Beyond Labeling: Intuitive Capability Assessment of Malware using Network Behavioral Profiles

Azqa Nadeem
Delft University of Technology
Delft, The Netherlands
azqa.nadeem@tudelft.nl

Christian Hammerschmidt
Delft University of Technology
Delft, The Netherlands
c.a.hammerschmidt@tudelft.nl

Carlos H. Gañán
Delft University of Technology
Delft, The Netherlands
c.hernandezganan@tudelft.nl

Sicco Verwer
Delft University of Technology
Delft, The Netherlands
s.e.verwer@tudelft.nl

ABSTRACT

Malicious software is still a leading threat in cybersecurity. Anti-Virus (AV) companies are pivotal in understanding and assigning labels to new malware samples. Currently, these labels are the sole source of ground truth information available to the security community to evaluate malware analysis methods. However, their adopted naming conventions are known to be inconsistent and unverifiable. The labels are also black box since they do not represent the capabilities of malware. We believe we need a white box way to determine the capabilities of malware based on their behavior, rather than black box family labels. The current state of the art in malware capability assessment contains largely manual approaches.

We propose a novel method called MalPaCA, which for a large part automates capability assessment by clustering temporal behavior observed in a malware's network traces. MalPaCA uses network traces since most malware uses internet to carry out its objectives. In doing so, we build behavioral profiles of malware capabilities that are significantly more descriptive than their black box family names. We also propose an intuitive, visualization-based evaluation method for the obtained clusters. We evaluate MalPaCA on 1.1k malware samples collected in the wild. MalPaCA shows promising results: (i) It correctly discovers capabilities, such as port scans and reuse of Command and Control servers; (ii) It discovers a number of discrepancies between behavioral clusters and traditional malware family designations; and (iii) It demonstrates the effectiveness of clustering unlabeled network traces using temporal features by producing a false positive rate of mere 8%.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**;

KEYWORDS

Behavioral modeling; Capability assessment; Malware analysis; Network traffic; Sequence Clustering.

1 INTRODUCTION

It has been over thirty years since the first malware was discovered. Yet, it is still one of the leading threats in cybersecurity today[1]. In 2017, Symantec reported a spike of 88% in detected malware variants[2]. According to Accenture, a malware attack on a company can cost $2.4M on average and can take 50 days to resolve[3]. The global cost of cyber attacks is expected to exceed $2 trillion this year [30]. These costs will keep increasing as the threat landscape evolves.

The rapid spike in the volume and types of malware requires the security community to understand malware in order to efficiently defend against it. Anti-Virus (AV) companies play a pivotal role in understanding malware by assigning labels to newly discovered samples. However, there are several shortcomings of malware family labels: (i) Each vendor has their own way of determining a malware family. Labels obtained from different vendors are often inconsistent [20]. (ii) The precise methods used by each vendor are proprietary and unstandardized [38]. (iii) The current labels are heavily based on static and system-level activity analysis. This poses a number of important problems for the security community: (a) The black-box nature of the used methods makes it impossible to verify assigned family labels, causing the evaluation of newer analysis methods to depend on unreliable ground truth labels [25]; (b) Although network traffic shows the core behavior of malware by showing direct interactions with the attacker or C&C server [8], this is rarely used to determine family labels. Hence, malware samples that have different code attributes but exhibit identical network behavior will end up in different families, see, e.g., Perdisci et al. [35]. To combat this, we propose descriptive, customizable, and white box identifiers for malware samples that represent capabilities and can be used in conjunction with assigned family labels.

Malware capability assessment has largely been a manual process [6, 39]. In this paper, we propose MalPaCA (Malware Packet Sequence Clustering and Analysis), that performs semi-automated capability assessment of malware. MalPaCA automates it by clustering similar temporal behaviors at the packet level, representing the observed capabilities. The extracted clusters are used to build behavioral profiles that are significantly more descriptive than black box family names. Network traffic is also easier to access than system-level activity logs [35]. The key intuition is that malware belonging to the same family will exhibit similar behaviors since malware authors share code and resources [42]. MalPaCA is novel as it adopts sequential features that keep the temporal nature of

[1] https://www.cybersecurity-insiders.com/top-15-cyber-threats-for-2019/

[2] https://www.varonis.com/blog/cybersecurity-statistics/

[3] https://www.accenture.com/us-en/insight-cost-of-cybercrime-2017

the traffic intact while identifying those behaviors. To this aim, it uses Dynamic Time Warping and N-grams to develop a distance measure between network connections. To a security analyst, MalPaCA presents visualizations of clusters of these connections, one for each capability. MalPaCA only utilizes easy-to-access features from packet headers and can be applied to encrypted traffic.

The last step of MalPaCA's capability assessment is assigning capability labels to clusters, which is manual. Each cluster is visualized using temporal heatmaps to determine which capability it captures. The temporal heatmaps provide a goal-driven approach to investigate the accuracy of MalPaCA's clustering by clearly showing the network connections that are grouped together. This eliminates the need to investigate thousands of network traces. Analysts can use these heatmaps to fine-tune MalPaCA's parameters, putting them back in the loop of malware data analysis. The temporal heatmaps also assist in understanding malware's attacking capabilities, and investigating behaviors that are similar across different malware families. The key advantage of this methodology is its white-box nature: it provides a visual representation to investigate MalPaCA's reasoning of finding behavioral similarity. In doing so, we address the interpretability problem of typical black box analysis methods, which will improve the stepping stones towards better detection methods.

We evaluate MalPaCA's performance on 1.1k malware samples coming from 15 families collected in the wild. We also compare the effectiveness of clustering using temporal sequential features by comparing with an existing method based on frequently used statistical aggregate features [5, 43]. The results are very promising: (i) MalPaCA automatically identifies several attacking capabilities, such as port scans and reuse of C&C servers; (ii) MalPaCA's capability assessment works even on low quality datasets, though additional traces result in more thorough profiles; (iii) MalPaCA demonstrates the effectiveness of clustering using temporal features by producing a false positive rate of mere 8%; and (iv) MalPaCA discovers a number of discrepancies between behavioral clusters and traditional malware family designations. We believe this happens either because their labels are incorrect or because the overlapping families share significant behavior.

In summary, our contributions are:

(1) A methodology called MalPaCA to perform semi-automated capability assessment by clustering malware's network behavior;
(2) A white-box visualization-based cluster evaluation method that works without ground truth labels and brings malware analysts back in the loop;
(3) A proof-of-concept of the proposed method that is evaluated on real-world malware samples collected in the wild;
(4) A demonstration of the effectiveness of clustering using temporal features, which shows less errors than an existing solution based on statistical aggregates.

