Cross-Network Behavioral Clustering for Managed Security Service Providers

Georgios Gkroumas
Technology R&D
PCCW Global
Athens, Greece
ggroumas@pccwglobal.com

Ilias Kotinas
Technology R&D
PCCW Global
Athens, Greece
ikotinas@pccwglobal.com

Kostas Giotis
Technology R&D
PCCW Global
Athens, Greece
kyiotis@pccwglobal.com

Theocharis Tsigkritis
Technology R&D
PCCW Global
Athens, Greece
ttsigkritis@pccwglobal.com

Paris Mermigkas
Technology R&D
PCCW Global
Athens, Greece
pmermigkas@pccwglobal.com

***Abstract*—Managed Security Service Providers (MSSP) oversee and protect customer networks often varying in the level of pre-installed security defense appliances and capabilities. This imbalance makes it challenging to offer a consistent level of service across clients. This paper presents an approach for raising the level of defense by indirectly utilizing the threat detection capabilities of secure networks based on behavioral similarity among cross-network entities. Initially, we present a holistic architectural view of the components we have deployed in order to efficiently ingest, process and analyze massive amounts of raw data from various perimeter network devices of MSSP customers. Our data analysis approach is based on entities clustering and a risk estimation routine leveraging on noisy labeling derived from advanced security appliances. Specifically, entities are clustered according to a list of aggregated metrics, characterizing the communication between local and remote network devices. Finally, we present the rationale behind the adoption of specific clustering approaches, as well as an optimization routine implemented to appropriately select the free parameters of the clustering sub-process.**

Keywords— clustering, risk assessment, MSSP, network security

I. INTRODUCTION

In this paper we propose a behavioral clustering approach tailored for Managed Security Service Providers (MSSPs). An MSSP provides outsourced monitoring and management of security devices and systems. Common services include managed firewall, intrusion detection, virtual private network, vulnerability scanning and anti-virus services as further detailed in [1]. Offering a consistent level of service across clients is a challenging goal for MSSPs when the managed equipment varies and is out of the control of the supervising security engineers. The monitored appliances for a given infrastructure may range from simple routers and commodity firewalls (defined in this paper as Capability Level 1 - *CL1*) up to Next Generation Firewalls (NGFW) and Intrusion Detection or Prevention systems (IPS/IDS) (respectively defined as *CL2*). The equipment is often pre-installed and pre-configured and often purely maintained in terms of up to date signatures and threat intelligence. The privacy requirements and data regulations along with a tendency of customers to prefer non-intrusive services usually do not allow providers to install traffic capture agents to increase their ability to analyze traffic and provide advanced security measures. Therefore, MSSPs are typically left with metadata-only access through audit logs which typically contain IP flow level information and policy violations (firewall allow/deny events).

The question is then what an MSSP could do to improve the security service to the more exposed clients? There are benefits in being a cross-network provider which have not been widely studied in the research community nor been employed in commercial solutions. The most important one is that a service provider is capable of employing cross-network metadata analysis techniques which may improve the predictive power of simple metadata like IP flows and policy violations for threat detection. The observation that malicious activities often leave network fingerprints can be exploited if used along with feedback (i.e. threat events) from advanced security appliances (*CL2*) in order to infer suspicious network behaviors for networks that might lack those security

978-3-903176-14-0 © 2018 IFIP

Line of research and development work within PCCW Global, the international operating division of HKT. (https://www.pccwglobal.com/en/about)

appliances (*CL1*). Another benefit for providers is offered by the advanced data infrastructure available in MSSPs, which allows to keep historical information for longer and draw more powerful conclusions from data.

In this paper, a behavioral clustering framework is presented, which takes advantage of the aforementioned cross-network access to metadata and infers suspicious behavior for internal entities (i.e. IP addresses of overseen assets) of CL1 networks by their similarity to entities in CL2 networks. More specifically, entities are clustered according to aggregated metrics of network traffic and the resulting clusters are separately evaluated and characterized by the number of threats reported in security appliances of the CL2 networks. The entities of the CL1 networks are then associated with the risk score of their behavioral cluster and appropriate alerts are raised for early notification or analysis prioritization.

The paper is organized as follows. Section II discusses related work. In section III we present the high-level architecture and the components comprising our suggested analysis pipeline. The section IV details our algorithmic approach for clustering, feature selection and parameters optimization. We present our experimental setup and evaluation results in section V. Finally, we conclude our study and present future directions in section VI.

