-01 00:00:06.754531,1662004806.754530907,0,5,2,2441781500,0,1024,6,5
2892600153,2654947512,6,50269,9954,54076,44,54321,241,2022-09-01 00:00:06.754532,1662004806.754532099,0,6,2,273719967,0,65535,6,5
908530134,2654948371,6,60000,22267,29945,40,52593,243,2022-09-01 00:00:06.757755,1662004806.757755041,0,7,2,3953383321,0,1024,5,5
2095761271,2654946164,6,35426,23,24909,40,47814,47,2022-09-01 00:00:06.757756,1662004806.757755995,0,8,2,2654946164,0,48210,5,5
3730698443,3221342855,6,11434,23,38491,40,51160,47,2022-09-01 00:00:06.760551,1662004806.760550976,0,9,2,3221342855,0,2186,5,5
1317040542,2654949154,6,43807,1411,47471,40,62717,240,2022-09-01 00:00:06.762365,1662004806.762365103,0,10,2,3589520988,0,1024,5,5
2421599656,2654980665,6,56987,30006,41599,44,54321,240,2022-09-01 00:00:06.764606,1662004806.764605999,0,11,2,260172763,0,65535,6,5
3742305278,2654946010,6,14538,23,44246,40,17510,47,2022-09-01 00:00:06.764607,1662004806.764606953,0,12,2,2654946010,0,7402,5,5
1547683205,2153349484,6,52039,31701,12732,40,37,246,2022-09-01 00:00:06.765188,1662004806.765187979,0,13,2,350419612,0,1024,5,5
2818215053,2654945833,6,36468,8080,18149,44,24498,243,2022-09-01 00:00:06.769355,1662004806.769355059,0,14,2,3947915783,0,14600,6,5
88053627,2654979997,6,7549,7549,32532,40,54321,243,2022-09-01 00:00:06.769356,1662004806.769356012,0,15,2,2367841818,0,65535,5,5
2727247193,2654945803,6,35550,20512,46005,44,9929,41,2022-09-01 00:00:06.770525,1662004806.770524979,0,16,2,2483843379,0,1024,6,5
2807990926,2153349393,6,22076,8448,12660,44,58409,41,2022-09-01 00:00:06.772981,1662004806.772980928,0,17,2,3664371124,0,1024,6,5
2421599570,2654949216,6,49316,30005,34946,44,54321,247,2022-09-01 00:00:06.772982,1662004806.772981882,0,18,2,1350262278,0,65535,6,5
2047137840,2654947447,6,14718,23,13381,40,46194,46,2022-09-01 00:00:06.772983,1662004806.772983074,0,19,2,2654947447,0,8806,5,5
2892600141,2654945594,6,53714,22460,62041,44,54321,241,2022-09-01 00:00:06.774654,1662004806.774653912,0,20,2,3155927281,0,65535,6,5
1542485749,2654980727,6,51406,4132,61600,40,21996,242,2022-09-01 00:00:06.777180,1662004806.777179956,0,21,2,385060799,0,1024,5,5
2892600176,2654947145,6,51110,9908,48324,44,54321,241,2022-09-01 00:00:06.781356,1662004806.781356096,0,22,2,3670731993,0,65535,6,5
866008546,2654980557,6,5599,23,55325,40,15520,53,2022-09-01 00:00:06.781357,1662004806.781357050,0,23,2,2654980557,0,32820,5,5
```

6 Running the Code

6.1 Setting the Parameters

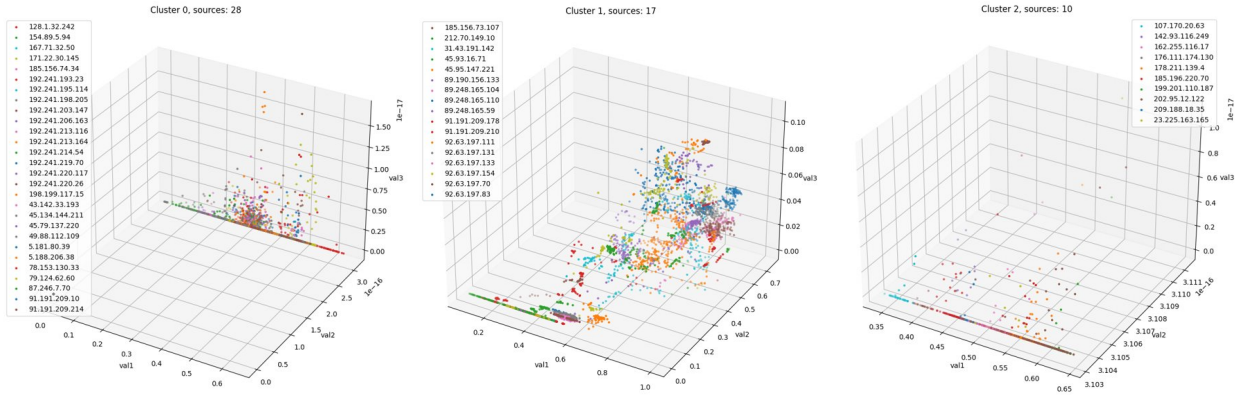
In the parameter file **`config.yml`**, specify the paths to CSV files, the CSV filenames, and the output filename. Specify the destination subnet, clustering type with parameters, and traffic type, which packets will be filtered by. Specify the max states and interpolation number, which are the maximum number of packets stored in memory and the number of packets to be used for interpolation. Lastly, define user preferences for running seeing intermediate outputs, saving figures, and saving the calculated parameters.