2 RELATED WORK

Malware family labels are known to be noisy, inconsistent and without a common vocabulary for all security companies to follow. They are also heavily based on static analysis [12] and system-level behavioral analysis [3, 41], rather than network-level behavioral analysis. These issues have been explored in literature. Maggi et al. [29] propose a method to find inconsistencies in malware family labels generated by Anti Virus (AV) scanners. Mohaisen et al. [31] are the first ones to measure the accuracy, consistency and completeness of AV scanners. Their results show that AV vendors produce inconsistent labels 50% of the time, on average. Popular tools, such as VirusTotal, run multiple AV scanners and return an array of labels predicted by each scanner, without a way to know which is correct. This led to research on dealing with the inconsistencies in the family labels. Kantchelian et. al [20] proposed an algorithm based on Expectation Maximization and Bayesian models that assigns weights to each vendor's trustworthiness. Sebastián et. al [38] developed a useful open source tool called AVClass that determines the likely family name after performing heavy filtering on all the predicted labels. However, these methods do not address the key underlying issue—malware family labels are black box with limited interpretability.

Utilizing behavioral profiles instead of family names solves these issues at the core. In order to characterize a malware family, capability assessment is done to determine the behaviors it can exhibit. Capability assessment has primarily been a manual effort, resulting in behavioral profiles that are quickly outdated. Black et al. [6] bridge the semantic gap between low-level API calls and high-level behaviors. They extract API calls by statically analyzing a banking malware dataset, and map them to high-level behaviors manually with the help of domain experts. Sharma et al. [39] recently proposed a method to build behavioral profiles. They select some high-level capabilities possessed by malware by investigating the literature, and map them to low-level behaviors extracted from the static analysis of only 56 malware samples. In addition, their method does not use any network traffic features. MalPaCA does semi-automated capability assessment by automatically extracting clusters. Contrary to Sharma et al. [39], our focus is on network traffic analysis and we perform our experiments on 1.1k malware samples.

There exist a vast number of clustering algorithms in literature for various objectives [3, 9, 17, 21, 28, 35, 43]. They aim to either distinguish malware families, perform behavioral analysis of malware, or extract syntactical signatures for detection purposes. One common problem with malware analysis methods is their complexity in both reproducing and understanding them, e.g. they generally involve multiple filtering phases turning them into black boxes, giving little control to malware analysts [5, 32]. Some of the existing approaches use features from protocol specific headers which limits their applicability. For example, there are methods only for HTTP-based malware using HTTP request and response queries [35], DNS-based malware using DNS queries [24, 36], and HTTPs-based malware using TLS message types [2, 27]. Existing work also places emphasis on using Deep Packet Inspection [18, 48], which will not work out-of-the-box when traffic is encrypted.

Most existing clustering approaches use statistical features, often relying on traffic analysis that uses sampled netflows [5, 14, 34, 43]. In such cases, only high-level features are available for behavior modeling. Garcia [14] builds a behavioral Intrusion Prevention System by using the size, duration and periodicity of flows. Tegeler et al.

[43] build a classification system to detect bot-infected network traffic using high-level features. Nevertheless, there also exist work that uses sequential features. Pellegrino et al. [34] learn state machines from sequential netflow data in order to detect bot-infected traffic. Hammerschmidt et al. [17] uses sequences of netflows to cluster host behavior over time. These methods require long uninterrupted sequences to provide a reasonable statistical distribution of data. In practice, malware-related data is often scarce and noisy. In contrast, we propose a method that operates directly on sequences keeping temporal information intact, and utilizes minimal sequence sizes.

3 METHODOLOGY

MalPaCA clusters malware's network behavior in an intuitive way that automatically identifies the capabilities of malware samples present in a dataset. The clustering also identifies samples that share common behaviors, providing a behavioral profile that is much more descriptive than mere family labels. The profiles are built using observed behavior since only the executed functionality is relevant for behavioral profiling. Profiles for individual families can be enriched further by capturing additional traffic.

Figure 1 illustrates the architecture of MalPaCA with its five phases (P1 to P5). Network traces (Pcap files) are given as input to the system, which are split into unidirectional streams (called *connections*) that are clustered based on temporal similarities. Each cluster is assigned a capability label by visualizing temporal heatmaps that show the various features of the connections.

3.1 Connection generation (P1)

Sequence clustering is a technique where input features are represented as sequences, which are clustered based on their mutual distances. The motivation for sequence clustering is explained with an example of a computer program that randomly generates a list of 9 numbers from 1 to 3, and the task is to model the different behaviors exhibited by this program. Suppose that it generates the following two sequences: [1,2,3,1,2,3,1,2,3] and [1,1,1,2,2,2,3,3,3]. It is evident that the behavior exhibited by the program was categorically different when it generated the two sequences. However, the aggregate of both these sequences is 2. Hence, if a statistical feature models this example, it would incorrectly group these two behaviors.

A *connection* is defined as an uninterrupted unidirectional list of all packets sent from source IP to destination IP address. This means 8.8.8.8 → 123.123.123.123 is a different connection than 123.123.123.123 → 8.8.8.8. We use unidirectional connections to pinpoint the origin of capabilities, which helps build more diverse behavioral profiles. We refer to them as *Outgoing* and *Incoming* connections based on their direction with respect to the `localhost`. It is noteworthy that we do not use IP address as a feature, but only to create connections.

Ideally, a connection captures one complete capability. But realistically, the length of the connection needs to be capped to a fixed threshold in order to avoid introducing artifacts due to the considerable variance in the lengths of the sequences. Following the guidelines of Korczyński at al. [22], we analyze only the first few packets of a connection, often referred to as the *handshake*. This is a fixed number denoted by the tunable parameter `len`. It should be large enough to allow the handshake to be modeled, the length of which is often unknown in network traffic analysis. Moreover, larger value of `len` also increases the computational resources required to process longer connections. Finding a trade-off between capturing most handshakes and limiting computation requirements is important.

3.2 Feature-set extraction (P2)

The choice of feature-set is crucial for determining the kind of behaviors that are identified by MalPaCA. Two considerations motivate our choice: 1) MalPaCA should be generalizable to more than one type of malware; 2) The feature set is small and easy to extract. Hence, we cannot use features extracted from the packet payload itself as they limit the applicability of the method. We also do not use IP addresses as they are easy to spoof and are considered Personally Identifiable Information[4] in countries like the Netherlands.

We use four sequential features: (i) packet size, (ii) time interval, (iii) source port, (iv) destination port. All four features are independent of the protocol type, making them available for every connection. Although these features are simplistic, their sequential nature captures behavioral changes effectively. Each feature is represented as a list of observations for individual time points, instead of aggregating them.