II. Related Work

Correlating logs, feeds and alerts from various sources has been a common practice for security professionals. Related research efforts include among others [2] and [3]. Such correlations proved to be useful in ultimately reducing the number of produced false alerts that an administrator or security analyst should go over. This possibility becomes even more apparent when looking through the perspective of an MSSP that usually holds a tremendous amount of potentially valuable but uncorrelated information.

Authors in [4] went one step further regarding the reduction of false alerts by establishing a normal baseline based on clusters of behavior profiles instead of the behavioral patterns of each distinct end-host. Authors argue that a clustered-based anomaly detection sensor will significantly decrease the volume of false alerts by producing more robust models, that will alert only when multiple hosts in a cluster of end-hosts demonstrate similar behavior, rather than just relying on the behavior of a given host. We believe as well that clustered network profiles can offer significant insights, and thus try to identify clusters containing hosts with anomalous network behavior and spread the knowledge by (i) prioritizing the related alerts that may be produced by other security components, and (ii) issuing early warnings about the rest of the hosts participating in a given cluster.

Xu et al. in [5] and [6] employ bipartite and one-mode projection graphs in an effort to analyze the social behavior characteristics and similarity of hosts belonging to the same /24 network prefix. Their approach further validates the existence of practical benefits (in the context of detecting anomalous traffic patterns) originating from the behavior clustering of network end-hosts. In our approach, we don't restrict the clustering within the limits of a particular network, but rather try to identify appropriate clusters throughout all hosts living within MSSP-monitored networks.

Moreover, [7] presents a host-based anomaly detection algorithm, namely Cluster Markov Networks (CMN), that combines a clustering-based model with a statistical approach. After grouping the training data to appropriate clusters, a separate Markov network is built from each cluster. Thus, outliers are identified as instances that do not fit into any of the aforementioned models. Authors of [7] argue that employing clustering on top of Markov networks allow the models to consider the local characteristics of data. However, data related to network security analysis are usually characterized by high dimensionality, a fact that usually hinders the use of cluster-based methods. An analysis that employs all features may miss important outliers, while on the other hand the significant reduction of features may fail to identify small clustered groups of outliers [8]. To counter this, we first applied expert knowledge on our system, by aggregating features and de-duplicating overlapping ones, thus significantly reducing the final number of features, while also increasing the robustness of our proposed mechanism.

Finally, apart from documents found in literature, clustering has also been used in the industry as well, for network security purposes. Imperva Defense Center describes in [9] an unsupervised machine learning approach to dynamically learn peer groups in order to detect access to critical data and/or file shares. Once the groups are identified, they are used to determine appropriate virtual permissions for which users should/should not be allowed to access files within an organization. Cylance also employs similarity and clustering [10] in order to provide the context around endpoint files, in an effort to protect against malware execution. Other platforms, like Palantir [11], feed information retrieved by malware sandbox tools (i.e. VirusTotal [12]) into a detection and clustering component in order to provide a dynamic threat intelligence and detection feedback loop.

III. Design Principles And Overall Approach

A. Motivation and Value Proposition

Malicious activities often have measurable network fingerprints that may be identified by perimeter monitoring devices and stored as network profiles. This observation can be exploited along with information (i.e. threat events) from advanced security appliances such as Intrusion Detection Systems (IDS) from *CL2* infrastructures in order to infer suspicious network behaviors that might occur in *CL1* infrastructures.

Moreover, we noticed that leveraging the line of reasoning above, we could also evaluate and prioritize security incidents based on historical data. Our motivation is then to alleviate the MSSPs engineers day-to-day burden for monitoring and incident identification tasks by prioritizing the entities with higher inferred risk indicating suspicious behavior.

Our overall approach consists of (i) a feature extraction component which aggregates network traffic data and

computes the behavioral attributes, (ii) a clustering algorithm which groups entities by their similarity across these features and (iii) a risk calculation routine which assigns threat scores to each resulting cluster taking into account threat events.

The motivation for using density-based clustering for the risk characterization of entities is based on our preliminary analysis applying t-distributed Stochastic Neighbor Embedding (t-SNE)[13] and visualizing the localization of behaviors in the low-dimensional embedding. T-SNE is a particularly well suited method for the visualization of high-dimensional datasets in two or three dimensions. When used along with a clustering algorithm it is straightforward to visualize the constructed clusters and compare them with the placement of threat-labeled entities. Fig.1 demonstrates a use case of t-SNE applied on the feature space of traffic-related events for internal IPs, as further detailed in section IV. The depicted entities have been distinctively colored according to the occurrence or lack of associated threat events.

