Analysis of sequential feature engineering and statistical features for malware behavior discovery

Mikhail Epifanov , Azqa Nadeem , and Sicco Verwer

Computer Science, Delft University of Technology

Abstract

Malware Packet-sequence Clustering and Analysis (MalPaCA) is a unsupervised clustering application for malicious network behavior, it currently uses solely sequential features to characterize network behavior. In this paper an extensive comparison between those features and statistical features is performed. During the comparison a better clustering performance achievable with statistical features for longer connection sequences is shown and advice on which features can be added to MalPaCA.

1 Introduction

Discovering malware behavior and labeling it correctly is currently a hard task to perform since it is mostly done manually which is a highly intensive and time consuming task. In, [7], statistical network flow analysis is used to detect malware families with a high success rate. However for this to succeed, the author had to use samples and labels from VirusTotal and Anubis which, even when using a tool such as AVClass only provides an accuracy of 93.9% to 62.3%. [6]

The importance of ground truth labels is made clear in [3], when accurate labels are available, they can be used to train future Machine Learning models to detect and stop malware on the network layer instead of application layer.

The automated clustering of malware behavior has already been done by the tool MalPaCA to a degree of success, the tool is able to correctly cluster malicious network behavior with an error rate of 8.3% [5]. This is achieved by using only the following properties: package size, interval, source port, and destination port. The current solution however is not complete and might benefit from extra properties to capture more complex network behavior.

The tool in its current state has some short comings. The use of properties like 'source port' adds a lot of noise to the clustering since there will never be an sequential overlap due to the random nature that the operating system uses for this property. By using an statistical approach the amount of unique source ports solves this problem.

Statistical features are also efficient to calculate, while dynamic time warping and N-grams, used sequential features, can be both be computationally expensive and memory heavy operations limiting the input size or sequence length.

By performing an comparative analysis between sequential features, and statistical features. This will show if certain statistical features can improve the explainability of a cluster, its accuracy in identifying/clustering a certain malware behavior, or better performance for longer sequence lengths.

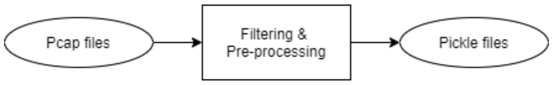


Figure 1: Data set pre-processing

2 Data Set

The IoT-23 [2] data set is used for this research. The data set contains the unfiltered pre-labeled internet traffic packets (PCAP) from 20 infected Internet of Things (IoT) devices and 3 non-infected devices. The captures were done over bigger periods of time

Labels	S-20	S-25	S-30	S-35	S-50	S-75
Benign	2329	2025	1818	1666	1399	1101
Attack	426	416	412	385	305	246
C&C	207	207	199	199	196	191
C&C-FileDownload	44	38	41	38	34	32
C&C-HeartBeat	58	58	58	59	56	53
C&C-HeartBeat-Attack	15	12	9	9	11	9
C&C-HeartBeat-FileDownload	4	7	8	6	5	9
C&C-Torii	40	40	40	40	40	40
DDoS	115	113	112	112	111	110
FileDownload	15	11	9	8	5	3
Okiru	17	5	3	3	1	-
PartOfAHorizontalPortScan	402	330	244	228	224	224
Total	3672	3262	2953	2754	2387	2018

Table 1: Average connections per label per sequence length

which gives a good insight in a real world scenario.

2.1 Filtering & Pre-processing

From the PCAP files the flows which are defined as an stream of packets from an source IP to and destination IP, uni-directional connections were used meaning that incoming packets of a flow and outgoing packets will be split into two separate flows.

Since MalPaCA requires an minimum of 20 packets, any flow with less than 20 packets will be ignored since it cannot be used for the sequential features extraction, only the first 5000 packets will be saved, since any more than that are usually DDoS packets which do not contain any extra info. Also only the source port, destination port, package length, package timestamp and malicious label are required, so all other information will be discarded from the data set when converting a PCAP file to a pickle file.

