Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge

Pedro Casas Hernandez, Johan Mazel, Philippe Owezarski

Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge

Pedro Casas[a,b,*], Johan Mazel[a,b], Philippe Owezarski[a,b]

[a] *CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France*
[b] *Université de Toulouse; UPS, INSA, INP, ISAE; UT1, UTM, LAAS; F-31077 Toulouse Cedex 4, France*

Abstract

Traditional Network Intrusion Detection Systems (NIDSs) rely on either specialized signatures of previously seen attacks, or on expensive and difficult to produce labeled traffic datasets for user-profiling to hunt out network attacks. Despite being opposite in nature, both approaches share a common downside: they require the knowledge provided by an external agent, either in terms of signatures or as normal-operation profiles. In this paper we present UNIDS, an Unsupervised Network Intrusion Detection System capable of detecting unknown network attacks without using any kind of signatures, labeled traffic, or training. UNIDS uses a novel unsupervised outliers detection approach based on Sub-Space Clustering and Multiple Evidence Accumulation techniques to pin-point different kinds of network intrusions and attacks such as DoS/DDoS, probing attacks, propagation of worms, buffer overflows, illegal access to network resources, etc. We evaluate UNIDS in three different traffic datasets, including the well-known KDD99 dataset as well as real traffic traces from two operational networks. We particularly show the ability of UNIDS to detect unknown attacks, comparing its performance against traditional misuse-detection-based NIDSs. In addition, we also evidence the supremacy of our outliers detection approach with respect to different previously used unsupervised detection techniques.

Keywords: NIDS, Unsupervised Machine Learning, Sub-Space Clustering, Evidence Accumulation, Outliers Detection, Network Attacks, DDoS, Buffer Overflows.

1. Introduction

The detection of network attacks is a paramount task for network operators in today's Internet. Botnets, Malwares, Distributed Denial of Service attacks (DDoS), buffer overflow attacks, network-scanning activities, and spreading worms or viruses are examples of the different threats that daily compromise the integrity and normal operation of the network. The principal challenge in automatically detecting network attacks is that these are a moving and ever-growing target [1, 2]. Network Intrusion Detection Systems (NIDSs) are the war-horses of network security. Two different approaches are by far dominant in the research literature and commercial IDS security devices: signatures-based or **misuse detection** (*detect what I know*) and **anomaly detection** (*detect what it is different from what I know*).

Misuse detection is the de-facto approach used in most IDSs. When an attack is discovered, generally after its occurrence during a diagnosis phase, the associated malicious pattern is coded as a *signature* by human experts, which is then used to detect a new occurrence of the same attack. To avoid costly and time-consuming human intervention, signatures can also be constructed by supervised machine-learning techniques, using instances of the discovered attack to build a detection model for it. Misuse detection systems are highly effective to detect those attacks which they are programmed to alert on. However, they cannot defend the network against new attacks, simply because they cannot recognize those attacks which do not match their lists of signatures. Indeed, networks protected by misused detection systems suffer from long periods of vulnerability between the diagnosis of a new attack and the construction of the new signature.

On the other hand, anomaly detection uses instances of normal-operation traffic to build normal-operation profiles, detecting anomalies as activities that deviate from this baseline. Such methods can detect new kinds of network attacks not seen before. Nevertheless, they require training to construct profiles, which is time-consuming and depends on the availability of anomaly-free traffic instances. In addition, it is not easy to maintain an accurate and up-to-date normal-operation profile, which induces high false-alarm rates.

Despite being opposite in nature, both misuse detection and anomaly detection share a common downside: they require the knowledge provided by an external agent to achieve their goal, either in terms of attack-signatures or as normal-operation profiles. As such, current network

*Corresponding author. Telephone: +33 (0)5 61 33 68 05 - Fax: +33 (0)5 61 33 64 11

Email addresses: pcasas@laas.fr (Pedro Casas), jmazel@laas.fr (Johan Mazel), owe@laas.fr (Philippe Owezarski)

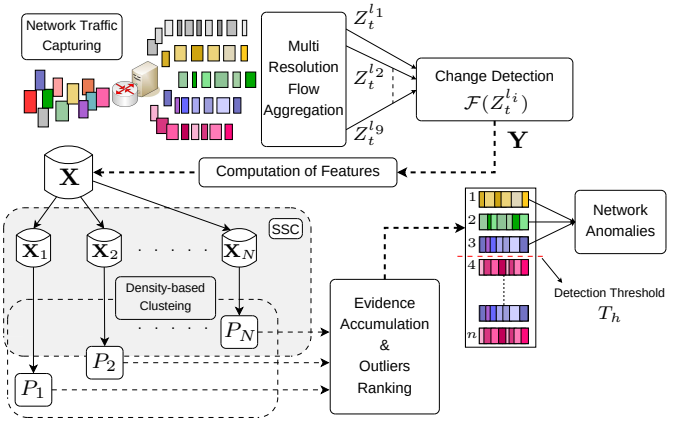


Figure 1: High-level description of the Unsupervised Network Intrusion Detection System (UNIDS).

security looks more like a reactive countermeasure than a proactive prevention mechanism. Over the past years we have, however, witnessed an increased interest within the network security community in shifting away from reactive defense towards more proactive security systems [3]. Our thesis behind this work is that reactive, knowledge-based approaches are not sufficient to tackle the network security problem, and that a holistic solution should also include proactive, knowledge-independent analysis techniques.

Armed with these ideas in mind, we present an Unsupervised Network Intrusion Detection System (UNIDS) capable of detecting network attacks without relying on signatures, training, or labeled traffic instances of any kind. Based on the observation that network attacks, and particularly the most difficult ones to detect, are contained in a small fraction of traffic instances with respect to normal-operation traffic [4] (we show that this hypothesis can always be verified by using traffic aggregation), their unsupervised detection consists in identifying *outliers*, i.e. instances that are remarkably different from the majority. UNIDS relies on robust clustering techniques based on Sub-Space Clustering (SSC) [17], Density-based Clustering [22], and Evidence Accumulation (EA) [21] to blindly extract the traffic instances that compose an attack.

UNIDS runs in three consecutive steps, analyzing packets captured in contiguous time slots of fixed length. Figure 1 depicts a modular, high-level description of this system. The first step consists in detecting an anomalous time slot in which the clustering analysis will be performed. For doing so, captured packets are first aggregated into *multi-resolution* traffic flows. Different time-series are then built on top of these flows, and any generic change-detection algorithm based on time-series analysis is finally used to flag an anomalous change. The second step takes as input all the flows in the time slot flagged as anomalous. At this step, outlying flows are identified using a robust multi-clustering algorithm, based on a combination of Sub-Space Clustering (SSC) [17], Density-based Clustering [22], and Evidence Accumulation Clustering (EAC) [21] techniques. The knowledge provided by this clustering algorithm is used to rank the degree of *abnormality* of all the identified outlying flows, building an *outliers ranking*. In the third and final step, the top-ranked outlying flows are flagged as anomalies, using a simple thresholding detection approach. As we will show through out the paper, the main contribution provided by UNIDS relies on its ability to detect unknown attacks in a completely unsupervised fashion.

The remainder of the paper is organized as follows. Section 2 presents a brief state of the art in the supervised and unsupervised attacks detection field, describing our main contributions. Section 3 describes the multi-resolution traffic aggregation and change-detection procedures used in the first step of the UNIDS system. Section 3 describes the core of UNIDS, presenting an in depth description of the different clustering techniques used to construct the outliers ranking. Section 6 presents the validation of our U-NIDS in real traffic traces obtained from two different networking datasets: the public MAWI traffic repository of the WIDE project [25], and the METROSEC project dataset [26]. In this section we also compare the performance of UNIDS against previous proposals for unsupervised detection of attacks available in the literature. Section 7 evaluates the ability of this system to detect unknown attacks in the well-known KDD99 network attacks dataset, comparing its performance with that obtained by an extensively investigated misuse NIDS based on decision trees. Finally, section 10 concludes the paper.

