MalPaCa Feature Combination: Which packet header features and combination thereof are the most generalizable, private and easy to extract to cluster malicious behavior?

Jonathan Garack , Sicco Verwer , Azqa Nadeem
TU Delft

MalPaCa is an unsupervised clustering tool, which the main purpose is to cluster unidirectional network connections based on network behavior. The clustering is only based on non-intrusive (private) packet features such as transport and network header fields, and thus it has a strong potential use-case. This paper focuses on feature extraction and finding the best combinations that provide best clustering results. The features should be generalizable to a wide range of malware families and follow an easy extraction process. To expand the research one additional packet-based feature is found, TCP flags, as well different variants of previously extracted features were employed, which improves the efficacy of the tool. Finally, a grid search is performed to determine the best combination of the features.

1 Introduction

In the field of malware detection, machine-learning-based approaches have been used for many years. Given the increasing dependence of data processes on network devices and the evermore emerging Internet of Things (IoT) devices, the protection of these systems against malware becomes of crucial importance. Networks that are affected by malware can often result in third party information-sharing or harm devices and consequently result in financial or other damages.
There are two approaches to detect malicious behavior in a network, one respecting the device's privacy more than the other. Deep Packet Inspection algorithms (DPI), which is considered the privacy-intrusive method and has a significant drawback as it has a limited domain of uses. In contrast, the other group is more privacy-friendly, which in turn makes the feature extraction more complex, as the available data is constrained.
MalPaCa was created to tackle this problem. This algorithm clusters unidirectional connections into groups that present similar network behavior. It does so, solely based on the available information in the Network or Transport protocols[10]. A significant advantage of MalPaCa is that it uses sequential features, instead of statistical ones for the input. This way the algorithm can discover malware behavior based on the temporal nature of a network connection. Although it is a novel approach to extracting data from network traffic, some similar use cases can be found in the literature. For example, Lin et. al [8] use the sequential features for securing cyber-physical systems. Conti et. al [2] were able to distinguish different actions performed by the mobile app just on the basis of encrypted network traffic capture.
The current implementation of the algorithm uses four features as its input (destination port, source port, time intervals, and packet size). The existing approach to Malware detection is based on statistical characteristics, therefore a critical and not yet well-explored area of research relates to sequential feature extraction from a network capture. In particular, this relates to the MalPaCa algorithm. Not all of the features which can be iterated upon were investigated. Research is required to determine the most general and easiest to extract features, that yield the lowest clustering error.
This paper will try to provide an answer to the central question, which reads as follows: "Which packet header features and a combination thereof are the most generalizable, private, and easy to extract to cluster malicious behavior?". Firstly in section 2, a short description of the MalPaca algorithm is be provided. Later the methodology that guides the research will be presented in section 3. The method is split into 3 subsections, data pre-processing, feature extraction, and feature combination. Section 4 will focus on the description of the malware that was captured in the dataset, and present the features taken into consideration. Part 5 will elaborate on the results of the grid search and substantiate the choice of the added feature. Last but not least, a discussion of the results will be provided and concluded with a brief summary of the findings and recommendations for future research.

2 Background

Malpaca[10]. is an unsupervised machine learning-based instrument that is able to cluster a network traffic capture into groups of similarly behaving connections. It focuses on unidirectional connections defined by source and destination IP addresses. The features are sequential whereby the metrics used to define the distance between two sequences are dynamic time warping and ngrams. The first measure is used when the sequence contains quantitative data, whereas the latter when qualitative data is employed. A python implementation of this algorithm already exists and it uses the destination port, source port, time intervals, and the packet size as its features. To cluster, the data HDBSCAN is used.

3 Methodology

3.1 Data preparation

The dataset used in this paper is IoT-23 [4]. It contains 23 malicious network captures, each one being a unique scenario, which comprise 20 malware and 3 benign network captures. The botnets captured in the traffic are Mirai, Okiru, Torii, Hakai, Gagfyt, IRC bots. Each botnet is described in detail in section 4.1. Additionally, each connection in the scenario was split into flows and labeled using the Zeek network analyzer. It is important to mention that MalPaca does not consider flows, only uni-directional connections. So each connection is assigned to a label based on the IP addresses and port numbers, therefore it might be possible that a connection has more than one label assigned to it.

To manage the size of the network captures, filtering is applied to the dataset. Let c_i be an arbitrary unidirectional connection consisting of packets p_{i_j}. Each uni-directional connection in the traffic capture is kept if the amount of $p_{i_j} \in c_i$ is greater than 20. This is the current standard threshold ($T = 20$) to qualify as a datapoint for MalPaca clustering. The connections such that $c_i > T$, are trimmed and only the first T packets are kept. This approach significantly shortens the size of the data,, which allows to process each scenario.

To capture the different behaviors of botnets that occurred later in the connection or to create more instances of the considered malware behavior, an additional step is taken. For each connection c_i with more than T packets, ranging between $[p_{i_kT}, p_{i_{(k+1)T}}]$, where k is an arbitrary positive number, new connections are established. For example, if a connection has 40 packets, 2 connections are created out of it. In appendix A table 7 a detailed overview of the used connections is presented.

From each scenario no more than 200 malicious connections with the same label are extracted (horizontal port scans are from different scenarios). This allows to retain a realistic proportion between malicious and benign connections. The distribution of different labels can be seen in table 1 below. The final sample of the dataset consists of 2059 connections, of which 1047 are malicious and 1012 benign.

Malware Label	Number Of Connections[1]
Benign	1012
Attack	39
HeartBeat-FileDownload:	4
Okiru	35
C&C-FileDownload	2
C&C	26
HorizontalPortScan	816
DDoS	70
Torii	40
FileDownload	2
C&C-HeartBeat	15

Table 1: Label distribution in dataset

[1] the number is the sum of the unidirectional connections, (requests + responses)

3.2 Feature Selection

A critical part of this research paper is the appropriate selection of features. The chosen attributes cannot be privacy intrusive, which limits the domain to only the transport and network protocol headers. To create the best set of variables to cluster the malware in the IoT-23 dataset several steps are taken.

Firstly each scenario from the dataset is studied with the help of the previous analysis available on the main website of the dataset. Next, each malware is described on the basis of its actions and divided into sets of common behaviors. To gather the relevant information several articles and blogs were examined, and in addition, the network traffic was analyzed in WireShark.

As already mentioned in the background section, the MalPaca algorithm in its default version uses 4 basic attributes: packet size, time interval, and source and destination port numbers.

3.3 Feature Combination

The final step of the research concerns the choice of the best combination of the studied variables. The attributes are combined using a weighted average of the distance between them. To find the weights that yield the best results, a grid search is performed. First, the Pearson and Spearman correlations are calculated for each pair of features. This shortens the domain of candidate variables. Under certain assumptions, described later in this paper, some features are seen having equal weights. After determining the final set of features, K_fold cross-validation is applied to the data to get the most reliable results. To assess the results of the grid search, an average of four metrics is used (the formulas for those metrics can be found in appendix D):

- **Cluster purity (CP)**: measures the extent to which a cluster contains a malicious class. 100% means it is either a fully benign or malicious cluster and 0 means that the amount of malicious and benign data points are equal.
- **Clustered data points (CDP)**: measures the percentage of clustered connections. The smaller the noise cluster, the larger the score.
- **Malicious purity (MP)**: measures the percentage of the most appearing malicious label in a cluster over the whole cluster size. It is only applied to clusters containing malicious labels.