The diagram demonstrates that entities with occurring threat events are localized in adjacent regions within the low-dimensional embedding, an observation which justifies density based clustering as an appropriate modeling technique for risk assessment by similarity.

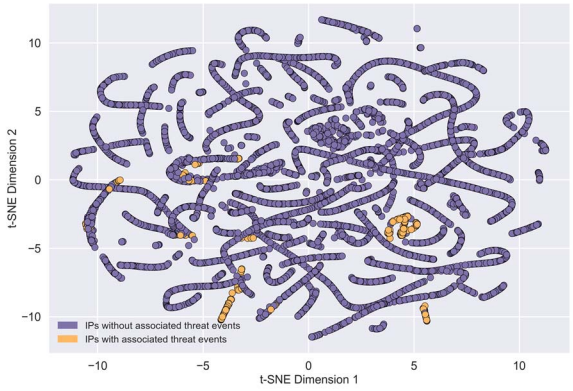

Fig. 1. Representation of internal IP behaviours in t-SNE two dimensional embedding

B. *Architectural Components*

The high level architecture of our approach has been specified in layers with respect to data ingestion, data engineering, and data analysis (Fig. 2). In order to detect suspicious behaviours of monitored entities, based on event logs from security appliances such as Intrusion Detection Systems (IDS), the aforementioned layers are specified as follows:

- Data ingestion: Front-end event listeners receive event logs from authorized devices. Event logs are forwarded to a publish-subscribe message queue to ensure traffic spikes absorption.

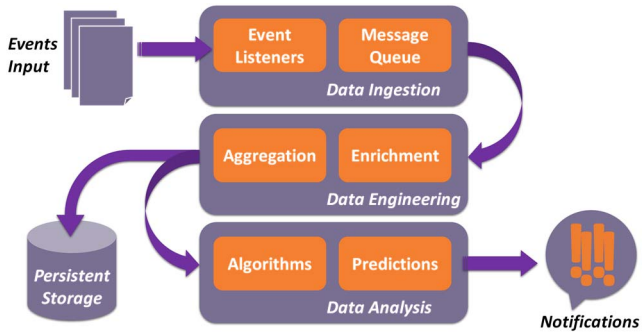

Fig. 2. High Level Architecture

- Data engineering: Back-end processors consume logs from the message queue and enrich them based on customer information, as well as normalizing them to a predefined schema. The output is forwarded back to the message queue to be consumed by our aggregation component and then forwarded to our persistent storage.
- Data analysis: This is the main modeling block which is also the focus of this paper. The internal entities (as identified by their IP addresses) are clustered based on the aggregated metrics produced in the previous layer and then are ranked and classified according to the estimated risk of their assigned cluster.

The components run periodically in a synchronized schedule and alerts are raised for internal entities with an estimated risk exceeding optimized thresholds. Additionally, security analysts have access to a risk-ranked list of all entities in order to review and prioritize their daily operations. The implementation details of each component are presented in the following section.

IV. Detailed Structure And Algorithmic Approach

Traffic data pass through network routers and security appliances, which in our case study include commercial off-the-shelf security appliances. All events gathered and processed for this study are structured as either a) vendor-implemented Syslog format based on [14], or b) Netflow version 9 [15].

A. *Data Ingestion*

Appliances are forwarding events to our edge listeners via User Datagram Protocol (UDP). Logstash[16] provides a solid data ingestion framework for collecting, processing, and forwarding events and log messages. Event listeners tag the events with appropriate information such as vendor/appliance based on listening port, and forward them to our message queue system. Apache Kafka[17] is a high-throughput low-latency distributed platform for handling real time data feeds. It is our primary message queue system that allows us to absorb ingestion spikes, as well as support component segregation.

B. *Data Engineering*

The properly-tagged events are then consumed by pre-processing nodes for enrichment with Geolocation information,

and conversion to a common schema in order to allow the aggregation process to operate on uniform data structures. After being pushed back to the message queue system, they are utilized by the aggregation component to produce the metrics listed in Table I.

In order to enable event filtering and stratified aggregation, two important semantics are established; *type* and *direction*.

Type of events falls under the following categories:

- Traffic events, which are events that are extracted by syslog and/or NetFlow messages describing a closed network session.
- Threat events, which are events that are generated by the various security modules of appliances in CL2 infrastructures.
- AAA(Authentication/Authorization/Accounting) events, are particular to user actions on the appliance itself, such as admin logins, configuration changes, etc.
- Access control events are relevant to user behavior based on appliance-defined policies (Website/application restrictions etc.)