2. Related Work & Contributions

The problem of network attacks and intrusions detection has been extensively studied in the last decade. Most NIDS are based on misuse-detection techniques, being Bro [27] and SNORT [28] the two most celebrated open-source systems in the literature. Most of the different techniques used in the NIDS field are summarized in [5, 6]. A particularly studied approach in recent years for misuse-detection that we shall use in the paper consists in the use of decision trees [7, 8]. Decision trees are one of the most powerful and simple data mining methods for decision-making; a decision tree uses a tree-like graph consisting of branch nodes that represent a choice among different alternatives, and leaves representing a class for the evaluated data (i.e. *attack* or *normal traffic*).

The problem of network anomaly detection has been extensively studied during the last decade. Most approaches analyze statistical variations of traffic volume descriptors (e.g., no. of packets, bytes, or new flows) and/or particular traffic features (e.g., distribution of IP addresses and ports), using either single link measurements or network wide data. A non-exhaustive list of standard methods includes the use of signal processing techniques (e.g., ARIMA modeling, wavelets-based filtering) on single-link traffic measurements [9], Kalman filters [12] for network-wide anomaly detection, and Sketches applied to IP-flows [14, 15].

Our approach falls within the unsupervised anomaly detection domain. The vast majority of the unsupervised detection schemes proposed in the literature are based on clustering and outliers detection, being [18, 19, 20] some relevant examples. In [18], authors use a single-linkage hierarchical clustering method to cluster data from the KDD99 data-set, based on the standard Euclidean distance for inter-patterns similarity. Reference [19] reports improved results in the same data-set, using three different clustering algorithms: Fixed-Width clustering, an optimized version of k-NN, and one class SVM. Reference [20] presents a combined density-grid-based clustering algorithm to improve computational complexity, obtaining similar detection results. PCA and the sub-space approach is another well-known unsupervised anomaly detection technique, used in [10, 11] to detect network-wide traffic anomalies in highly aggregated traffic flows, and more recently in [13] for anomaly detection in single-router traffic metrics.

UNADA presents several advantages with respect to current state of the art. First and most important, it works in a completely unsupervised fashion, which means that it can be directly plugged-in to any monitoring system and start to work from scratch, without any kind of calibration and/or training step. Secondly, it uses a robust density-based clustering technique to avoid general clustering problems such as sensitivity to initialization, specification of number of clusters, detection of particular cluster shapes, or structure-masking by irrelevant features. Thirdly, it performs clustering in very low dimensional spaces, avoiding sparsity problems when working with high-dimensional data [16]. Finally, we show that UNADA clearly outperforms previously proposed methods for unsupervised anomaly detection in real network traffic.

3. Multi-resolution Flow Aggregation & Change-Detection

UNADA performs unsupervised anomaly detection on single-link packet-level traffic, captured in consecutive time slots of fixed length ΔT and aggregated in IP flows (standard *5-tuples*). IP flows are additionally aggregated at different *flow-resolution* levels, using 9 different *aggregation keys* l_i. These include (from coarser to finer-grained resolution): *traffic per Time Slot* (l_1:tpTS), *source Network Prefixes* ($l_{2,3,4}$: IPsrc/8, /16, /24), *destination Network Prefixes* ($l_{5,6,7}$: IPdst/8, /16, /24), *source IPs* (l_8: IPsrc), and *destination IPs* (l_9: IPdst). The 7 coarsest-grained resolutions are used for change-detection, while the remaining 2 are exclusively used in the clustering step.

To detect an anomalous time slot, time-series $Z_t^{l_i}$ are constructed for simple traffic metrics such as number of bytes, packets, and IP flows per time slot, using aggregation keys $i = 1, \ldots, 7$. Any generic change-detection algorithm $\mathcal{F}(.)$ based on time-series analysis is then applied to $Z_t^{l_i}$. At each new time slot, $\mathcal{F}(.)$ analyses the different time-series associated with each aggregation key, going from coarser (l_1) to finer resolution (l_7). Time slot t_0 is flagged as anomalous if $\mathcal{F}(Z_{t_0}^{l_i})$ triggers an alarm for any of the traffic metrics at any of the 7 aggregation levels. Tracking anomalies at multiple aggregation levels provides additional reliability to the change-detection algorithm, and permits to detect both single source-destination and distributed anomalies of very different intensities.

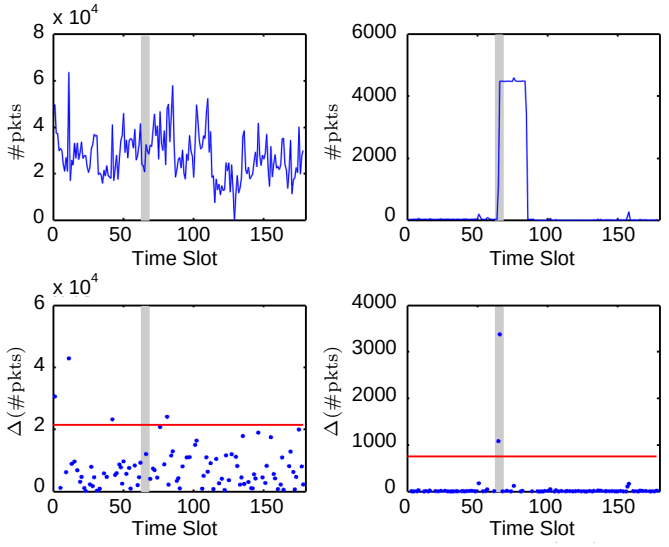


Figure 2: Low-intensity anomalies might be hidden inside highly aggregated traffic, but are visible at finer-grained aggregations. The DDoS attack is evident at the victim's network.

Figure 2 shows how a low intensity DDoS attack might be dwarfed by highly-aggregated traffic. The time-series associated with the number of packets, namely $Z_t = \#\text{pkts}_t$, does not present a perceptible change $\Delta(\#\text{pkts}_t)$ at tpTS aggregation (left); however, the attack can be easily detected using a finer-grained resolution, e.g., at the victim's network (IPdst/24 aggregation, on the right).

4. Unsupervised Anomaly Detection through Clustering

The unsupervised anomaly detection step takes as input **all** the IP flows in the flagged time slot. At this step UNADA ranks the degree of abnormality of each flow, using clustering and outliers analysis techniques. For doing so, IP flows are analyzed at two different resolutions, using either IPsrc or IPdst aggregation key. Traffic anomalies can be roughly grouped in two different classes, depending on their spatial structure and number of impacted IP flows: *1-to-N* anomalies and *N-to-1* anomalies. *1-to-N* anomalies involve many IP flows from the same source towards different destinations; examples include network scans and spreading worms/virus. On the other hand, *N-to-1* anomalies involve IP flows from different sources towards a single destination; examples include DDoS attacks and flash-crowds. *1-to-1* anomalies are a particular case of these classes, while *N-to-N* anomalies can be treated as

multiple *N-to-1* or *1-to-N* instances. Using IPsrc key permits to highlight *1-to-N* anomalies, while *N-to-1* anomalies are more easily detected with IPdst key. The choice of both keys for clustering analysis ensures that even highly distributed anomalies, which may possibly involve a large number of IP flows, can be represented as outliers. Without loss of generality, let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n aggregated-flows (at IPsrc or IPdst) in the flagged slot. Each flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m traffic attributes or *features*, like num. of sources, destination ports, or packet rate. Let $\mathbf{x}_i \in \mathbb{R}^m$ be the vector of traffic features describing flow $\mathbf{y}_i$, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times m}$ the complete matrix of features, referred to as the *feature space*.

