Sub-Space Clustering and Evidence Accumulation for Unsupervised Anomaly Detection in IP Networks

Pedro Casas, Johan Mazel, Philippe Owezarski, Yann Labit

Sub-Space Clustering and Evidence Accumulation for Unsupervised Anomaly Detection in IP Networks

Pedro Casas, Johan Mazel, Philippe Owezarski, Yann Labit
CNRS; LAAS; 7 Avenue du colonel Roche, F-31077 Toulouse, France
Universite de Toulouse; UPS, INSA, INP, ISAE; LAAS
{pcasashe,jmazel,owe,ylabit}@laas.fr

ABSTRACT

Network traffic anomaly detection and analysis has been a hot research topic for many years. Current detection systems employ two different approaches to tackle the problem, even using signature-based detection methods or supervised machine-learning techniques. However, both approaches present serious ground limitations. The former fails to detect new unknown anomalies, the latter highly relies on labeled data for training, which is difficult and expensive to produce. These limitations become highly restrictive in current Internet traffic scenario, characterized by emerging network applications and new variants of network attacks. In this paper, we introduce a novel approach to detect network traffic attacks in a completely unsupervised fashion. The proposed method does not assume any anomaly signature or particular model for anomaly-free traffic, which allows for detection of previously unseen attacks. By combining the multiple evidence of traffic structure provided by sub-space clustering techniques, we show that our method can efficiently isolate and extract unknown anomalies buried inside large amounts of traffic. Apart from discovering new anomalies, the method automatically generates a new and easy-to-interpret signature for the novel detected anomaly, easing network administrator tasks. This new unsupervised anomaly detection method is a powerful means to detect zero-day attacks in a changing environment, where signature-based or supervised learning may fail. We evaluate the ability of our promising proposal to discover a distributed attack in real traffic from the public MAWI traffic repository, discussing future directions and ongoing work.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations - *Network monitoring*; G.3 [**Probability and Statistics**]: Time series analysis; I.5.3 [**Artificial Intelligence**]: Clustering - *Algorithms, Similarity measures.*

General Terms

Measurement, Algorithms, Performance, Security.

Keywords

Unsupervised Anomaly Detection, Zero-day attacks, Subspace Clustering, Evidence Accumulation, Fisher Score.

1. INTRODUCTION

Network traffic anomaly detection has become a vital component of any IP network in today's Internet. Ranging from non-malicious unexpected events such as flash-crowds and failures, to network attacks such as denials-of-service and network scans, network traffic anomalies can have serious detrimental effects on the performance and integrity of the network. This is why anomaly detection and analysis is currently one of the main tasks for every network administrator. The principal challenge in automatically detecting and analyzing traffic anomalies is that these are a moving target. It is difficult to precisely and permanently define the set of possible anomalies, especially in the case of network attacks, because new attacks as well a new variants to already known attacks are continuously emerging. A general anomaly detection and analysis system should therefore be able to detect a range of anomalies with diverse structure, using the least amount of previous information.

The problem of anomaly detection in IP networks has been extensively studied during the last decade. Two different approaches are by far dominant in current research literature and commercial detection systems: signature-based detection and supervised machine-learning-based detection. Signature-based detection methods have been widely used in Intrusion Detection Systems (IDSs). When an attack is discovered, generally after its occurrence in a diagnosis phase, the associated anomalous traffic pattern is coded as a signature by human experts, and it is then used to detect a new occurrence of the same attack. Signature-based detection methods cannot defend the network against zero-day attacks, i.e., new unknown attacks for which there are no signatures to search for yet. Additionally, as the number of signatures grows, the efficiency and cost of detection raise, becoming a difficult-to-avoid bottleneck. Even more, defining new signatures is a resources-consuming task, because they involve deep traffic inspection by human experts.

To alleviate these problems, supervised machine-learning techniques have been extensively applied to the detection problem. Supervised learning-based methods use labeled traffic data to build and train a model for normal-operation traffic (i.e., traffic free of anomalies), detecting traffic anoma-

lies as patterns that deviate from this anomaly-free model. Such methods can detect new types of network attacks not seen before, basically because these new attacks will naturally deviate from what the normal-operation traffic model dictates. Nevertheless, supervised learning requires a set of purely anomaly-free traffic, which is generally unavailable. Labeling traffic as anomaly-free is not only time consuming and expensive, but also very hard to achieve in practice, since it is difficult to guarantee that there are no attacks buried inside the collected data. Additionally, it is not easy to maintain an accurate and up-to-date model for anomaly-free traffic, particularly when new services and applications are constantly appearing.

