Behavior-Aware Network Segmentation using IP Flows

Juraj Smeriga
jsmeriga@mail.muni.cz
Institute of Computer Science, Masaryk University
Brno, Czech Republic

Tomas Jirsik
jirsik@ics.muni.cz
Institute of Computer Science, Masaryk University
Brno, Czech Republic

ABSTRACT

Network segmentation is a powerful tool for network defense. In contemporary complex, dynamic, and multilayer networks, network segmentation suffers from lack of visibility into processes in the network, which results in less strict segment definition and loosen network security. Moreover, the dynamics of the networks makes the manual identification of network segments nearly impossible. In this paper, we inspect the possibilities of the behavior-aware network segmentation using IP flows and machine learning approaches that would enable to identify segments automatically, even in a complex network. We evaluate the suitability of clustering algorithms for identification of behavior-consistent segments in a network. We show that the clustering algorithms can identify relevant behavior-consistent clusters that overlap with those identified manually by experts. Apart from the segment identification, we investigate the other essential task of network segmentation process: assignment of an unknown host to an existing segment. We evaluate the performance of four different classification mechanisms on a real-world dataset. We show that it is possible to assign an unknown host to an appropriate network segment with up to 92% precision. Moreover, we release the whole dataset and experiment steps available for public use.

CCS CONCEPTS

• **Security and privacy** → *Network security*; • **Networks** → *Network measurement*.

KEYWORDS

network, segmentation, classification, clustering, IP flows, machine learning

ACM Reference Format:
Juraj Smeriga and Tomas Jirsik. 2019. Behavior-Aware Network Segmentation using IP Flows. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3339252.3339265

ARES '19, August 26–29, 2019, Canterbury, United Kingdom

ACM ISBN 978-1-4503-7164-3/19/08…$15.00
https://doi.org/10.1145/3339252.3339265

1 INTRODUCTION

Network segmentation has proven to be an effective strategy for network defense. It limits the impacts of network intrusion and improves network security. Using the firewall and routing rules, the network segmentation divides a network into smaller sub-networks that include hosts with common risk profiles [9], e.g., isolates development and production systems from each other. With the emergence of dynamic cloud environments, the high-granular segmentation often referred to as *micro-segmentation* has become vital for maintaining the network's security. The micro-segments can contain only a few hosts with similar privileges or can be limited even to individual applications.

A fundamental task in the segmentation process is a definition of rules for assigning hosts into network segments. For example, segmentation can be based on the intention-based security [12], or on the identification of data flows in a network [9]. The intention-based security assigns hosts to segments based on their behavior, while the latter approach identifies significant data flows in a network and the segments are aligned with these data flows. The above-mentioned approaches for assigning the hosts into network segments are facing several challenges when used in the context of contemporary complex and dynamic network environments, e.g., IoT networks. The complexity of these environments, the number of the connected hosts, and the dynamics of the networks (e.g., automated instance deployment) make it nearly impossible to create rules for the segmentation manually. Moreover, there is usually limited access to the hosts in the network, which reduces the visibility into relations in the network and hinders the collection of information needed for the segmentation decision. The lack of network visibility, network dynamics, and the number of hosts in a network results to a less strict definition of a segment and looser firewall rules, which opens potential sideways for adversaries to break out from a segment.

Visibility into a network, even into the complex one, can be obtained via passive network monitoring techniques. IP flow network monitoring is an effective passive monitoring technique to gain visibility in large-scale, high-speed networks [11]. An IP flow is an abstraction of a network connection that includes information from packet headers on who is communicating with whom, for how long, and on which ports, etc. IP flows can be observed at centralized observation points in a network with no necessity to access the monitored hosts. Nevertheless, the number of observed IP flows is still too high to be manually analyzed to determine relations in a network and to provide a basis for segmentation rules definition. Moreover, the IP flows represent a connection in a network, while for the host behavior-based segmentation, a host-oriented view on network traffic is required.

While the IP flow network monitoring can provide the desired visibility into a network, the rules for network segmentation still

need to be derived from the IP flows. Due to the volume of the IP flows (over thousands of IP flows per second in a mid-sized network), the derivation of the rules from IP flows becomes a big data problem, where a manual identification these rules is non-effective. Machine learning methods for automated relation discovery are expected to be suitable tools for the rules derivation from IP flow data.

The general goal of this paper is to *explore the possibilities of utilizing machine learning techniques on information from IP flows to create behavior-consistent network segments*. Specifically, we propose the following hypotheses:

(1) *The network can be divided into behavior-consistent network segments using machine learning techniques.*
(2) *It is possible to assign an unknown host to an existing network segment based on its behavior.*

The *Hypothesis 1* inspects, whether IP flow data contains behavior-consistent network segments that can be identified using machine learning, namely clustering, techniques. The discovery of such segments would enable to define segmentation rules that would reflect intention-based security network segmentation principles. Having defined the segmentation rules, the *Hypothesis 2* inspects approaches that can be used for assigning an unknown host to an existing segment using its behavior profile created from information retrieved via IP flow network monitoring. The evaluation of both above-mentioned hypotheses is carried out on the real-world data from the campus network.

The major contributions of this paper are four-fold: (1) we identify relevant features from IP flow network traffic that are suitable for host behavior modeling and can be used for network segmentation, (2) we shed light on possibilities of using clustering techniques on the selected features for creating a behavior-consistent segments and evaluate the resulting segments, (3) we compare and evaluate several classification techniques that can assign a previously unknown host to an existing segment, and (4) we make all our experiments and data available for research re-evaluation in a public repository.