UNADA is based on clustering techniques applied to $\mathbf{X}$. The objective of clustering is to partition a set of unlabeled samples into homogeneous groups of similar characteristics or *clusters*, based on some measure of similarity. Samples that do not belong to any of these clusters are classified as *outliers*. Our particular goal is to identify those outliers that are remarkably different from the rest of the samples, additionally ranking how much different these are. The most appropriate approach to find outliers is, ironically, to properly identify clusters. After all, an outlier is a sample that does not belong to any cluster. Unfortunately, even if hundreds of clustering algorithms exist [16], it is very difficult to find a single one that can handle all types of cluster shapes and sizes. Different clustering algorithms produce different partitions of data, and even the same clustering algorithm provides 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 avoid such a limitation, we have developed a divide & conquer clustering approach, using the notions of *clustering ensemble* and *multiple clusterings combination*. The idea is novel and appealing: why not taking advantage of the information provided by multiple partitions of $\mathbf{X}$ to improve clustering robustness and identification of outliers? A clustering ensemble $\mathbf{P}$ consists of a set of multiple partitions P_i produced for the same data. Each partition provides an independent evidence of data structure, which can be combined to construct a new measure of similarity that better reflects natural groupings and outliers. There are different ways to produce a clustering ensemble. We use Sub-Space Clustering (SSC) [17] to produce multiple data partitions, doing Density-based clustering in N different sub-spaces $\mathbf{X}_i$ of the original space (see figure **??**).

4.1. Clustering Ensemble and Sub-Space Clustering

Each of the N sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ is obtained by selecting k features from the complete set of m attributes. To deeply explore the complete feature space, the number of sub-spaces N that are analyzed corresponds to the number of k-combinations-obtained-from-m. Each partition P_i is obtained by applying DBSCAN [22] to sub-space $\mathbf{X}_i$. DBSCAN is a powerful clustering algorithm that discovers clusters of arbitrary shapes and sizes [16], relying on a density-based notion of clusters: clusters are high-density regions of the space, separated by low-density areas. This algorithm perfectly fits our unsupervised traffic analysis, because it is not necessary to specify a-priori difficult to set parameters such as the number of clusters to identify. Results provided by applying DBSCAN to sub-space $\mathbf{X}_i$ are twofold: a set of $p(i)$ clusters $\{C_1^i, C_2^i, .., C_{p(i)}^i\}$ and a set of $q(i)$ outliers $\{o_1^i, o_2^i, .., o_{q(i)}^i\}$. To set the number of dimensions k of each sub-space, we take a very useful property of monotonicity in clustering sets, known as the downward closure property: if a collection of elements is a cluster in a k-dimensional space, then it is also part of a cluster in any $(k-1)$ projections of this space. This directly implies that, if there exists any interesting evidence of density in $\mathbf{X}$, it will certainly be present in its lowest-dimensional sub-spaces. Using small values for k provides several advantages: firstly, doing clustering in low-dimensional spaces is more efficient and faster than clustering in bigger dimensions. Secondly, density-based clustering algorithms such as DBSCAN provide better results in low-dimensional spaces [16], because high-dimensional spaces are usually sparse, making it difficult to distinguish between high and low density regions. Finally, clustering multiple low-dimensional sub-spaces provides a finer-grained analysis, which improves the ability of UNADA to detect anomalies of very different characteristics. We shall therefore use $k = 2$ for SSC, which gives $N = m(m-1)/2$ partitions.

4.2. Ranking Outliers using Evidence Accumulation

Having produced the N partitions, the question now is how to use the information provided by the multiple clusters and outliers identified by density-based clustering to detect traffic anomalies. A possible answer is provided in [21], where authors introduced the idea of Evidence Accumulation Clustering (EAC). EAC uses the clustering results of multiple partitions P_i to produce a new inter-samples similarity measure that better reflects their natural groupings.

UNADA implements a particular algorithm for Evidence Accumulation, called Evidence Accumulation for Ranking Outliers (EA4RO): instead of producing a similarity measure between the n different aggregated flows described in $\mathbf{X}$, EA4RO constructs a dissimilarity vector $D \in \mathbb{R}^n$ in which it accumulates the distance between the different outliers o_j^i found in each sub-space $i = 1, .., N$ and the centroid of the corresponding sub-space-biggest-cluster $C_{\max}^i$. The idea is to clearly highlight those flows that are far from the normal-operation traffic at each of the different sub-spaces, statistically represented by $C_{\max}^i$.

Algorithm 1 presents a pseudo-code for EA4RO. The different parameters used by EA4RO are automatically set by the algorithm itself. The first two parameters are used by the density-based clustering algorithm: $n_{\min}$ specifies

Algorithm 1 Evidence Accumulation for Ranking Outliers (EA4RO)

Initialization:
  Set dissimilarity vector D to a null $n \times 1$ vector
  Set smallest cluster-size $n_{\min} = \alpha \, . \, n$
for $i = 1 : N$ **do**
  Set density neighborhood δ_i for DBSCAN
  P_i = DBSCAN $(\mathbf{X}_i, \delta_i, n_{\min})$
  Update $D(j)$, $\forall$ outlier $o_j^i \in P_i$:
    $w_i \leftarrow \dfrac{n}{(n - n_{\max_i}) + \epsilon}$
    $D(j) \leftarrow D(j) + \mathrm{d_M}(o_j^i, C_{\max}^i)\, w_i$
end for
Rank flows: $D_{rank} = \mathrm{sort}(D)$
Set anomaly detection threshold: T_h = find-slope-break(D_{rank})

the minimum number of flows that can be classified as a cluster, while δ_i indicates the maximum neighborhood distance of a sample to identify dense regions. $n_{\min}$ is set at the initialization of the algorithm, simply as a fraction α of the total number of flows n to analyze (we take $\alpha = 5\%$ of n). δ_i is set as a fraction of the average distance between flows in sub-space $\mathbf{X}_i$ (we take a fraction 1/10), which is estimated from 10% of the flows, randomly selected. This permits to fast-up computations. The weighting factor w_i is used as an outlier-boosting parameter, as it gives more relevance to those outliers that are "less probable": w_i takes bigger values when the size $n_{\max_i}$ of cluster $C_{\max}^i$ is closer to the total number of flows n. Finally, instead of using a simple Euclidean distance as a measure of dissimilarity, we compute the Mahalanobis distance $\mathrm{d_M}$ between outliers and the centroid of the biggest cluster. The Mahalanobis distance takes into account the correlation between samples, dividing the standard Euclidean distance by the variance of the samples. This permits to boost the degree of abnormality of an outlier when the variance of the samples is smaller.

In the last part of EA4RO, flows are ranked according to the dissimilarity obtained in D, and the anomaly detection threshold T_h is set. The computation of T_h is simply achieved by finding the value for which the slope of the sorted dissimilarity values in D_{rank} presents a major change. In the evaluation section we explain how to perform this computation with an example of real traffic analysis. Anomaly detection is finally done as a binary thresholding operation on D: if $D(i) > T_h$, UNADA flags an anomaly in flow $\mathbf{y}_i$.