4 Feature Extraction

To extract the best set of features a good understanding of the malware in the dataset needed to be achieved. This section will first describe the botnets from which traffic was captured in the IoT-23 scenarios and then focus on the determining the most common behaviors within.

4.1 Botnets in the dataset

In this section, a description of each botnet will be given. The description is based on the malware behavior catalogs, research articles, and cyber-security blogs, extended by a Wireshark inspection of the traffic.

4.1.1 Mirai

Mirai is a malware with the goal to infect a large amount of IoT devices. After creating a botnet, a bot-master can later perform large-scale DDoS attacks (the scale of the attacks is so heavy that it is measured in Tbit/s). It has a common pattern of behavior that can be described in a few steps[6]. To infect a vulnerable machine it performs a horizontal port scan to find vulnerable open ports. When it finds an open port it uses a set of hard-coded usernames and passwords to try and brute-force the IoT device to gain access to it. When a weakly secured device is discovered and infected, it starts reporting to the C&C server using various other ports. The C&C server communicates with the bot (infected device) to find new vulnerable machines to infect. An interesting feature of this malware is that it wipes out all of the existing malware from the infected device. The device also downloads a variety of executable files from the C&C and runs them on the machine. Upon gathering a significant amount of bots, the C&C server is able to give a command, which forces the botnet to perform one of many types of DDoS floods (HTTP, TCP, or others) under the specified address. The labels given by the Zeek analyzer to scenarios infected by this malware are C&C, DDoS, PartOfAHorizontalPortScan, C&C-FileDownload, FileDownload, C&C-HeartBeat.

4.1.2 Okiru

This malware was developed on the basis of Mirai. The behavior is similar to the above-described mechanism, but instead of focusing on basic CPU architectures, this botnet is able to target ARP processors, which makes this variant more dangerous (as almost 1.5 billion products per year are created with this CPU architecture [5]). The labels given by the Zeek analyzer for scenarios where the devices are infected by this malware are C&C-HeartBeat, Okiru, Okiru-Attack.

4.1.3 Torii

This Botnet differs from the previous two as it uses more advanced techniques to hide its existence. An interesting thing to notice in this botnet is that it does not perform a DDoS attack. Instead, it extracts sensitive information and performs different commands on the infected device. In summary, it takes virtually full control over the device. To hide its communication with the C&C server it uses the TCP port 443 (HTTPS) [7]. As this malware does not perform any type of attack, the only malware given by the Zeek analyzer is C&C-Torii (communication with the C&C server).

4.1.4 Hakai

Hakai is another botnet based on Mirai[1]. The IoT-23 dataset only captures its communication with the C&C server. This malware exploits ThinkPHP vulnerability, so its main focus is web servers built using this open-source framework.

4.1.5 Other botnets

Some botnes from the data set are omitted and are not described in this section. This was done because i) the network capture consists only of few long connections (and thus would not be captured by the MalPaCa tool) and ii) it was almost identical to other types of botnets in terms of behavior.

4.2 Most important behavior in the botnets

Based on the aforementioned malware description, the most interesting behavior is extracted and described in this section. The description in table 2 lays out the features best describing the given behavior. Additionally in appendix C heat maps (Figure 12-21) supporting the feature selection for each behavior are presented. The graphs backing up TCP flags are found in further section 5.2.1.

4.2.1 Horizontal port scan

This label is a type from the port scans family. Two main port scans can be recognized, vertical and horizontal. The first one focuses on a single device and scans for an open port. The latter one tries to exploit a particular service on many different devices, it is the most commonly used type of port scan [11]. This information-gaining technique is relatively old, and a lot of variations exist. The main protocol used for port scans is TCP. Most of the known port scan attacks use different kinds of combinations of the *TCP flags*. A single horizontal port scan attack is usually performed by a single machine. The latency, as well as the execution of the code, is therefore similar, which emphasizes the usefulness of the *Time interval* feature. Furthermore, it is noticed that in the connections between the adversary and the attacked device, the payload of the respective packets in each connection follows the same pattern (*packets size*). Finally, both *destination source ports* have a huge classifying potential for this behavior as horizontal port scans check the same port over different devices.

4.2.2 DDoS

This is one of the most known attacks that is performed to prevent legitimate use of service [9]. In the IoT-23 dataset,the main purpose of Okiru or Mirai is to perform this type of attack. There many different types of DDoS attacks. The attacks can be performed on different layers from the TCP/IP model. The ones captured in the IoT-23 dataset were UDP/TCP floods. The TCP attacks can be also divided into more groups based on the flags set in a TCP packet. In most cases when a DDoS flood is performed the packets sent to the attacked device are the same. Therefore the *packets size*, *Destination source port* and *TCP flags* are useful for categorizing and labelling of the behaviour.

4.2.3 File download

Usually, when a device is already infected by malware, files are downloaded, to be later executed. For example, Mirai-like botnets download executable files to wipe out coexisting malware on the infected device in order to be in full control. In the .labeled file created by the Zeek analyzer, there are 3 different types of labels for filedownload (C&C-FileDownload, FileDownload, C&C-HeartBeat-FileDownload). Although it is a common and important part of the attack, it was difficult to distinguish the most impactful features as in the IoT-23 data set there are too few connections labeled as filedownload.

4.2.4 C&C communication

This type of behavior varies depending on the malware. For example, the Torii botnet tries to cover this behavior by mim-

icking the HTTPS communication standard. The C&C communication has also a lot of different types, in the case of the Torii botnet, it's used for information gathering, however this communication is mainly kept to give further commands to the botnet. This behavior is best described by four features *Time intervals*, *packet size* and *TCP flags*.

Malicious behavior	Feature
1. Horizontal port scan	Time Interval
	TCP Flags
	Packet Size
	Dest. & Source port
2. DDoS	Packet Size
	Dest. & Source port
	TCP flags
3. File download	—
	—
4. C&C	Time Interval
	TCP Flags
	Packet Size

Table 2: Features best describing chosen malicious behaviors

4.3 Features extracted

After the data inspection elaborated on in the previous two sections, 5 features are extracted for further examination. In this section, each attribute is described and the different variants thereof that were considered are discussed. The first four features are taken from the default version of MalPaCa[10] and were analyzed to understand their impact on the clustering. The last is an additional feature that was discovered based on the WireShark inspection and the behavior of the botnets.

4.3.1 Time Interval

The time periodicity in botnets communication is a well-known and studied phenomenon [3] [12]. The assumption is made that bots use a constant (or random, but within some boundaries) time interval when communicating with the C&C server. The downside of this feature when used to describe single packets, is the error in measurement that can occur due to high latency or low bandwidth. This feature is considered in two different variants:

- First Time Difference (1TD) - the formula for this feature is:

$$g(i) = \begin{cases} pat^2[i] - pat[i+1], & \text{if } i+1 < 20. \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

- Second Time Difference(2TD) - This feature was considered as it includes additional information. If the periodicity of a communication was linear, the value is equal to 0. The formula to calculate this attribute is:

$$sg(i) = \begin{cases} g(i) - g(i+1), & \text{if } i+1 < 20. \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

[2]packet arrival time

Additionally, both variances are analyzed to determine if the normalization impacts the clustering result. To normalize the time interval sequences a min-max normalization was applied to each single sequence.