After performing this action, Figure 1, a mere 60MB of data is left instead of the original 100GB.

3 Methodology

3.1 Background

The base version of MalPaCA operates by utilizing sequential features from a connection and comparing them using dynamic time warping (DTW), this operation is also done for the ports by first constructing N-grams from the port sequence. To make the compare as fair as possible, all base features, "Sequential feature generation", "Distance measurements" and "HDBScan clustering" from Figure 2 are left unchanged, these can be found in chapters 4.2, 4.3 and 4.4 of "Beyond Labeling" respectively.

3.2 Random subset selection

A connection is a fixed amount of packets from a flow, since MalPaCA is only able to process a limited number of connections, not all data from the previous step can be used. From each file of the data set 200 connections per label were extracted, e.g. benign, DDoS.

The proportion of benign/malicious behavior is not represented by a real world scenario, but since the focus mostly lies on identifying different network behavior and being able to detect malicious clusters, this was an compromise that had to be made, to get an as big of set of different (malicious) behaviors.

3.3 Statistical features generation

The statistical features were selected from "BotMark" [8], a summary of each feature and its origin can be found there.
Since MalPaCA uses uni-directional connection, and has a fixed amount of packets per connection some features presented in "BotMark" are not relevant. To account for the ports information and account for better distances three properties were added, the total amount of unique source ports (USP), the total amount of unique destination ports (UDP), and the amount of common ports (for IoT devices: 25, 53, 80, 119, 123, 143, 161, 443, and 5353). These ports were selected to better

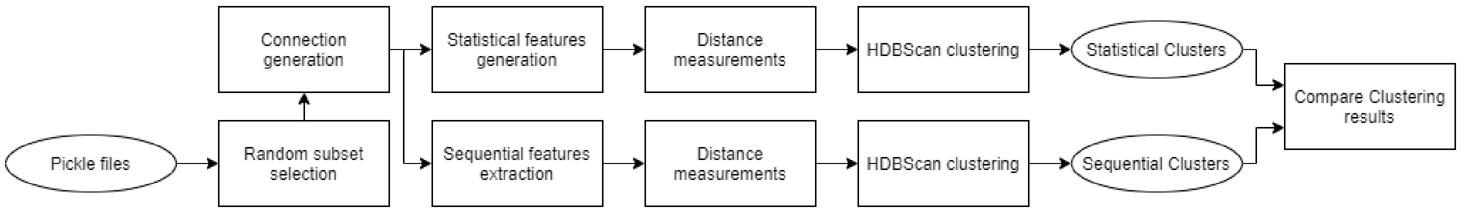


Figure 2: Data flow for comparing statistical features with sequential features

identify a connection. Benign behavior has a highly likely hood to contain exclusively common ports.

All features used can be seen in Table 2. Each of these features was calculated for a connection, the 2.5th percentile was used as lower bound and the 97.5th percentile as upper bound per feature and then normalized to ensure that each feature accounts equal for the Euclidean distance.

4 Experimental Setup

4.1 Data selection

For the experiment the following selection of connection per labels were made: Table 1. This selection was done using a sliding window with the correct sequence length over all connections available using the method described in subsection 3.2 and as can be seen in the code [1].

The only variable which was changed, with the increasing growth sequence length. was the 'min_cluster_size' and 'min_samples' for the HDBScan algorithm, which both were set to equal $\frac{1}{100}$th of the total connections.