5. Experimental Evaluation of UNADA

We evaluate the ability of UNADA to detect different attacks in real traffic traces from the public MAWI repository of the WIDE project [25]. The WIDE operational network provides interconnection between different

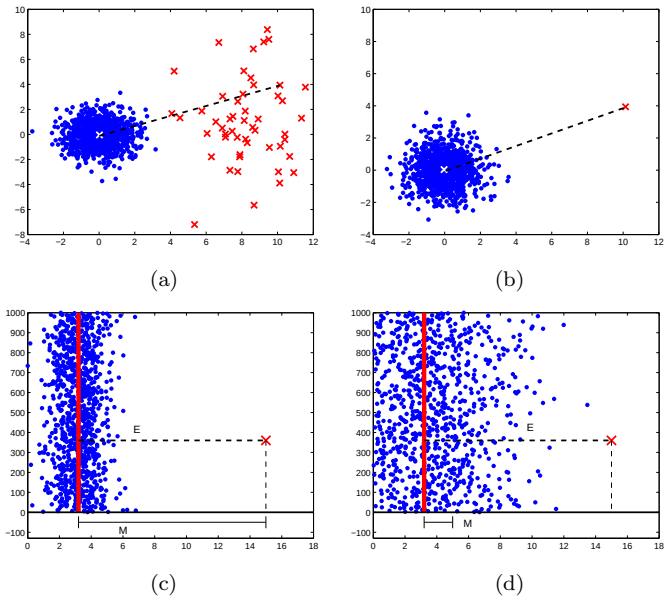


Figure 3: Evidence Accumulation for Ranking Outliers.

research institutions in Japan, as well as connection to different commercial ISPs and universities in the U.S.. The traffic repository consists of 15 minutes-long raw packet traces daily collected for the last ten years. The traces we shall work with consist of traffic from one of the trans-pacific links between Japan and the U.S.. MAWI traces are not labeled, but some previous work on anomaly detection has been done on them [15, 24]. In particular, [24] detects network attacks using a signature-based approach, while [15] detects both attacks and anomalous flows using non-Gaussian modeling. We shall therefore refer to the combination of results obtained in both works as our *ground truth* for MAWI traffic.

We shall also test the true positive and false positive rates obtained with UNADA in the detection of flooding attacks in traffic traces from the METROSEC project [26]. These traces consist of real traffic collected on the French RENATER network, containing simulated attacks performed with well-known DDoS attack tools. Traces were collected between 2004 and 2006, and contain DDoS attacks that range from very low intensity (i.e., less than 4% of the overall traffic volume) to massive attacks (i.e., more than 80% of the overall traffic volume). In addition, we compare the performance of UNADA against some previous methods for unsupervised anomaly detection presented in section 2.

5.1. Features Selection for Detection of Attacks

The selection of the m features used in $\mathbf{X}$ to describe the aggregated flows in $\mathbf{Y}$ is a key issue to any anomaly detection algorithm, but it becomes critical and challenging in the case of unsupervised detection, because there is no additional information to select the most relevant set. In general terms, using different traffic features permits

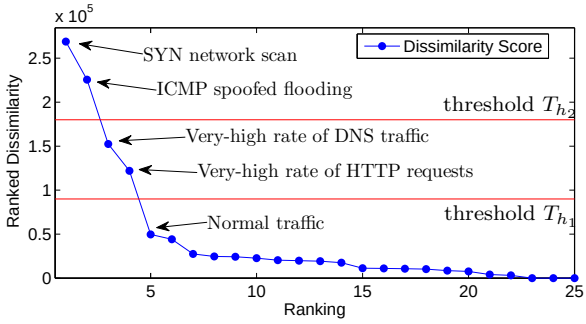
to detect different types of anomalies. In this paper we shall limit our study to detect well-known attacks, using a set of standard traffic features widely used in the literature. However, the reader should note that UNADA can be extended to detect other types of anomalies, considering different sets of traffic features. In fact, more features can be added to any standard list to improve detection results. For example, we could use the set of traffic features generally used in the traffic classification domain [23] for our problem of anomaly detection, as this set is generally broader; if these features are good enough to classify different traffic applications, they should be useful to perform anomaly detection. The main advantage of UNADA is that we have devised an algorithm to highlight outliers respect to any set of features, and this is why we claim that our algorithm is highly applicable.

In this paper we shall use the following list of $m = 9$ traffic features: number of source/destination IP addresses and ports (nSrcs, nDsts, nSrcPorts, nDstPorts), ratio of number of sources to number of destinations, packet rate (nPkts/sec), fraction of ICMP packets (nICMP/nPkts) and SYN packets (nSYN/nPkts), and average packet size (avgPktsSize). According to previous work on signature-based anomaly characterization [24], such simple traffic descriptors permit to describe standard network attacks such as DoS, DDoS, scans, and spreading worms/virus.

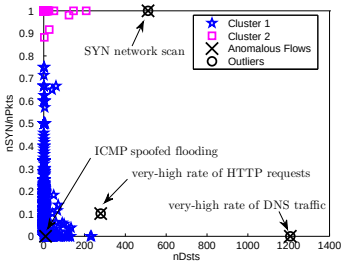
Table 1 describes the impacts of different types of attacks on the selected traffic features. All the thresholds used in the description are introduced to better explain the evidence of an attack in some of these features. DoS/DDoS attacks are characterized by many small packets sent from one or more source IPs towards a single destination IP. These attacks generally use particular packets such as TCP SYN or ICMP echo-reply. echo-request, or host unreachable packets. Port and network scans involve small packets from one source IP to several ports in one or more destination IPs, and are usually performed with SYN packets. Spreading worms differ from network scans in that they are directed towards a small specific group of ports for which there is a known vulnerability to exploit (e.g. Blaster on TCP port 135, Slammer on UDP port 1434, Sasser on TCP port 455), and they generally use slightly bigger packets. Some of these attacks can use other types of traffic, such as FIN, PUSH, URG TCP packets or small UDP datagrams.

5.2. Detecting Attacks in MAWI traffic

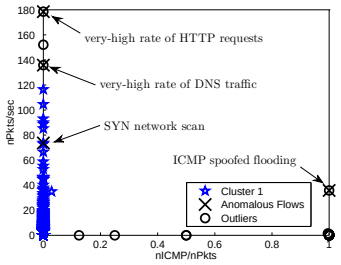
We begin by analyzing the performance of UNADA to detect network attacks and other types of anomalies in one of the traces previously analyzed in [15]. IP flows are aggregated with IPsrc key. Figure 4.(a) shows the ordered dissimilarity values in D obtained by the EA4RO method, along with their corresponding manual classification. The first two most dissimilar flows correspond to a highly distributed SYN network scan (more than 500 destination hosts) and an ICMP spoofed flooding attack directed to a small number of victims (ICMP redirect traffic towards port 0). The following two flows correspond to



(a) Unsupervised detection of network attacks and elephant flows.



(b) Scan and flooding attacks (1/2)



(c) Scan and flooding attacks (2/2)

Figure 4: Detection and analysis of network attacks in WIDE.

unusual large rates of DNS traffic and HTTP requests; from there on, flows correspond to normal-operation traffic. The ICMP flooding attack and the two unusual flows are also detected in [15]; the SYN scan was missed by their method, but it was correctly detected with accurate signatures [24]. Setting the detection threshold according to the previously discussed approach results in T_{h_1}. Indeed, if we focus on the shape of the ranked dissimilarity in figure 4.(a), we can clearly appreciate a major change in the slope after the $5th$ ranked flow. Note however that both attacks can be easily detected and isolated from the anomalous but yet legitimate traffic without false alarms, using for example the threshold T_{h_2} on D.

Figures 4.(b,c) depict the corresponding four anomalies in two of the N partitions produced by the EA4RO method. Besides showing typical characteristics of the attacks, such as a large value of nPkts/sec or a value 1 for attributes nICMP/nPkts and nSYN/nPkts respectively, both figures permit to appreciate that the detected attacks do not necessarily represent the largest elephant flows in the time slot. This emphasizes the ability of UNADA to detect attacks of low intensity, even of lower intensity than normal traffic.