4.3.2 Bytes Size

The bytes size feature, measured as the IP header total length field, is also added to the domain of consideration. After an inspection in WireShark, some attacks like DDoS UDP floods have a constant byte size, but some attacks like Syn/Ack horizontal port scans have periodic byte size.

4.3.3 Destination & Source Port

The destination and source port both are of similar importance when considering unidirectional connections. This feature is certainly important when detecting horizontal port scans, or DDoS attacks. However, botnets like Torii often use known services like HTTP (port 443) to disguise their communication with the C&C server, so the effect of this feature on the clustering should be carefully evaluated.

4.3.4 TCP Flags

The final feature that shows significant results and was included in this study is the TCP flags feature. This feature was chosen based on the literature review [11]. The flag is presented as a decimal number, based on the binary representation of the flag order. To minimize the number of categories it was decided to only consider the most significant flags. To choose the flags a literature review was made. Furthermore, based on the results obtained from calculating the distribution of the flags (results can be seen in appendix E table 8) used in each malware, a decision was made to only use the first 6 flags which are: FIN, SYN, RES, PUSH, ACK, URG. These are also the main flags used to attack the machines during various port scans or floods. One of the main flood attacks that can be detected using this feature are TCP ACK flood (flood with TCP packets that have the ACK flags set), TCP Xmas (flood with TCP packets that have the FIN, URG, PUSH flag set), and others (TCP NULL flood, TCP FIN flood, TCP URG Flood, etc.) As it can be observed, most of the TCP protocol attacks can be defined by the combination of flags. With this additional information, one can distinguish between different malicious anomalies.
As UDP packets do not include flag header fields, this paper assigns a unique value of -1, which allows to distinguish them more efficiently. The calculated correlation (Spearman and Pearson) between TCP flags and the type of protocol (6-TCP, 17-UDP) is above 0.8, which allows us to conclude not to consider the type of protocol as a candidate feature.

5 Results

This section focuses on the discussion of the experiment and results thereof. In the first part, a detailed description of the experimental setup is given, followed by the results obtained by performing the grid search. The latter part of this section provides an in depth description of the newly added feature.

5.1 Grid Search

5.1.1 Experimental setup

To find the best combination of the Features a Grid search is performed. The experiment runs on a machine with Intel Core i5 9th gen, 8 cores and 16GB RAM. The parameters chosen for the MalPaCa tool in this experiment are: *order of the ngrams* (equal to 3), *length of packet sequence* (equal to 20 which was the default value used in [10]), *minimum cluster size* (equal to 10) and *minimum sample size* (equal to 8). The data set used in the experiment is described in section 3.1.

The grid search is performed for 7 hyper parameters presented in table 3. It is important to mention that the source port and destination port weights were always equal, as the functionality of those features is equally important depending on whether its an in or out connection.

variable name	name in experiment	range
Source port	s_value	0-3
Destination port	d_value	0-3
Flags set	f_value	0-3
Bytes size	b_value	0-3
Time interval	t_value	0-3
Second time difference	two_td	True/False
normalized	normalized	True/False

Table 3: Range of weights tested for each feature/variance.

To get reliable results and prevent the overfitting of the model a K_fold cross validation is applied (with k parameter equal to 3). As a result six data sets were created from the original one. All potential combinations from the grid search are validated against each of the aforementioned sets. The score of each combination is the average of the metrics obtained in one of the six datasets.

5.1.2 Experiment outcome

This subsection provides an analysis of the results obtained in the experiment. In table 4 the summary statistics for each metric is presented. The abbreviations used in the table are described in section 3.3.

	Average	MP	CP	CDP[3]
mean	0.822938	0.900236	0.819618	0.748961
Std	0.040764	0.017095	0.032015	0.107022
min	0.701532	0.848483	0.683801	0.527718
max	0.871636	0.934950	0.880020	0.895501

Table 4: Basic statistics of the grid search.

Firstly the individual influence of the variables on the different metrics is presented. For each feature the most significant findings are shown in the graphs. The graphs are created using *matplotlib* library. To create the visualization, the scores are grouped by the weight used in the combination (each line in the figures presents a single weight). The top 100 results of each aggregation are selected and are presented in figures 1-6. It is important to mention that the standard deviation of the metrics does not show any significance in the analysis, therefore the mean was chosen as the most accurate general description for each weight.

Destination & Source ports

The weight of the port number sequences has most significant impact on the *clustered data points* and *malware purity* metrics. As we can observed in figure 1, the weight of this features has a positive relationship with the Malware purity metric. The impact of this feature's weight on the MP metric shows an interesting behavior. While the score does not show any significant differences between nonzero weights (1, 2, 3), for which the mean and std are virtually equal, the difference from zero weights has a significant effect on MP. When the zero weight is applied to this feature the average of MP is approximately 0.03 lower than the mean for non zero weights.

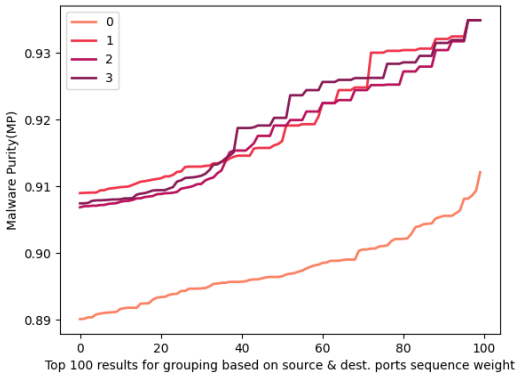

Figure 1: Top 100 MP metric results sorted in an ascending order and grouped by port sequence weight

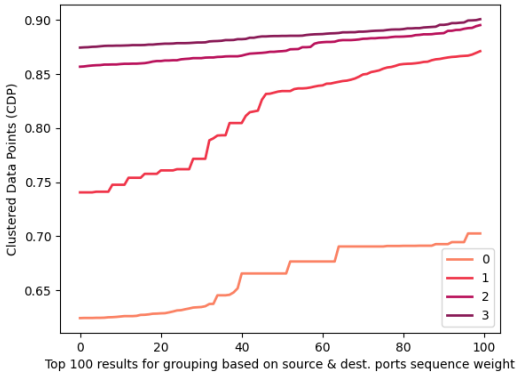

Figure 2: Top 100 CDP metric results sorted in an ascending order and grouped by port sequence weight

The *Clustered Data Points* results are presented in figure 2.

[3]Respectively Malware Purity, Cluster Purity, Clustered Data Points

It can be noticed that the port sequence weight is positively correlated with the *Clustered data points* score. With each increase of weight the score increases significantly therefore the Port Sequence has a material influence on CDP metric. The mean difference between 0 and 3 weight value is equal to 0.23 which is one of the strongest results found.
The Cluster Purity score is not found to be significantly influenced by the weight of port sequences. The outcome is illustrated in graph 23 of appendix G.

Time intervals
The results from the time interval feature can be found in graph 3. The chart is generated based on the default MalPaCa definition without using any of the aforementioned variants (which are presented later in this section). As a result only 255 combinations are taken into account. Thus the x-axis of the analyzed graphs ranges between 0 and 63/64.
The time gap has a noteworthy impact on the *cluster purity* (CP) metric. The highest scores for this metric around found for zero weights. The score difference between zero and non-zero weights for CP equals to 0.04 on average.