In this paper, we introduce a novel approach to detect network traffic anomalies without relying on signatures or labeled traffic data for training issues. Our approach falls within the Unsupervised Anomaly Detection (UAD) domain, a new research area that has drawn quite a lot of interest in the research community, but that still represents a rather immature field. The proposed method permits to detect traffic anomalies and to automatically produce easy-to-interpret signatures both in an on-line fashion, analyzing traffic in a temporal sliding-window basis. The anomaly detection and analysis is performed in three consecutive steps: (i) the first step consists in a sliding-window-based change detection algorithm, which marks traffic from a certain temporal bin as anomalous, using the notions of absolute deltoids [1]; (ii) in the second step, robust clustering techniques based on sub-space clustering [9] and evidence accumulation clustering [19] are applied to the traffic inside this anomalous bin, blindly extracting the anomalous traffic instances that raised the alarm in the first step; (iii) finally, the evidence of traffic structure provided by the second step is further used to produce filtering rules for the detected anomaly, which are ultimately combined into a new anomaly signature, easy-to-visualize and to interpret by a human network administrator.

The remainder of this paper is organized as follows. Section 2 presents the state of the art in the Unsupervised Anomaly Detection field and describes our main contributions. Section 3 briefly describes the first part of our algorithm, originally introduced in [25]. In section 4 we introduce the core of the proposal, presenting an in depth description of the different clustering techniques used by our algorithm. Section 5 presents the automatic generation rules algorithm, which builds easy-to-interpret signatures for the detected anomaly. Section 6 presents a primary validation of the proposed algorithm on real traffic data from the WIDE project [28], showing how it can accurately detect and isolate a distributed network attack without any prior knowledge of its existence. Finally, section 7 concludes this paper and presents ongoing work for what we believe is a promising approach.

2. RELATED WORK & CONTRIBUTIONS

Most work on unsupervised anomaly detection in data networks has been devoted to the IDS field, generally targeting the detection of network intrusions in the well known KDD'99 dataset [27]. The majority of the detection schemes proposed in the literature so far are based on clustering techniques and outliers detection [10–15]. The objective of clustering is to partition a set of unlabeled patterns into homogeneous groups of "similar" characteristics, based on some similarity measure. Outliers detection consists in identifying those patterns that do not belong to any of these groups or "clusters" produced by the clustering algorithm.

In [10], authors use a simple single-linkage hierarchical clustering method to cluster data from the KDD'99 dataset, based on the standard Euclidean distance for inter-pattern similarity. [11] reports improved results in the same dataset, using three different clustering algorithms: the Fixed-Width clustering algorithm, an optimized version of the k-Nearest Neighbors algorithm, and the one class Support Vector Machine algorithm. [12] carried out further research in the same direction, presenting an extension of the Fixed-With clustering algorithm to handle time-varying traffic patterns. In [13], authors introduce another method for unsupervised anomaly detection, using an extension of an existing clustering method for large datasets proposed in [22]. [14] presents a combined density-based and grid-based clustering algorithm to improve computational complexity w.r.t. previous works, obtaining similar detection results. More recent work proposed in [15] uses an extension of the celebrated k-means clustering algorithm, using a Gaussian mixture model and the Expectation-Maximization algorithm to estimate the optimal number of clusters to use.

Our Unsupervised Anomaly Detection and Analysis method presents several advantages w.r.t. current state of the art. Firstly, we perform anomaly detection based on small-clusters identification, rather than conducting outliers detection. This can be simply achieved by using different levels of traffic aggregation, transforming outliers in small clusters. Results based on outliers detection are more difficult to generalize and to extend for future analysis, as outliers represent, by definition, few isolated patterns. Secondly, we avoid the lack of robustness of general clustering approaches, by combining the notions of sub-space clustering and multiple evidence accumulation. In particular, our algorithm is immune to general clustering problems like sensitivity to initialization, fixed specification of number of clusters, or structure-masking by irrelevant features. Thirdly, the algorithm performs clustering in low-dimensional feature spaces, avoiding the well known "curse" of high-dimensional data. Additionally, it permits to automatically generate a compact and easy-to-interpret signature for the detected anomaly, alleviating network administration tasks. Finally, the algorithm is designed to work in an on-line fashion, and rather than testing its performance off-line in an out-of-date and simulated dataset, we detect real network attacks in real network traffic. This work is still under development, but the novelty of the proposal as well as the primary results reported in this paper are exciting and promising.