6.2 Running the Analysis

Run the **'Timeframe Anomaly Clustering.ipynb'** that will take all the parameters from config.yml
This will:

- Compute or load pre-computed features
- Run clustering
- Plot the clusters
- Print statistics about the clusters

Note that each cluster can be visualized in 3D using the first three principal component values however there are more. These PC values do not have any meaning, but the 3 that are plotted hold the most meaningful parts of the original data. The clusters may not always look separated when only 3 of the values are plotted

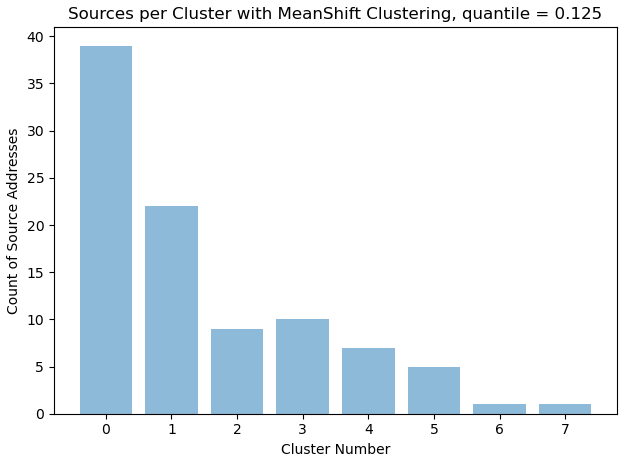


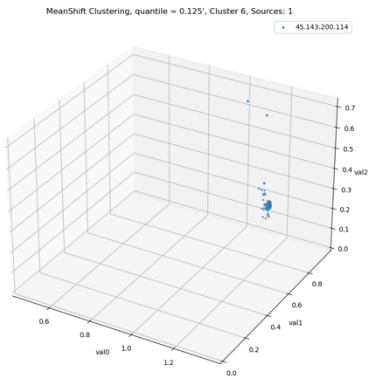
7 Interpreting the Results

This code utilizes multiple ways to detect anomalies

7.1 Number of Sources in a Cluster

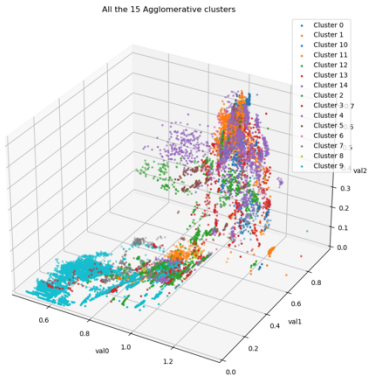
Sources that are similar will group together in same cluster. Any sources that do not group with other clusters and are in their own clusters can be anomalous. This can be seen in the code by looking at the plots for number of sources per cluster or individual cluster plots as shown below.





7.2 How Clusters Overlap

A plot (see example below) shows all the different clusters plotted on the same graph in different colors. Any clusters that look more separated from the other clusters (e.g. the light blue one here) may be anomalous. Please note that sometimes clusters may all look overlapping, that is because we can only plot 3 of all the features that are used for separating the clusters



7.3 Distances between Cluster Centers

This is done by taking all the points (packets) in a cluster and computing their center (average). Then the distance from each cluster center is computed to all the other cluster centers. The results look like the table below:

	0	1	2	3	4	5	6	7
0	0.00	0.76	0.38	0.47	1.00	0.98	1.12	0.81
1	0.76	0.00	0.55	0.38	0.29	0.52	0.79	0.58
2	0.38	0.55	0.00	0.26	0.78	0.71	0.88	0.65
3	0.47	0.38	0.26	0.00	0.63	0.56	0.75	0.48
4	1.00	0.29	0.78	0.63	0.00	0.69	0.95	0.81

5	0.98	0.52	0.71	0.56	0.69	0.00	0.30	0.32
6	1.12	0.79	0.88	0.75	0.95	0.30	0.00	0.38
7	0.81	0.58	0.65	0.48	0.81	0.32	0.38	0.00

Here the diagonal values are 0 since the distance from a cluster center to itself is 0. For any cluster that have distances to other clusters much larger than others, could be anomalous.

The code also outputs the average distance from each cluster to other clusters by averaging each row in the table above, not counting the 0. Any cluster with a very high distance to other clusters can be anomalous

```
Cluster:  0  Average Distance to Other Clusters:  0.7880152863987087
Cluster:  1  Average Distance to Other Clusters:  0.5547307682556666
Cluster:  2  Average Distance to Other Clusters:  0.602851015644925
Cluster:  3  Average Distance to Other Clusters:  0.5044554716226052
Cluster:  4  Average Distance to Other Clusters:  0.7367039760053636
Cluster:  5  Average Distance to Other Clusters:  0.5833974560858931
Cluster:  6  Average Distance to Other Clusters:  0.7409750757182839
Cluster:  7  Average Distance to Other Clusters:  0.5751807711444057
```

7.4 Distances between Cluster Points

Lastly, the distances between the points of one cluster to other clusters are computed. If a cluster has too many points for each source, than the max number of limit_val (in config file) packets are used. cluster. Here the diagonal values are not 0 because these are distances between points in the same cluster. Ideally, the distances within each cluster should be smaller than the distances between points of two different clusters. Here also if the distances between one cluster and others are large, than that may be an anomalous cluster.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0.36	0.95	0.49	0.63	1.07	1.03	1.15	0.89
1	0.95	0.64	0.73	0.72	0.6	0.69	0.85	0.78
2	0.49	0.73	0.26	0.49	0.87	0.79	0.92	0.74
3	0.63	0.72	0.49	0.57	0.81	0.76	0.87	0.71
4	1.07	0.6	0.87	0.81	0.41	0.78	1	0.91
5	1.03	0.69	0.79	0.76	0.78	0.38	0.4	0.53
6	1.15	0.85	0.92	0.87	1	0.4	0.13	0.51
7	0.89	0.78	0.74	0.71	0.91	0.53	0.51	0.43