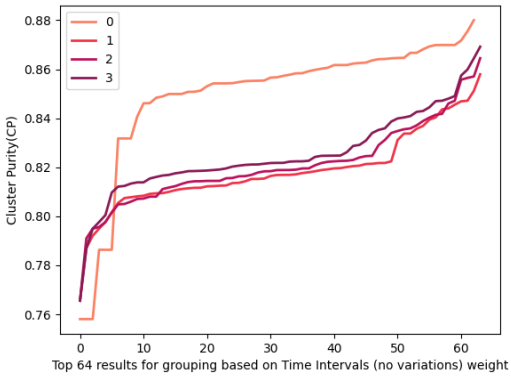

Figure 3: Top 100 CP metric results sorted in an ascending order and grouped by Time Interval weights

Time interval weights are not found to be of significant influence on the CDP and MP metrics and are therefore not presented in this section. The respective graphs are listed in appendix G.

Bytes size
The bytes size feature has a weak impact on the metrics. Based on the averages none of the metrics shows any significance, whereby most of the weights had equal impact. Nevertheless, it is noticed that when the weight is equalized to zero, the average result decreases.

TCP Flags
The TCP Flags weights show promising results across all metrics. Figures 4 and 5 present the CP and CDP metrics results plotted agains the weight of TCP flags. In graph 4, it can be observed that that combinations having a zero weight perform worse than the nonzero ones. The average difference between the means is equal 0.03.

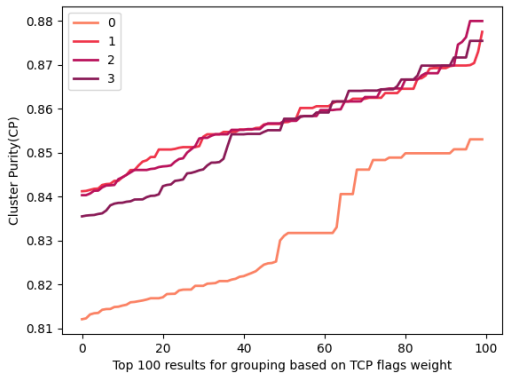

Figure 4: Top 100 CP metric results sorted in an ascending order and grouped by TCP flags weights

In figure 5 we can note that the mid-weights (1,2) show better results than using the boundary weights (0,3). This indicates that this feature is of great importance, however as explained later in this paper it performs better if one of the other 4 variables has a higher weight.

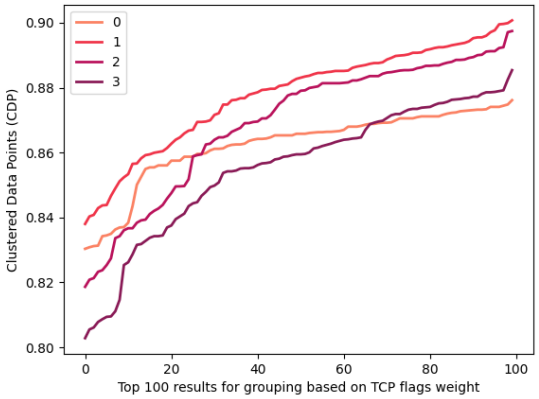

Figure 5: Top 100 CDP metric results sorted in an ascending order and grouped by TCP flags weights

Time intervals variants
As there are 4 other possible variants of the time gap feature, an analysis is performed to investigate which one yields the highest result (combinations with t-value = 0 are rejected). The graph presented below is created by grouping normalized and two_td variants. Figure 6 shows that the when two_td variants are employed (0,1 and 1,1) the MP results are impacted negatively. However, when the normalization is applied the average of the metrics is 0.03 higher. As the variants had the strongest impact on MP, the CP and CDP metrics are presented in appendix G.

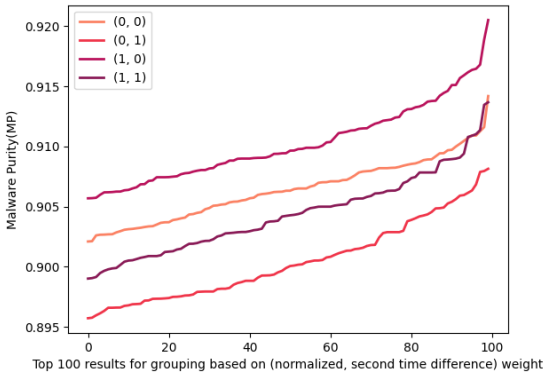

Figure 6: Top 100 MP metric results sorted in an ascending order and grouped by 2TD and normalization variants. The first value of the tuple corresponds to the normalized variant and the second to 2TD. 1 means that the variant is employed

5.1.3 Grid's search best combinations

To determine the best combination of features the arithmetic average of each metric for each combination is used. The top 5 combinations are presented in table 5, which lists each outcome per metric as well as the aforementioned mean. Noticeably, for each mix of features, the average outcome is comparable. However, some patterns within the presented combinations can be seen. First of all, only min-max normalized time intervals are mentioned, implying that the 2TD variant is less significant. Secondly, source and destination imports are of great importance as the weights are set to 3 in each of the 5 cases. Last but not least, for each case TCP flags range between 1 and 2, proving that it adds value to the clustering.

AVG	MP	CP	CDP	comb.[4]
0.876597	0.906461	0.822591	0.900740	2,1,1,3,3,T,F
0.876530	0.912451	0.830648	0.886491	1,1,1,3,3,T,F
0.876157	0.914618	0.826876	0.886976	2,0,1,3,3,T,F
0.874460	0.906756	0.819535	0.897088	2,1,2,3,3,T,F
0.873935	0.905694	0.818418	0.897694	2,2,1,3,3,T,F

Table 5: Top 5 results based on the arithmetic average of MP, CP and CDP.

It is not reasonable to choose the most effective combination based solely on the average of the metrics. However, by analyzing the top results for each metric certain observations can are made.
First, Cluster Purity (CP) is the highest when the weights corresponding to packet size and TCP flags are high, but the port sequences and first-time interval is low.
Secondly, the highest malware purity (MP) is obtained by setting the b_value and t_value to 0 and assigning an equal weight for the remaining 3 attributes. Looking at the top 10 scores for this metric, each combination follows the same scheme as explained before.

[4]Characters in the combination correspond to (in the same order): t_value, b_value, f_value, s_value, d_value, normalized, two_td

Lastly, the highest Clustered Data Points were obtained by combining higher values of port sequences and lower b_value, t_value, and f_value.

5.2 Evaluation of new feature - TCP Flags

To isolate the impact of this feature on the clustering a special combination (s_value= 1, d_value=1, t_value=1, b_value=1, two_td=False, normalized=False) is created. The HDBSCAN cluster and sample size parameters are both set to 7. After the clustering, the result and effectiveness of adding this feature is investigated by comparing the chosen behavior of the connections within the same cluster. Two additional clusters that captured interesting behavior and are not included in this section are provided in appendix C (Figures 10 and 11).

- **SYN port scan** One of the scenarios has captured the most common type of port scan, SYN, also known as a half-open port scan. This family of port scans can perform a large number of scans on different devices within a short time frame, by first sending a packet with SYN flag set, and when getting the response it replies with an RST packet. This behavior is observed in figure 7, whereby the black color represents the SYN flag set and the white color (corresponding to 4) implies that the RST flag is set.