Direction refers to the direction of the event with respect to customer's network. Possible values are:

- in: Event is inbound to customer's network(s)
- out: Event is outbound from customer's network(s)
- prv: Both event initiator and recipient are inside customer's networks(s)

The main entity used for aggregation is the internal IP within a given network. We chose a tumbling time window of 15 minutes for our aggregation component. The 15-minutes time-window was chosen after an analysis of the number of unique internal entities that appear in a sparsely populated communication log. We noticed that around that time period, there were diminishing returns in waiting more to collect measurements for more entities. Computed features are split into two categories: counts and counts of unique occurrences. Counts are computed precisely, while counts of unique occurrences are approximated using the HyperLogLog algorithm [18]. The aggregation process takes place with a 60-seconds delay to absorb accepted levels of latency in the pipeline. Any events exceeding this latency tolerance threshold are ignored in subsequent computations. TABLE I represents the aggregated metrics, from which features are extracted for each applicable direction.

TABLE I. METRICS

Metrics			
Names	***Definition***	***Direction***	***Events Type***
connections_count	number of network connections as measured by counting terminated syslog traffic events for a given internal IP	in, out, prv	traffic
firewall_blocks_count	number of firewall blocks as measured by counting blocked syslog traffic events for a given internal IP	in,out, prv	traffic
logins_count	number of login attempts at a network/security device	In	AAA
failedLogins_count	number of failed login attempts at a network/security device	in	AAA
accCtrl_count	number of total events in access-control logs	out	access control
blockedAccCtrl_count	number of blocked application-level events as reported in access-control logs	out	access control
threats_count	number of threat events in the corresponding logs from security devices	in, out, prv	threat
services_unique	number of unique services, as translated from IANA-registered destination ports	in, out	traffic
countries_unique	number of unique countries a given entity communicates with (from GeoIP translation)	in, out	traffic
asns_unique	number of unique Autonomous Systems a given internal IP communicates with	in, out	traffic
externalIPs_unique	number of unique external IPs a given internal IP communicates with	in, out	traffic

C. Data Analysis

1) Feature Selection

Feature selection was based on knowledge provided by network security field experts and we concluded into the following categories: (i) Network Features, (ii) Policy Features, (iii) Security Features.

Network features characterize the behavior of monitored IPs which is the foundation of our clustering approach. Security features constitute the ground truth for risk estimation and evaluation. While we initially considered policy features (e.g. firewall blocks) as potential response variables, they have been finally excluded since they are commonly biased due to arbitrary policies that network administrators enforce within organizations.

2) Features Transformation

In order to apply cross-network clustering for differently distributed traffic patterns, we have transformed volume-based features to ratios, dividing the measurement for each direction by the total number of events for each metric. The resulting list of features (see Table II) includes one ratio for each direction (e.g. 25% incoming, 50% outgoing, 25% private) and one absolute sum for the total connections. The absolute number is still useful for volumetric attacks, but it is log-transformed to attenuate the effect of right-skewed traffic patterns.

Our risk calculation routine needs to tally the number of entities with security events in each cluster. However, several entities have duplicate or recurring incidents which would lead to skewed results if we used the total sums of threat events. Therefore, our response variables have been transformed to

TABLE II. FEATURES

Features		
Names	***Definition***	***Type***
connections_count_any	sum of connections related metrics	network
connections_ratio_in	ratio of inbound connections to sum of connections	network
connections_ratio_out	ratio of outbound connections to sum of connections	network
connections_ratio_prv	ratio of 'prv' connections to sum of connections	network
logins_count_in	number of login attempts	network
accCtrl_count_out	number of outbound events in access-control logs	network
services_unique_in	number of unique services in inbound traffic events	network
services_unique_out	number of unique services in outbound traffic events	network
countries_unique_in	number of unique countries in inbound traffic events	network
countries_unique_out	number of unique countries in inbound traffic events	network
asns_unique_in	number of unique Autonomous Systems in inbound traffic events	network
asns_unique_out	number of unique Autonomous Systems in outbound traffic events	network
externalIPs_unique_in	number of unique external IPs in inbound traffic events	network
externalIPs_unique_out	number of unique external IPs in outbound traffic events	network
threats_bin	binary feature representing internal IPs exceeding zero threat events	security
failedLogins_bin	binary feature representing internal IPs exceeding zero denied login attempts	security
firewallBlocks_bin	binary feature representing internal IPs exceeding zero total blocked connections	security

binary labels, singly indicating the presence of absence of security events. The presence of threat events, firewall blocks and failed logins are the three final response variables as listed in Table II.