Packet size measures the size of the *IP datagram* of each packet in bytes. **Time interval** captures the inter packet arrival time in milliseconds. We use time interval because malware tends to show a periodic behavior, e.g. bots send periodic heartbeat packets[5] to inform the C&C server about the infected host. MalPaCA is meant to be used on a single network at a time since using inter-arrival time makes connections collected on different latency networks incomparable. **Port numbers** can be considered as the *doors* that hosts use to communicate with the outside world. We use both source and destination port numbers because the connections are unidirectional. We particularly use source port for analysts to limit the use of problematic ports in case of outgoing connections. We can also identify the protocol used by the malware based on port information, e.g. Port 80 indicates HTTP-based malware while 53 indicates DNS-based malware. Moreover, usage of certain vulnerable ports can indicate suspicious activity. We build one separate sequence for each feature, resulting in four sequences per connection. Figure 2 shows that each connection is represented as a quadruple $(f1, f2, f3, f4)$, where each element is a sequential feature.

3.3 Distance measure (P3)

In order to reason about behavioral similarity, we must be able to measure distance between sequences. Three considerations motivate our choice of the distance measures used: 1) Different distance measures are applicable on nominal and ratio data types; 2) The distance measure should be intuitive to help understand the results; 3) The distance measure must produce results that are resilient to delays and noise, which are common characteristics of network traces. The last consideration was added after observing distance measures producing results that were artifacts of network delays, instead of behavioral differences. MalPaCA uses a combination of

[4] https://www.enterprisetimes.co.uk/2016/10/20/ecj-rules-ip-address-is-pii/
[5] https://www.ixiacom.com/company/blog/mirai-botnet-things

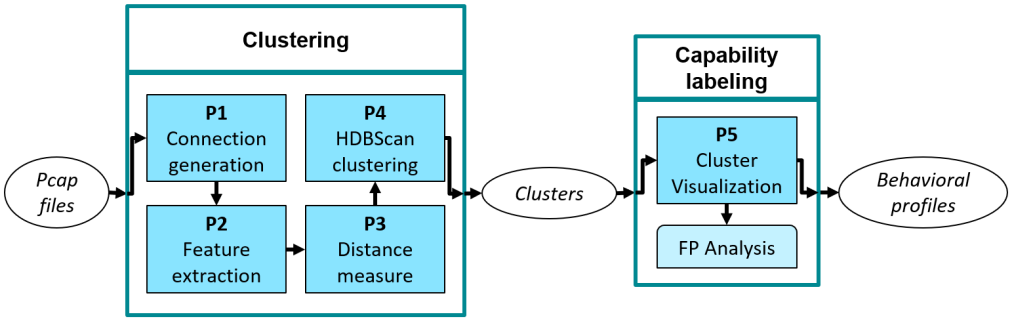


Figure 1: The MalPaCA framework. Connections are clustered on behavioral similarities. Each cluster captures a capability.

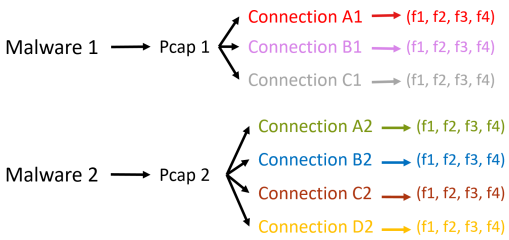


Figure 2: Sequence generation: Each malware sample is broken into multiple connections. Each connection is represented by 4 features, each of which is a sequence. MalPaCA clusters connections based on similar feature sequences.

Dynamic Time Warping (DTW) and N-gram analysis to measure distance between two connections.

DTW has applications in shape-matching and time-series classification, such as in fingerprint verification [23], and characterizing DDoS attack dynamics [46]. Bio-informatics and Computational linguistics have used Ngrams long before their application in cybersecurity, e.g., in modeling genomic sequences [45] and file matching [26]. They have also been used to classify malicious code [1].

Dynamic Time Warping. DTW [4] is used to measure distances between numeric sequences (packet size and time interval) due to its robustness to delays and noise, which is a common characteristic of network traffic. It aligns two time-series that may contain distortions (or warps) in the time-axis. It finds local substructures in one sequence and maps them to those of the other sequence. A visual representation of DTW is shown in Figure 3.

The output of DTW is a *similarity* score. It is normalized using Eq. (1), and converted to distance using Eq. (2).

$$normed_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

where $x = [x_1, x_2, x_3, \ldots, x_n]$ *and* $normed_i$ *is the normalized value of* x_i.

$$distance = 1.0 - similarity \tag{2}$$

where the similarity score has been scaled to the range [0-1].

Ngram analysis. Port numbers are represented as Ngrams and the distance between them is measured in the vector space using

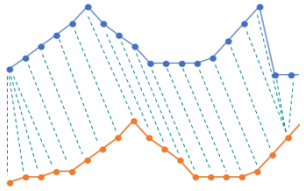


Figure 3: Dynamic Time Warping aligns sequences to minimize distance between two time-series. Blue and orange are two time series and the green dotted line shows which points are mapped together by DTW.

Cosine similarity. An *Ngram* is defined as the set of n (called *order* henceforth) consecutive items in a given sequence. Ngrams capture the structure of a sequence from a statistical point of view. The larger the value of *order* is, the more structure is captured within a single ngram. A sequence is converted into a set of ngrams, called its *Ngram profile*, by using a sliding window of length *order*. An example for $order = 2$ is shown in Table 1, where A, B, C, D are hypothetical port numbers. Next, a set of distinct Ngrams, called C, is generated from the whole dataset. For each sequence, a numeric list equal to the length of C is generated. For each Ngram in C, its frequency in the sequence's Ngram profile is appended to the list. If the Ngram does not exist in the Ngram profile, 0 is appended instead. This step transforms the profiles into their vector representation.

Table 1: Example of Measuring Distance Between Two Categorical Sequences Using Ngrams and Cosine Similarity.

Input	Ngram profiles	C=[AB,BC,CB,DA,CA]	Cosine distance
ABCBC	*AB, BC, CB, BC*	[1,2,1,0,0]	0.3876
DABCA	*DA, AB, BC, CA*	[1,1,0,1,1]	

In the vector space, *Cosine similarity* is one of the most frequently used similarity score. It is determined by the angle between two non-zero vectors. The similarity value lies between 0 and 1, where 1 means that the two vectors are the same (parallel to each other) and 0 means they are completely different (orthogonal to each other). It is converted to distance using Eq. (2). Cosine similarity is selected as the distance measure because of its robustness.

Finally, the DTW and cosine distances are combined to calculate the final distance between two connections using Eq. (3).

$$d_{conn}(a,b) = \frac{d_{packetSize}(a,b) + d_{interval}(a,b) + d_{sourcePort}(a,b) + d_{destPort}(a,b)}{4} \quad (3)$$

where a and b are two connections; $d_{conn}(a,b)$ is the final calculated distance between a and b; $d_{packetSize}(a,b)$ is the DTW distance between the sequences of packet sizes of a and b; $d_{interval}(a,b)$ is the DTW distance between the sequences of intervals between a and b; $d_{sourcePort}(a,b)$ and $d_{destPort}(a,b)$ are the Cosine distances between the sequences of source ports and destination ports between a and b, respectively.