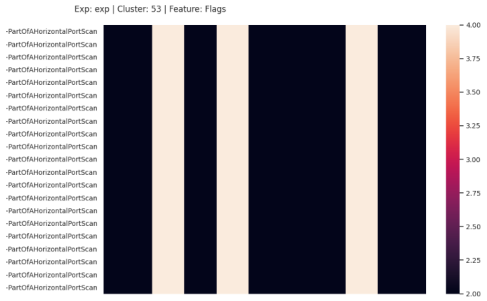

Figure 7: Heatmap of TCP flags value for a cluster that captured SYN port scan

- **DDos** There are three different DDoS attacks discovered in the dataset. The detected floods were: UDP flood, URG-ACK-PSH Flood, and the most basic SYN flood. A common pattern noticed in all of the DDoS floods captured by the dataset is that each packet in their connection is similar. Figure 8 represents two different DDoS attacks. The first heatmap virtually only consists of -1 which is the default value of the TCP flag feature, if the packet is part of a UDP attack. The second heatmap represents a DDoS attack that uses the combination of four flags (SYN, PUSH, ACK, URG). The authors of the analysis of the employed data set suspect the attack is an XMas flood, however, this is not consistent with the existing DDoS literature.

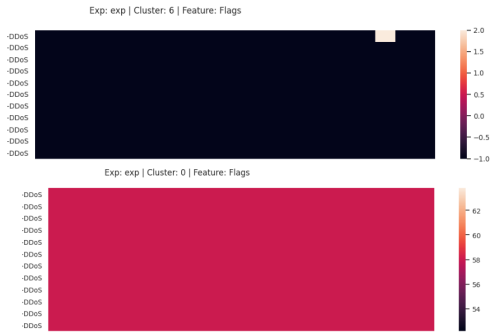

Figure 8: The first heat map presents TCP Flags in a cluster that captured a UDP flood. The second heat map presents TCP flags in a cluster that caught PUSH-ACK-URG-DDoS flood

- **C&C-Torii** As described in the beginning, this botnet attempts to hide itself by mimicking HTTPS connections. Adding the TCP flag feature to the attribute mix resolves this problem. The use of flags when communicating with a C&C server shows a pattern across different connections. It is noteworthy to mention that the communication from C&C-server to the infected device is clustered into a pure cluster, but the outgoing connections are classified in the same cluster as some HTTPS responses in the dataset. Nevertheless, the purity of the responding cluster is still high.

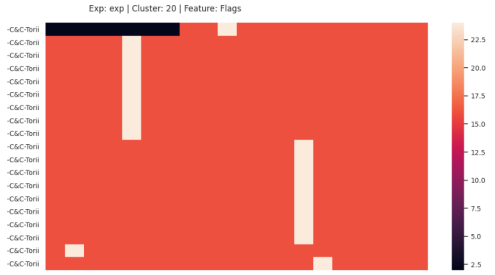

Figure 9: Heatmap of TCP flags in a cluster that captured C&C-Torii communication.

6 Responsible Research

Malicious network activity comprises a significant ethical problem in the XXI century. Such adverse activity can often result in the leakage of private data, misuse of public or private resources and interaction with IoT devices which may result in both physical and non-physical harm. This research, particularly when combined with the MalPaCa tool, should counteract the aforementioned malicious network activity and thus the negative externality which could arise from it. As a result this research does carry positive ethical aspects. The most significant ethical risk could be the application of privacy intrusive features which could be misused. However, the research specifically refrains from using such features and with that it mitigates any ethical risks of misusing private information.

The study is easily reproducible as the code will be shared publicly in its entirety on github. Furthermore, the dataset comprises open source raw data and thus can be accessed by anyone interested in reproducing the results. The same holds true for any of the libraries used which are all widely available to the public. To make such replication of results as easy as possible, each step has been clearly and precisely described within the scope of the document.

7 Discussion

To compare the results to known MalPaCa results two types of clustering are performed. The hyper parameters that were chosen for the tool are HDBSCAN (size=11, sample=7), seq_length=20. The weight-parameters for the default MalPaCa algorithm were t_value=1, b_value=1, f_value=1, s_value=1, d_value=1, none of the available variants was used. The second combination of features was chosen based on the grid search and its weights are t_value=2, b_value=1, f_value=2, s_value=3, d_value=3 (the time interval feature was normalized). The most outstanding findings were chosen and are presented in this section. The whole table is presented in Appendix F (table 9 and 10). Below (table 6) the summarized metrics are presented for both clusterings:

	AVG	MP	CP	CDP	CS	nC
Def. malpaca	0.88	0.92	0.82	0.90	50.40	37
Best comb.	0.90	0.94	0.87	0.91	47.71	39

Table 6: Metric scores for default and new feature combinations

Based on the aforementioned metrics we can notice that the new combination performs better in the aforementioned setup. Both combinations perform well in the noise metric (CDP), however, a difference in purity metrics is noticed.

The reason the purity metric is significantly lower is the use of port sequence features. For example cluster 12 of the default combination clustering has purity close to 50%. This can be explained by the fact that one group attempts to scan port 23, whereas the other tries to perform a brute force attack. With the use of the new combination, this cluster gets split up into several sub-clusters as the TCP flags further define the behavior. A similar situation can be observed in cluster 2 of the old combination. As mentioned before, the C&C-Torii is blended with 37 benign connections, which is a significant downside of the current implementation. In this cluster, the communication is handled using the 443 port, which is assigned to HTTPS communication. However, the use of flags allows recognizing the malicious connections into a separate cluster. The main contributor to this finding is the fact that the flags used by C&C-Torii show a unique pattern shown in figure 9.

Another improvement that is visible after adding the TCP flags feature to the tool is the ability to recognize a larger variety of DDoS attacks, which in the default combination was predominantly qualified into the Noise cluster (-1).

Concluding, the use of TCP flags provides additional information for clustering and does make the MalPaCa tool less prone to manipulations relating to the mimicking of network communication service. Additionally, the heat maps generated for TCP flag feature add a lot of value to the explainabil-

ity of the cluters, especially in case of well-studied behaviors like DDoS or port scans.

8 Conclusions

The main focus of this research is to explore the feature space to extract the most relevant ones, and later to find the best combination of those features to use in the MalPaCa tool. The tool is currently pre-implemented and by is default clustering based on 4 features.

After data inspection, an additional feature and two different variances of a preexisting features are found. Adding the novel *TCP flags* feature positively impacts the clustering metrics, specifically the purity of the clusters. The two variations that are considered are *time interval* feature normalization and the second time interval. Only the first shows a significant impact on the clustering performed by MalPaCa.

To find the best combination of features, a grid search was performed. No single combination of features is leading, however, a pattern in the best-performing metrics is found. First of all, the use of Source and destination ports has an important role in the overall clustering. The weights of the source and destination port sequences in the top 10 clustering were equal to 3. Another observation shows that adding the TCP flags feature improves the clustering and its weight ranges between 1 and 2. The first weight produced less noise, however, the second weight adds to the recognition between benign and malicious connections, which try to mimic the common type of harmless connections (HTTPS, telnet). Lastly, both time intervals and packet sizes were considered meaningful and their values vary between 1 and 2.

The above summarized findings provide address the central question of this paper. The research has produced a generalizable, easy to extract and privacy-friendly set of features. The generalizability is accomplished by validating the variables over a large variety of different botnets. The variables can be extracted form the packet headers in a simple and efficient way, as evidenced by the MalPaCa execution time. All extracted features are not privacy intrusive, because the focus was put only into the network and transport protocols header features (excl. IP addresses).

