Computer Communications 35 (2012) 772–783

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge

Pedro Casas *, Johan Mazel, Philippe Owezarski

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

ARTICLE INFO

*Article history:*
Received 31 August 2011
Received in revised form 10 January 2012
Accepted 20 January 2012
Available online 28 January 2012

*Keywords:*
NIDS
Unsupervised Machine Learning
Sub-Space Clustering
Evidence Accumulation
Outliers detection

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. Finally, we show that the algorithms used by UNIDS are highly adapted for parallel computation, which permits to drastically reduce the overall analysis time of the system.



## 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.

* Corresponding author at: CNRS, LAAS, 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France. Tel.: +33 (0)5 61 33 68 05; fax: +33 (0)5 61 33 64 11.
*E-mail addresses:* pcasas@laas.fr (P. Casas), jmazel@laas.fr (J. Mazel), owe@laas.fr (P. Owezarski).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2012.01.016

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 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 [6] (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 to blindly extract the traffic instances that compose an attack. This unsupervised security system runs in three consecutive steps, analyzing packets captured in contiguous time slots of fixed length. Fig. 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. In this paper we shall use a standard change-detection algorithm [4] on three very simple and traditionally used volume metrics, consisting of # bytes, # packets, and # flows per time slot. The choice of volume metrics is based on [11], but change-detection can be performed on any other traffic metric sensitive to anomalies. The algorithm basically flags an anomaly when the derivative of any of these metrics exceeds a detection threshold, dynamically computed from the variance of previous anomaly-free measurements. The reader should bear in mind that this change-detection step is not a critical part of UNIDS, but that it is merely used to limit the frequency of usage of the clustering step, which is certainly more expensive in terms of computational resources.

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) [18], Density-based Clustering [23], and Evidence Accumulation Clustering (EAC) [22] 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 step**, the top-ranked outlying flows are flagged as anomalies, using a simple thresholding detection approach.

As we show through out the paper, the main contribution of UNIDS relies on its ability to detect unknown attacks in a completely unsupervised fashion, avoiding the need for signatures, training, or labeled traffic flows. This paper represents a continuation of our previous work on unsupervised anomaly detection [5]. In particular, we show that UNIDS can be used to detect unknown network attacks of very different nature, outperforming traditional misuse-detection-based systems; as such, we provide more evidence and solid results on the quality and relevance of our proposals for unsupervised anomaly detection. In addition, we show that the computational time involved in the unsupervised traffic analysis can be drastically reduced w.r.t the system presented in [5], by simply taking advantage of the parallel structure of the multi-clustering algorithm used in the core of UNIDS.

The remainder of the paper is organized as follows. Section 2 presents a brief state of the art in the network intrusion and anomaly detection fields, 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 to identify an anomalous time-slot. Section 4 describes the core of UNIDS, presenting an in depth description of the different clustering techniques used to construct the outliers ranking. Section 5 presents the validation of UNIDS in real traffic traces obtained from two networking datasets: the public MAWI traffic repository of the WIDE project [25], and the METROSEC project dataset [27]. In this section we also compare the performance of UNIDS against previous proposals for unsupervised detection of attacks available in the literature. Section 6 evaluates the ability of UNIDS 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. Implementation related issues of UNIDS, including evaluation of computational