4.2 Compare clustering results

The clusters $C_{0\dots n}$ generated by the statistical features and sequential features will be compared based on the following measurements, where set c is the set containing all connections:

- Noise score
- Average cluster size & count
- Silhouette coefficient
- Cluster purity
- Cluster malicious purity

4.2.1 Noise score

The clustering algorithm allocates points which have no clear cluster to a noise cluster C_n, meaning those points did not get clustered and will be disregarded in the calculations. To accommodate for this the amount of points that did not get placed in the noise cluster is tracked. This is calculated as follows:

$$\text{Noise score} = \frac{|c| - |C_n|}{|c|}$$

4.2.2 Average cluster size & count

MalPaCA clusters network behaviors, since there is a limited amount of different behaviors, especially in this data set, the cluster size should be maximized and the cluster count minimized to the amount of behaviors that there are expected to be in the data set. The noise cluster is excluded from this count. For this the following metric will be used:

$$\text{Average cluster size} = \frac{\sum_{i=0}^{i=m} |C_i|}{|C| - 1} * 100$$
$$\text{Cluster count} = |C| - 1$$

4.2.3 Silhouette coefficient

Silhouette coefficient or index is a metric to validate the consistency of clusters. It measures an connections cohesion to its own cluster and separation to other clusters. The index ranges from -1 to 1 with a high score meaning an cluster is dense and has a good separation to other clusters.
To calculate the Silhouette index for connection $i \in C_i$ the following equation was used:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$
$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$
$$s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \Leftrightarrow |C_i| > 1$$

A score between -1 and 0 should never be possible with HDBScan because any points that could lead

Feature	Description	Source
NSP	Number of small packets (len of 63 - 400 bytes)	[8]
AIT	Average arrival time of packets	[8]
TBT	Total number of transmitted bytes	[8]
APL	Average payload packet length for time interval	[8]
PV	Standard deviation of payload packet length	[8]
DPL	The total of number of different packet sizes	[8]
MX	Size of largest package	[8]
MP	The number of maximum packets	[8]
PPS	Number of packets per second	[8]
BPS	Average bits-per-second	[8]
USP	Total number of unique Source ports	-
UDP	Total number of unique Destination ports	-
CP	Common ports in Source and Destination ports	-

Table 2: Statistical Feature List

to such a score are discarded to the noise cluster, therefor only the range of 0 to 1 will be used and the average over all clusters will be taken.

4.2.4 Cluster purity

Since the primary objective of MalPaCA is to distinguish malicious from benign network behavior, the purity of a cluster must be calculated. A cluster is pure if either all connections within are benign or malicious.
Let p_i^m be the percentage of connections with a malicious label in cluster C_i, then the cluster purity is defined as followed:

$$\text{Cluster purity} = \frac{|p_i^m - 0.5|}{0.5}$$

The result of this metric is not a perfect measurement for behavior discovery, since benign data is not fully labeled, e.g. benign file-download behavior will be labeled with malicious file-download behavior which in practice would be a good behavioral cluster and it also relies on labels only containing one behavior, which is not true for labels like 'C&C-HeartBeat-FileDownload' and thus is a multi-class problem.

4.2.5 Cluster malicious purity

To improve on the shortcomings of the Cluster purity metric, the purity of malicious cluster are also taken into account, which are clusters with more than 60% malicious connections. This is done to check if all those malicious connections have the same label, since its highly likely that a connection with the same label executes the same network behavior.
Let $c_i^{m_k}$ be the amount of connections with a malicious label k in cluster C_i and K a set containing all malicious labels:

$$\text{Cluster malicious purity} = \frac{\max c_i^{m_k} \forall k \in K}{|c_i^k|}$$

4.3 Running experiments

The experiments were run with different sequence lengths (S) and executed 100 times to ensure the result was significant and not dependant on the random subset selection of connections.

5 Results

The arrows indicates wherever the value should be maximize or minimize.