3.4 HDBScan Clustering (P4)

A key strength of MalPaCA is the clustering algorithm itself. There exists a familial structure among malware samples [40, 44]. Therefore, it makes sense to use hierarchical clustering to model their relationships. We have used Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBScan) [7] for this purpose. The key strengths of HDBScan are twofold: a) it automatically determines thre optimal number of clusters, (b) its generated clusters remain stable over time. In addition, at the time of writing this paper, HDBScan is the best clustering algorithm, especially for Python implementations[6].

It requires a pairwise distance matrix as input. It does not force data points to become part of clusters—all data points whose membership to a cluster cannot be determined are considered to be *noise*. In the current context, *noise* refers to behaviors that are either too different from all the others or cannot be clearly assigned to one cluster. An ideal dataset with clear cluster boundaries will have no noise. Hence, in the presence of a less ideal dataset, noise is discarded to extract high-quality clusters. Keep in mind that discarding excessive connections as noise can also be counterproductive. We discuss this limitation in Section 6.

Evasion resilience. Knowing MalPaCA's internals, attackers can attempt to evade it by designing malware that exhibits a wide range of randomized behavior to avoid falling into a specific cluster. If the updated malware still accomplishes the same tasks, it has to show the same core network behaviors. The usage of Dynamic Time Warping distance makes our system resilient to random delays [11] and due to the relative distance measures used in HDBScan, randomized port numbers [13] will also be clustered together. Nonetheless if they manage to evade MalPaCA, the malware sample will end up with a different behavioral profile making analysts more prone to analyze it.

3.5 Cluster visualization (P5)

Formalizing cluster quality without ground truth is a fundamental challenge in clustering. Although some metrics exist that capture cluster quality (i.e. Silhouette index [37] and DB Index [10]), they are not applicable to MalPaCA. They require a notion of a point's distance from its cluster centroid, which does not work well with sequences since each data point (or connection) is represented as a vector of its relative distances from the other points. Since each connection is represented by four sequences, a single number does not adequately capture the intricacies that we can otherwise see via visualizations. We validate MalPaCA using temporal heatmaps since we do not have any ground truth at capability level.

We define the following properties to be indicative of good clustering: (1) Cluster homogeneity is high—a cluster contains only similar connections. (2) Cluster separation is high—each cluster captures a different capability. (3) Clusters are small and specific so they only capture the core capability. The first two properties ensure that we obtain meaningful capability-based clusters, the third ensures that only the core capabilities are captured in the clusters.

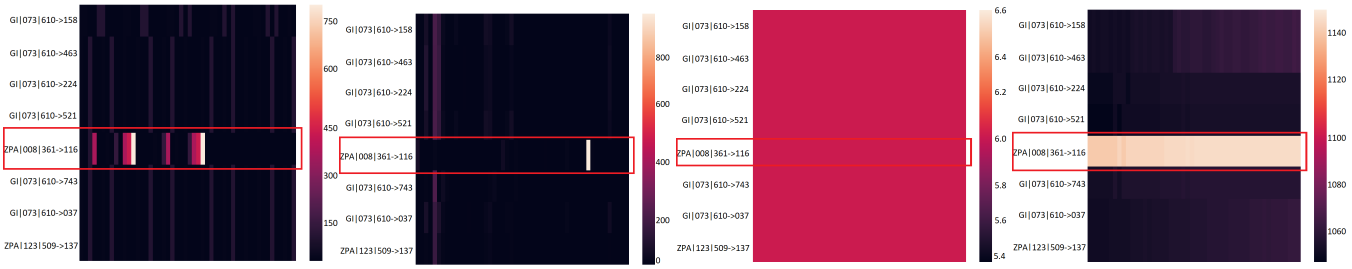
We use temporal heatmaps for a white box cluster analysis. The analyst can inspect heatmaps to determine which attacking capability is captured in a cluster. This gives malware analysts more control over the cluster debugging phase. We leave the automation of this process as future work.

Four temporal heatmaps are associated to each cluster, one corresponding to each feature. Each row in a heatmap shows the corresponding feature sequence of the first `len` packets in a connection contained in that cluster. A set of example heatmaps is shown in Figure 4. The figure highlights one dissimilar connection among the eight in the cluster.

False Positive Analysis. Because we are in an unsupervised setting without a factual ground-truth, evaluation is subjective and the resulting clusters need to be manually analyzed to determine if they are meaningful. Visualizing the cluster content helps to identify which connections do not belong in a cluster. A *false positive* (FP) is defined as a connection that is placed in cluster `X` despite half of its features being different from the remaining connections in the cluster. Since each feature currently holds equal weight, we only consider a connection as FP if more than two features *differ*. We consider two features *different* if more than 50% of their sequences differ so significantly that a different color appears on the temporal heatmap. Figure 4 shows a cluster containing one FP, highlighted in red. It shows that three out of four feature values of this connection are significantly different from other connections in the same cluster. The FP rate is calculated as $\frac{FPs}{Clustersize}$, i.e., $\frac{1}{8}$. We measure the FP rate of each cluster similarly, and calculate the *average percentage of FPs per cluster* as a notion of cluster quality.

In practice, we first establish the common majority by finding two or more connections that are most similar to each other, i.e., the ones that have the least mutual distance. The pair-wise distance matrix computed during clustering is used as a lookup table for finding such connections. We consider them the rightful owners of that cluster. Figure 4 shows a simple case where the *rightful owners of a cluster* are easily visible since 7 out of 8 connections are very similar. The rest of the connections are compared with the rightful owners and are either considered as FPs or TPs, depending on how many feature sequences differ. We do not calculate false negatives because they can be derived (over-estimated) from FPs—a FP connection in cluster `X` has its feature values much similar to cluster Y, so it is a FN for cluster `Y`.

[6] http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html



(a) Packet sizes (b) Interval (c) Source Port (d) Destination Port

Figure 4: A false positive: one connection does not belong in the cluster it is assigned

Table 2: Composition of Malware Families and Their Contribution in the Experimental Dataset.