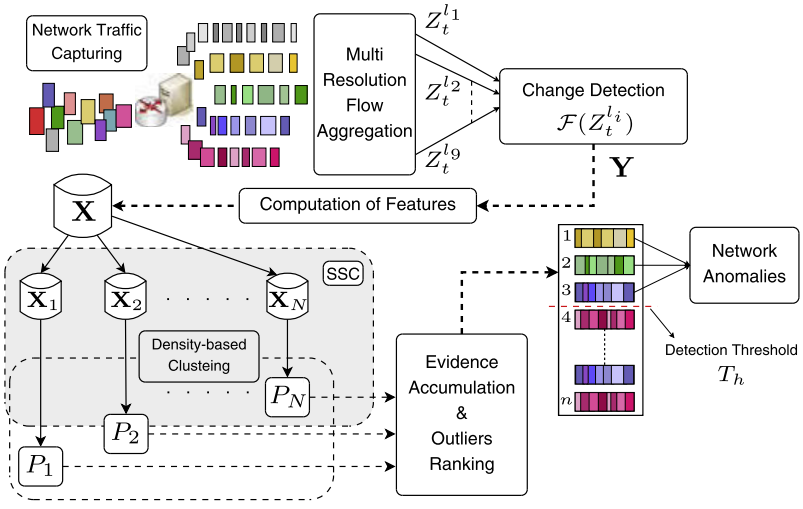



**Fig. 1.** High-level description of the Unsupervised Network Intrusion Detection System (UNIDS).

time and scalability, are discussed in Section 7. Finally, Section 8 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 [28] and SNORT [29] the two most celebrated open-source systems in the literature. Most of the different techniques traditionally applied in the NIDS field are summarized in [7,8]. A particularly studied approach for misuse detection that we shall include in this paper for performance-comparison consists in the use of decision trees [9,10]; in simple words, a tree-like graph is constructed from a training set of labeled *attack*/*normal operation* flows and then used to classify the nature of new incoming traffic flows.

Our approach falls within the unsupervised attacks detection domain. The vast majority of the unsupervised detection schemes proposed in the literature are based on clustering and outliers detection, being [19–21] some relevant examples. In [19], authors use a single-linkage hierarchical clustering method to cluster data from the KDD99 dataset, based on the standard Euclidean distance for inter-patterns similarity. Reference [20] reports improved results in the same dataset, using three different clustering algorithms: Fixed-Width clustering, an optimized version of *k*-NN, and one class SVM. Ref. [21] presents a combined density-grid-based clustering algorithm to improve computational complexity, obtaining similar detection results. PCA-based anomaly detection represents another well-known unsupervised technique, used in [12,13] to detect network wide traffic anomalies in highly aggregated traffic flows, and more recently in [14] for anomaly detection in single router traffic metrics.

The Unsupervised Network Intrusion Detection System presented in this paper 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 [17]. Finally, we shall evidence that UNIDS clearly outperforms some previously proposed methods for unsupervised detection in both artificial and real network traffic.

## 3. Multi-resolution flows & change-detection

UNIDS performs unsupervised detection of network attacks on packet-level traffic, captured at a single monitoring point in consecutive time slots of fixed length $\Delta T$, and aggregated in IP flows (standard *5-tuples*). UNIDS could be extended to work on network-wide traffic captured at multiple vantage points, but only in the case of accurate time-synchronization between measuring probes, and having access to the aggregated packet traces from all monitoring points. In any case, this paper considers only traffic captured at a single monitoring point for the evaluations. IP flows are additionally aggregated at different *flow-resolution* levels, using 9 different *aggregation keys* $l_i$, please refer to Fig. 1. 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 the simple aforementioned traffic metrics including number of bytes, packets, and 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.

Fig. 2 shows how a low intensity DDoS attack might be dwarfed by highly-aggregated traffic. The time-series associated with the number of packets, $Z_t = \#\text{pkts}_t$, 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 NIDS through clustering

The unsupervised detection step takes as input all the IP flows in the flagged time slot. At this step UNIDS 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

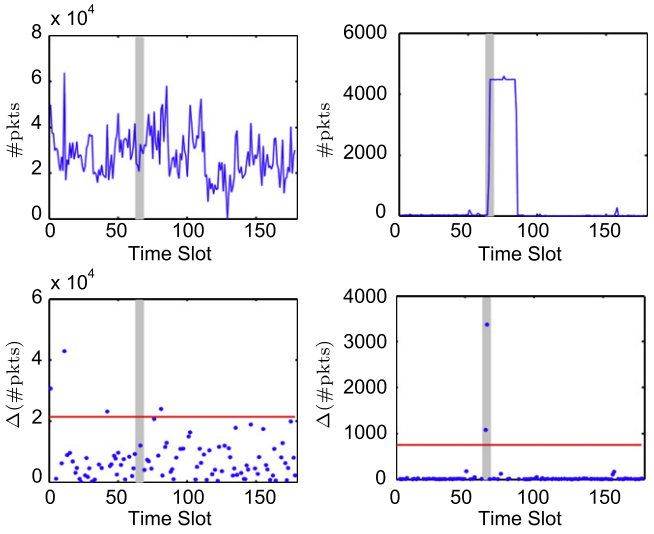



**Fig. 2.** Low-intensity anomalies might be hidden inside highly aggregated traffic, but are visible at finer-grained aggregations.