3. DELTOIDS FOR CHANGE DETECTION

The first part of our unsupervised anomaly detection algorithm was originally introduced in a previous work of our own [25], and consists basically in abrupt change-detection in time-series. The algorithm works as follows. Network traffic is constantly captured and analyzed in temporal consecutive bins of length ΔT, using a temporal sliding-window. Every ΔT seconds, three different volume attributes defined as $\mathbf{z}_i = \{\#\text{SYN}_i, \#\text{pkts}_i, \#\text{bytes}_i\}$ are computed for the traffic lying inside temporal bin i, corresponding to traffic captured between times t_i and $t_i + \Delta T$. These three attributes correspond to the number of SYN packets, the total number of packets, and the total number of bytes found

in temporal bin i. Additionally, the *absolute deltoids* [1] $\mathbf{d}_i = \mathbf{z}_i - \mathbf{z}_{i-1}$ are computed for current temporal bin i. The anomaly detection algorithm flags an anomalous traffic behavior at temporal bin i if any of the deltoids associated to the three volume attributes exceeds a detection threshold λ_d. A different detection threshold λ_d^j, $j = 1, \ldots, 3$ is built for each volume attribute, using the standard deviation of the corresponding deltoid, obtained from a set of N past measurements:

$$\lambda_d^j = k \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(\mathbf{d}_i(j)^2 - \bar{\mathbf{d}}(j)^2 \right)} = k\, \sigma_d^j$$

where k is a scaling factor that permits to adjust the sensitivity of detection. Our choice of volume metrics is based on [2], but the algorithm can be used with any other traffic metric sensitive to anomalies. In order to cope with normal traffic variations, each detection threshold λ_d^j is periodically updated: if no anomalous behavior was flagged during the last N temporal bins, the variance of each deltoid is recomputed from these N measurements.

In order to detect low intensity anomalies buried in highly aggregated traffic, we apply the detection algorithm to different aggregation levels at the same time. Aggregation is done based on destination IP address and network mask. In this paper we shall use four different aggregation levels: /0 (i.e., whole traffic), /8, /16, and /24. As with any other detection algorithm, this increase in sensitivity generates a higher rate of false positives. Although this would generally be overkilling for any anomaly detection approach, this is not a serious issue for our method, because the final anomaly decision is taken at the second stage of the process, which is described next.

4. UNSUPERVISED AD

The unsupervised anomaly detection step begins immediately after an anomalous traffic behavior is detected. This step takes as input all the traffic that belongs to the temporal bin flagged as anomalous. Without loss of generality, let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a representation of the n traffic *patterns* $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ found in this temporal bin. We use the generic term pattern because its exact nature will depend on the particular level of aggregation that we shall use for the traffic analysis in this particular bin. For example, $\mathbf{y}_i$ can represent all the IP packets with the same origin or destination IP address or IP network. Each vector $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional vector of traffic attributes or *features* that describe $\mathbf{y}_i$. Examples of standard traffic features are the number of different origin or destination hosts (i.e., IP address /32 in IPv4), or the number of different destination ports. We shall refer to the d-dimensional space defined by the d traffic features as the *feature space*.

Our algorithm makes three assumptions about network traffic: (i) The majority of the network traffic corresponds to normal-operation traffic [10], (ii) the attack traffic is statistically different from normal-operation traffic [26], and (iii) it is possible to find a traffic aggregation in which anomalous traffic lies in small-size clusters. If any of these three assumptions fail, the performance of the algorithm will deteriorate.

Our unsupervised algorithm is based on data clustering, an extremely challenging problem. While hundreds of clustering algorithms exist [3–8], it is very difficult to find a single clustering algorithm that can handle all types of cluster shapes and sizes, or even decide which algorithm would be the best one for a particular data set [16, 17]. Different clustering algorithms produce different partitions of data, and even the same clustering algorithm produces different results when using different initializations and/or different algorithm parameters. This is in fact one of the major drawbacks in current cluster analysis techniques: the lack of robustness.

To circumvent the limitations of current clustering techniques, we integrate in our algorithm a multiple clustering combination approach, via the notion of *clustering ensemble* [20]. A clustering ensemble consists in a set of multiple partitions produced for the same data. The idea is to combine the multiple partitions into a single data partition, obtaining more consistent and robust clustering results. Each partition of the clustering ensemble provides an independent evidence of data organization, which can be exploited to find a proper separation between normal traffic and anomalies.

There are many different ways of producing a clustering ensemble. For example, multiple data partitions can be generated by using different clustering algorithms, or by applying the same clustering algorithm with different values of parameters and/or initializations. There are also many different ways of combining the information provided by the different partitions. In our particular problem, we generate multiple partitions by modifying the feature space used to represent the traffic patterns, applying the same clustering algorithm on each of the modified spaces. In order to preserve the meaning of the different traffic features, we shall modify the feature space using Sub-Space Clustering (SSC) techniques [9].