Moreover, since our features range from fractional values (ratios) up to several orders of magnitude (event counts), that would affect the clustering of entities with the highest-valued ones dominating the process. Therefore, we apply minmax normalization which is a strategy for linearly transforming a variable x to a variable y according to the following equation:

$$Y = \frac{x - min}{max - min} \quad (1),$$

where min and max are the minimum and maximum values in X, where X is the set of observed values of x.

3) Features Weights

As we have no a priori knowledge of the relevant significance of each feature, feature weights are introduced to our model and computed utilizing the Pearson correlation coefficient [19] of each feature with the variable representing the presence of threat events. The optimization of feature weights occurs once a day and is applied considering the occurring behaviors in all possible time windows.

4) Clustering Behaviors

The decision about the clustering algorithm that would be applied to our dataset has been taken considering the scalability [20] of the algorithm and the fitness to our modeling requirements. As demonstrated previously based on t-SNE representation, density-based clustering appears to be an appropriate clustering algorithm for modeling network entities behaviors. Therefore, we concluded to use a density based clustering algorithm and specifically DBSCAN [21]. The intuition behind DBSCAN algorithm is that it finds core samples of high density and expands clusters from them. It is well-suited for data that contain clusters of similar density. The main parameters to optimize are *epsilon* and *minimum samples*, where *minimum samples* represent the number of neighbors that shape a cluster and *epsilon* specifies how close points should be to each other to be considered a part of a cluster. For the optimization of parameters (*epsilon* and *minimum samples*), we applied grid search [22] to the dataset as described in the parameters optimization section.

5) Threat Risk Scores

Our behavioral modeling approach includes a risk calculation routine which assigns threat scores to each resulting cluster taking into account the number of threat events. The absence of verified ground truth rules out the building a dataset for classification. However, the presence of entities with recent threat events and their assignment to clusters with similar traffic behaviors justifies the employment of a risk indicator for a given cluster as the percentage of its entities with verified threat occurrences. This score will be referred in this paper as Cluster Threat Risk Score (CTRS). CTRS is calculated as follows:

$$CTRS = {}^{m}/_{n} \quad (2),$$

where *m* is the count of the cluster's internal IPs that have at least one occurring threat event and *n* is the total count of the cluster's internal IPs. Outliers resulting from DBSCAN are considered as a cluster and a score for this cluster is computed as well.

Each internal IP's behavior inherits the Cluster Threat Risk Score of the cluster it is assigned to. This score is referred as ETRS or Entity Threat Risk Score. For any internal IP exceeding an optimized ETRS threshold, notifications are raised and forwarded to security engineers. The threshold is re-evaluated in every aggregation time window (15 minutes) and applied to the corresponding next day's time window. The procedure for the threshold selection is described in the following section.

D. Parameters Optimization

The presence of several free parameters have led us to implement an optimization routine regarding the clustering subprocess. *Positive Likelihood Ratio* [23] is the metric used and needs to be maximized. DBSCAN is applied to the dataset using grid search to find the *epsilon* and the *minimum samples*. Grid search is simply an exhaustive searching through a manually specified subset of parameters optimizing

a given metric. During each run of DBSCAN, the ETRS threshold that maximizes positive likelihood ratio is extracted, and is applied as the decision boundary for the next day's corresponding time window. This ensures that the parameters are continuously evolving and also that the parameters applied on subsequent time windows are independent of the current run of the algorithm.

V. EVALUATION

A. Evaluation Dataset

In order to evaluate the performance of our approach, we selected a representative data sample and extracted the features that constitute the behaviors of the internal IPs, as listed in Table II. The challenging facet of the evaluation is to collect valid ground truth for the response variable, which indicates whether our predictions for suspicious behavior co-occurred with a security event in the subsequent time period. For evaluation purposes, we can only collect ground truth (threat labels) from *CL2* networks with IPS (Intrusion Prevention System) or IDS (Intrusion Detection System) capabilities, and therefore in this section we are only considering behaviors from this class of networks. What should be an appropriate time window for collecting that ground truth is not a trivial problem and we had to experiment with different settings for the construction of a representative dataset. Finally, since this constitutes a noisy labeling approach and there is no authoritative answer on when or whether malicious behaviors are detected by a stateless Intrusion Detection System (IDS), we settled to an 1-hour time window which was concluded to be as long as necessary for collecting as many threat events as at least the 1% of the total monitored entities.