The remainder of this paper is organized as follows. Section 2 describes the methodology used to examine the defined hypotheses, including problem description, algorithm selection, dataset description, and evaluation approaches. The network segment discovery is inspected in Section 3, and the network segment assignment is addressed in Section 4. Both sections describe the model training, results of the application of the relevant machine learning techniques to the defined problem, and the discussion of the results. The related works are presented in Section 5. and Section 6 concludes the paper.

2 METHODOLOGY

This section describes in detail our approach to test our hypotheses. First, we describe the problems that our hypotheses refer to. Next, we describe a dataset used to test hypotheses. Based on the problem and the dataset description, we select and describe relevant algorithms that would enable to accept or reject the hypotheses. Last, we define methods for algorithm evaluation, which enables us to draw the conclusion on the hypotheses.

2.1 Problem description

Hypothesis 1 aims to face a problem of network segment discovery. For a defined network range, we want to identify behavior-consistent clusters of machines that can be transformed into network segments. There is no prior knowledge of the network organization, no information on hosts in a network, or what services are provided on the network. The only information available is the IP flows obtained from the central network monitoring system. This setup reflects a not so rare situation of a computer security incident team that wants to introduce or revise the network security segmentation for a company. The company's network is divided into many subnets with various administrators, documentation of the networks and services are outdated or are not available, so there is only a little information on the hosts in the network at hand for the team. We are aware that in reality, some fragments of other information or data sources are usually available. Nevertheless, we choose the above-described simplified setup where only IP flows are available as it does not pose any other assumptions on information available and should work in the majority of real-work scenarios.

Given the above-described setup, we face the following challenges when testing *Hypothesis 1*. First, we need to transform the connection-oriented IP flows to host-oriented view on the network and select relevant features for network segment discovery. Second, we need to identify algorithms for segment discovery concerning their suitability for the given problem and expected performance. Next challenge is to determine a suitable number of segments to create. The optimal number of segments can be determined based on a given measure, e.g., the similarity of the behavior of all hosts within the selected segment, or a distance between the identified network segments.

Hypothesis 2 aims to inspect the problem of network segment assignment. Given the existing network segmentation, we want to categorize a new host that connects to a network into an existing network segment. This setup is an analogy to a situation, when an employee brings own device and connects it to a computer network or when a new virtual host is provisioned in a cloud environment, and security firewall rules need to be determined for this host. The host can be first placed into "quarantine" segment, where its behavior is observed, and then the host is allocated into existing segment according to its observed behavior and firewall rules are set based on the existing segment policies.

The main challenge for the above-described setup for *Hypothesis 2* is to determine the suitable algorithm for the host assignment to a network segment. The algorithm should be able to assign new hosts to the correct segment based on their network behavior and minimize the number of host misplacement. Moreover, the algorithm should be general enough to be able to place the majority of the connecting host into a segment.

2.2 Dataset

Dataset used to test the hypotheses was collected over one month period in January 2019. The observation points for the collection of IP flows were located at the borders of the university campus network. The campus university network has /16 CIDR IPv4 network range at disposal and contains various network segments from segments connecting dormitories, over server segments, to a

segment containing working stations of university administrative workers. The size of the raw IP flows used to create the dataset was over 860GB. A host in our dataset is identified by its source IPv4 address.

2.2.1 Features. First, we identified features that represent host behavior and are retrievable from IP flows. Moore et al. [18] presented a list of IP flow features that can be used for IP flow based classification. We considered features that represent host behaviors. The selected features are presented in Table 1. Since we are interested in the host's behaviour, all features represent traffic that originates at the host.

Type	Name	Aggregation
Aggregations	# of flows (FL)	src IP
	# of packets (PKT)	src IP
	# of bytes (BYT)	src IP
	Flow duration (DUR)	src IP
Distinct counts	# of peers (PEER)	src IP, dst IP
	# of ports (PORT)	src IP, dst ports
	# of protocols (PROTO)	src IP, dst ports
	# of AS numbers (AS)	src IP, dst AS number
	# of countries (CTRY)	src IP, dst country

Table 1: Host-related IP Flow Features

For each of the selected features, we computed a 24 dimensional vector, where each vector element represents a given hour in a day for each host and each day in the observed month, i.e. $FL_{(j,k)} = (fl_{(j,k),1}, \ldots, fl_{(j,k),24})$ where $j = 1, \ldots, 65536$ is host identification, and $k = 1, \ldots, 31$ is a day in January. The aggregation features sums the given feature a given variable over the given hour interval, i.e. $fl_{(j,k),1} = \sum($ all flows with src IP = host j in hour 1 of day $k)$. The distinct count features counts unique pairs described in Aggregation column of Table 1 over the given hour, i.e. $peer_{(j,k),1}$ = number of unique pairs (*src IP of host j*, *dst IP*) in hour 1 of day k. For the aggregation of the values for a given hour and host over the month, we used only business day to avoid weekend bias. Then maximum, minimum, and average functions were applied to get only one value for a given hour and host, i.e. $fl^{avg}_{j,1} = 1/n \sum_{i=1}^{n} fl_{(j,i),1}$ for average, and $fl^{min}_{j,1} = min\{fl_{(j,k),1}, k = 1, \ldots, n\}$ for minimum, where n is a number of business days in January 2019. The resulting dataset contains following tuples for each of the selected features: $(FL^{avg}_j, FL^{min}_j, FL^{max}_j)$, $j = 1, \ldots, 65536$ where $FL^{avg}_j = (fl^{avg}_{j,1}, \ldots, fl^{avg}_{j,24})$, and FL^{min}_j, FL^{max}_j analogously. This results in 648 features for each observed host. The observed features for a sample host are depicted in Figure 1.