5.3. Detecting Attacks with Ground Truth

Figure 5.3 depicts the True Positives Rate (TPR) as a function of the False Positives Rates (FTR) in the detection of different attacks in MAWI and METROSEC. Figure 5.3.(a) corresponds to the detection of 36 anomalies in MAWI traffic, using IPsrc as key. These anomalies include network and port scans, worm scanning activities (Sasser and Dabber variants), and some anomalous flows consisting on very high volumes of NNTP traffic. Figure 5.3.(b)

Type of Attack	Class	Agg-Key	Impact on Traffic Features
DoS (ICMP/SYN)	1-to-1	IPdst	nSrcs = nDsts = 1, nPkts/sec > λ_1, avgPktsSize < λ_2, nICMP/nPkts > λ_3, nSYN/nPkts > λ_4.
DDoS (ICMP/SYN)	N-to-1	IPdst	nDsts = 1, nSrcs > α_1, nPkts/sec > α_2, avgPktsSize < α_3, nICMP/nPkts > α_4, nSYN/nPkts > α_5.
Port scan	1-to-1	IPsrc	nSrcs = nDsts = 1, nDstPorts > β_1, avgPktsSize < β_2, nSYN/nPkts > β_3.
Network scan	1-to-N	IPsrc	nSrcs = 1, nDsts > δ_1, nDstPorts > δ_2, avgPktsSize < δ_3, nSYN/nPkts > δ_4.
Spreading worms	1-to-N	IPsrc	nSrcs = 1, nDsts > η_1, nDstPorts < η_2, avgPktsSize < η_3, nSYN/nPkts > η_4.

Table 1: Features used by UNADA in the detection of DoS, DDoS, network/port scans, and spreading worms. For each type of attack, we describe its impact on the selected traffic features.

also corresponds to anomalies in MAWI traffic, but using IPdst as key. In this case, there are 9 anomalies, including different kinds of flooding DoS/DDoS attacks. Finally, figure 5.3.(c) corresponds to the detection of 9 DDoS attacks in the METROSEC data-set. From these, 5 correspond to massive attacks (more than 70% of traffic), 1 to a high intensity attack (about 40%), 2 are low intensity attacks (about 10%), and 1 is a very-low intensity attack (about 4%). The detection is performed using traffic aggregated with IPdst key. In the three evaluation scenarios, the ROC plot is obtained by comparing the sorted dissimilarities in D_{rank} to a variable detection threshold.

We compare the performance of UNADA against three previous approaches for unsupervised anomaly detection: DBSCAN-based, k-means-based, and PCA-based outliers detection. The first two consist in applying either DBSCAN or k-means to the complete feature space $\mathbf{X}$, identify the largest cluster $C_{\max}$, and compute the Mahalanobis distance of all the flows lying outside $C_{\max}$ to its centroid. The ROC is finally obtained by comparing the sorted distances to a variable detection threshold. These approaches are similar to those used in previous work [18, 19, 20]. In the PCA-based approach, PCA and the sub-space methods [10, 11] are applied to the complete matrix $\mathbf{X}$, and the attacks are detected by comparing the residuals to a variable threshold. Both the k-means and the PCA-based approaches require fine tuning: in k-means, we repeat the clustering for different values of clusters k, and take the average results. In the case of PCA we present the best performance obtained for each evaluation scenario.

Obtained results permit to evidence the great advantage of using the SSC-Density-based algorithm in the clustering step with respect to previous approaches. In particular, all the approaches used in the comparison generally fail to detect all the attacks with a reasonable false alarm rate. Both the DBSCAN-based and the k-means-based algorithms get confused by masking features when analyzing the complete feature space $\mathbf{X}$. The PCA approach shows to be not sensitive enough to discriminate different kinds of attacks of very different intensities, using the same representation for normal-operation traffic.

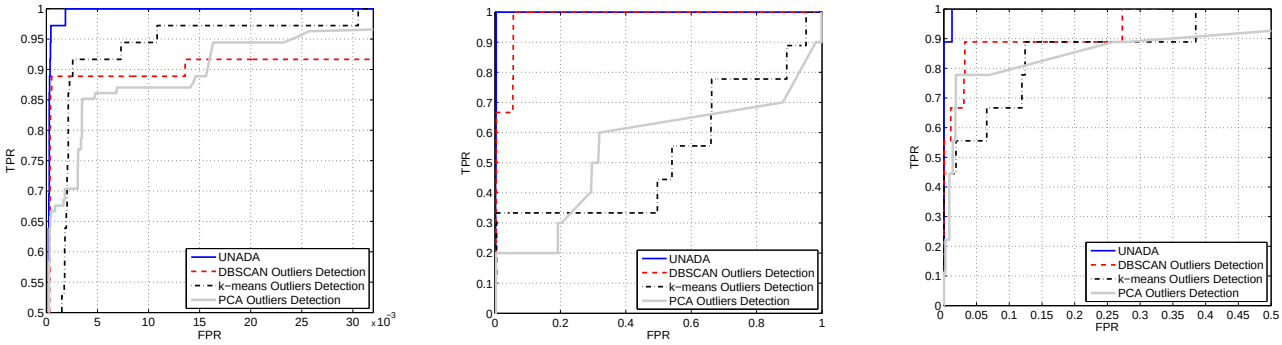
6. Experimental Evaluation in KDD99

In this section we evaluate the performance of the U-NIDS to detect network attacks in the well-known and widely used KDD99 network attacks dataset [**?**]. In addition, we compare its performance with that obtained by an extensively investigated misuse NIDS based on decision trees, as well as against some other methods for unsupervised detection available in the literature.

The KDD99 dataset contains a wide variety of intrusions simulated in a military network environment. Traffic consists of packets aggregated into connections, being a connection a flow of TCP packets between a source and a destination IP address. Simulated attacks include DoS attacks, unauthorized access from a remote machine - R2L attacks (e.g. password guessing), unauthorized access to super-user privileges - U2R attacks (e.g. buffer overflows), and probing attacks (e.g. port scanning). Each connection or flow is described by a set of $m = 41$ features (e.g. number of bytes, TCP flags, failed remote-login attemps, etc.) and a label indicating either the name of the attack or if the flow corresponds to normal-operation traffic.

The KDD99 dataset has become widely discredited. This letter is intended to briefly outline the problems that have been cited with the KDD Cup'99 dataset, and discourage its further use. Pero como lo nico que hacemos es comprar uno contra el otro, de ltima no pasa nada. Adems es mas evidencia de que funciona.

In order to compare the performance of the unsupervised system against a decision-tree-based NIDS we take two different data sub-sets, the first used for training issues and the second one for testing. The testing dataset corresponds to instances of different attacks that are not present in the training dataset, which will permit us to evidence the paramount advantage of doing unsupervised detection when new attacks arise. DoS and probing attacks in KDD99 are represented by a large number of flows, in some cases even more flows than those corresponding to normal-operation traffic. While this issue limits the use of the outliers detection technique (i.e. the attack is not an outlier in those cases), we shall see in the following section that real network traffic can be aggregated either at



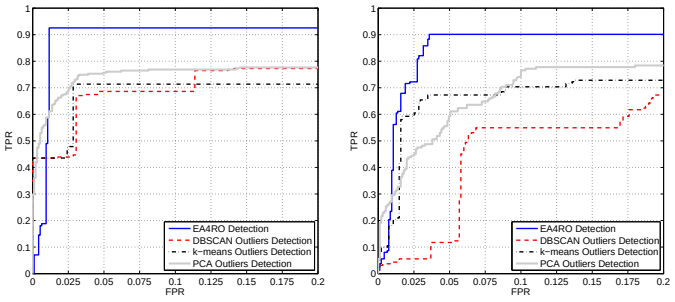
(a) MAWI, IPsrc key. (b) MAWI, IPdst key. (c) METROSEC, IPdst key.

Figure 5: True Positives Rate vs False Alarms in MAWI and METROSEC.

source or destination IP address to dramatically reduce the number of flows that compose a highly distributed and/or large volume attack. In order to avoid this limitation in the already processed flows of KDD99 (which can not be aggregated because we do not have the corresponding IP addresses), we shall select only a small fraction of flows for both DoS and probing attacks in our training and testing datasets. The training dataset has 950 normal flows and 255 attacks, while the testing dataset consists of 950 normal flows and 162 attacks.