4.1 Clustering Ensemble and SSC

Any general clustering algorithm takes $\mathbf{X}$ as input and organizes the n patterns into different clusters, forming a data partition P. In multiple clustering combination, we generate N different partitions of the same n patterns, building a cluster ensemble $\mathbf{P} = \{P_1, P_2, \ldots, P_N\}$. Producing a clustering ensemble leads to an exploration of distinct views of inter-pattern relationships. From a computational perspective, multiple partitions produced in an independent way facilitate efficient data analysis. In our algorithm, the N partitions are generated by applying a particular clustering algorithm to each of the sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ that result from the combinations of k attributes taken from the d original attributes. N is therefore the number of k-combinations obtained from d. Each partition P_i is obtained by applying DBSCAN [21] to sub-space $\mathbf{X}_i$. DBSCAN is a powerful density-based clustering algorithm, which permits to accurately discover clusters of arbitrary shapes [4]. The reader should note that in our sub-space approach we do not loose the *meaning* of each of the dimensions of the feature space, keeping tractable attributes to facilitate traffic analysis and comprehension of results.

Using small values for k provides several advantages: doing clustering in low-dimensional data is more efficient and less time-consuming than clustering in bigger dimensions, which partially reduces the increased cost of multiple clustering. Additionally, density-based clustering algorithms such as DBSCAN provide better results in low-dimensional feature spaces [23], simply because high-dimensional feature

spaces are usually sparse, making it difficult to distinguish between high and low density regions. In this sense, we can avoid the well known "curse of dimensionality" problem.

However, there is a clear trade-off between data dimensionality d, sub-spaces dimension k, and computational cost: the bigger the value of d and the lower the value of k, the bigger the cluster ensemble P becomes. This trade-off may render the problem computationally infeasible, but for the values of d and k we shall work on, this is not really an issue. For example, if we consider the complete list of attributes defined in [25], we have an initial dimension $d = 20$. To set the value of k, we take into account a very useful property of monotonicity in clustering sets, known as the downward closure property: "if a collection of points is a cluster in a k-dimensional space, then it is also part of a cluster in any $(k-1)$ projections of this space" [23]. This directly implies that, if there exists any evidence of density in the data $\mathbf{Y}$, we are sure that this evidence will be present in low-dimensional spaces. For this reason, we have decided to use a small value of k, usually $k = 2$. For $d = 20$ and $k = 2$, the number of partitions N is equal to 190, i.e., a reasonably small cluster ensemble. Additionally, the computation of multiple partitions can be done in paralel, which certainly speeds-up analysis.

4.2 Evidence Accumulation Clustering for AD

In [18, 19], authors introduced the idea of Evidence Accumulation Clustering (EAC). EAC is a multiple clustering combination algorithm that uses the clustering results of the multiple partitions P_i to produce a new inter-patterns similarity measure which better reflects their natural groupings. The EAC algorithm follows a split-combine-merge approach to discover the underlying structure of data. In the **split** step, the N partitions of the clustering ensemble $\mathbf{P}$ are generated, which in our case corresponds to the multiple sub-space clustering result. In the **combine** step, a new measure of similarity between patterns is produced, using a simple *voting* mechanism to combine the multiple clustering results. The underlying assumption in EAC is that patterns belonging to a "natural" cluster are very likely to be co-located in the same cluster in different partitions. Taking the co-occurrences of pairs of patterns in the same cluster as votes for their association, the N partitions are mapped into a $n \times n$ co-association matrix A, such that $A(i,j) = n_{ij}/N$. The value n_{ij} simply corresponds to the number of times that pattern pair $(\mathbf{y}_i, \mathbf{y}_j)$ was assigned to the same cluster through the N partitions.

This voting mechanism for evidence accumulation can be improved for our particular task of unsupervised anomaly detection. In fact, according to our assumptions, traffic anomalies lie in small-size clusters. In this sense, it would be beneficial to assign a different weight to the vote that a pattern pair $\{\mathbf{y}_i, \mathbf{y}_j\}$ gets, taking into account not only the membership to the same cluster, but also the size of the cluster where both patterns lie together. Given a certain partition P_i formed by $k = 1, \ldots, K$ clusters, and defining $n(k)$ as the number of patterns inside cluster C_k, we shall use the *evidence importance function* $w_k(n(k))$ as a weighting function that takes bigger values for small values of $n(k)$, and tends to zero for large values of $n(k)$. Using this function, we modify the voting mechanism described before, multiplying the assigned vote by weight w_k when two patterns are found in the same cluster C_k of size $n(k)$. The pseudo-code for our evidence accumulation voting mechanism depicted in algorithm 1 better explains this simple idea. The parameter $n_{\min}$ simply specifies the minimum number of patterns that can be classified as a cluster by the DBSCAN algorithm. The parameter γ permits to set the slope of w_k. Even tuneable, we shall work with fixed values for $n_{\min}$ and γ. The final part of the combine step consists in transforming the co-association matrix A into a similarity matrix S, which is simply its complement to 1 in the off-diagnoal values.