2.2.2 Labels. All hosts in the dataset were labeled to be able to test the *Hypothesis 2*. We retrieved the list of existing segmentation of our campus network manually maintained by local CSIRT team. The existing segmentation contains the list of IP network ranges along with the description of the administrative unit and subunit to which a segment belongs. The labels contain 596 network ranges operated by the 22 different administrative units. Since the list of

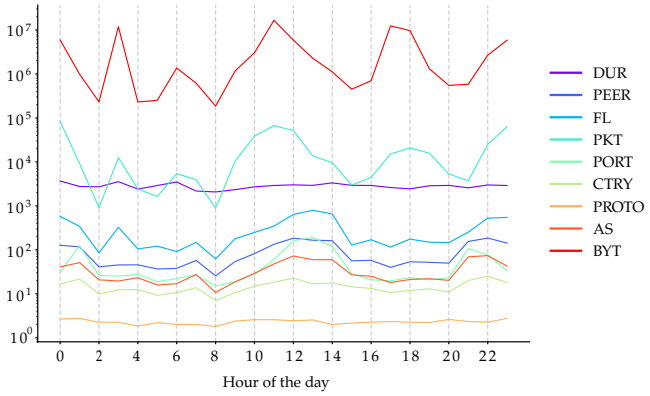

Figure 1: Observed Features for a Sample Host

the segments is manually operated and updated, it does not cover all hosts in university /16 network address range.

The obtained list was cleaned - all labels were converted to lower case to remove duplicity, we removed white spaces, and united the contact format. All hosts from the dataset were assigned with the labels according to a subnet from the list they belonged to. The hosts that did not fit into any network range in the list were labeled with NA values. Available labels for hosts in our datasets are summarized in the Table 2.

Name	Description
Range (RNG)	A network range a host belongs to
Unit (UNT)	An administrative unit owning the network range
Sub-unit (SUB-UNT)	An sub-unit of the unit
Contact (CONT)	Contact to an administrator

Table 2: Dataset Labels

2.2.3 Preprocessing. The raw dataset created from IP flow was further preprocessed so it could be used for machine learning tasks. First, we removed missing values and outliers. Next, we standardized the numerical variables and sampled the unit counts to provide a balanced dataset. Last, we anonymized values in the dataset so it could be publicly released.

The dataset contained two types of missing values before preprocessing: the host's records with missing labels and records with labels but missing from one to all observations. There were 6 016 hosts of all hosts in /16 range (9.18 %) with missing labels, and 28 012 hosts with missing values in all features but labels (42.74 %) (i.e. hosts that were assigned to a subnet but did not communicated during the whole observation period). All these records were removed from the dataset leaving 31 501 hosts in the dataset. These hosts communicated at least once during the observation period. Other missing values, e.g., missing observations in only one hour, were replaced with zeros.

Next, we removed outliers observations from the dataset using the 0.95 quantile threshold. Having removed the extreme values,

we standardized the dataset as the standardization improves the performance of distance-based clustering techniques. Each feature was scaled to zero mean and unit variance. During the exploration analysis of the dataset, we discovered, that 44.45% hosts belongs to the one unit, while shares of other units were about 4%. Hence, we undersampled this unit to achieve a more balanced dataset for clustering. We employed a random approach where we randomly pick records from the majority class to reduce it to 25% of its original size.

Last, we anonymized the dataset for public use. The host's IP addresses were hashed, and so were the network ranges. Further, we removed labels containing the contact information for a network subnet administrator. We share the anonymized dataset with the public at https://github.com/CSIRT-MU/BehaviorNetworkSegmentation.

2.3 Algorithms

The description of the problems linked to our hypotheses displays that there are two different algorithms needed. While the problem of *Hypothesis 1* suggests to use clustering algorithms as the task is to divide the hosts into previously unknown clusters, the problem related to *Hypothesis 2* implies a classification task where an item (a host) needs to be assigned to the one from the existing categories (segments). Hence, we provide a description of the algorithm selection process and the algorithm's basics for each hypothesis separately in the paragraphs below.

2.3.1 Network Segment Discovery Algorithms. For the network segment discovery task, we apply clustering algorithms. We choose to employ the *K-Means* [17] and *Density-based spatial clustering of applications with noise* (DBSCAN) [8] algorithms as both algorithms can process data with a high number of samples, are commonly applied in wide range of applications, and provide a different approach to the determination of the number of the cluster. The K-means algorithm needs to specify the number of clusters a priori, while the DBSCAN does not need such information and estimates the number of clusters by itself.

K-Means algorithm aims to divide set on n samples X into k disjoint groups C with equal variance while minimizing the intra sum of squared criterion [2]:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (\| x_i - \mu_i \|) \tag{1}$$

The iterative process repositions the initial cluster centers to be in the center of the updated clusters created by assigning points to its nearest cluster center. The above-mentioned implementation uses Euclidean distance and takes each feature to determine a distance from a cluster center. However, in our setup, where a feature a vector of 24 values, the Euclidean-based K-means can suffer from the curse of the dimensionality.