- S-20 results: Table 3
- S-20 Sequential Clusters: Figure 3
- S-20 Statistical Clusters: Figure 3
- S-25 results: Table 4
- S-30 results: Table 5
- S-35 results: Table 6
- S-35 Sequential Clusters: Figure 5
- S-35 Statistical Clusters: Figure 5
- S-50 results: Table 7
- S-75 results: Table 8

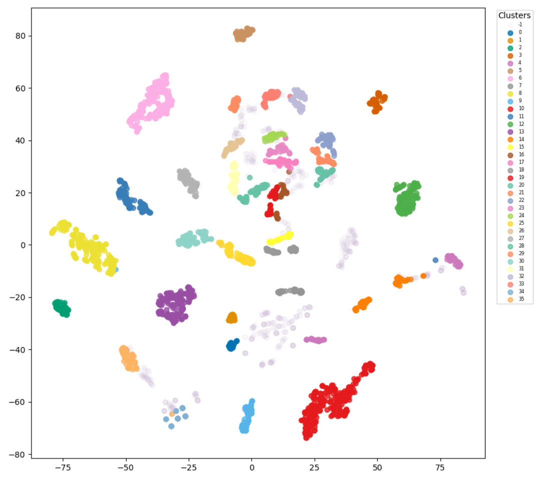


Figure 3: Sequential result (S-20)

Metric	Sequential	Statistical
↓Number of clusters	37.95	32.58
↑Average cluster size	2.0513	2.1145
↑Noise score	0.77776	0.68753
↑Silhouette Index	0.63957	0.76883
↑Purity	0.85634	0.86133
↑Malicious purity	0.89316	0.86141

Table 3: Results S-20

6 Responsible Research

This research has minor ethical impact, it can be solely used to improve the performance of (not limited to) MalPaCA. Which in terms will be used to identify malicious behavior to identify bad actors in a network.

The reproducibility of this research was an very important aspect during the research, therefor a lot of steps were taken to make it as reproducible as possible. The data set used, is open-source [2], which should be accessible for a long time to come and all source code has been published on Github [1].

All (performance) benchmarks were performed on the authors local computer and with the help of the Numba python library [4] executed within an hour with the IoT-23 data set, meaning that anyone should be able to reproduce the result without needing access to a powerful server.

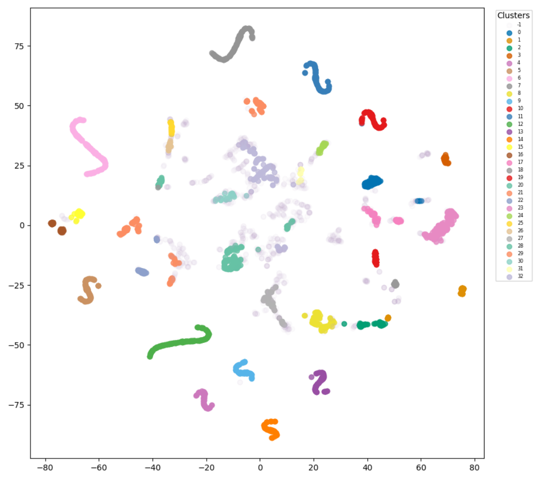


Figure 4: Statistical result (S-20)

Metric	Sequential	Statistical
↓Number of clusters	39.22	36.4
↑Average cluster size	1.9415	1.8912
↑Noise score	0.76033	0.68648
↑Silhouette Index	0.6282	0.76968
↑Purity	0.8653	0.86872
↑Malicious purity	0.9063	0.91626

Table 4: Results S-25

7 Discussion

The statistical features give quite a good representation if a cluster is good or bad, in case a cluster is good, it is more likely to see all points within a small range and very little outliers, as can be seen in Figure 10. However bad clusters can also be detected. There will be a lot of outliers and variance in many features, as can be seen in Figure 11. While when using sequential features this cluster probably wouldn't have been formed since in the sequential feature a clear difference can be seen Figure 12 demonstrates this clearly, there, outlined in red, benign behavior was clustered together with C&C-Torii, outlined in green. However the ability to see this quickly and clearly does give extra insight into a cluster, and its also clear to understand why these certain connections were clustered together, that's because it nearly a perfect match on features like TBT, APL, DPL, BPS UDP, and CP.