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 number of sources and ports, or packet rate. Let $\mathbf{x}_i \in \mathbb{R}^m$ be the vector of features describing flow $\mathbf{y}_i$, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{m \times m}$ the complete matrix of features, referred to as the *feature space*.

UNIDS 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 defined for the underlying samples' space. Very different notions of similarity or *distance* are used in the clustering domain, ranging from simple Euclidean distance to more complex notions such as statistical-based similarity. Samples that do not belong to any of the generated 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 [17], 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*. These ideas are well-known within the machine-learning community, but the application of these techniques for anomaly and intrusions detection 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. For example, multiple partitions can be generated by using different clustering algorithms, or by applying the same clustering algorithm with different setting parameters or initializations. We particularly use Sub-Space Clustering (SSC) [18] to produce multiple data partitions, doing density-based clustering in $N$ different sub-spaces $\mathbf{X}_i$ of the original space, please refer to Fig. 1.

### 4.1. Clustering ensemble and Sub-Space Clustering

Each of the $N$ sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projection of $\mathbf{X}$ into $k$ features out of the $m$ attributes, resulting in $N$ $k$-dimensional sub-spaces. 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 $m$. Fig. 3 explains this approach; in the example, a 3-dimensional feature space $\mathbf{X}$ is projected into $N = 3$ 2-dimensional sub-spaces $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$, which are then independently partioned via density-based clustering. Each partition $P_i$ is obtained by applying DBSCAN [23] to sub-space $\mathbf{X}_i$. DBSCAN is a powerful clustering algorithm that discovers clusters of arbitrary shapes and sizes [17], 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.

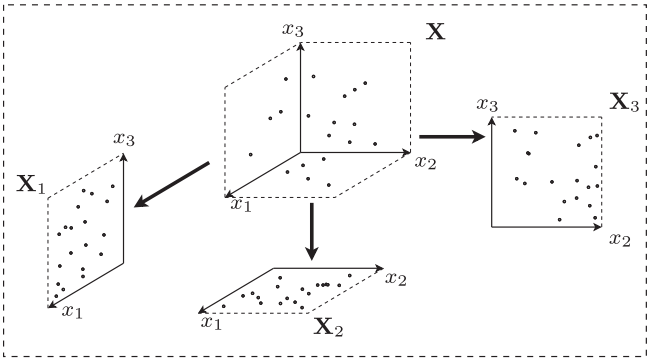



**Fig. 3.** Sub-Space Clustering: 2-dimensional sub-spaces $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$ are obtain from a 3-dimensional feature space $\mathbf{X}$ by simple projection. Units in the graph are irrelevant.

The results obtained by applying DBSCAN to sub-space $\mathbf{X}_i$ are twofold: a set of $p(i)$ clusters $\{C_1^i, C_2^i, \ldots, C_{p(i)}^i\}$ and a set of $q(i)$ outliers $\{o_1^i, o_2^i, \ldots, o_{q(i)}^i\}$. 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 [17], 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 UNIDS to detect attacks of very different characteristics. We shall therefore use $k = 2$ for Sub-Space Clustering, which gives $N = m(m-1)/2$ partitions.

**Algorithm 1:** Ranking Outliers via EA4RO

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

### 4.2. Ranking Outliers using Evidence Accumulation

Having produced the $N$ partitions, the question now is how to use the multiple identified clusters and outliers to detect network attacks and other anomalies. A possible answer is provided in [22], 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.

UNIDS implements a particular algorithm for Evidence Accumulation, which we shall refer to as 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 subspace $i = 1, \ldots, N$ and the centroid of the corresponding subspace-biggest-cluster $C_{\max}^i$. The idea behind EA4RO is to clearly highlight those aggregated flows that are remarkably different from the normal-operation traffic at each of the different sub-spaces, statistically represented by $C_{\max}^i$. The reader should remember here that even highly distributed attacks composed of thousands of IP flows can be represented as outliers using traffic aggregation.