Algorithm 1 Evidence Accumulation for UAD

Initialization:
  Set co-association matrix A to a null $n \times n$ matrix.
  Set minimum cluster size $n_{\min}$ and slope γ.
for $t = 1 : N$ **do**
  P_t = DBSCAN $(X_t, n_{\min})$
  Update $A(i,j)$, $\forall$ pair $\{\mathbf{y}_i, \mathbf{y}_j\} \in C_k$ and $\forall C_k \in P_t$:
    $w_k \leftarrow e^{-\gamma \frac{(n_t(k) - n_{\min})}{n}}$
    $A(i,j) = A(i,j) + \frac{w_k}{N}$
end for
Compute a similarity matrix S from A: $S \leftarrow 1 - A$

In the last **merge** step, any clustering algorithm can be used on the new similarity matrix S so as to obtain a final data partition P^*. In our case, we are only interested in finding a small cluster, which by assumption will represent a traffic anomaly. Therefore, the detection consists in finding those pattern pairs $\{\mathbf{y}_i, \mathbf{y}_j\}$ that have the smallest dissimilarity, according to S.

5. SIGNATURES GENERATION

Selecting the best attributes for sub-space clustering is a difficult task. Many papers in the literature attempt to do so by using extensive search heuristics through data [24] (greedy forward and greedy backward, pruning, bottom-up, iterative top-down, etc.), additionally using some measure of goodness of clustering to assess the relevance of a particular feature. In our case, we do not intend to perform attributes selection for the sub-space clustering step, but for automatically generating filtering rules that permit to clearly identify the anomalous cluster detected by EAC. The basic idea is to combine these filtering rules into an easy-to-interpret anomaly signature which helps the network administrator, and that could be eventually used to easily detect this anomaly in the future. We follow a similar approach to those proposed in the feature selection literature, but using the already generated clustering ensemble $\mathbf{P}$. Basically, we select those partitions P_i in which the anomalous cluster is clearly isolated from the rest of the traffic patterns.

We shall define two different types of filtering rules: *absolute* rules and *splitting* rules. Absolute rules do not depend on the relative separation between clusters. This kind of rules corresponds to the presence of dominant attributes in the patterns of the anomalous cluster. For example, if one of the attributes in $\mathbf{X}$ corresponds to a strong presence of SYN packets in a certain traffic aggregation specified for $\mathbf{Y}$ (e.g., 90% of the packets are SYN packets), it is likely that the majority of the patterns in the anomalous cluster of a SYN scan attack will have a value equal to 1 for this attribute. Absolute rules are rules of type `(attribute == value)`. On

the other hand, splitting rules consist in isolation rules that depend on the relative separation between clusters. Briefly speaking, if the anomalous cluster is well separated from the rest of the clusters in a certain partition P_i, then the attributes of the corresponding sub-space X_i are good candidates for defining a filtering rule. Splitting rules are rules of type `(attribute </> threshold)`.

Absolute rules are important rules, because they define inherent characteristics of the anomaly. As regards splitting rules, it is clear that some of them will be more important than others, based on the degree of separation between clusters. In order to assess the importance of splitting rules, we use the notions of Linear Discriminant Analysis, through the computation of the Fisher Score (FS). The FS is a measure of separation between clusters, relative to the total variance within each cluster. Given two clusters C_1 and C_2, the FS $F(i)$ for attribute i can be computed as follows:

$$F(i) = \frac{(\bar{x}_1(i) - \bar{x}_2(i))^2}{\sigma_1(i)^2 + \sigma_2(i)^2} \quad (1)$$

where $\bar{x}_j(i)$ and $\sigma_j(i)^2$ are the mean and variance of attribute i in cluster C_j. Therefore, in order to select the most important splitting rules, we shall keep those features with largest Fisher score.