The next thing which shows a big difference is the Silhouette coefficient. Figure 8 shows a

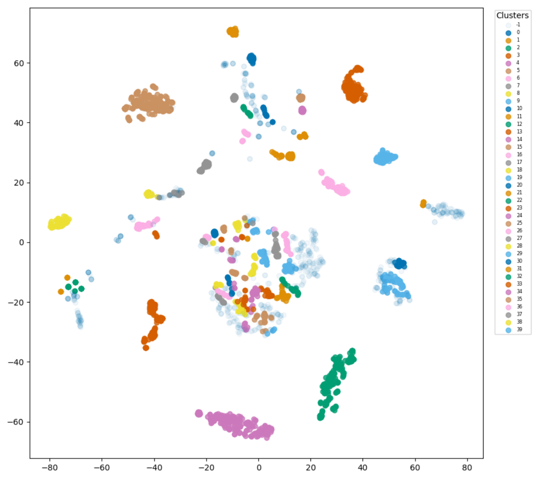


Figure 5: Sequential result (S-35)

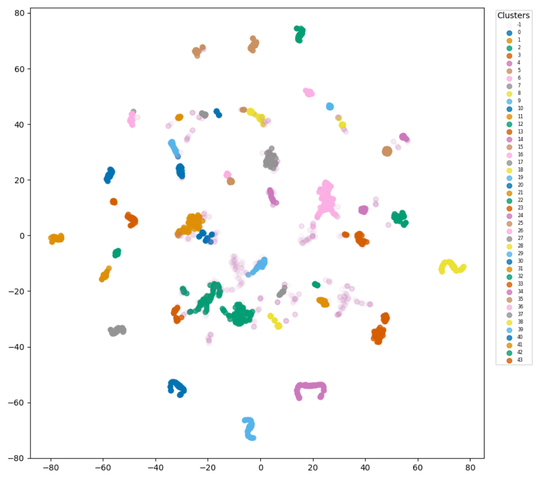


Figure 6: Statistical result (S-35)

Metric	Sequential	Statistical
↓Number of clusters	38.79	37.57
↑Average cluster size	1.9373	1.7135
↑Noise score	0.7508	0.64237
↑Silhouette Index	0.61049	0.74372
↑Purity	0.86427	0.86844
↑Malicious purity	0.91732	0.93366

Table 5: Results S-30

Metric	Sequential	Statistical
↓Number of clusters	37.55	41.78
↑Average cluster size	1.9225	1.6013
↑Noise score	0.7211	0.66805
↑Silhouette Index	0.60309	0.76829
↑Purity	0.85654	0.89404
↑Malicious purity	0.91903	0.91425

Table 6: Results S-35

clear distinction between Sequential and Statistical clusters, my belief is that this discrepancy is caused by the 'source port' feature of the sequential feature set. This causes each nearly identical connection to have a huge distance due to the cohesion part of the silhouette coefficient, statistical features do not suffer of this and have a big gain (on average about 0.2) over all sequence lengths.

Another interesting pattern was the stable noise score which was achieved with the statistical features, however the noise is high at lower sequence lengths, there seems to be an big increase in noise when using longer sequence lengths for sequential features as can be seen in Figure 7. From this can be said that with the nearly same scores in Purity and Malicious purity seen in Table 3 sequential features seem superior with short sequence lengths but are around equal at the point when the sequence length reaches 35. There was no substantial difference in purity and malicious purity discovered from those experiments, indicating that the clusters are the same quality.

An attempt at comparing the clusters generated by the sequential features and statistical features was made, but this yielded to good results. On average only 25% of the noise clusters was overlapping meaning that both methods used quite different subsets of connections from the available ones, leading to quite distinct clusters except an occasionally overlapping cluster (Containing mostly benign data).

The generation and clustering of statistical features were also around 10 to 50 times faster than the sequential features, however part of this might be to partially to blame to code optimizations. On average the statistical features took between 0.5-3 seconds to generate and sequential between 20-35 seconds dependant on sequence length. However both algorithms are $O(n)$ and the biggest bottleneck will always be the distance matrix calculation which is $O(n^2)$.