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 the minimum number of flows that can be classified as a cluster, while $\delta_i$ 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 $\alpha$ of the total number of flows $n$ to analyze (we take $\alpha = 5\%$ of $n$). $\delta_i$ 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 speed-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". Figs. 4(a) and (b) explain this idea: $w_i$ takes bigger values when the size $n_{\max_i}$ of cluster $C_{\max}^i$ is closer to the total number of flows $n$, implying that this outlier is particularly different from the majority.

Finally, instead of using a simple Euclidean distance as a measure of dissimilarity, we compute the Mahalanobis distance $d_M$ 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. Figs. 4(c) and (d) permit to evidence the advantage of using such a measure. The Euclidean distance between the centroid of the cluster and the outlier is the same in both figures. However, variance in Fig. 4(c) is bigger than in Fig. 4(d), which intuitively should reduce the abnormality degree of the outlier. Such a notion is correctly captured by the Mahalanobis distance, which is smaller in Fig. 4(c).

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. Detection is finally done as a binary thresholding operation on $D_{rank}$: if $D_{rank}(i) > T_h$, UNIDS flags an anomaly in flow $\mathbf{y}_i$.

## 5. UNIDS performance detection in real traffic

We evaluate the ability of UNIDS 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 research institutions in Japan, as well as connection to different commercial ISPs and universities in the US. The traffic repository consists of 15 min-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 US. MAWI traces are not labeled, but some previous work on anomaly detection has been done on them [16,24]. In particular, [24] detects network attacks using a signature-based approach, while [16] 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 UNIDS in the detection of flooding attacks in traffic traces from the METROSEC project [27]. 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 UNIDS 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 intrusion detection system, 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, as shown in Section 6, UNIDS can be extended to detect different types of anomalies, considering different sets of traffic features. In fact, more features can be added to any standard list to improve detection results. The main advantage of UNIDS 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 section 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. In Section 6 we show how UNIDS can efficiently detect other kinds of attacks such as buffer overflows, password guessing attacks, and unauthorized access to network resources when considering other types of traffic descriptors.