Family name	# Malware samples
Blackmoon (B)	887 (74.10%)
Gozi ISFB (GI)	122 (10.19%)
Citadel (C)	70 (5.85%)
Zeus VM AES (ZVA)	29 (2.42%)
Ramnit (R)	22 (1.83%)
Dridex Loader (DL)	15 (1.25%)
Zeus v1 (Zv1)	10 (0.83%)
Zeus Panda (ZPa)	10 (0.83%)
Gozi EQ (GE)	7 (0.58%)
Dridex RAT Fake Pin	7 (0.58%)
Dridex (D)	6 (0.50%)
Zeus P2P (ZP)	4 (0.33%)
Zeus (Z)	3 (0.25%)
Zeus OpenSSL	2 (0.17%)
Zeus Action	2 (0.16%)
Total	**1,196** (100%)

4 EXPERIMENTAL SETUP

4.1 Dataset

MalPaCA was evaluated on malware samples collected in the wild. We worked in collaboration with a security company that specializes in malware analysis and threat intelligence. The dataset contained approximately 47k malware samples that were collected over one year. The samples were labeled using a set of YARA[7] rules proprietary to the company. Each detected sample was executed in a sandboxed environment containing several virtual machines. The resulting network traffic was stored in a Pcap file. One Pcap was stored per sample. If a sample detected it was being executed in a VM, we re-executed it in a different VM with different settings. Figure 2 depicts how connections are extracted from malware samples. In the end, we were able to collect approximately 1.1k Pcap files belonging to 15 malware families. The Pcap files generated 8997 connections. Table 2 summarizes the dataset.

4.2 Parameters

MalPaCA has four tunable parameters, i.e. *order* of the Ngrams used on port numbers, *len* of packet sequences for features, and the two parameters of HDBScan clustering algorithm: *Minimum_Cluster_Size* and *K_nearest_neighbors*.

In our experiments, we have used trigrams ($order = 3$) to represent nominal sequences. Trigrams form a good trade-off between performance and data sparsity, based on the results of Kalgutkar et al. [19]. The length of connections in the dataset is highly skewed towards shorter sequences, with a mean of 20 packets. This mean is used as *len*. Out of 8997 connections, only 733 connections (8%) are longer than the threshold. Connections from many families are discarded because of they are not adequately long for behavioral modeling, e.g. only 80 connections are retained out of 5819 Blackmoon connections. A pairwise distance matrix of dimensions 733x733 is generated using the selected distance measures. The HDBScan clustering algorithm uses $Minimum\_Cluster\_Size = 7$ and $K\_nearest\_neighbors = 7$. These parameters were selected by tuning MalPaCA on a configuration dataset, which formed 5% of the dataset. The experiments were performed on a machine with Intel Xeon E3-12xx v2 processor, 8 cores and 64GB of RAM.

4.3 Comparison with statistical features

MalPaCA clusters the temporal network behavior for semi-automated capability assessment of malware families. The clustering pipeline utilized by MalPaCA is novel in that it uses sequential features to represent behavior, uses noise-resilient distance measures and uses a robust hierarchical clustering algorithm to group similar capabilities.

Clustering methods have been used for malware classification very frequently in research [3, 16, 32, 35, 43]. A majority of these studies use statistical features for behavioral modeling. We compare the performance of using sequential versus statistical features. We use Tegeler et al. [43] as a baseline to compare our results since they not only use statistical features, but also incorporate periodic behavior using Fourier transform to detect bot-infected network traffic. Although the goal of their study diverges from ours, their feature selection approach is aligned with ours. For objectivity, we keep the rest of the pipeline as explained in Section 3. Taking guidelines from Tegeler et al. [43] and adapting them to our problem statement, each connection in the baseline is characterized by 1) average packet size, 2) average interval between packets, 3) average duration of a connection and 4) the maximum Power Spectral Density (PSD) of the FFT obtained by a binary sampling of the C&C communication.

[7] https://virustotal.github.io/yara/

Table 3: For Each Cluster, (i) No. of connections, (ii) No. of malware families, (iii) The capability label, and (iv) The traffic direction.

Cluster	# Conns	# Families	Behavior	Direction
c1	39	***9 (Common)***	SSDP traffic	Out
c2	90	***9 (Common)***	Broadcast traffic	Out
c3	9	4	LLMNR traffic	Out
c4	49	5	Systematic port scan	In
c5	56	5	Randomized port scan	Out
c6	25	***1 (Rare)***	Connection spam	In
c7	23	***1 (Rare)***	Connection spam	Out
c8	16	***1 (Rare)***	Malicious subnet	Out
c9	11	***1 (Rare)***	Connection spam	Out
c10	9	2	HTTPs traffic	Out
c11	8	2	C&C Reuse	In
c12	18	4	HTTPs traffic	In
c13	25	5	Misc.	In
c14	10	3	Misc.	In
c15	20	3	Misc.	In
c16	12	3	Misc.	Out
c17	19	3	Misc.	Out
c18	10	4	Misc.	Out

5 RESULTS AND DISCUSSION

MalPaCA produces 18 clusters from the dataset. There are, on average, 25 connections in each cluster. The algorithm discards 284 connections as noise. Table 3 enumerates the number of connections in each cluster.

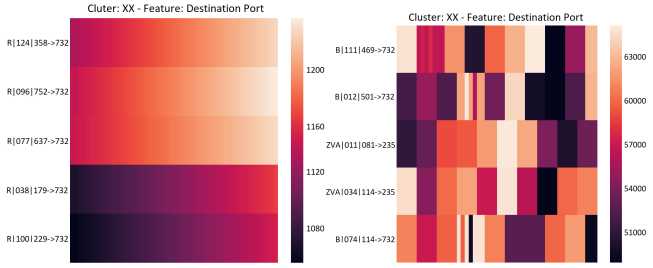
Table 3 also lists the behaviors captured by each cluster along with how popular that behavior is among the malware families. We describe a few of the interesting behaviors obtained by MalPaCA. We also contrast our clustering results with IP-based blacklisting [15], which is a very common practice in security companies.

1. Connection Direction Identification. MalPaCA successfully identifies the direction of traffic flow even though no such feature is used. The clusters and their traffic direction are listed in Table 3. Interestingly, we continue to see this pattern even when port-related features are removed from the clustering. Hence, the sequence of packet sizes and their inter-arrival time are collectively indicative of the flow direction. This important trait identifies whether the suspicious behavior is originating from inside the network or from outside it.

2. Device Probing. Some clusters capture connections that connect to the same host. For example, one cluster contains all connections broadcasting to 239.255.255.250, which is used by the SSDP protocol to find Plug and Play devices. Another cluster captures all connections broadcasting to 224.0.0.252, which is used by the Link-Local Multicast Name Resolution (LLMNR) protocol to find local network computers. These clusters could easily have been obtained by using IP-based blacklist, but they would not have clustered behaviorally similar hosts with different IP addresses.

3. Split-personality C&C Servers. In several instances, an infected host was observed responding differently to the same request, so much so that the resulting connections ended up in different clusters. For example, two connections of Gozi-ISFB contact 46.38.238.XX, which has been reported as a malicious server located in Germany. The outgoing connections are identical as they both request for the same resource. However, the responses received are very different—the first response contains a small packet followed by a series of 1200-byte packets, while the second one contains a periodic list of small and large packets in the range of 600 to 1800 bytes. This insight portrays a better picture of the behavior of said C&C server. In contrast, a blacklist would have grouped these connections since they belong to the same host.