Metric	Sequential	Statistical
↓Number of clusters	32.23	43.88
↑Average cluster size	1.9678	1.5581
↑Noise score	0.63206	0.68279
↑Silhouette Index	0.50182	0.74882
↑Purity	0.83776	0.88695
↑Malicious purity	0.9266	0.9266

Table 7: Results S-50

Metric	Sequential	Statistical
↓Number of clusters	30.43	40.69
↑Average cluster size	1.9944	1.6469
↑Noise score	0.60462	0.6687
↑Silhouette Index	0.46494	0.71629
↑Purity	0.83475	0.86692
↑Malicious purity	0.9332	0.91982

Table 8: Results S-75

8 Conclusions & Future Work

From the discussion the following points can be concluded, a quick overview can be found in Figure 9:

- Both sequential and statistical features generate clusters which are distinguishable network behaviors, however the labels provided are not precious enough to identify the actual network behavior.

- For larger sequential lengths ($S \geq 45$) statistical features perform better than sequential features. There was an significant change in the noise score while Silhouette index, purity and malicious purity stayed on par.

- There is an considerable reduction of memory usage when using statistical features compared to sequential, since each connection can be reduced to one float/int for each feature.

- MalPaCA might benefit from removing the source/destination port to increase the cluster cohesion and replacing it with and statistical feature like USP or UDP. CP is currently handcrafted for the current data set and might not be useful in general data sets, future research required.

Possible points for future research:

- There was a nearly 10% variance in cluster performance from run to run, as can be seen in Figure 13. This was seen in all scores since they are very intertwined with each other. This might mean that the performance of the two feature sets in question might depend on the data set used.

- There is a big difference in the number of clusters & cluster size between the two methods, leading to the question if different clustering parameters might improve the performance of either statistical or sequential features.

- The validation/error score is currently not a good indication on the clustering validity, there is still a lot of manual labor involved in

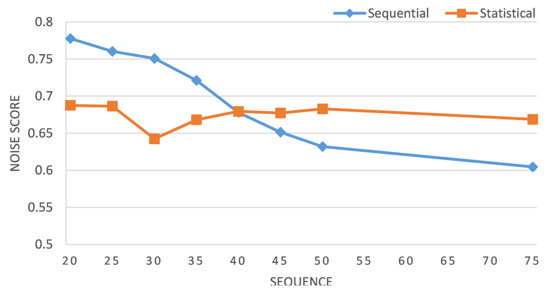


Figure 7: Noise score over sequence lengths

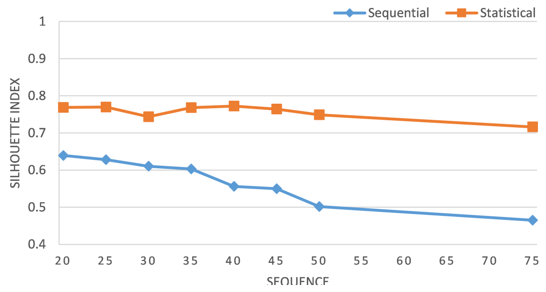
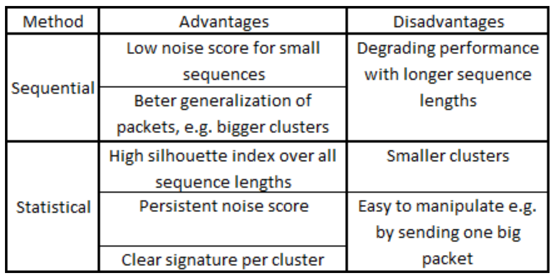


Figure 8: Silhouette index over sequence lengths



Method	Advantages	Disadvantages
Sequential	Low noise score for small sequences	Degrading performance with longer sequence lengths
	Beter generalization of packets, e.g. bigger clusters	
Statistical	High silhouette index over all sequence lengths	Smaller clusters
	Persistent noise score	Easy to manipulate e.g. by sending one big packet
	Clear signature per cluster	