6. EXPERIMENTAL EVALUATION

We shall evaluate the ability of our Unsupervised Anomaly Detection algorithm to detect and to automatically generate a signature for a distributed network attack in real traffic data from the Japanese MAWI traffic repository [28]. This real-traffic dataset comes from the WIDE Internet operational network, a test-bed network developed under the WIDE project (http://www.wide.ad.jp/). The WIDE network provides interconnection between different research institutions in Japan, as well as connection to different commercial ISPs and universities in the U.S. The MAWI traffic repository consists of 15-minutes-long raw packet traces collected daily at 14:00 (Japan time) since 1999. These traces are provided publicly after being anonymized and stripped of payload data. The trace we shall work with consists in traffic captured in January 2004 at one of the trans-pacific links between Japan and the U.S., measured at sample point-B. The line is a 100-Mbps link with 18-Mbps CAR.

In this evaluation, traffic is aggregated on a destination IP address /24 basis. The sliding-window time-scale granularity used by the change-detection deltoid-based algorithm is 20 seconds. Absolute-deltoids are computed for the three volume metrics previously described, namely #SYN, #pkts, and #bytes. The network attack that we shall analyze consists in a distributed network SYN scan directed to many victim hosts under IP network address 162.225.0.0/16, originated at an attacker host with origin IP address 204.243.26.29. The attack starts at time 14:06:43 and it is directed towards multiple hosts. Figure 1 depicts this situation.

The change-detection algorithm detects an anomalous temporal bin at time 14:07:00 due to an anomalous absolute deltoid in the three volume metrics. Figure 2 depicts the brutal modification in all of the metrics when the attack is deployed. The Unsupervised Anomaly Detection algorithm is therefore *fed* with all the traffic that belongs to the 21-th sliding-window. Given the traffic aggregation level used, each pattern $\mathbf{y}_i \in \mathbf{Y}$ consists of all the IP packets directed

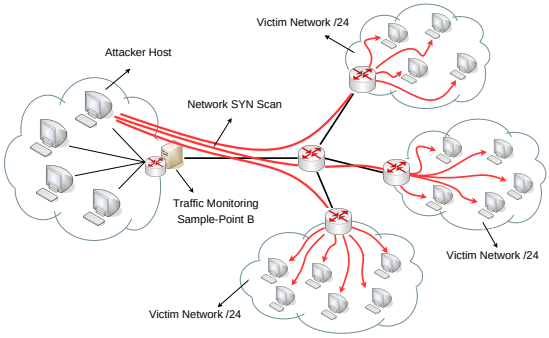


Figure 1: Network SYN Scan Attack in WIDE.

to a certain IP network destination address /24. To describe each of these patterns, we shall use some of the attributes that were used in [25] to classify general network attacks (DoS, DDoS, and Scans). The idea is to show that our unsupervised algorithm can automatically detect and build a signature without any previous knowledge about the attack under analysis. The list of $d = 8$ attributes includes the n° of different source and destination IP addresses (nSrcs and nDsts), as well as the proportion of SYN packets (nSYN/nPkts) among others.

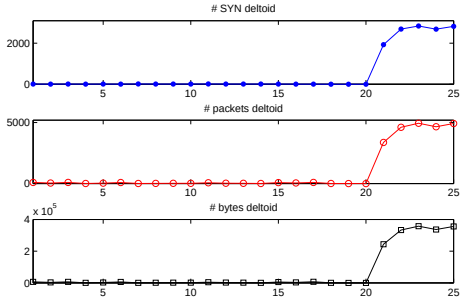
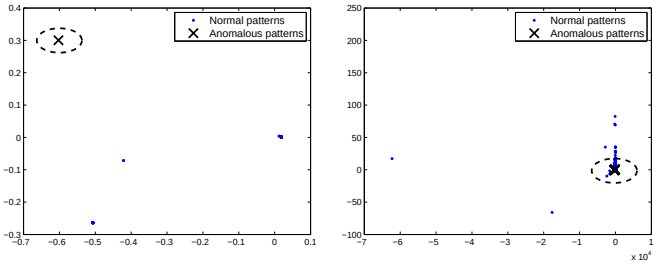


Figure 2: Anomaly detection based on absolute-deltoids change-detection in WIDE.

In order to appreciate the great advantage of using our sub-space and EAC algorithm w.r.t. a traditional clustering approach, based on the information provided by the complete feature space of 8 attributes, we shall compute a similarity matrix S_{tra} for the n patterns in $\mathbf{X}$. Each element (i, j) in S_{tra} represents inter-patterns similarity by means of the Euclidean distance between patterns i and j. Figure 3 depicts a two-dimensional plot of the information provided by both similarity matrices S and S_{tra}, using a Multi-Dimensional Scaling analysis. In order to assess the power of discrimination provided by each similarity matrix, we assume that the anomalous patterns are known in advance. In the case of S_{tra}, we can appreciate that the anomalous patterns are mixed-up with the normal ones, and the discrimination using all the attributes at the same time becomes difficult. In the case of S, the anomalous patterns are perfectly isolated from the rest of the patterns, providing a powerful discrimination measure of similarity.