4. Port Scan Detection. Some clusters capture a *Port Scan*[8], which is a method for determining open ports on a device in a network. Port scans are usually a part of the reconnaissance phase in the attack kill chain [47]. Utilizing sequences of port numbers enables us to detect any suspicious temporal behavior before an attack happens. The clusters identify two types of port scans: (i) *Systematic port scan* where ports are swept incrementally, which is seen as a gradient in the corresponding heat map; and (ii) *Randomized port scan* where ports are contacted randomly, which shows up in the heat map as a checkered pattern. See Figure 5. Port scans carried out by different connections are clustered together if they contact the same range of port numbers, which increases their mutual similarity. This result is in direct contrast with Mohaisen et al. [32] who conclude that port numbers are the least useful features in distinguishing malware families.



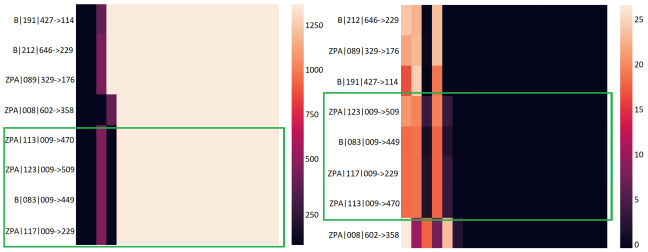
(a) Systematic port scan. **(b) Randomized port scan.**

Figure 5: Example clusters showing systematic and randomized port scans

5. C&C Reuse by Multiple Families. One cluster contains connections from different families that contact the same C&C server, and their temporal heatmaps look behaviorally identical. The cluster includes three Zeus-Panda (ZPA) connections and one Blackmoon (B) connection who contact a single IP address (encoded as 009), which has been reported as malicious. Figure 6 shows the temporal heatmaps of this cluster associated to the packet size and interval features. The said connections are highlighted in green. The source port of 6 and destination port of 80 remains constant for the whole cluster. This result suggests that either the YARA rules mislabeled one of the samples or that the authors of these samples shared the C&C server.

6. Malicious Subnet Identification. In some instances, several connections contact IP addresses that fall in the same subnet. For example, two Zeus-VM-AES connections contact one host from 62.113.203.XX subnet, while another connection detected 15 days later contacts another host in the said subnet. Similarly, two Zeus-Panda connections and one Blackmoon connection contact two hosts in 88.221.14.XX subnet. This gives actionable intelligence to

[8]https://whatismyipaddress.com/port-scan



(a) Packet sizes (b) Interval

Figure 6: Zeus Panda and Blackmoon connections reusing the same C&C server.

ISPs to investigate if other IPs in these subnets are also hosting C&C servers.

5.1 Malware Capability Assessment

By visualizing the behavioral trend of various features using temporal heatmaps, we successfully assigned labels to 12 clusters. Six clusters were left unlabelled since we could not identify the captured capability simply by exploring temporal heatmaps. It is interesting to note that these particular clusters were also the source of false positives. Table 3 shows that *SSDP* and *Broadcast traffic* are the most common behaviors and are both specific to Windows OS. Since the dataset is composed of Windows-based malware, it explains why 9 out of 12 families have connections in these two clusters. On the contrary, *Connection Spam* and *Malicious Subnet* are the rarest behaviors. *Malicious Subnet* only captures Zeus-VM-AES. Gozi-ISFB opens numerous connections, creating a *Connection Spam*. The incoming connections are stored in one cluster, while the outgoing traffic is split into two clusters due to the difference in the type of requests. This detailed behavioral analysis enables the identification of interesting clusters to analyze further. MalPaCAâĂŹs goal is to identify different behaviors in the network traffic and it does so regardless of their maliciousness and origin. Therefore, the resulting clusters can contain benign behaviors as well as malicious ones. The common clusters can be discarded if they contain known-benign behaviors, drastically reducing the number of connections to analyze.

Behavioral profiling. MalPaCA helps build behavioral profiles of malware families using the capabilities they were observed exhibiting. Table 4 lists the behavioral profiles for each malware family in our dataset. Dridex, Gozi-EQ, Zeus-P2P and Zeus-v1 only generate either *SSDP* or *Broadcast traffic*. Since this traffic is obtained from standard Windows services, it is likely that the malware was not activated when the associated Pcap files were recorded. Hence, the only connections observed from these families seem benign. On the contrary, Gozi-ISFB has the most diverse profile. Its connections are found in 16 out of 18 clusters, which exhibit attacking capabilities such as *Port Scans* and *Connection Spamming*. Specifically, the *Connection Spamming* behavior is never exhibited by any other malware family in the dataset. There are two reasons for Gozi-ISFB's diversity: (i) Gozi-ISFB is the largest family under consideration with longer sequences on average, so many of its behavioral aspects are captured; and (ii) Gozi-ISFB opens more connections per sample compared to other families. For example, one sample of Gozi-ISFB opens 111 connections, while the average number of connections for other families is 3.

Performance Analysis. The temporal heatmaps show that on average, 8.3% connections per cluster are incorrectly placed—their feature sequences are different from their member connections in a cluster. Upon further analysis, majority of the FPs originate from the last 6 clusters. The FPR is not significantly high for an unsupervised setting since not all of these connections require manual revision. The behavioral profiles are not intended for automated deployment at scale but human-in-the-loop exploration. Additionally, the overview is easy to understand and reproduce in contrast to malware family labels.

5.2 Effectiveness of Sequential Features

MalPaCA's strength is in its use of sequential features for clustering. We compare the effectiveness of MalPaCA with an existing method by Tegeler et al. [43] that is based on statistical aggregates. The baseline method results in 22 clusters, with an average of 21.2 connections per cluster. 265 connections are discarded as noise. These results are in comparison with sequence clustering – 18 clusters; on average 25 connections per cluster; 284 connections discarded as noise.

Baseline seems to perform better with smaller cluster size on average and discarding fewer connections as noise. However, their temporal heatmaps reveal important insights:

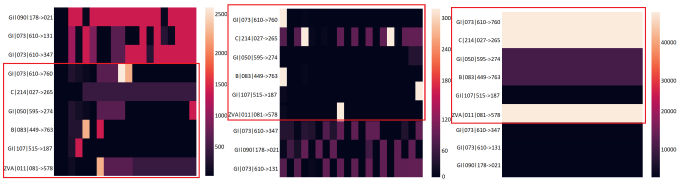
1. With statistical features, connections present in most clusters appear very different from their member connections. On average, 57.5% connections per cluster have visually different temporal heatmaps, compared to 8.3% for sequential features. Figure 7 shows a cluster from the baseline. It has nine connections, out of which six are FPs based on their behavior. The *rightful owners* of the cluster are the connections that have the least mutual distance, i.e. `GI|090|178→021`, `GI|073|610→131`, `GI|073|610→346`. The other six connections have minor differences in all features, except the source port which is 6 for all. They were primarily clustered together because their statistical features had the least mutual distance in the dataset, i.e. $average\_time\_interval = 19.77 \pm 3.11$; $fft = 0.07 \pm 0.05$; $average\_duration = 397.7 \pm 61.7$; $average\_bytes = 573.3 \pm 113.8$. The temporal heatmaps clearly show behavioral differences missed by the statistical features.