Figure 9: Summary of each method

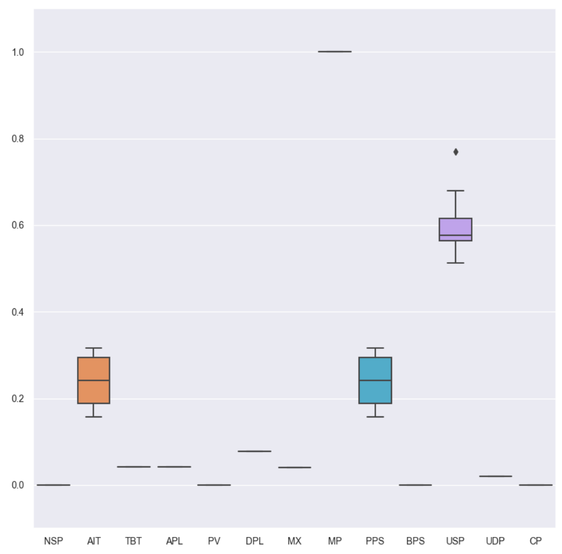


Figure 10: Box plot of good clustering result (Little variant in few features, little to non outliers)

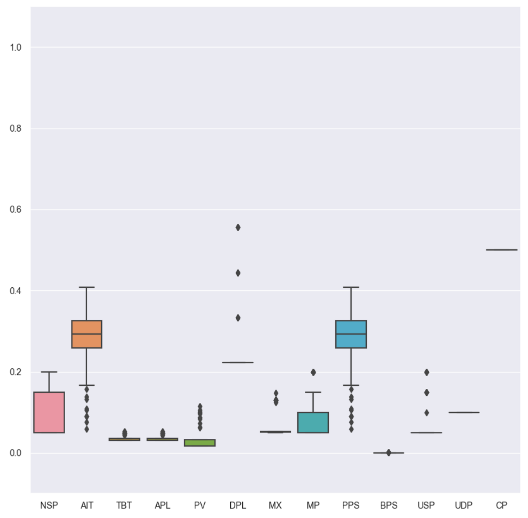


Figure 11: Box plot of bad clustering result (More variance in multiple features, many outliers)

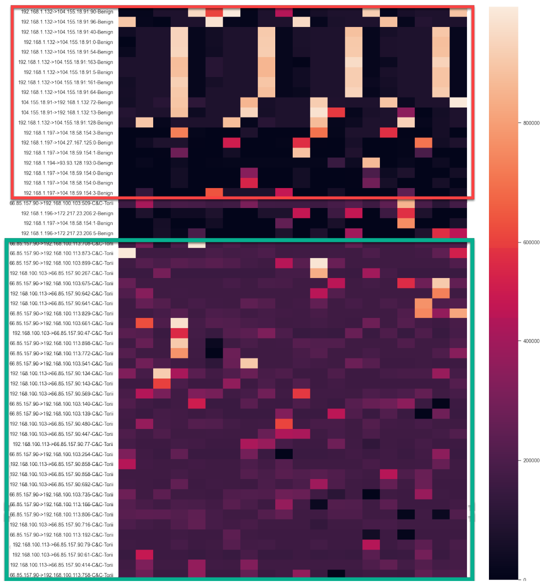


Figure 12: Sequence of bad statistical cluster Two distinct behaviours

verifying whenever or not a cluster is good.

- Optimizations need to be made since calculating the distance metric is an $O(n^2)$ problem which doesn't scale well with a big amount of connections for both methods.
- The labels are not precise enough, since to discover network behavior the labels need to be way simpler and not a collection of multiple behaviors. The benign data set should also preciously labeled so malicious network behaviors can be differentiated and validated with the same non-malicious behavior.
- Future research can also be conducted into whenever Deep Package Inspection labels can be used in combination with current labels to create a better validation set.