Hence, we decided to investigate also an alternative approach and handle the features as a time series. This approach reduces the dimension to the number of features. We used Dynamic Time Wrapping (DTW) method [4], specifically LB_Keogh distance measure [15] to determine a distance from the cluster center. Such that, we can cluster the features as time series using the K-Means algorithm. We expect an improved performance compared to simple K-Means as the time series approach takes time causality into account.

DBSCAN algorithm creates the clusters based on the distance to the points in the neighborhood. The DBSCAN algorithm requires the size of the neighborhood ϵ and the minimum number of points in a neighborhood *minPts*. DBSCAN algorithm can be simplified as [25]:

(1) For a random non-noise point retrieve ϵ-neighborhood. If the neighborhood contains more than *minPts* points, a cluster is started. Otherwise, the point is marked as noise.
(2) All points from the ϵ-neighborhood are part of the cluster, and so are points in their ϵ-neighborhood. This process continues until the whole cluster is found.
(3) A new unvisited random non-noise point is selected, and the process restarts.

The advantage of the DBSCAN is that it does not require to define the number of the cluster as input. Moreover, it is robust to outliers and can find arbitrary shaped clusters, compared to linear cluster separation used in K-Means.

2.3.2 Network Segment Assignment Algorithms. The segment assignment task used to test *Hypothesis 2* implies the application of the classification algorithms. We choose to employ *k-nearest neighbors* (KNN), *support vector machine* (SVM), and decision trees (DTs) algorithms to label new host into an existing segment based on its behavior. These algorithms are selected to be the representatives of the common approaches to classification tasks.

K-nearest neighbors [7] assigns a new object by a plurality vote of it's k neighbors. During the training phase, the algorithm stores the class labels and feature vectors of the training objects. During the classification of an object, the algorithm searches the k-nearest objects to the classified object based on the given distance measure. Analogous to clustering algorithms, we employ both Euclidean distance and Dynamic Time Wrapping with LB_Keogh distance. The number of neighbors k used to classify a new observation need to be determined as input and influences the performance of the classification algorithm. The optimal number of the neighbors can be determined from the comparison of prediction error rate and k-value. Usually, there exists a threshold for k, beyond which there is not a significant drop in the error rate.

Support vector machine classification [6] constructs a set of hyper-planes with the largest distance to the nearest training data points of any class. These planes are then used for the classification of the new object. The larger distance from the data points, the lower is the error of the classifier. SVM is usually used as a non-probabilistic binary linear classifier. Nevertheless, the non-linear classification is possible as well by using the kernel trick that maps the input feature vector into high-dimensional space where hyper-planes can be found.

Decision tree classifiers are a non-parametric method that creates a tree-like model where each leaf node represents a class label. The classification rules are then paths from the root to a given node. The advantages of the decision trees are that they are simple to understand, require little data preprocessing leading to better performance on datasets with violated assumptions, and they can handle multi-output problems. The disadvantage of the decision

trees is that they are unstable as a small variation in the data can result in a different decision tree. The optimal decision tree is the tree that has greater coverage while minimizing the number of levels.

2.4 Evaluation

The evaluation of the algorithm performance is conducted separately for classification and clustering tasks.

The evaluation of the clustering algorithms does not implicitly require labels in the data. The clusters can be evaluated with respect to their density, homogeneity, or inter-cluster distance, for example. Nevertheless, when original labels are available, there exist metrics to compare original labels with the original clusters.

For the evaluation, where no labels are available, we employ the *Silhouette score* [23], to show the clustering performance. The Silhouette coefficient uses the mean intra-cluster distance a and mean inter-cluster distances b. The coefficient is computed as:

$$s = \frac{b - a}{\max\{a, b\}} \tag{2}$$

The Silhouette coefficient return values in $< -1, 1 >$. Negative values of the coefficient imply that the sample has been assigned to the wrong cluster, values near 0 mean overlapping clusters and values near 1 indicate the best clustering results.

Using the labels, we evaluate clustering methods using the *adjusted Rand index* [22]. The Rand index (RI) is used to compare two clusterings. It takes all pairs of samples from the compared clusterings and counts pairs assigned to the same or different clusters. The RI is dent adjusted for permutations to compare the clustering to random clustering as:

$$ARI = \frac{RI - \text{expected\_RI}}{\max(RI) - \text{expected\_RI}} \tag{3}$$

The ARI is expected to return values around 0 for random labeling, negative values independent clustering and values close 1 for the exact match for between the clusterings. Assuming that one of the compared clusterings are the actual labels, we can compare the clustering results to the real-world state.

For the evaluation of the classification methods, we first split our dataset to train and test data sample in 80 : 20 ratio. The train and test samples are chosen randomly from the original dataset. The classification model is estimated on the train data set and then applied to test dataset to predict the labels. The predicted and actual labels of the testing dataset are compared, and the precision, recall, and F-score metrics are computed. The overview of the algorithms used to test hypotheses and their evaluation methodology is provided in Table 3.

Hypothesis	Algorithms	Evaluation
Hyp. 1	K-Means K-Means with LB_Keogh DTW DBSCAN	Silhouette (s), adj. Rand index (ARI)
Hyp. 2	KNN KNN with LB_Keogh DTW SVM DTs	precision, recall, F-score

Table 3: Methodology Overview

3 NETWORK SEGMENT DISCOVERY

3.1 Model Training

Both algorithms for testing *Hypothesis 1* were applied on the whole data set described in Section 2.2.