2. Statistical features are also unable to identify the direction of network traffic. In the cluster shown in Figure 7, there is one incoming connection in the cluster along with eight outgoing ones. A similar trend is observed for 19 out of 22 clusters. In contrast, sequences of packet size and inter-arrival time are enough to identify traffic direction in sequence clustering.

In summary, while statistical features may be simple to use, they lose behavioral information that plays a crucial role in accurately determining similarities in malware behavior. Different behavioral profiles may look the same from a statistical viewpoint. However, sequence clustering performs significantly better in modeling behavior than statistical features.

Table 4: Malware families' behavioral profiles. Columns are Malware Families. Rows are capabilities extracted by MalPaCA.

	B	C	D	DL	GE	GI	R	Z	ZP	ZPa	Zv1	ZVA
SSDP traffic	X	X	X	X	X	X	X	X	-	X	-	X
Broadcast traffic	X	X	-	X	-	X	X	-	X	-	X	X
LLMNR traffic	X	X	-	X	-	X	-	-	-	-	-	-
System. port scan	X	X	-	-	-	X	X	-	-	-	-	X
Random. port scan	X	X	-	-	-	X	X	-	-	-	-	X
In conn spam	-	-	-	-	-	X	-	-	-	-	-	-
Out conn spam	-	-	-	-	-	X	-	-	-	-	-	-
Malicious Subnet	-	-	-	-	-	-	-	-	-	-	-	X
In HTTPs	-	X	-	X	-	X	-	-	-	X	-	-
Out HTTPs	-	-	-	-	-	X	-	-	-	X	-	-
C&C reuse	X	-	-	-	-	-	-	-	-	X	-	-
Misc.	X	X	-	X	-	X	-	X	-	X	-	X
# Clusters	7	11	***1***	8	***1***	***16***	4	2	***1***	7	***1***	7



(a) Packet sizes **(b) Interval** **(c) Destination Port**

Figure 7: Six out of nine behaviorally different connections clustered together in baseline version.

6 LIMITATIONS AND FUTURE WORK

Limitations. Performance optimizations are needed to make sequence clustering more efficient and scalable. In MalPaCA, DTW forms the main bottleneck as the length of sequences grows longer. There exist streaming versions of DTW that compute results in real-time. One such technique is presented by Oregi et al. [33]. Moreover, using Locality Sensitive Hashing [3] can make MalPaCA scalable by reducing distance computations.

Secondly, density-based clustering discards rare events as noise. It makes sense if the dataset is noisy. However, in the presence of a purely malicious dataset, the connections that lie in lower-density regions may represent rare attacking capabilities, which will be discarded in the current implementation.

Lastly, the specificity of the identified behaviors is highly dependent on the length of sequences. The shorter sequences only capture the handshake, while longer sequences can capture the additional behavior too. At longer lengths, significantly more clusters are formed, each highly specific to a certain kind of behavior. At shorter lengths, those differences diminish, and the clusters start to merge. For example, at $len = 50$ several clusters capture slightly different variations of port scans, while at $len = 20$ all those variations merge to form only a few clusters.

Future work. There are several research directions this work can take: (i) We will work on fully automating the capability assessment of malware by building a directory of observed behaviors, which will be used for cluster labeling. (ii) We will test and improve MalPaCA's adversarial evasion resilience. (iii) We will integrate additional behavioral data sources in MalPaCA so the profiles are based on all static, system-level and network behavior. (iv) Since MalPaCA is a generic technique, we will test its applicability in building behavioral profiles for everyday-use software.

7 CONCLUSIONS

In this paper, we propose MalPaCA, an intuitive network traffic-based method to perform malware capability assessment. It clusters malware according to its capabilities using sequence clustering. We also propose a visualization-based cluster evaluation method whose key advantage is its white-box nature, allowing malware analysts to investigate, understand, and even correct labels, if necessary. We implement MalPaCA and evaluate it on real-world financial malware samples collected in the wild. MalPaCA identifies various attacking capabilities automatically, such as port scans and reuse of C&C servers. We discover a number of samples that do not adhere to their family names, either because of incorrect labeling by black-box solutions or extensive overlap in the families' behavior. We also show that sequence clustering outperforms existing methods based on statistical features by making only 8% mistakes as opposed to 57%.

MalPaCA, with its visualizations and capability assessment, can actively support the understanding of malware samples. The resulting behavioral profiles give malware researchers a more informative and actionable characterization of malware than current family designations.

REFERENCES

[1] Tony Abou-Assaleh, Nick Cercone, Vlado Keselj, and Ray Sweidan. 2004. Detection of New Malicious Code Using N-grams Signatures. In *PST*. 193–196.

[2] Blake Anderson, Subharthi Paul, and David McGrew. 2017. Deciphering malware's use of TLS (without decryption). *Journal of Computer Virology and Hacking Techniques* 14, 3 (2017).

[3] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. 2009. Scalable, behavior-based malware clustering.. In *NDSS*, Vol. 9. Citeseer, 8–11.

[4] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. *KDD workshop* 10, 16 (1994), 359–370.

[5] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, and Christopher Kruegel. 2012. Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In *ACSAC*. ACM, 129–138.

[6] Paul Black, Iqbal Gondal, and Robert Layton. 2017. A survey of similarities in banking malware behaviours. *Computers & Security* (2017).

[7] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *PAKDD*. Springer, 160–172.

[8] Lorenzo Cavallaro, Christopher Kruegel, Giovanni Vigna, Fang Yu, Muath Alkhalaf, Tevfik Bultan, Lili Cao, Lei Yang, Heather Zheng, Christopher C Cipriano, et al. 2009. Mining the network behavior of bots. *Technical Report 2009-12* (2009).

[9] Neil Wong Hon Chan. 2015. *SCANNER: Sequence Clustering of resource Access to find Nearest Neighbors.* Master's thesis. Rochester Institute of Technology.

[10] David L Davies and Donald W Bouldin. A cluster separation measure. In *TPAMI 1979*.

[11] Mohamed G Elfeky, Walid G Aref, and Ahmed K Elmagarmid. 2005. WARP: time warping for periodicity detection. In *Data Mining*. IEEE, 8–pp.