Our initial planning involved using either firewall block events or threat events as our response variable. However, although a high occurrence of firewall blocks may correlate with threat events, we concluded that it was too policy-sensitive and network-specific that could not support drawing general and reproducible conclusions. Therefore, in our evaluation dataset, the response variable which expresses the collected ground truth is singly based on the occurrence of threat events.

In the resulting dataset, there are 8981 internal IPs, from which 80 internal IPs have occurring threat events in the behavioral sample and 142 entities have occurring threat events during the subsequent 1-hour ground-truth time window.

B. Metrics

In this section we define the evaluation metrics both for the quality of the behavioral clustering and the final predictions (threat notifications). The evaluation results will be detailed in the following section.

DBSCAN effectiveness on clustering the internal IPs behaviors was evaluated using the average Silhouette Coefficient. Silhouette coefficient[24] is a metric used to evaluate the consistency of a clustering algorithm and expresses how similar a data point is to its own cluster compared with other clusters. It ranges between -1 and 1, with negative values representing poorly assigned data points.

For the evaluation of our predictions, we have to transform the ETRS to binary predictions utilizing the ETRS threshold as a decision boundary and thus predictions are categorized into positives and negatives. As positives are considered the internal IPs that are characterized as suspicious, while as negative those that are not. Based on the correctness of our predictions, they are split into true and false. True are the cases that our algorithm correctly issued an early notification regarding suspicious internal IPs, based on ground truth and false those that our algorithm wrongly misclassified.

Based on the confusion matrix consisting of the *True positive*, *False positive*, *False negative* and *True negative* counts, the metrics used to evaluate our algorithm are *Recall*, *False Positive Rate*, *Precision* and *Positive Likelihood Ratio or LR+*, which can be defined as:

$$Recall = {TP}/{(TP + FN)} \quad (3),$$

$$False\ Positive\ Rate = {FP}/{(FP + TN)} \quad (4),$$

$$Precision = {TP}/{(TP + FP)} \quad (5),$$

$$LR+ = \frac{Recall}{False\ Positive\ Rate} \quad (6),$$

for FP, TN, TP and FN, respectively *False Positive*, *True Negative*, *True Positive* and *False Negative*.

For a more qualitative interpretation, *Recall* indicates how many occurrences of actually malicious behaviors of internal IPs in the dataset we are able to correctly classify as suspicious; respectively. *Precision* indicates how many of our predicted suspicious behaviors of internal IPs, are actually malicious. *False Positive Rate* indicates how many times we misclassified the behavior of an internal IP as suspicious.

Even though above metrics are used and assist in our algorithm evaluation, the main metric optimized is *Positive Likelihood Ratio (LR+)*. The actual value of *LR+* measures how many times more efficient is our classification when compared to random selection of internal entities for inspection (i.e. the method we assume a security analyst would opt for).

C. Results

Applying grid search to our dataset we derive the optimal DBSCAN parameters, *epsilon* of 0.25 and *minimum samples* of 5, with a resulting *LR+* of 55.8. DBSCAN forms six clusters including a dummy (cluster '0' in Table III) cluster

TABLE III. CLUSTERS INFORMATION

Clusters info		
Clusters	***Number of Internal IPs***	***CTTS***
outliers	101	0.129
0	8531	0.004
1	55	0.4
2	60	0.05
3	222	0.023
4	6	0
5	5	0

consisting of the majority of the internal IPs whose behaviors consist only of a low value of 'connections_count_any' feature presented in Table II. Note that internal IPs not belonging to any cluster (given the selected minimum samples and radius), are characterized as outliers by DBSCAN and we consider them as a new cluster and compute their ETRS. Given the above, the measured average Silhouette Coefficient is 0.423, which is a good indicator of the separability in our dataset.

In the context of our evaluation, we utilize the t-SNE analysis we described in section III, now also depicting the localizations of the resulting clusters from DBSCAN in order to confirm the fitness to our modelling assumptions. Fig.3 depicts on the left the t-SNE embedding of the internal IPs behaviors colored based on the occurrence of threat events and on the right their embedding colored based on the DBSCAN cluster they form. We observe that the clusters formed by DBSCAN contain closely-localized entities in the 2-dimensional embedding. The alert-generating clusters (exceeding the CTRS threshold) also appear to correlate well enough with the localization of the threat-labeled entities in the pre-evaluation t-SNE representation.