For the K-Means algorithm, we determined the number of clusters parameter by taking the total amount of administrative units in our data set, i.e., 22. This choice allowed us to evaluate the results against the ground truth representing the real-world segmentation by the administrative units. The initial cluster was selected randomly, and the maximum number K-Means iterations was set to 300. K-means with LB Keogh as a distance metric is computationally difficult. Therefore we limited the maximum number of iterations to 30.

The training the DBSCAN required to determine values for ϵ and $minPts$. We began by taking $minPts = 2 * number\_of\_features$ and then computing ϵ by running KNN algorithm with $k = minPts$ as suggested in [25]. The Figure 2, we plotted a graph of distances to k nearest neighbors in ascending order and chose ϵ by recognizing an elbow point at a distance equal to 160.

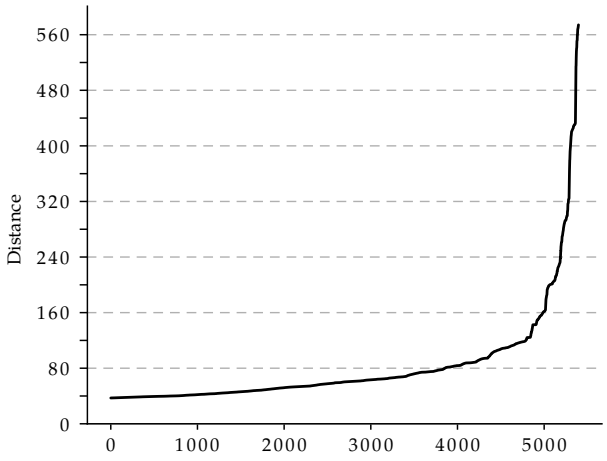

Figure 2: ϵ selection from KNN distances

However, these parameters did not lead to the identification of coherent and clear clusters. Therefore we shifted to parameter grid search to determine which parameters show the best results. We performed multiple grid searches, then gradually refined the grid parameters according to the accuracy of the results. Initial grid search parameters were: $minPts = [25, 50, 100, 200, 400]$, $\epsilon = [1, 2, 4, 8, 16, 32, 64, 128]$. The grid search settled with $minPts$ set to 40 and ϵ set to 5.

Dynamic time warping is generally computationally expensive with its $O(nm)$ time complexity, $n(m)$ being the length of the first (second) times series. LB_Keogh provides an improvement in speed. It takes in parameter r (reach). Based on our findings we decide to set $r = 5$.

3.2 Results

The results of network segment discovery are presented in Table 4. The Silhouette coefficient for all clustering algorithms suggests that there are overlapping clusters present. The DBSCAN algorithm provides the best clustering results. For the K-Means algorithm, the overlapping clusters can be explained by the fact, that $k = 22$ was set to the total number of the administrative units. Since the administrative units represent the organizational division of the network, it is possible, that in reality, there are less behavioral-based clusters in the network. Given the higher number of the administrative-based clusters and lower number of the actual behavior-based clusters, it is natural, that the identified clusters will overlap. This reasoning is supported by the fact, that DBSCAN algorithm estimated the optimal number of behavior-based clusters to 7.

Considering the ARI metrics, that compares the estimated clustering with the ground truth clusters, DBSCAN algorithm performed even an independent clustering from the ground truth. The best performance with ARI equal to 0.3 was recorded by the K-Means algorithm. These results reflect the fact that there are hosts with similar behavior presented in multiple administrative units. When considering the hour features as individual variables, the similarity among hosts over multiple administrative units is higher than when a feature is taken as the a time series of 24 observation in LB_K_DTW clustering.

Algorithm	Silhouette	ARI
K-Means	0.02	0.30
K-Means_LB_K-DTW	0.07	0.16
DBSCAN	0.08	-0.13

Table 4: Network Segment Discovery Results

3.3 Discussion

The results computed on the dataset representing the whole network show that the clustering algorithms tend to identify lover number of behavioral clusters in the network compared to the number of logical segments identified in the network. When the number of the cluster to create is set to be equal to the (higher) number of the segments presented in a network, the clusters overlap. Hence, the behavior-based network segmentation based on clustering is suitable for networks, where segmentation to a lower number of segments is sufficient.

Given the results above, we decided to further inspect the performance of the clustering algorithms on the subset of a given network. From the original dataset, we selected five administrative units (*reduced segment)*, that has different agenda and purpose, e.g., a segment for cloud computing, faculty segment, a segment for bureaucracy services, etc. Hence, the behavior of the hosts is expected to be more different from each other. To test an extreme case, when a segmentation to only two segments is required, we also tested the binary clustering (*binary segment* on two selected segments from the original dataset. The results of our further analysis are presented in Table 5.

Considering the clustering evaluation without labels, the Silhouette coefficient did not improve significantly. However, we can

Segment	Algorithm	Silhouette	ARI
Reduced	K-Means	0.001	0.92
	K-Means_LB_K-DTW	0.13	0.22
	DBSCAN	0.19	-0.04
Binary	K-Means	0.001	0.95
	K-Means_LB_K-DTW	0.03	0.13
	DBSCAN	0.001	-0.14

Table 5: Reduced Segment Discovery Results

observe major improvement in values of the adjusted Rand Index that compares the estimated clustering with the clustering based on real-world labels. Both for reduced and binary segments, the ARI for K-Means is near one (0.92, 0.95 respectively), which implies near-perfect match with the label-based clustering. We explain the major improvement by the setting the lower number of clusters k for K-Mean compared to the original experiment on the whole dataset and making it equal to the number of labels. So, the algorithm can perform reasonable clustering aligned with real-world settings.