Let us first evaluate the detection properties of the EA4RO-based detection algorithm used by the U-NIDS when changing the detection threshold T_h. Figure 6 depicts the True Positives Rate as a function of the False Positives Rates (i.e., the ROC curve) for both training and testing datasets. Figure 6.(a) shows the results obtained by applying the U-NIDS to the training dataset, while figure 6.(b) shows the results obtained when analyzing the testing dataset. In both cases we can appreciate that the EA4RO detection algorithm is able to detect a large fraction of attacks (more than 90%) with very low false positive rates (less than 1% and 3.5% respectively).

Figure 6 additionaly compares the detection performance obtained with EA4RO against three previous approaches for unsupervised anomaly detection proposed in the literature: DBSCAN-based, k-means-based, and PCA-based outliers detection. The first two consist in applying either DBSCAN or the celebrated k-means clustering algorithm [16] to the complete feature space $\mathbf{X}$, identify the largest cluster $C_{\max}$, and compute the Mahalanobis distance of all the flows lying outside $C_{\max}$ to its centroid. The ROC is finally obtained by comparing the sorted distances to a variable detection threshold. These approaches are similar to those used in previous work [18, 19, 20]. In the PCA-based approach, PCA and the sub-space methods [10, 11] are applied to the complete matrix $\mathbf{X}$, and the attacks are detected by comparing the residuals to a variable threshold. Both the k-means and the PCA-based approaches require fine tuning: in k-means, we repeat the clustering for different values of clusters k, and take the average results. In the case of PCA we present the best performance obtained for each evaluation scenario. In both cases we can appreciate the outperforming ability of the EA4OR detection algorithm w.r.t. these traditional approaches, which fail to detect as many attacks as EA4OR with a reasonable false alarms rate.

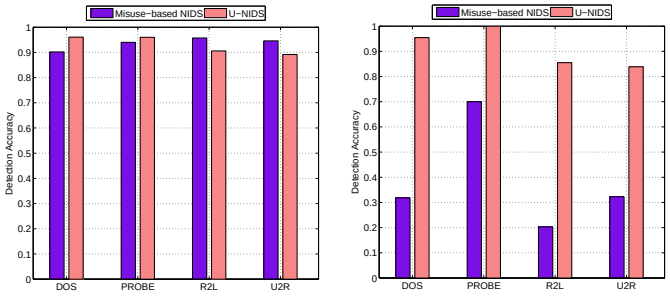


(a) Training dataset (b) Testing Dataset

Figure 6: True Positives Rate vs False Alarms in KDD99.

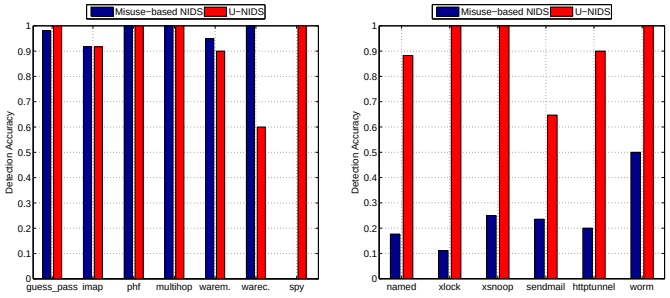
Now that we have shown the detection properties of the algorithms used in the U-NIDS, we shall compare its performance with a largely studied misuse-based NIDS built through decision trees. We shall build a different decision tree for each of the four different categories of attacks (DoS, probe, R2L, and U2R), using the training dataset and standard C4.5 decision trees [7, 8]. To train each of the trees for each different category of attacks, we consider that the flows belonging to the rest of the categories of attacks as well as the normal operation flows correspond to the "negative" class (there is no attack of the corresponding category). For example, let us suppose that we want to build a decision tree to detect R2L attacks; in that case, all the flows in the trainig dataset which belong to the R2L category belong to the "positive" class (there is a R2L attack), while the normal-operation flows as well as the DoS, probe, and U2R flows compose the negative class.

We shall compare the ability of the U-NIDS to detect unknown attacks with respect to a traditional misused-detection NIDS based on decision trees. Decision trees



(a) Training dataset (b) Testing Dataset

Figure 7: U-NIDS vs Misuse-based NIDS in KDD99.



(a) Training dataset (b) Testing Dataset

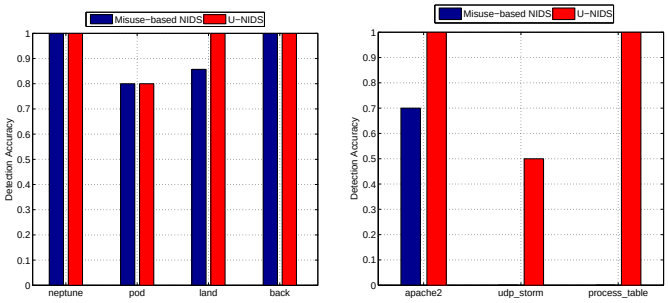
Figure 8: U-NIDS vs Misuse-based NIDS - R2L attacks.

permit to construct comprehensive signatures for network attacks in the form of multiple filtering-rules, using a graph structure.

Figure 7 presents the detection accuracy (number of correctly detected attacks) obtained either with the U-NIDS or with the misuse-based NIDS previously described in both the training and testing datasets. Results are individually presented for each of the four categories of attacks. As expected, results obtained by both systems are very similar in the training dataset: for the case of U-NIDS, we have already shown in figure 6 that more than 90% of the attacks can be correctly detected; in the case of the misuse-based NIDS, it is quite obvious that using the system to detect the attacks which has been programmed to alert on shall provide performant results. What it is interesting to appreciate is what happens with both systems when we try to detect unknown attacks. Figure 7.(b) evidences the limitations of misuse-based NIDS to detect unknown attacks, and more importantly, the paramount advantage of using our U-NIDS for detecting new previously unseen attacks. We refer the reader to figures 8, 9, 10, and 11 to appreciate the detection accuracy obtained with both systems for the different attacks on each of the four different categories.

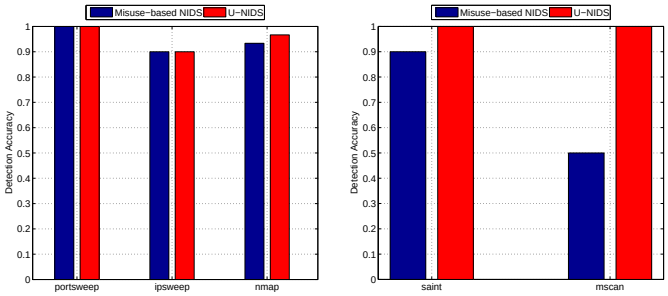
7. Detecting Attacks in Real Traffic Traces

Let us now evaluate the ability of the U-NIDS to detect different attacks in real traffic traces from the public



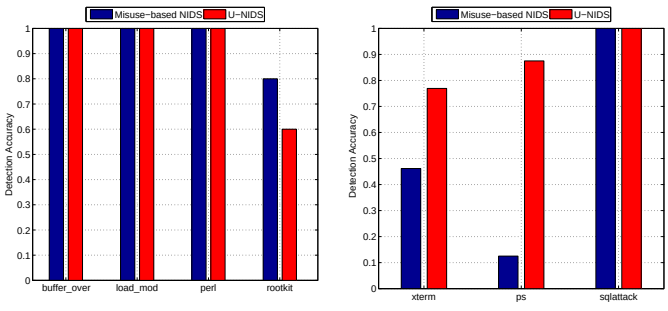
(a) Training dataset (b) Testing Dataset

Figure 9: U-NIDS vs Misuse-based NIDS - DoS attacks.