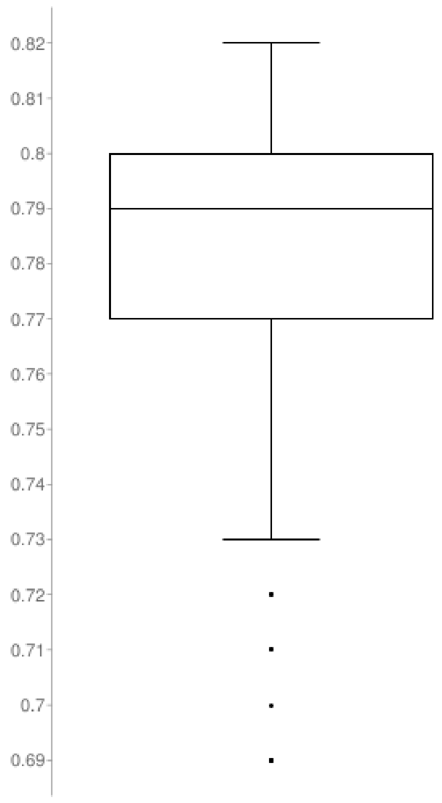


Figure 13: Variance over 100 runs in Noise Score for S-20 Sequential features

References

[1] Mikhail Epifanov. *Fork of tudelft-cda-lab/malpaca-pub*. original-date: 2021-04-29T10:40:12Z. Apr. 2021. URL: `https://github.com/Mikhail5555/malpaca-pub` (visited on 06/03/2021).

[2] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. eng. type: dataset. Jan. 2020. DOI: `10.5281/zenodo.4743746`. URL: `https://zenodo.org/record/4743746` (visited on 05/27/2021).

[3] Christian A. Hammerschmidt et al. "Reliable Machine Learning for Networking: Key Issues and Approaches". In: *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. ISSN: 0742-1303. Oct. 2017, pp. 167–170. DOI: `10.1109/LCN.2017.74`.

[4] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba: a LLVM-based Python JIT compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. New York, NY, USA: Association for Computing Machinery, Nov. 2015, pp. 1–6. ISBN: 978-1-4503-4005-2. DOI: `10.1145/2833157.2833162`. URL: `https://doi.org/10.1145/2833157.2833162` (visited on 06/03/2021).

[5] Azqa Nadeem et al. "Beyond Labeling: Using Clustering to Build Network Behavioral Profiles of Malware Families". en. In: *Malware Analysis Using Artificial Intelligence and Deep Learning*. Ed. by Mark Stamp, Mamoun Alazab, and Andrii Shalaginov. Cham: Springer International Publishing, 2021, pp. 381–409. ISBN: 978-3-030-62582-5. DOI: `10.1007/978-3-030-62582-5_15`. URL: `https://doi.org/10.1007/978-3-030-62582-5_15` (visited on 04/22/2021).

[6] Marcos Sebastián et al. "AVclass: A Tool for Massive Malware Labeling". en. In: *Research in Attacks, Intrusions, and Defenses*. Ed. by Fabian Monrose et al. Vol. 9854. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 230–253. ISBN: 978-3-319-45718-5. DOI: `10.1007/978-3-319-45719-2_11`. URL: `http://link.springer.com/10.1007/978-3-319-45719-2_11` (visited on 05/06/2021).

[7] Florian Tegeler et al. “BotFinder: finding bots in network traffic without deep packet inspection”. In: *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. CoNEXT ’12. New York, NY, USA: Association for Computing Machinery, Dec. 2012, pp. 349–360. ISBN: 978-1-4503-1775-7. DOI: `10 . 1145 / 2413176 . 2413217`. URL: `http://doi.org/10.1145/2413176.2413217` (visited on 04/22/2021).

[8] Wei Wang et al. “BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors”. en. In: *Information Sciences* 511 (Feb. 2020), pp. 284–296. ISSN: 0020-0255. DOI: `10.1016/j.ins.2019.09.024`. URL: `https://www.sciencedirect.com/science/article/pii/S0020025519308758` (visited on 05/14/2021).