4 NETWORK SEGMENT ASSIGNMENT

4.1 Model Training

KNN algorithm uses classes of k nearest neighbors of new data point to classify it by majority voting. Since it is difficult to estimate this parameter in advance accurately, Figure 2 shows the process of finding the elbow point on a graph by plotting various choices of k against the error rate. We settled with $k = 5$, since higher values did not provide significantly better classifier accuracy.

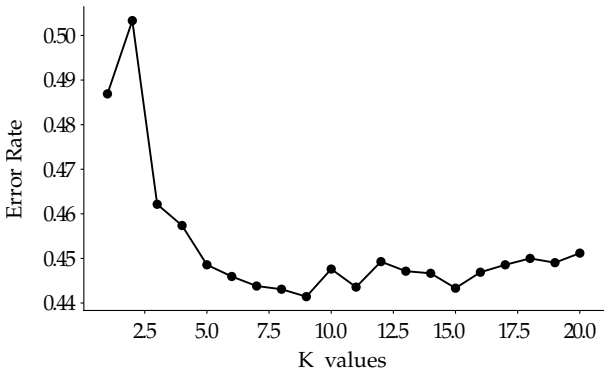

Figure 3: KNN - Error Rate vs. k values

Regarding SVM, we used a polynomial kernel with a degree of 3. Penalty parameter, C, and kernel coefficient, *gamma*, were both picked by executing grid search. Grid parameters were set as following: $C = [1, 10^{-1}, 10^{-2}, 10^{-3}]$, $gamma = [auto, 1, 0.1]$, $kernel = [rbf, polynomial]$. $C = 0.01$ and $gamma = 1$ provided the most accurate results. We did not set a hard limit on number of iterations and all classes had weight set to 1.

When implementing DTs we chose the following parameters: Gini impurity as criterion to measure split quality, *bestsplit* as

splitting strategy, maximum features to consider when splitting node set to number of all available, we did not set any *max_depth* of the tree, minimum number of samples required to split an internal node was set to 2, minimum samples required to be a leaf node was set to 1.

4.2 Results

The results for the classification of an unknown host to an existing segment using machine learning techniques are presented in Table 6.

In general, the classification algorithms were able to correctly classify around 50% of hosts that according to the ground truth belonged to a given segment. The best recall was observed for the SVM algorithm. The SVM also showed the best where 62% of the hosts assigned to a segment belonged to the segment according to the labels. The worst results showed the KNN with LB_Keogh DTW metric, which suggests that time causality captured by this similarity measure is not essential for clustering hosts by their behavior.

Algorithm	Precision	Recall	F-Score
KNN	0.56	0.52	0.52
KNN_LB_K-DTW	0.40	0.41	0.36
SVM	0.62	0.62	0.60
DTs	0.61	0.61	0.61

Table 6: Network Segment Assignment Results

The precision and recall near 50% can be explained by the fact that the labels were obtained from the administrative-based segmentation of the network. The administrative segments represent logical segregation of a network and do not entirely reflect the behavior-based network segmentation. Hence, hosts with similar behavior are present in multiple administrative segments. For the classification mechanism is then hard to identify a robust segregation rule, that would assign these hosts to the right cluster.

4.3 Discussion

As discussed above, the lower performance of the classification methods applied on the dataset is explained by the facts that the administrative segregation does not entirely reflect the behavior-aware segmentation of a network. The noise into the in classification process is introduced mainly by a small and "fuzzy" administrative units present in our dataset.

To inspect the performance of the classification methods in detail, we decided to run the experiment on the dataset, where the noise of the small administrative units is removed. Analogous to the network segmentation identification experiment, we selected a subset of 5 network segments (*reduced segment*), where both administrative-based and behavior-based segregation overlap. Using this subset, we reviewed the results of all clustering algorithms except the KNN_LB_K-DTW as the time causality aware measure showed as unnecessary in original results. Analogously, we executed the experiment of binary classification where hosts were assigned to two different network segments (*binary segment*). The results of our further analyses are presented in Table 7.

Segment	Algorithm	Precision	Recall	F-Score
Reduced	KNN	0.75	0.74	0.74
	SVM	0.82	0.81	0.81
	DTs	0.78	0.78	0.78
Binary	KNN	0.87	0.87	0.87
	SVM	0.90	0.90	0.90
	DTs	0.92	0.92	0.92

Table 7: Reduced Segment Assignment Results

We observe that the performance of the classification methods on the reduced segment improved significantly. The hosts from the classified host, the 82% were classified to the correct cluster with 81% recall in the case of SVM. For the binary classification, the best performance showed the DT algorithms with the precision and recall of 92%.

5 RELATED WORKS

Techniques for network traffic classification using machine learning tools are surveyed by Nguyen and Armitage in [19]. The authors describe techniques both for clustering and classification of network traffic and identify challenges for operational deployment of the ML tools. Buczak and Guven surveys machine learning techniques for cyber security intrusion detection in [5]. Besides the inspection of ML methods for cyber security, the survey also provides information on the analysis workflow and cyber security data sets.