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. The aforementioned

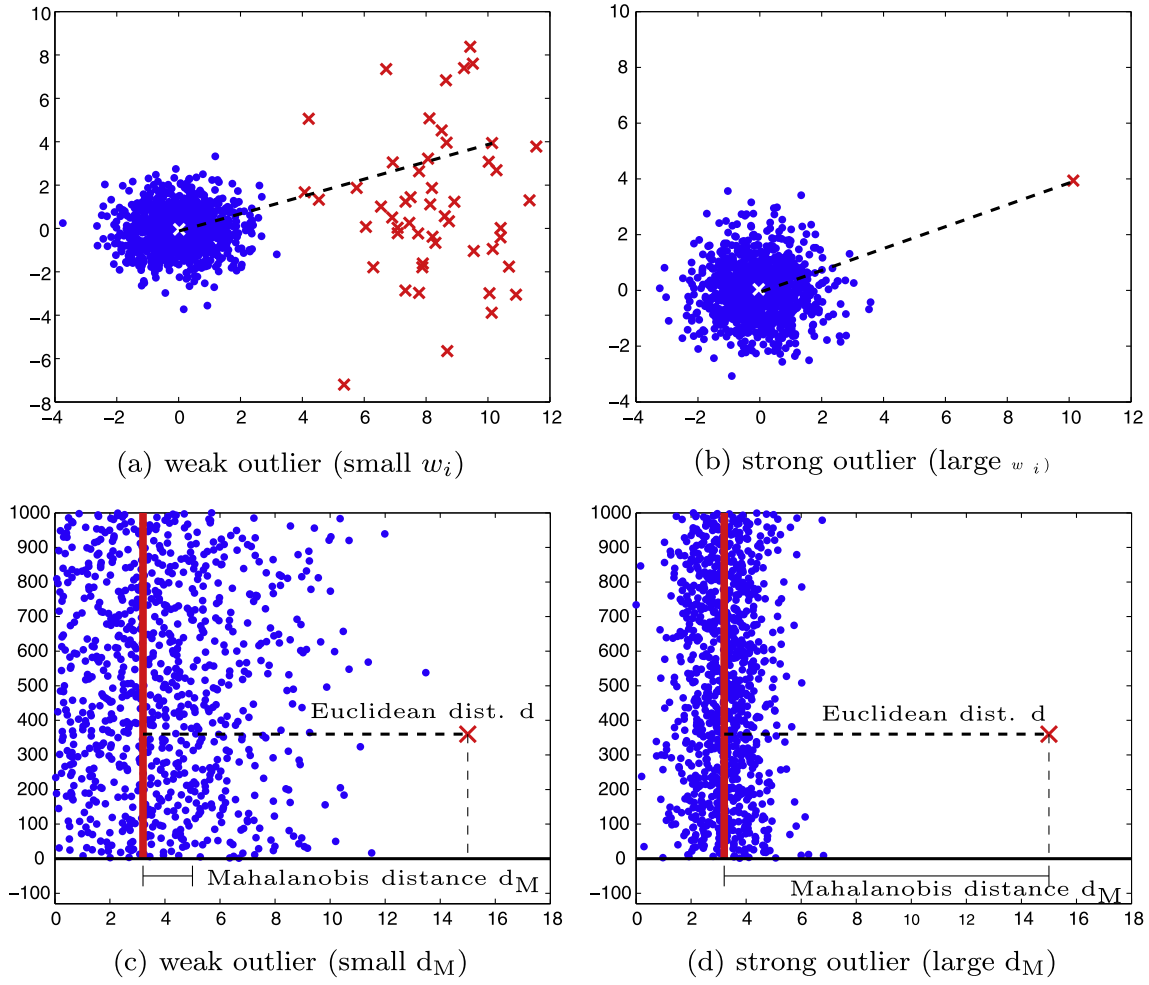



**Fig. 4.** Evidence Accumulation for Ranking Outliers. Weighting factor $w_i$ and Mahalanobis distance $d_M$ both permit to boost the degree of abnormality of an outlying flow. Units in these graphs are irrelevant.

**Table 1**
Features used by UNIDS 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.

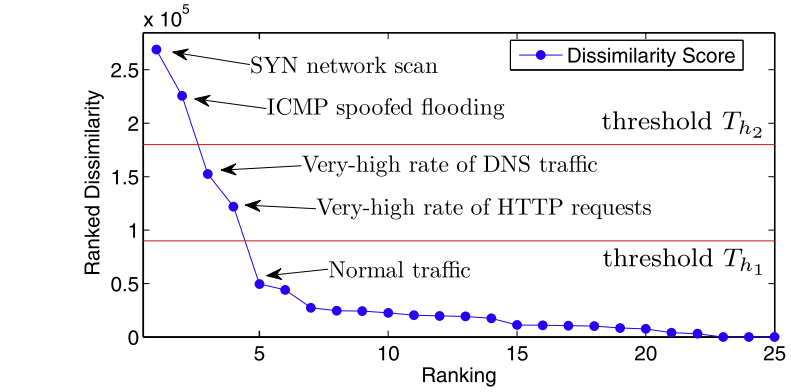
| Type of Attack | Class | Agg-Key | Impact on traffic features |
|---|---|---|---|
| DoS (ICMP/SYN) | 1-to-1 | IPdst | $nSrcs = nDsts = 1,\ nPkts/sec > \lambda_1,\ avgPktsSize < \lambda_2,\ nICMP/nPkts > \lambda_3,\ nSYN/nPkts > \lambda_4$ |
| DDoS (ICMP/SYN) | N-to-1 | IPdst | $nDsts = 1,\ nSrcs > \alpha_1,\ nPkts/sec > \alpha_2,\ avgPktsSize < \alpha_3,\ nICMP/nPkts > \alpha_4,\ (nSYN/nPkts > \alpha_5$ |
| Port scan | 1-to-1 | IPsrc | $nSrcs = nDsts = 1,\ nDstPorts > \beta_1,\ avgPktsSize < \beta_2,\ nSYN/nPkts > \beta_3.$ |
| Network scan | 1-to-N | IPsrc | $nSrcs = 1,\ nDsts > \delta_1,\ nDstPorts > \delta_2,\ avgPktsSize < \delta_3,\ nSYN/nPkts > \delta_4.$ |
| Spreading worms | 1-to-N | IPsrc | $nSrcs = 1,\ nDsts > \eta_1,\ nDstPorts < \eta_2,\ avgPktsSize < \eta_3,\ nSYN/nPkts > \eta_4.$ |

particularities permit to differentiate these kinds of distributed attacks from other distributed yet legitimate traffic (e.g., P2P traffic). Section 5.2 shows practical evidence on this claim.