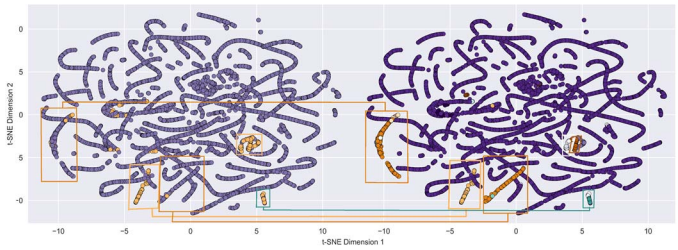

Fig. 3. t-SNE embedding colored based on the occurrence of threat events and on the formed DBSCAN clusters respectively

In order to get an understanding of how *Recall* varies for different levels of the *False Positive Rate* and thus selecting an optimal range for our decision boundaries (ETRS threshold), we compute the ROC (Receiver Operating Characteristic) Curve of the response variable, which is depicted as graphical plot (Fig.4). In our analysis, we observe a data point representing *Recall* of 0.415 and *False Positive Rate* of 0.043 that is associated with an ETRS threshold greater than 0.004, below which internal IPs cannot be separated since all internal IPs would be classified as suspicious (positive). For this threshold level, the corresponding *Precision* is 0.183.

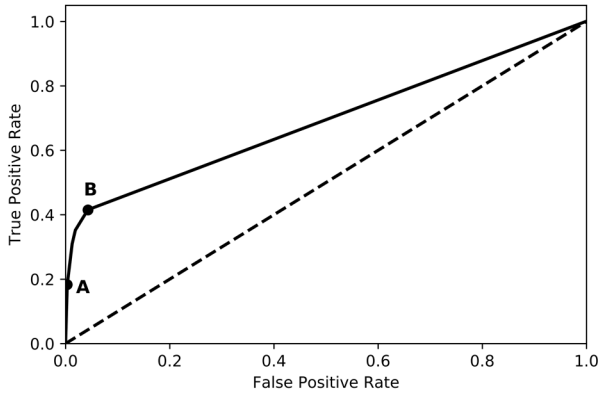

Fig. 4. ROC Curve of the algorithm

Our maximum achieved *LR+* is 55.8, which means that our approach is 55.8 times more efficient than random selection of internal entities for inspection (i.e. the method we assume a security analyst would opt for). It is associated with a *CTRS* of 0.4 (Table III), for which *Recall* is 0.183, false positive rate is 0.003 and precision is 0.473. *Recall* and *False Positive Rate* can also be depicted in point A at Fig.4. *LR+* in this scenario, is actually the slope of the ROC Curve.

By plotting the maximum *LR+* as a function of *ETRS* threshold (see fig. 5), we observe a monotonic relationship between *LR+* and the ETRS threshold (up to ETRS of 0.4, above which all internal IPs behaviors are classified as non-suspicious), which has been consistent during our experiments. This observation supports our assumption that the selected risk score (*ETRS*) is an appropriate proxy variable for quantifying how suspicious the behavior of internal IPs are, since a higher risk correlates with a greater likelihood of being truly malicious. Moreover, although alerts are raised above the aforementioned ETRS threshold, by moving in lower levels of the *ETRS-LR+* curve (increasing *Recall*, but also *False Positive Rate*) we are able to provide security analysts with a risk-ranked list of all entities in order to review and prioritize their daily operations.

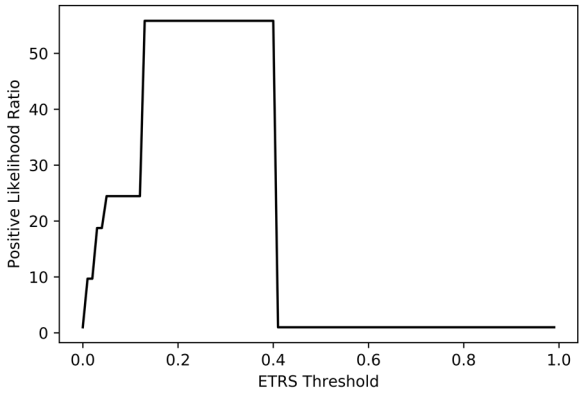

Fig. 5. ETRS Threshold Optimization

VI. Conclusion And Future Work

We presented a reasoning framework for inferring suspicious activity based on the similarity of entities behavior occurring within (i) infrastructures protected by advanced security appliances, with entities behavior in (ii) infrastructures behind simple firewall appliances. The framework allows MSSPs engineers to prioritize subsets of data and specific entities of a monitored network with a higher risk of suspicious behavior. While the attained levels of accuracy are not up to par with solutions having access to network packet payloads, the suggested approach still manages to add value by reducing the amount of information an analyst has to work through in day to day operations.