As we explained before, the anomaly detection simply consists in identifying the most similar patterns in S. Figure 4 shows a histogram on the distribution of inter-pattern sim-



(a) EAC Similarity Matrix (b) Traditional Similarity Matrix.

Figure 3: MDS for Traditional and EA Clustering.

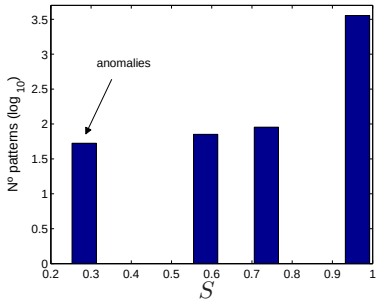
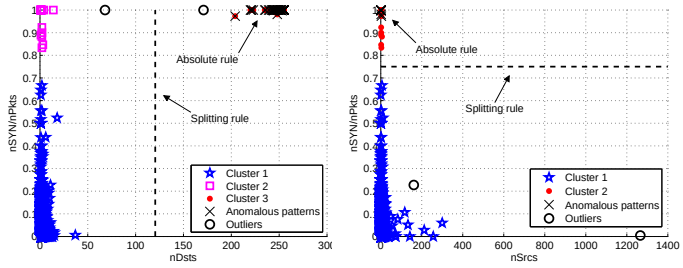


Figure 4: Anomaly detection in S.

ilarity, according to S. Selecting the most similar group of patterns results in a compact cluster of 53 patterns. A further analysis of the IP packets that compose each of these patterns reveals the same origin IP address, which corresponds to the attacker's host. Each individual pattern corresponds to a flow of SYN packets directed towards a different destination IP network address /24 in 162.225.0.0/16.

The last step of the algorithm consists in automatically building easy-to-understand and easy-to-visualize discrimination rules that can be combined into a signature for the anomaly that has been detected. Figure 5 depicts some of the partitions where both absolute and splitting rules where found. The partition depicted in figure 5.a presents three clusters, one of them exclusively composed of anomalous patterns. This partition builds two rules; the former is a splitting rule of the form `<nDsts > `λ_1`>`, relative to the number of different destination IP addresses found in the IP packets aggregated into each pattern. Given the distributed nature of the network scan attack, it is clear that the number of destination IPs must be much larger than in the case of normal-operation traffic. The latter is an absolute rule `<nSYN/nPkts == 1>`, in which almost every pattern of the anomalous cluster has the SYN flag activated for all of its IP packets. This rule makes perfect sense, because the network scan uses SYN packets. The partition depicted in figure 5.b shows two clusters, but in this case the smallest one contains not only anomalous patterns but also some other patterns not identified as such. Given that the filtering rules are defined in a per-cluster basis, a new splitting rule to separate both clusters is generated for attribute nSYN/nPkts, which permits to relax the absolute rule previously defined. The new rule for nSYN/nPkts is therefore `<nSYN/nPkts >` λ_2`>`. Given that this new rule covers the previous one, the



(a) (b)

Figure 5: Filtering rules for automatic anomaly signature generation.

absolute rule is discarded. Another rule is also identified in this partition, consisting in an absolute rule for attribute nSrcs. This absolute rule specifies that all the packets come from the same origin IP address, which corresponds to the attacker's IP address. Adding this rule to both previously generated rules finally produces a new anomaly signature that can be expressed as:

`<(nDsts > `λ_1`) & (nSYN/nPkts > `λ_2`) & (nSrcs == 1)>`

The most interesting observation of this step of the algorithm is that the generated anomaly signature permits to effectively isolate all of the patterns that conform the network SYN scan, correctly classifying all the corresponding IP packets that are aggregated into each pattern.

7. CONCLUSIONS & PERSPECTIVES

The Unsupervised Anomaly Detection algorithm that we have proposed presents many interesting advantages w.r.t. previous proposals in the field of Anomaly Detection. It uses exclusively unlabelled data to detect traffic anomalies, without assuming any particular model or any canonical data distribution. This allows to detect new previously unseen network attacks. Despite using ordinary clustering techniques to identify traffic anomalies, the algorithm avoids the lack of robustness of general clustering approaches, by combining the notions of sub-space clustering and multiple evidence accumulation. The sub-space clustering approach also permits to obtain easy-to-interpret and tractable results, providing insight and explanations about the detected anomalies to a human network manager. Additionally, clustering in low-dimensional feature spaces provides results that can be visualized by standard techniques, which improves the assimilation of results.