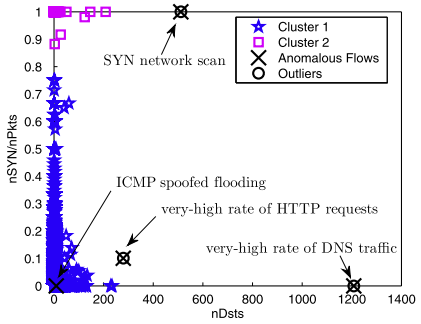
### 5.2. Detecting attacks in MAWI traffic

We begin by analyzing the performance of UNIDS to detect network attacks and other types of anomalies in one of the traces previously analyzed in [16]. IP flows are aggregated with IPsrc key, i.e., each aggregated flow corresponds to many IP flows coming from the same IP address. Fig. 5(a) shows the ordered dissimilarity values in $D_{rank}$ obtained by the EA4RO algorithm, 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 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 [16]; 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 Fig. 5(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 $D_{rank}$.

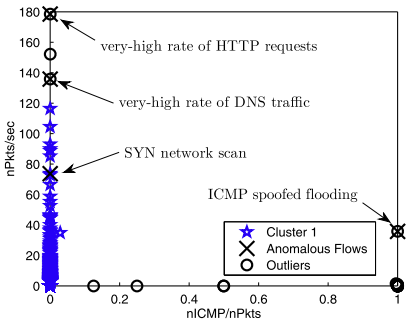
Figs. 5(b) and (c) depict the corresponding four anomalous aggregated flows in two of the $N$ partitions produced by Sub-Space Clustering and the EA4RO algorithm. Besides showing typical characteristics of the attacks, such as a large value of nPkts/sec or a




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




(b) Scan and flooding attacks




(c) Scan and flooding attacks

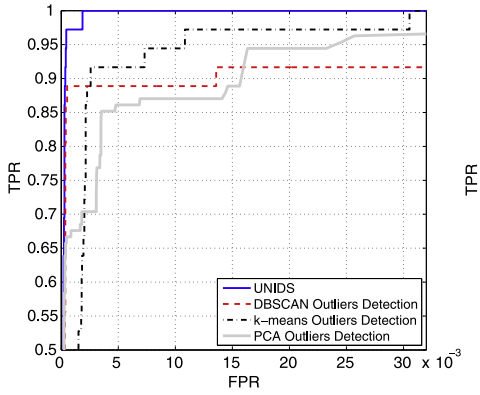
**Fig. 5.** Detection and analysis of network attacks in WIDE.

100% of ICMP/SYN packets, 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 UNIDS to detect attacks of low intensity, even of lower intensity than normal traffic.

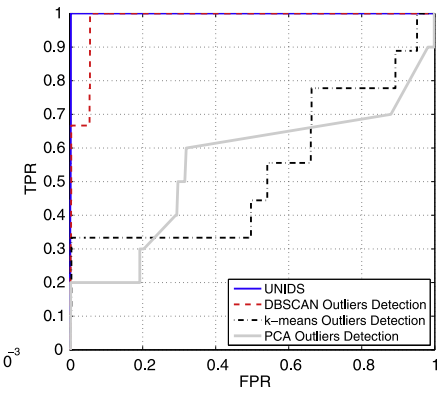
### 5.3. Detecting attacks with ground truth

Fig. 6 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. Fig. 6(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. Fig. 6(b) 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, Fig. 6(c) corresponds to the detection of 9 DDoS attacks in the METROSEC dataset. 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.