(a) Training dataset (b) Testing Dataset

Figure 10: U-NIDS vs Misuse-based NIDS - Probing attacks.



(a) Training dataset (b) Testing Dataset

Figure 11: U-NIDS vs Misuse-based NIDS - U2R attacks.

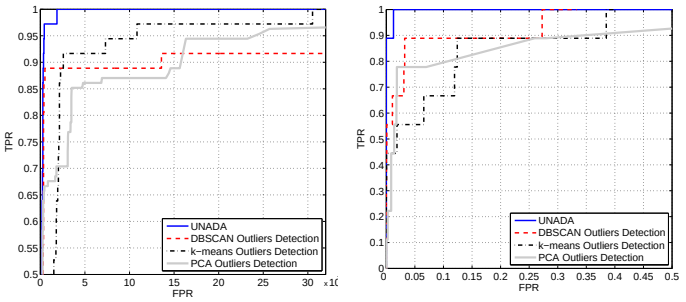
MAWI repository of the WIDE project [25]. The WIDE operational network provides interconnection between different research institutions in Japan, as well as connection to different commercial ISPs and universities in the U.S.. The traffic repository consists of raw packet traces daily collected for the last ten years. The traces we shall work with consist of traffic from one of the trans-pacific links between Japan and the U.S.. MAWI traces are not labeled, but some previous work on anomaly detection has been done on them [15, 24]. In particular, [24] detects network attacks using a signature-based approach, while [15] detects both attacks and anomalous flows using non-Gaussian modeling. We shall therefore refer to the combination of results obtained in both works as our *ground truth* for MAWI traffic.

We shall also test the true positive and false positive rates obtained in the detection of flooding attacks in traffic traces from the METROSEC project [26]. These traces consist of real traffic collected on the French RENATER network, containing simulated attacks performed with well-known DDoS attack tools. Traces were collected between 2004 and 2006, and contain DDoS attacks that range from very low intensity (i.e., less than 4% of the overall traffic volume) to massive attacks (i.e., more than 80% of the overall traffic volume). In addition, we compare the performance of the U-NIDS against the same unsupervised detection approaches presented in previous section.

In the following evaluations, we aggregate traffic either at source or destination IP address to show that the hypothesis of having a reduced number of flows as attacking flows is valid even in the case of distributed attacks. For example, if we define a flow as all the packets that are directed towards a single destination IP address, then this flow will be an outlier in the case of a DDoS, in which many different attackers send traffic to the same victim. As traffic features, we shall use in these evaluations basic traffic descriptors such as number of sources and destinations, number of source and destination ports, packet rate, fraction of SYN and ICMP packets, etc.

Figure 12 depicts the ROC curves obtained in the detection of different attacks in MAWI and METROSEC. Figure 12.(a) corresponds to the detection of 36 anomalies in MAWI traffic, using packets aggregated at source IP address. These anomalies include network and port scans, worm scanning activities (Sasser and Dabber variants), and some anomalous flows consisting on very high volumes of NNTP traffic. Figure 12.(b) corresponds to the detection of 9 DDoS attacks in the METROSEC dataset, using packets aggregated at destination IP address. From these, 5 correspond to massive attacks (more than 70% of traffic), 1 to a high intensity attack (about 40%), 2 are low intensity attacks (about 10%), and 1 is a very-low intensity attack (about 4%).

As before, obtained results permit to evidence the great advantage of using the EA4RO algorithm in the clustering step with respect to previously proposed approaches. In particular, all the approaches used in the comparison



(a) MAWI, IP source flows. (b) METROSEC, IP dest. flows.

Figure 12: True Positives Rate vs False Alarms in MAWI and METROSEC.

generally fail to detect all the attacks with a reasonable false alarm rate. Both the DBSCAN-based and the k-means-based algorithms get confused by masking features when analyzing the complete feature space $\mathbf{X}$. The PCA approach shows to be not sensitive enough to discriminate different kinds of attacks of very different intensities, using the same representation for normal-operation traffic.

8. Implementation Issues

The SSC-EA-based algorithm performs clustering in $N = m(m-1)/2$ low-dimensional sub-spaces $\mathbf{X}_i \in \mathbb{R}^2$. As we have shown, this provides a high discrimination power to detect and characterize different types of network attacks. However, the multiple clusterings computation increases the total Computational Time (CT) of the algorithm, imposing scalability issues for on-line detection of network attacks in very-high-speed networks. Scalability should be addressed regarding both the number of features used to describe traffic flows (m) and the number of flows to analyze (n). In the real traffic evaluations that we have presented, the number of flows captured in a time slot of $\Delta T = 20$ seconds rounds $n = 2500$ flows. For the $m = 9$ features that we have used, the total number of clusterings to compute is $N = 36$, which takes about 14.4 seconds in a standard single-processor machine.

Two key features of the SSC-EA-based algorithm can be exploited to reduce scalability problems in m and n. Firstly, clustering is performed in low-dimensional sub-spaces ($\mathbb{R}^2$), independently of the number of features that are used. Clustering in low-dimensional feature spaces is faster than in high-dimensional spaces [16], which partially alleviates the overhead of multiple clusterings computation. Secondly, the clustering of each sub-space $\mathbf{X}_i$ can be performed independently of the analysis on the other sub-spaces, which is perfectly adapted for parallel computing architectures. Parallel computing has become the dominant paradigm for accelerating specific tasks and represents a booming domain, driven by the availability of strong computational-power entities at low costs. Parallelization can be achieved in different ways: using a single multi-processor and multi-core machine, using GPU

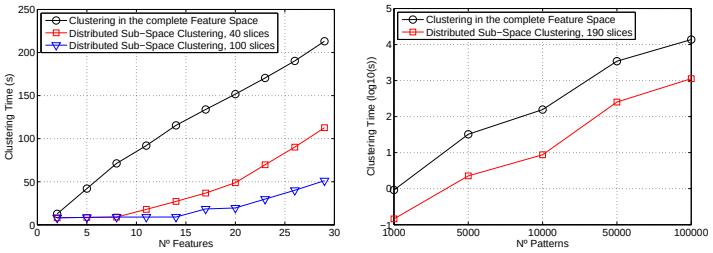
(Graphic Processor Unit) capabilities, using network processor cards, using a distributed group of machines, or combining these techniques. We shall use the term "slice" as a reference to a single computational entity.

Modern network processor cards are able to perform traffic monitoring even in 10 Gbps network connections. In a average-loaded 10 Gbps link (about 50%-60%) there are about 500.000 packets per second; if we consider traffic flows with an average rate of 500 kbps (about 50 pkts/sec) and a average duration of at least 20 seconds, then we have about $n = 10.000$ flows to analyze in each time slot of $\Delta T = 20$ seconds. From our experimentations, we know that we can analyze this number of flows using as much as $m = 20$ traffic descriptors in less than 20 seconds, using a parallel architecture with about 100 slices. Current network processor cards vendors offer multi-core solutions for high-performance networking with as much as 64 general purpose cores [**?**], which are perfectly adapted to deploy our unsupervised detection algorithm for very high speed knowledge-independent traffic monitoring.

9. Computational Time and Parallelization

The last issue that we analyze is the Computational Time (CT) of the algorithm. The SSC-EA-based algorithm performs multiple clusterings in $N(m)$ low-dimensional sub-spaces $\mathbf{X}_i \subset \mathbf{X}$. This multiple computation imposes scalability issues for on-line detection of attacks in very-high-speed networks. Two key features of the algorithm are exploited to reduce scalability problems in number of features m and the number of aggregated flows n to analyze. Firstly, clustering is performed in very-low-dimensional sub-spaces, $\mathbf{X}_i \in \mathbb{R}^2$, which is faster than clustering in high-dimensional spaces [16]. Secondly, each sub-space can be clustered independently of the other sub-spaces, which is perfectly adapted for parallel computing architectures. Parallelization can be achieved in different ways: using a single multi-processor and multi-core machine, using network processor cards and/or GPU (Graphic Processor Unit) capabilities, using a distributed group of machines, or combining these techniques. We shall use the term "slice" as a reference to a single computational entity.