[12] Yu Feng, Saswat Anand, Isil Dillig, and Alex Aiken. 2014. Apposcopy: Semantics-based detection of android malware through static analysis. In *SIGSOFT*. ACM, 576–587.

[13] Jayant Gadge and Anish Anand Patil. 2008. Port scan detection. In *ICON*. IEEE, 1–6.

[14] Sebastian Garcia. 2015. Modelling the Network Behaviour of Malware To Block Malicious Patterns. The Stratosphere Project: a Behavioural Ips. *Virus Bulletin* (2015).

[15] Ibrahim Ghafir and Vaclav Prenosil. 2015. Blacklist-based malicious ip traffic detection. In *2015 Global Conference on Communication Technologies (GCCT)*. IEEE, 229–233.

[16] Margaret Gratian, Darshan Bhansali, Michel Cukier, and Josiah Dykstra. 2018. Identifying Infected Users via Network Traffic. *Computers & Security* (2018).

[17] Christian Hammerschmidt, Samuel Marchal, Radu State, and Sicco Verwer. 2016. Behavioral clustering of non-stationary IP flow record data. In *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 297–301.

[18] ThienLuan Ho, Seong-Je Cho, and Seung-Rohk Oh. 2018. Parallel multiple pattern matching schemes based on cuckoo filter for deep packet inspection on graphics processing units. *IET Information Security* (2018).

[19] Vaibhavi Kalgutkar, Natalia Stakhanova, Paul Cook, and Alina Matyukhina. 2018. Android authorship attribution through string analysis. In *ARES*. ACM, 4.

[20] Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Brad Miller, Vaishaal Shankar, Rekha Bachwani, Anthony D Joseph, and J Doug Tygar. 2015. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *AISec*. ACM, 45–56.

[21] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. 2007. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB journal* 16, 4 (2007), 507–521.

[22] Maciej Korczyński and Andrzej Duda. 2014. Markov chain fingerprinting to classify encrypted traffic. In *Infocom*. IEEE, 781–789.

[23] Zsolt Miklos Kovacs-Vajna. 2000. A fingerprint verification system based on triangular matching and dynamic time warping. *TPAMI* 22, 11 (2000), 1266–1276.

[24] Jehyun Lee and Heejo Lee. 2014. GMAD: Graph-based Malware Activity Detection by DNS traffic analysis. *Computer Communications* 49 (2014), 33–47.

[25] Peng Li, Limin Liu, Debin Gao, and Michael K Reiter. 2010. On challenges in evaluating malware clustering. In *RAID*. Springer, 238–255.

[26] Wei-Jen Li, Ke Wang, Salvatore J Stolfo, and Benjamin Herzog. 2005. Fileprints: Identifying file types by n-gram analysis. In *IAW, SMC*. IEEE, 64–71.

[27] Yao LI and Wen HAO. 2009. Research of encrypted network traffic type identification. *Journal of Computer Applications* 29, 6 (2009), 1662–1664.

[28] Yuping Li, Jiyong Jang, Xin Hu, and Xinming Ou. 2017. Android malware clustering through malicious payload mining. In *RAID*. Springer, 192–214.

[29] Federico Maggi, Andrea Bellini, Guido Salvaneschi, and Stefano Zanero. 2011. Finding non-trivial malware naming inconsistencies. In *International Conference on Information Systems Security*. Springer, 144–159.

[30] James Moar. 2018. Juniper Research: The Future of Cybercrime & Security. *Computer Fraud & Security* 2018, 9 (2018), 4. https://doi.org/10.1016/s1361-3723(18)30082-4

[31] Aziz Mohaisen, Omar Alrawi, Matt Larson, and Danny McPherson. 2013. Towards a methodical evaluation of antivirus scans and labels. In *International Workshop on Information Security Applications*. Springer, 231–241.

[32] Aziz Mohaisen, Omar Alrawi, and Manar Mohaisen. 2015. Amal: High-fidelity, behavior-based automated malware analysis and classification. *computers & security* 52 (2015).

[33] Izaskun Oregi, Aritz Pérez, Javier Del Ser, and José A Lozano. 2017. On-Line Dynamic Time Warping for Streaming Time Series. In *ECML-PKDD*. Springer, 591–605.

[34] Gaetano Pellegrino, Qin Lin, Christian Hammerschmidt, and Sicco Verwer. 2017. Learning behavioral fingerprints from netflows using timed automata. In *IFIP*. IEEE, 308–316.

[35] Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces.. In *NSDI*, Vol. 10.

[36] Oksana Pomorova, Oleg Savenko, Sergii Lysenko, Andrii Kryshchuk, and Kira Bobrovnikova. 2015. CN. In *ICCN*. Springer, 127–138.

[37] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987).

[38] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. Avclass: A tool for massive malware labeling. In *RAID*. Springer, 230–253.

[39] Arushi Sharma, Ekta Gandotra, Divya Bansal, and Deepak Gupta. 2019. Malware Capability Assessment using Fuzzy Logic. *Cybernetics and Systems* (2019), 1–16.

[40] Guillermo Suarez-Tangil, Juan E Tapiador, Pedro Peris-Lopez, and Jorge Blasco. 2014. Dendroid: A text mining approach to analyzing and classifying code structures in android malware families. *Expert Systems with Applications* 41, 4 (2014), 1104–1117.

[41] Mingshen Sun, Xiaolei Li, John CS Lui, Richard TB Ma, and Zhenkai Liang. 2017. Monet: a user-oriented behavior-based malware variants detection system for android. *TIFS* 12, 5 (2017).

[42] ST Tajalizadehkhoob, Hadi Asghari, Carlos Gañán, and MJG Van Eeten. Why them? Extracting intelligence about target selection from Zeus financial malware. In *WEIS*.

[43] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. 2012. Botfinder: Finding bots in network traffic without deep packet inspection. In *CoNEXT*. ACM, 349–360.

[44] Ronghua Tian, Lynn Batten, Rafiqul Islam, and Steve Versteeg. 2009. An automated classification system based on the strings of trojan and virus families. In *MALWARE*. IEEE.

[45] George Volis, Christos Makris, and Andreas Kanavos. 2016. Two Novel Techniques for Space Compaction on Biological Sequences. *WEBIST* (2016).

[46] An Wang, Aziz Mohaisen, Wentao Chang, and Songqing Chen. 2015. Capturing DDoS attack dynamics behind the scenes. In *DIMVA*. Springer, 205–215.

[47] Tarun Yadav and Arvind Mallari Rao. 2015. Technical aspects of cyber kill chain. In *SSCC*.

[48] Changhe Yu, Julong Lan, JiChao Xie, and Yuxiang Hu. 2018. QoS-aware Traffic Classification Architecture Using Machine Learning and Deep Packet Inspection in SDNs. *Procedia computer science* 131 (2018), 1209–1216.