The majority of the previous works focus on the profiling network behavior of individual hosts [3, 10, 26], their classification [14, 16] and clustering [21]. The authors of [10] use the change point detection techniques and the indicator of "freshness" to cluster the hosts according to its different activities over time. IP flow records are used as a data source, and the authors test their approach to modeling the different behavior of botnet hosts from real-world data observation. IP flow statistics are also used for user behavior identification in [3]. Author record user behavior of 43 users as they are using nine different applications (e.g., Youtube, Facebook) in the form of IP flows. Using statistical aggregations such as average, skewness, or variance mean ratio, they aim to devise rules for host behavior identification. For that task, they employ SVM with recursive feature extraction with over 80% average accuracy to identify application a user is using. Xu et al. [26] proposes a new perspective to profiling network host by identification of behavior clusters of hosts in the same network prefixes. They create bipartite graphs of network communications with one node projection and supplies them to the spectral clustering algorithm to discover behavior clusters in network blocks. Applications in domains such as network traffic pattern discovery or detection of anomalous behavior are presented.

BLINC [14] is a tool for the identification of host behavior patterns at the transport level. It analyses host patterns at three levels - the social, functional, and application level. The authors of the tool impose additional constraints on the source data such as no access to the payload or the fact that well-know port does not have to link to the application directly. For the classification, they

employ cumulative distribution functions and bipartite graphs for modeling host social behavior, and graphlets for application level host modeling. The BLINC is claimed to classify approximately 80% flows with 99% accuracy. Performance of various machine learning techniques for host roles classification is evaluated in [16]. The authors try to classify hosts to different binary groups, e.g., client vs. server, web non-email server vs. web email server, or the personal office vs. public place connected host. The compared ML techniques are on-line SVM and decision trees on the data retrieved from sFLows. Qin et al. describes a multilevel user cluster mining (MUCM) to identify user cluster in the monitored network with different lengths of network prefixes in [21]. The clustering uses user's behavior similarity measure, adaptive selection of weights and K-Means to identify the clusters in the network segment. The identified clusters are further analyzed from different perspectives. Authors analyze the number of users in the clusters, traffic volumes per cluster, user's behavior dynamics. Application of the clustering on botnet detection is demonstrated in this paper.

The related works on the identification of host communities, or network segments include [1, 13, 20, 24]. Proença et al. create a digital signature of the network segments using real-world data in [20]. The digital signature is computed BLGBA algorithm that takes the distribution of elements in frequencies based on the difference between the greatest and smallest element in a sample. In [1], the authors explored the possibilities of identification of the maximal cluster of hosts based on the proposed data fusion and clustering algorithms. They aim to create a cluster of hosts whose network behavior is similar with regards to intrusion detection. For the identification of intrusion flows of a host, the authors employed supervised learning algorithms such as Decision Trees, KNN, and SVM. Jakalan et al. propose a methodology of supervised clustering of IP addresses based on IP communication structure in [13]. For clustering, they create bipartite networks from IP flow record, apply one-mode projection, and build a social similarity graph of the inside IP addresses. Network segregation for Industry 4.0 is investigated in [24]. The authors identify communication patterns in an Industry 4.0 Cyber-Physical Production Systems. They apply DTs, Naive Bayes, Bayesian network, SVM, and KNN on the manually labeled dataset to learn the regularities in communication pattern. The communication patterns are used to classify the hosts into segments in the network.

6 CONCLUSION

The primary goal of this paper was to explore the possibilities of utilizing machine learning techniques on information from IP flows to create behavior-consistent network segments. In the paper, we describe relevant features for host behavior modeling, including the formal description of their retrieval from IP flow records, their preprocessing. From these features, we create an annotated dataset that is used to evaluate further hypotheses.

Having created the dataset, we evaluated three clustering techniques to test the hypothesis that a network can be divided into behavior-consistent network segments using machine learning techniques. Our findings showed that it is possible to cluster network into behavior-consistent segments. However, we discovered that the number of behavior-consistent clusters identified in a network is significantly lower than the number of administrative-based clusters. Compared to real-world settings, the clustering identifies a lower number of network segments. Hence, the clustering approach is suitable when a lower number of network segments is sufficient for real-world network segmentation. When we tested the clustering methods on a set of administrative units that are also behavior-consistent, then the clustering performed well and identified nearly the same clusters as were the administrative ones.

Next, we test the hypothesis, that it is possible to assign an unknown host to an existing network segment based on its behavior. Using the identified host behavior features, we employed three classification algorithms to test this hypothesis. We discovered that the precision and recall of the classification algorithms used for classification to all administrative clusters present in the dataset are 62% at best as the behavior characteristics of hosts in different administrative units overlaps. Further analysis, where we selected only behavior-different administrative units showed, that the precision can reach the rate of 92% correctly classified hosts.

Our further research will include the identification and investigation of additional host behavioral features, that can be retrieved from IP flow records. We will mainly focus on application level information present in the IP flow records. We aim to inspect further characteristics of host features that may influence network segment identification and assignment, the stability of host behavior in time, for example. We also plan to evaluate the deep learning classification approaches as they are becoming the mainstream in classification research.

The analyzed dataset, including all executed experiments in the form of Jupyter notebooks, is available in public repository at https://github.com/CSIRT-MU/BehaviorNetworkSegmentation to enable a complete re-validation of our research.