We have verified the effectiveness of this unsupervised detection approach to detecte and isolate a real distributed network attack in a completely blind fashion, without assuming any particular traffic model, detection threshold, significant clustering parameters, or even clusters structure beyond a basic definition of what an anomaly is. However, this is an on-going work which still requires validation as regards the detection of a larger variety of network attacks. A comprehensive evaluation of the algorithm is part of current on-going work. As regards computational speed of the approach, both SSC and EAC can run in a paralel basis, and thus we are implementing both in a distributed framework.

8. REFERENCES

[1] G. Cormode and S. Muthukrishnan, "What's New: Finding Significant Differences in Network Data Streams", in *IEEE Trans. on Networking*, vol. 13 (6), pp. 1219-1232, 2005.

[2] A. Lakhina, M. Crovella, C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows", in *Proc. ACM IMC*, 2004.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review", in *ACM Computing Surveys*, vol. 31 (3), pp. 264-323, 1999.

[4] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means", in *Pattern Recognition Letters*, vol. 31 (8), pp. 651-666, 2010.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification - second edition", Wiley Publisher, 2001.

[6] L. Kaufman and P. J. Rosseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis", John Wiley & Sons, 1990.

[7] B. Everitt, "Cluster Analysis", John Wiley & Sons, 1993.

[8] S. Theodoridis and K. Koutroumbas, "Pattern Recognition", Academic Press, 1999.

[9] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: a Review", in *ACM SIGKDD Explorations Newsletter*, vol. 6 (1), pp. 90-105, 2004.

[10] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion Detection with Unlabeled Data Using Clustering", in *Proc. ACM DMSA Workshop*, 2001.

[11] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data", in *Applications of Data Mining in Computer Security*, Kluwer Publisher, 2002.

[12] J. Oldmeadow, s. Ravinutala, and C. Leckie, "Adaptive Clustering for Network Intrusion Detection", in *Proc. Int. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2004.

[13] K. Burbeck and S. Nadjm-Tehrani, "ADWICE - Anomaly Detection with Real-Time Incremental Clustering", in *Proc. 7th Int. Conf. on Information Security and Cryptology*, 2004.

[14] K. Leung and C. Leckie, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clustering", in *Proc. Australasian Computer Science Conference, ACSC05*, 2005.

[15] W. Lu and I. Traore, "Unsupervised Anomaly Detection using an Evolutionary Extension of K-means Algorithm", in *Int. Journal of Information and Computer Security*, vol. 2 (2), pp. 107-139, 2008.

[16] C. Fraley and A. E. Raftery, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis", in *The Computer Journal*, vol. 41 (8), pp. 578-588, 1998.

[17] R. Dubes and A. K. Jain, "Clustering Techniques: The User's Dilemma", in *Pattern Recognition*, vol. 8, pp. 247-260, 1976.

[18] A. Fred and A. K. Jain, "Data Clustering Using Evidence Accumulation", in *Proc. 16th Int. Conf. Pattern Recognition*, pp. 276-280, 2002.

[19] A. Fred and A. K. Jain, "Combining Multiple Clusterings Using Evidence Accumulation", in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27 (6), pp. 835-850, 2005.

[20] A. Strehl and J. Ghosh, "Cluster Ensembles - A Knowledge Reuse Framework For Combining Multiple Partitions", in *Journal on Machine Learning Research*, vol. 3, pp. 583-617, 2002.

[21] M. Ester, H. P. Kriegel, J. Sander, X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *Proc. ACM SIGKDD*, 1996.

[22] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an Efficient Data Clustering Method for Very Large Databases", in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1996.

[23] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", in *Proc. ACM SIGMOD: Management of Data Conference*, 1998.

[24] H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery & Data Mining", Kluwer Academic Publishers, 1998.

[25] G. Fernandes and P. Owezarski, "Automated Classification of Network Traffic Anomalies", in *Proc. SecureComm'09*, 2009.

[26] E. Denning, "An Intrusion-Detection Model", in *IEEE Trans. Software Engineering*, vol. 13 (2), pp. 222-232, 1987.

[27] UCI Knowledge Discovery in Databases Archive, "The Third International Knowledge Discovery and Data Mining Tools Competition Dataset, KDD99 Cup", in http://kdd.ics.uci.edu/databases/kddcup99

[28] K. Cho, K. Mitsuya and A. Kato, "Traffic Data Repository at the WIDE Project", in *USENIX Annual Technical Conference*, 2000.