9 Future Work

The current experiment was performed on an IoT network. Therefore the malware in this dataset comprised mainly IoT botnets. It will be important that future research will focus on different network captures and other types of malicious behaviors. Specifically, it would be worthwile studying the set of flags used by the TCP flags feature, which would determine the external validity of the conclusions in this paper. More focus should also be put on the normalization of the features, as it already showed some promising results when applied to *time interval*. Last but not least, other kind of distance metrics should be considered with the goal to extract even more features. .

References

[1] By: Augusto Remillano II January 25, Authors, Augusto Remillano II Threats Analyst, Augusto Remillano II, Threats Analyst, and Contact Us. Thinkphp vulnerability abused by botnets, Jan 2019.

[2] Mauro Conti, Luigi V Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 297–304, 2015.

[3] Sebastian Garcia. Modelling the network behaviour of malware to block malicious patterns. the stratosphere project: a behavioural ips. *Virus Bulletin*, pages 1–8, 2015.

[4] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. Stratosphere laboratory. a labeled dataset with malicious and benign iot network traffic.

[5] Rommel Joven and David Maciejak. Iot botnet: More targets in okiru's cross-hairs, Jan 2018.

[6] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[7] Jakub Kroustek, Vladislav Iliushin, Anna Shirokova, and Jan Neduchal. Torii botnet - not another mirai variant, Sep 2018.

[8] Qin Lin, Sridhar Adepu, Sicco Verwer, and Aditya Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems.(2018). 2018.

[9] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[10] Azqa Nadeem, Christian Hammerschmidt, Carlos H Gañán, and Sicco Verwer. Beyond labeling: Using clustering to build network behavioral profiles of malware families. In *Malware Analysis Using Artificial Intelligence and Deep Learning*, pages 381–409. Springer, 2021.

[11] Stuart Staniford, James A Hoagland, and Joseph M McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1-2):105–136, 2002.

[12] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 349–360, 2012.

A Dataset in detail

IoT-23 code	mal. name	pseudo-conn. [5]	no. conn.
34-1	Mirai	2	10M:36B
43-1	Mirai	1	5M:8B
44-1	Mirai	-	-
49-1	Mirai	-	-
52-1	Mirai	-	-
20-1	Torii	10	20M:132B
21-1	Torii	10	20M:105B
42-1	Trojan	10	50M:118B
60-1	Gagfyt	1	3M:4B
17-1	Kenjiro	1	634M:56B
36-1	Okiru	2	5M:44B
33-1	Kenjiro	1	6M:51B
8-1	Hakai	10	20M:103B
35-1	Mirai	2	17M:135B
48-1	Mirai	2	55M:24B
39-1	IRCBot	-	-
7-1	Linux, Mirai	4	15M:42B
9-1	Linux, Hajime	-	-
3-1	Muhstik	-	-
1-1	Hide and Seek	1	162M:9B

Table 7: The usage of the scenarios from IoT-23. (In the fourth column the *Y*M:*X*B, should be understood as *X* benign and *Y* malicious connections extracted)

B Clusters obtained by TCP Flags feature

Custom port scan Some more interesting TCP port scans were also detected and can be seen in figure 10. This attack uses a combination of ACK, SYN, PSH, FIN flags. The distribution thereof has not been determined, and it might be a custom set attack.

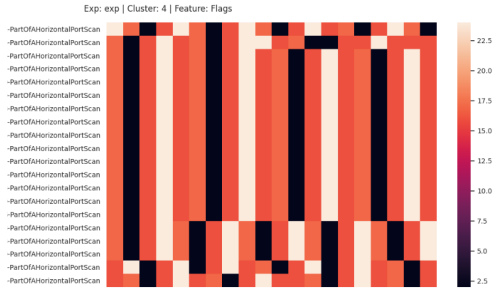

Figure 10: Heatmap of the TCP flags value in a cluster that caught Custom port scan

Okiru-Attack This is a DDoS flood performed using IoT device already infected by the Okiru botnet. It was calssified to the same cluster with some DDoS connections, although when only analyzing the flags it uses we can notice on Figure 11 a pattern in the flags set.

[5] Subconnections created out a single long (¿Threshold) connections.

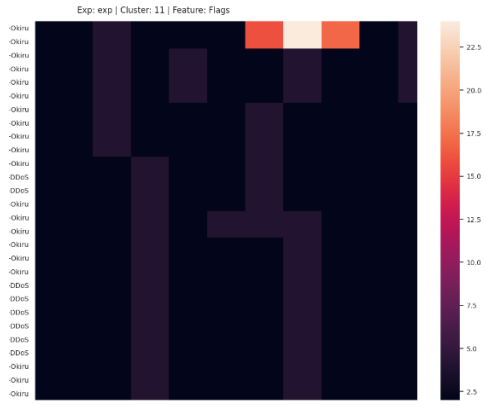

Figure 11: Heatmap of the TCP flags value in a cluster that caught Okiru attack

C Features best describing behaviour

C.1 Horizontal Port Scan

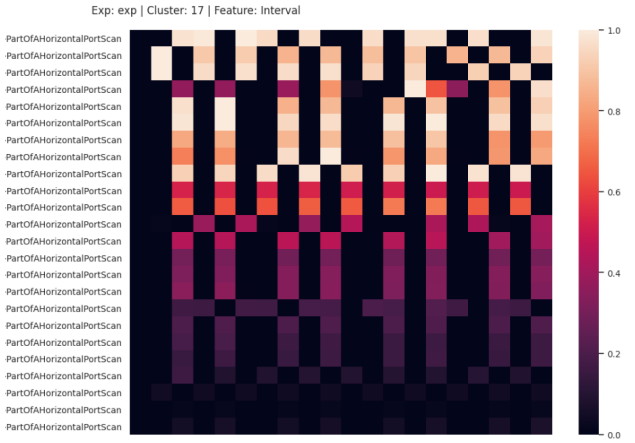

Figure 12: Heatmap showing the periodicity of the time interval feature in horizontal port scan cluster

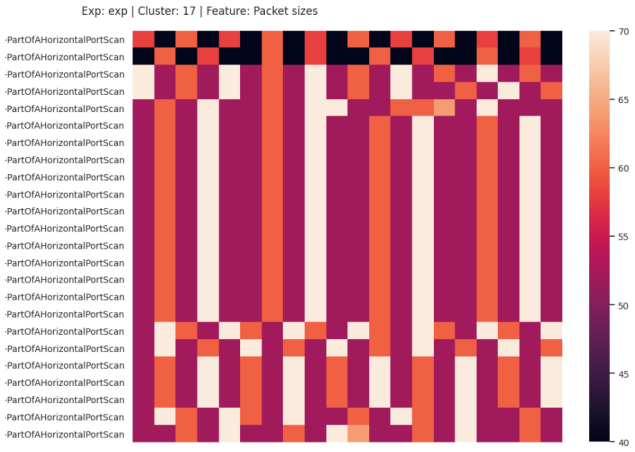

Figure 13: Heatmap showing the periodicity of the packet size feature in horizontal port scan cluster