Figure 13 depicts the CT of the SSC-EA-based algorithm, both (a) as a function of the number of features m used to describe traffic flows and (b) as a function of the number of flows n to analyze. Figure 13.(a) compares the CT obtained when clustering the complete feature space $\mathbf{X}$, referred to as CT($\mathbf{X}$), against the CT obtained with SSC, varying m from 2 to 29 features. We analyze a large number of aggregated flows, $n = 10^4$, and use two different number of slices, $M = 40$ and $M = 100$. The analysis is done with traffic from the WIDE network, combining different traces to attain the desired number of flows. To estimate the CT of SSC for a given value of m and M, we proceed as follows: first, we separately cluster each of the $N = m(m-1)/2$ sub-spaces $\mathbf{X}_i$, and take the worst-case of



(a) Time vs. n features. (b) Time vs. n patterns.

Figure 13: Computational Time as a function of n of features and n of flows to analyze. The number of aggregated flows in (a) is $n = 10000$. The number of features and slices in (b) is $m = 20$ and $M = 190$ respectively.

the obtained clustering time as a representative measure of the CT in a single sub-space, i.e., $\text{CT}(\mathbf{X}_{\text{SSCwc}}) = \max_i \text{CT}(\mathbf{X}_i)$. Then, if $N \leqslant M$, we have enough slices to completely parallelize the SSC algorithm, and the total CT corresponds to the worst-case, $\text{CT}(\mathbf{X}_{\text{SSCwc}})$. On the contrary, if $N > M$, some slices have to cluster various sub-spaces, one after the other, and the total CT becomes $(N\%M + 1)$ times the worst-case $\text{CT}(\mathbf{X}_{\text{SSCwc}})$, where % represents integer division. The first interesting observation from figure 13.(a) regards the increase of CT($\mathbf{X}$) when m increases, going from about 8 seconds for $m = 2$ to more than 200 seconds for $m = 29$. As we said before, clustering in low-dimensional spaces is faster, which reduces the overhead of multiple clusterings computation. The second paramount observation is about parallelization: if the algorithm is implemented in a parallel computing architecture, it can be used to analyze large volumes of traffic using many traffic descriptors in an on-line basis; for example, if we use 20 traffic features and a parallel architecture with 100 slices, we can analyze 10000 aggregated flows in less than 20 seconds.

Figure 13.(b) compares CT($\mathbf{X}$) against $\text{CT}(\mathbf{X}_{\text{SSCwc}})$ for an increasing number of flows n to analyze, using $m = 20$ traffic features and $M = N = 190$ slices (i.e., a completely parallelized implementation of the SSC-EA-based algorithm). As before, we can appreciate the difference in CT when clustering the complete feature space vs. using low-dimensional sub-spaces: the difference is more than one order of magnitude, independently of the number of flows to analyze. Regarding the volume of traffic that can be analyzed with this 100% parallel configuration, the SSC-EA-based algorithm can analyze up to 50000 flows with a reasonable CT, about 4 minutes in this experience. In the presented evaluations, the number of aggregated flows in a time slot of $\Delta T = 20$ seconds rounds the 2500 flows, which represents a value of $\text{CT}(\mathbf{X}_{\text{SSCwc}}) \approx 0.4$ seconds. For the $m = 9$ features that we have used ($N = 36$), and even without doing parallelization, the total CT is $N \times \text{CT}(\mathbf{X}_{\text{SSCwc}}) \approx 14.4$ seconds.

10. Conclusions

The Unsupervised Network Anomaly Detection Algorithm that we have proposed presents many interesting advantages with respect to previous proposals in the field of unsupervised anomaly detection. It uses exclusively unlabeled data to detect traffic anomalies, without assuming any particular model or any canonical data distribution, and without using signatures of anomalies or training. Despite using ordinary clustering techniques to identify traffic anomalies, UNADA avoids the lack of robustness of general clustering approaches, by combining the notions of Sub-Space Clustering, Density-based Clustering, and multiple Evidence Accumulation. We have verified the effectiveness of UNADA to detect real single source-destination and distributed network attacks in real traffic traces from different networks, all in a completely blind fashion, without assuming any particular traffic model, clustering parameters, or even clusters structure beyond a basic definition of what an anomaly is. Additionally, we have shown detection results that outperform traditional approaches for outliers detection, providing a stronger evidence of the accuracy of UNADA to detect network anomalies.

Acknowledgments

This work has been done in the framework of the European project ECODE, funded by the European Commission under grant FP7-ICT-2007-2/223936.

References

[1] S. Hansman and R. Hunt, "A Taxonomy of Network and Computer Attacks", in *Computers and Security*, vol. 24 (1), pp. 31-43, 2005.

[2] Symantec Enterprise Security, "Symantec Internet Security Threat Report, Trends for 2010", *Symantec Reports*, vol. 16, April 2011.

[3] N. Wyler, "Aggressive Network Self-Defense", ISBN 978-1-931836-20-3, Syngress - Elsevier, 2005.

[4] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network Anomaly Detection and Classification via Opportunistic Sampling", in *IEEE Network*, vol. 23 (1), 2009.

[5] R. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview", in *IEEE Security & Privacy Mag.*, vol. 35 (4), 2002.

[6] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection", in *IEEE Communications Surveys & Tutorials*, vol. 12 (3), 2010.

[7] C. Kruegel and T. Toth, "Using Decision Trees to Improve Signature-Based Intrusion Detection", in *Proc. RAID*, 2003.

[8] J. Lee, H. Lee, S. Sohn, J. Ryu, and T. Chung, "Effective Value of Decision Tree with KDD99 Intrusion Detection Datasets for IDS", in *Proc. ICACT*, 2008.

[9] P. Barford, J. Kline, D. Plonka, and A. Ron, "A Signal Analysis of Network Traffic Anomalies", in *Proc. ACM IMW*, 2002.

[10] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies", in *Proc. ACM SIGCOMM*, 2004.

[11] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions", in *Proc. ACM SIGCOMM*, 2005.

[12] A. Soule, K. Salamatian, and N. Taft, "Combining Filtering and Statistical Methods for Anomaly Detection", in *Proc. ACM IMC*, 2005.

[13] D. Brauckhoff, K. Salamatian, and M. May, "Applying PCA for Traffic Anomaly Detection: Problems and Solutions", in *Proc. INFOCOM*, 2009.

[14] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based Change Detection: Methods, Evaluation, and Applications", in *Proc. ACM IMC*, 2003.

[15] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, "Extracting Hidden Anomalies using Sketch and non Gaussian Multiresolution Statistical Detection Procedures", in *Proc. LSAD*, 2007.

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

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

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

[19] 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.

[20] K. Leung and C. Leckie, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clustering", in *Proc. ACSC05*, 2005.

[21] A. Fred and A. K. Jain, "Combining Multiple Clusterings Using Evidence Accumulation", in *IEEE Trans. Pattern Anal. and Machine Int.*, vol. 27 (6), 2005.

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

[23] N. Williams, S. Zander, and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification", in *ACM SIGCOMM Computer Communication Review*, vol. 36 (5), 2006.

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

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

[26] "METROlogy for SECurity and QoS", at `http://laas.fr/METROSEC`

[27] "Bro Intrusion Detection System", at `http://www.bro-ids.org`

[28] "SNORT: an Open Source Network Intrusion Prevention and Detection System", at `http://www.snort.org`