Our future work includes optimizing and re-evaluating our approach in an augmented dataset with cleaner annotations and more accurate labels. We are also planning to utilize meta-features from applying anomaly detection on the aggregated network metrics for which we have some preliminary promising results. Our noisy labeling technique could also be extended to take into account feedback from external Threat Intelligence appliances, which may provide additional information (e.g. related to specific attack vectors) instead of limiting our sources to only data reported by Next Generation Firewalls and/or IDS/IPS appliances.

Acknowledgment

The presented approach is related to intellectual property protected by the US patents with Nos.: US 2018/0183821Al (non provisional) and 62/466,179 (provisional). An implementation of the presented approach is used by MOREAL[25].

References

[1] IT-Glossary:Managed-Security-Service-Provider, Available at: https://www.gartner.com/it-glossary/mssp-managed-security-service-provider/

[2] P. A. Porras and P. G. Neumann. Event Monitoring Enabling Responses to Anomalous Live Disturbances, NISS, 1997

[3] J. Ullrich DShield, Available at: https://secure.dshield.org/howto.html

[4] Vanessa Frias-Martinez, Salvatore J. Stolfo, and Angelos D. Keromytis, 'Behavior-Profile Clustering for False Alert Reduction in Anomaly Detection Sensors', IEEE Computer Security Applications Conference, 2008

[5] Kuai Xu, Feng Wang, Lin Gu, 'Network-aware behavior clustering of Internet end hosts', Proceedings of IEEE INFOCOM 2011, Shanghai, China, 2011

[6] Kuai Xu, Feng Wang, Lin Gu , 'Behavior Analysis of Internet Traffic via Bipartite Graphs and One-Mode Projections', IEEE/ACM Transactions on Networking, Volume: 22, Issue: 3, June 2014

[7] J.B. Ingram, K. Chiang, A. Mustafa, M. Solaimani, J. Sahs, L. Khan, 'Host-based Anomalous Behavior Detection Using Cluster-Level Markov Networks', Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States); Sandia National Laboratories, Livermore, CA, 2013

[8] C. Aggarwal, Outlier Analysis. Springer-Verlag New York Incorporated, 2013. [Online]. Available: https://books.google.com/books?id=900CkgEACAAJ

[9] Imperva Hacker Intelligence Initiative, 'Today's File Security is So '80s', 2017, Available at: https://www.imperva.com/docs/HII_Todays_file_security_is_so_80s.pdf

[10] CylancePROTECT, Malware Execution Control, Available at: https://www.cylance.com/content/dam/cylance/pdfs/feature-focus/Feature_Focus_PROTECT_Malware_Control.pdf

[11] Palantir Cyber, 'An End-to-End Cyber Intelligence Platform', Available at: https://www.palantir.com/wp-assets/wp-content/uploads/2014/03/Solution-Overview-Palantir-Cyber.pdf, 2014

[12] VirusTotal, Available at: https://www.virustotal.com

[13] van der Maaten, L.J.P.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9:2579-2605, 2008.

[14] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009

[15] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004

[16] Logstash, Available at :https://www.elastic.co/guide/en/logstash/current/index.html

[17] Apache Kafka, Available at: https://kafka.apache.org

[18] P. Flajolet, Éric Fusy, O. Gandouet, and F. Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In Analysis of Algorithms (AOFA), pages 127–146, 2007

[19] Karl Pearson (20 June 1895) "Notes on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, 58 : 240–242.

[20] "Benchmarking Performance and Scaling of Python Clustering Algorithms", Available at: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html

[21] Martin Ester Kriegel, Hans-Peter & Sander, Joerg & Xu, Xiaowei. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD. 96. 226-231.

[22] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13:281–305, 2012.

[23] Swets JA. (1973). "The relative operating characteristic in Psychology". *Science*. 182 (14116): 990–1000

[24] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. 20, 53–65 (1987)

[25] MOREAL Threat Intelligence Platform, Available at: https://docs.moreal.co/behavioural-clustering/