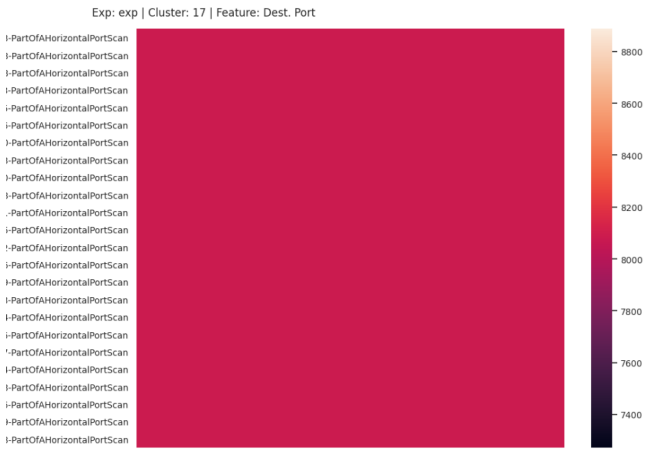

Figure 14: Heatmap showing the continual use of the same destination port in a horizontal port scan cluster

C.2 DDoS

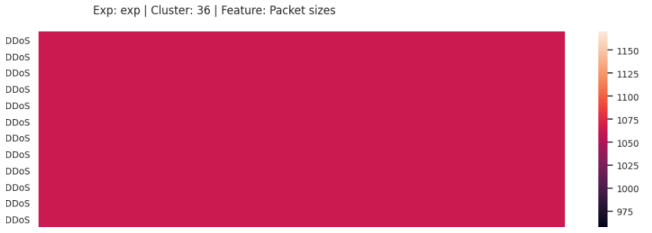

Figure 15: Heatmap showing the fixed packet size in DDoS cluster

Figure 16: Heatmap showing the fixed packet size in a different DDoS cluster

Figure 17: Heatmap showing the fixed destination port used in a DDoS cluster

Figure 18: Heatmap showing the fixed destination port used in a different DDoS cluster

C.3 C&C

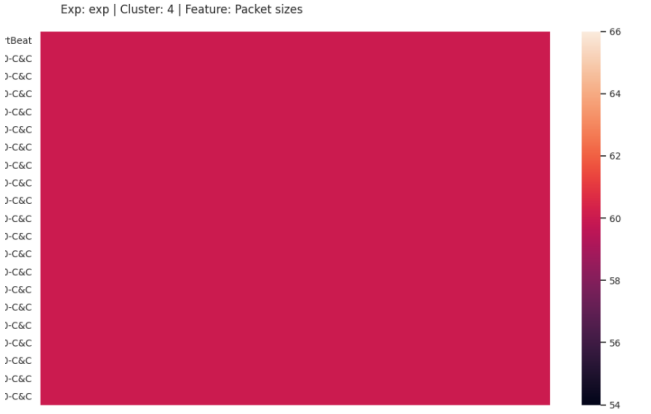

Figure 19: Heatmap showing the fixed packet size used in a C&C cluster

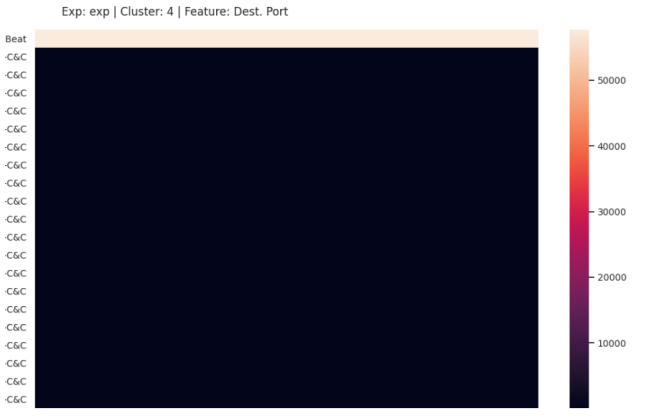

Figure 20: Heatmap showing the fixed destination port used in a C&C cluster

Figure 21: Heatmap shows the periodicity of the time interval feature in a C&C cluster

D Formulas of metrics

Cluster purity (CP):

$$CP = \frac{\sum 4 \times (clustering\_purity\_i - 0.5)^2}{\#clusters\_in\_clustering}$$

Clustered data points (CDP):

$$CDP = 1 - \frac{\#connections\_in\_noise\_cluster}{\#all\_connections}$$

Malicious purity (MP):

$$MP = \frac{\sum \#connections\_of\_the\_cluste\_owner}{\#connections\_in\_cluster}$$

E Distribution of flags over each malware

	Flag	Count		Flag	Count
DDoS		0	Okiru		0
	RES	0		RES	0
	NON	0		NON	0
	CONG	0		CONG	0
	ECN	0		ECN	0
	URG	0		URG	0
	ACK	242		ACK	0
	PUSH	242		PUSH	0
	RST	242		RST	0
	SYN	42		SYN	110
	FIN	534		FIN	172
		0			0
PartOfAHorizontalPortScan		0	C&C-HeartBeat		0
	RES	0		RES	0
	NON	0		NON	0
	CONG	0		CONG	0
	ECN	0		ECN	0
	URG	0		URG	0
	ACK	0		ACK	0
	PUSH	261		PUSH	27
	RST	116		RST	6
	SYN	3499		SYN	15
	FIN	2281		FIN	50
		11			2
Attack		0	C&C-Torii		0
	RES	0		RES	0
	NON	0		NON	0
	CONG	0		CONG	0
	ECN	0		ECN	0
	URG	0		URG	0
	ACK	0		ACK	0
	PUSH	324		PUSH	775
	RST	135		RST	40
	SYN	39		SYN	0
	FIN	36		FIN	27
		0			0
C&C		0	FileDownload		0
	RES	0		RES	0
	NON	0		NON	0
	CONG	0		CONG	0
	ECN	0		ECN	0
	URG	0		URG	0
	ACK	0		ACK	0
	PUSH	33		PUSH	36
	RST	2		RST	20
	SYN	0		SYN	0
	FIN	430		FIN	0
		0			0

Table 8: TCP Flag count for all recognized malicious behaviors in the dataset

F Clustering results used for comparison

cluster no.	Malicious behavior captured
0	'None': 0, 'C&C': 20
1	'None': 37, 'C&C-Torii-resp': 20
2	'None': 23
3	'None': 0, 'PartOfAHorizontalPortScan': 12
4	'None': 0, 'DDoS': 12
5	'None': 28, 'Attack-resp': 14, 'PartOfAHorizontalPortScan-resp': 55
6	'None': 2, 'Attack-resp': 1, 'PartOfAHorizontalPortScan-resp': 29
7	'None': 22, 'FileDownload': 1, 'DDoS': 2
8	'None': 13, 'DDoS-resp': 2, 'PartOfAHorizontalPortScan-resp': 1
9	'None': 12, 'C&C-resp': 3, 'C&C-HeartBeat-FileDownload-resp': 2
10	'None': 0, 'Okiru': 27, 'DDoS': 8
11	'None': 19, 'Okiru': 2, 'PartOfAHorizontalPortScan': 58
12	'None': 48, 'Attack': 23, 'PartOfAHorizontalPortScan': 62
13	'None': 3, 'C&C-Torii': 17
14	'None': 27
15	'None': 20, 'FileDownload-resp': 1
16	'None': 22
17	'None': 30
18	'None': 48
19	'None': 60
20	'None': 78
21	'None': 3, 'Attack': 1, 'PartOfAHorizontalPortScan': 27
22	'None': 0, 'PartOfAHorizontalPortScan': 24
23	'None': 14
24	'None': 20
25	'None': 88
26	'None': 53
27	'None': 60
28	'None': 76
29	'None': 38
30	'None': 5, 'PartOfAHorizontalPortScan': 525, 'Okiru': 2, 'DDoS': 2
31	'None': 12
32	'None': 12
33	'None': 0, 'PartOfAHorizontalPortScan': 12
34	'None': 0, 'DDoS': 12
35	'None': 0, 'DDoS': 12
-1 (noise)	'None': 133, 'C&C': 3, 'DDoS': 20, 'C&C-Torii': 3, 'C&C-HeartBeat-FileDownload': 2, 'C&C-HeartBeat': 9, 'C&C-HeartBeat-resp': 6, 'PartOfAHorizontalPortScan': 9, 'Okiru': 4