ACKNOWLEDGMENTS

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

REFERENCES

[1] Lena Ara and Xiao Luo. 2018. Identify the Maximal Cluster of Hosts Based on Data Fusion and Machine Learning Algorithms for Intrusion Detection. In *Proceedings - 4th IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2018, 4th IEEE International Conference on High Performance and Smart Computing, HPSC 2018 and 3rd IEEE International Conference on Intelligent Data and Securit.* IEEE, 42–46. https://doi.org/10.1109/BDS/HPSC/IDS18.2018.00022

[2] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: the Advantages of Careful Seeding. In *Proc ACM-SIAM symposium on discrete algorithms.*, Vol. 8. 1027–35. https://doi.org/10.1145/1283383.1283494

[3] Shingo Ata, Yusuke Iemura, Nobuyuki Nakamura, and Ikuo Oka. 2017. Identification of user behavior from flow statistics. In *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS).* IEEE, 42–47. https://doi.org/10.1109/APNOMS.2017.8094176

[4] D Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD-94 Workshop on Knowledge Discovery in Databases (AAAIWS'94).* AAAI Press, 359—-370. https://doi.org/10.1016/j.cgh.2009.10.031

[5] Anna L. Buczak and Erhan Guven. 2016. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

[6] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (sep 1995), 273–297. https://doi.org/10.1007/BF00994018

[7] T. M. Cover and P. E. Hart. 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13, 1 (jan 1967), 21–27. https://doi.org/10.

1109/TIT.1967.1053964

[8] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 226–231.

[9] Monica Givati. 2018. What is Micro-Segmentation. Retrieved March 10, 2019 from https://www.guardicore.com/micro-segmentation/what-is-micro-segmentation/

[10] Christian Hammerschmidt, Samuel Marchal, Radu State, and Sicco Verwer. 2017. Behavioral clustering of non-stationary IP flow record data. In *2016 12th International Conference on Network and Service Management, CNSM 2016 and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016*. IEEE, 297–301. https://doi.org/10.1109/CNSM.2016.7818436

[11] Rick Hofstede, Pavel Celeda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. 2014. Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys and Tutorials* 16, 4 (2014), 2037–2064. https://doi.org/10.1109/COMST.2014.2321898

[12] Nic Jackson. 2018. Network segmentation in modern environments. Retrieved March 11, 2019 from https://www.hashicorp.com/blog/network-segmentation-in-modern-environments

[13] Ahmad Jakalan, Jian Gong, Qi Su, and Xiaoyan Hu. 2015. Community Detection in large-scale IP networks by Observing Traffic at Network Boundary. In *WORLD CONGRESS ON ENGINEERING AND COMPUTER SCIENCE, WCECS 2015, VOL I*. 59–64. https://www.semanticscholar.org/paper/Community-Detection-in-large-scale-IP-networks-by-Jakalan-Gong/8ab13041d18dc16d4c71bfc30f2e05efcbe9b11e

[14] T Karagiannis, K Papagiannaki, and M Faloutsos. 2005. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*. ACM, New York, NY, USA, 229–240. https://doi.org/10.1145/1080091.1080119

[15] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7, 3 (mar 2005), 358–386. https://doi.org/10.1007/s10115-004-0154-9

[16] Bingdong Li, Mehmet Hadi Gunes, George Bebis, and Jeff Springer. 2013. A supervised machine learning approach to classify host roles on line using sFlow. In *Proceedings of the first edition workshop on High performance and programmable networking - HPPN '13*. ACM Press, New York, New York, USA, 53. https://doi.org/10.1145/2465839.2465847

[17] J. Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Mathematical Statist. Probability*. University of California Press, 281–297.

[18] Andrew Moore, Denis Zuev, and Michael Crogan. 2005. Discriminators for Use in Flow-based Classification. *Queen Mary and Westfield College, Department of Computer Science* 16, August (2005), 2005. https://doi.org/10.1.1.101.7450

[19] Thuy T.T. Nguyen and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials* 10, 4 (2008), 56–76. https://doi.org/10.1109/SURV.2008.080406

[20] M. Lemes Proença, C. Coppelmans, M. Bottoli, A. Alberti, and L. S. Mendes. 2004. The Hurst Parameter for Digital Signature of Network Segment. In *Telecommunications and Networking - ICT 2004: 11th International Conference on Telecommunications, Fortaleza, Brazil, August 1-6, 2004. Proceedings*. Springer, Berlin, Heidelberg, 772–781. https://doi.org/10.1007/978-3-540-27824-5_103

[21] Tao Qin, Xiaohong Guan, Chenxu Wang, and Zhaoli Liu. 2015. MUCM: Multilevel User Cluster Mining Based on Behavior Profiles for Network Monitoring. *IEEE Systems Journal* 9, 4 (dec 2015), 1322–1333. https://doi.org/10.1109/JSYST.2014.2350019

[22] William M. Rand. 1971. Objective Criteria for the Evaluation of Clustering Methods. *J. Amer. Statist. Assoc.* 66, 336 (dec 1971), 846–850. https://doi.org/10.1080/01621459.1971.10482356

[23] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20, C (nov 1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[24] Firooz B. Saghezchi, Georgios Mantas, José Ribeiro, Alireza Esfahani, Hassan Alizadeh, Joaquim Bastos, and Jonathan Rodriguez. 2019. Machine Learning to Automate Network Segregation for Enhanced Security in Industry 4.0. In *9th International Conference on Broadband Communications, Networks, and Systems (Broadnets)*. Springer, Cham, Faro, Portugal, 149–158. https://doi.org/10.1007/978-3-030-05195-2_15

[25] Erich Schubert, Martin Ester, Xiaowei Xu, Hans Peter Kriegel, and Jörg Sander. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems* 42, 3 (jul 2017), 1–21. https://doi.org/10.1145/3068335

[26] Kuai Xu, Feng Wang, and Lin Gu. 2011. Network-aware behavior clustering of Internet end hosts. In *Proceedings - IEEE INFOCOM*. IEEE, 2078–2086. https://doi.org/10.1109/INFCOM.2011.5935017