Table 9: Clusters obtained by classifying based on the default combination of features

cluster no.	Malicious behavior captured
0	'None': 0, 'PartOfAHorizontalPortScan': 12
1	'None': 0, 'DDoS': 12
2	'None': 26, 'C&C-resp': 3, 'DDoS-resp': 2, 'C&C-HeartBeat-FileDownload-resp': 2, 'PartOfAHorizontalPortScan-resp': 1
3	'None': 1, 'PartOfAHorizontalPortScan-resp': 26
4	'None': 48, 'Attack': 23, 'PartOfAHorizontalPortScan': 62
5	'None': 28, 'Attack-resp': 14, 'PartOfAHorizontalPortScan-resp': 55
6	'None': 0, 'C&C': 20, 'C&C-HeartBeat': 1
7	'None': 22, 'FileDownload': 1, 'DDoS': 2
8	'None': 0, 'Okiru': 27, 'DDoS': 10
9	'None': 23
10	'None': 19, 'Okiru': 2, 'PartOfAHorizontalPortScan': 58
11	'None': 0, 'DDoS': 11
12	'None': 28
13	'None': 19, 'FileDownload-resp': 1
14	'None': 1, 'C&C-Torii-resp': 20
15	'None': 13
16	'None': 1, 'C&C-Torii': 18
17	'None': 16
18	'None': 0, 'PartOfAHorizontalPortScan': 20
19	'None': 22
20	'None': 30
21	'None': 48
22	'None': 60
23	'None': 78
24	'None': 14
25	'None': 20
26	'None': 88
27	'None': 53
28	'None': 60
29	'None': 76
30	'None': 38
31	'None': 13
32	'None': 3, 'Attack': 1, 'PartOfAHorizontalPortScan': 26
33	'None': 2, 'PartOfAHorizontalPortScan': 524, 'Okiru': 2, 'DDoS': 2
34	'None': 0, 'DDoS': 12
35	'None': 0, 'DDoS': 12
36	'None': 12
37	'None': 0, 'PartOfAHorizontalPortScan': 12
-1 (noise)	'None': 144, 'C&C': 3, 'DDoS': 7, 'C&C-Torii': 2, 'C&C-HeartBeat-FileDownload': 2, 'C&C-HeartBeat': 8, 'C&C-HeartBeat-resp': 6, 'PartOfAHorizontalPortScan': 15, 'Attack-resp': 1, 'Okiru': 4, 'PartOfAHorizontalPortScan-resp': 5

Table 10: Clusters obtained by classifying based on the newly selected combination of features (+TCP flags)

G Supplementary graphs from grid search results

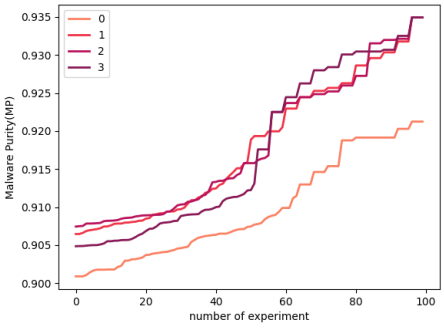

Figure 22: Top 100 Malware Purity(MP) metric results sorted in an ascending order and grouped by TCP flags weights

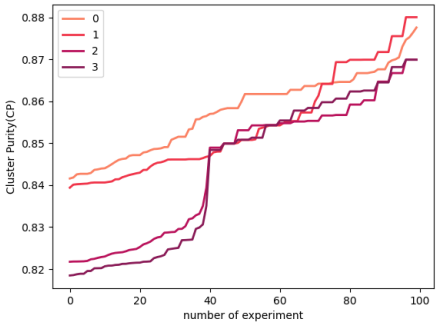

Figure 23: Top 100 Cluster Purity(CP) metric results sorted in an ascending order and grouped by source & dest. ports sequence weights

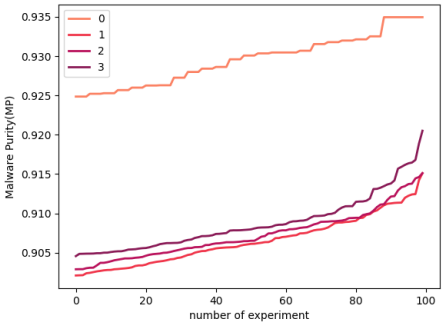

Figure 24: Top 100 Malware Purity(MP) metric results sorted in an ascending order and grouped by Time Intervals (no variations) weights

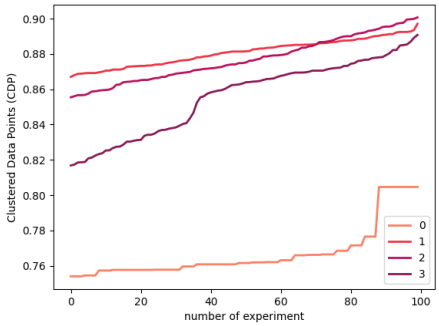

Figure 25: Top 100 Clustered Data Points (CDP) metric results sorted in an ascending order and grouped by Time Intervals (no variations) weights

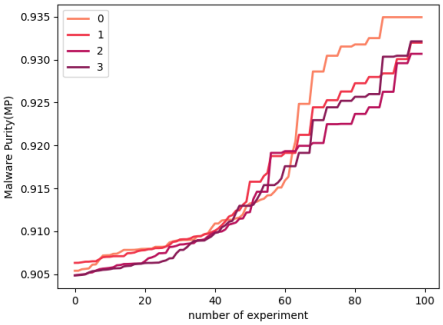

Figure 26: Top 100 Malware Purity(MP) metric results sorted in an ascending order and grouped by Packets sizes weights

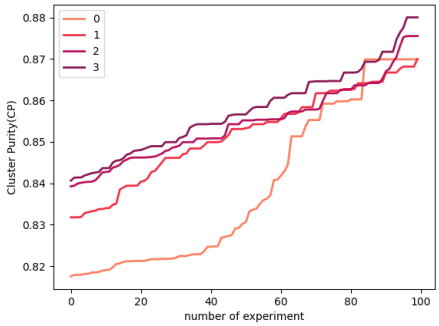

Figure 27: Top 100 Cluster Purity(CP) metric results sorted in an ascending order and grouped by Packets sizes weights

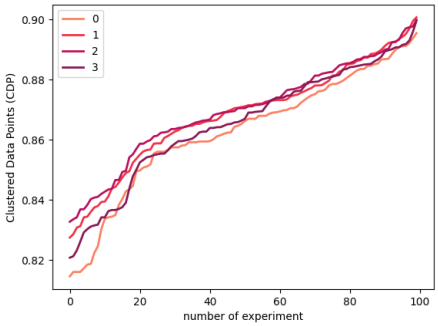

Figure 28: Top 100 Clustered Data Points (CDP) metric results sorted in an ascending order and grouped by Packets sizes weights