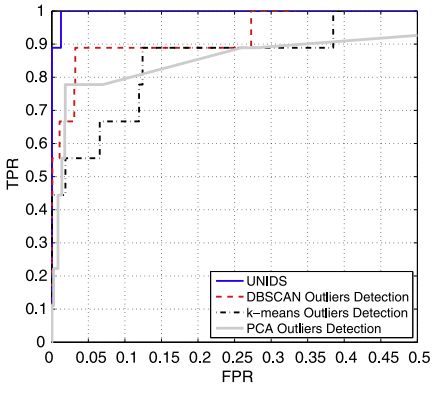
We compare the performance of UNIDS against three previously used approaches for unsupervised detection of network attacks and anomalies: DBSCAN-based, *k*-means-based, and PCA-based outliers detection. The first two consist in applying either DBSCAN or *k*-means to the complete feature space **X**, identify the largest




(a) MAWI, IPsrc key.




(b) MAWI, IPdst key.




(c) METROSEC, IPdst key.

**Fig. 6.** True Positives Rate vs false alarms in MAWI and METROSEC.

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 [19–21]. In the PCA-based approach, PCA and the sub-space methods [12,13] are applied to the complete matrix **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 EA4RO algorithm for outliers detection 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 **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. This limitation has already been detected in previous work [15].

## 6. UNIDS vs NIDS in KDD99

In this section we evaluate the ability of UNIDS to detect network attacks in the well-known and widely used KDD99 network attacks dataset [26]. The main interest of this evaluation is to show that UNIDS can detect other kinds of attacks when considering other types of traffic descriptors. As we did in Section 5, we shall compare UNIDS against the previous methods used for unsupervised detection, namely DBSCAN-based, *k*-means-based, and PCA-based detection. In addition, we compare UNIDS performance with that obtained by an extensively investigated misuse NIDS based on decision trees. Decision trees permit to construct comprehensive signatures for network attacks in the form of multiple filtering-rules, using a graph structure.

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 attempts, etc.) and a label indicating either the name of the attack or if the flow corresponds to normal-operation traffic.

In order to compare the performance of UNIDS 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 by the way 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 was not an issue for UNIDS when evaluated in previous section with real traffic traces, KDD99 flows provided at [26] are already pre-processed and can not be aggregated because the corresponding IP addresses are not available. To avoid this limitation in the already processed KDD99 flows, we shall select only a small fraction of flows for both DoS and probing attacks in our training and testing datasets. In any case, the reader should remember that we have already proved that UNIDS is able to detect both DoS and probing attacks in real traffic traces when using flows aggregation. 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 True Positives and False Positives rates obtained by UNIDS in both training and testing datasets. Fig. 7 depicts the corresponding ROC curves. Fig. 7(a) shows the results obtained by applying UNIDS to the training dataset, while Fig. 7(b) shows the results obtained when analyzing the testing dataset. In both cases we can appreciate that UNIDS 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). Fig. 7 additionally compares the obtained detection performance against the three previous approaches used for unsupervised detection. In both cases we appreciate once again the outperforming ability of UNIDS w.r.t. these approaches, which fail as before to detect as many attacks as UNIDS with a reasonable false alarms rate.

Let us now compare UNIDS against 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 [9,10]. 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 training 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. In order to avoid over-fitting problems when training the decisions trees, we have decided to use early-stopping learning. For this reason, not all the attacks present in the training dataset are detected by the decision-trees-based NIDS.

Fig. 8 presents the detection accuracy (number of correctly detected attacks) obtained either with UNIDS or with the 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: in the case of UNIDS, we have already shown in Fig. 7 that more than 90% of the attacks can be correctly detected; in the case of the NIDS, it is quite obvious that using the system to detect the attacks which has been programmed to alert on shall provide proper results. What it is interesting to appreciate is what happens with both systems when we try to detect unknown attacks. Fig. 8(b) evidences the limitations of the NIDS to detect unknown attacks, and more importantly, the paramount advantage of using our UNIDS for detecting new previously unseen attacks. We refer the reader to Fig. 9 to appreciate the detection accuracy obtained with both systems for the different attacks on each of the four different attacks categories available in the KDD99 dataset.

## 7. Computational Time and Parallelization

The last issue that we analyze is the Computational Time (CT) of UNIDS. The EA4RO algorithm performs multiple clusterings in $N = m(m-1)/2$ low-dimensional sub-spaces $\mathbf{X}_i \subset \mathbf{X}$. This multiple computation imposes scalability issues for detection of attacks in 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 [17]. Secondly,

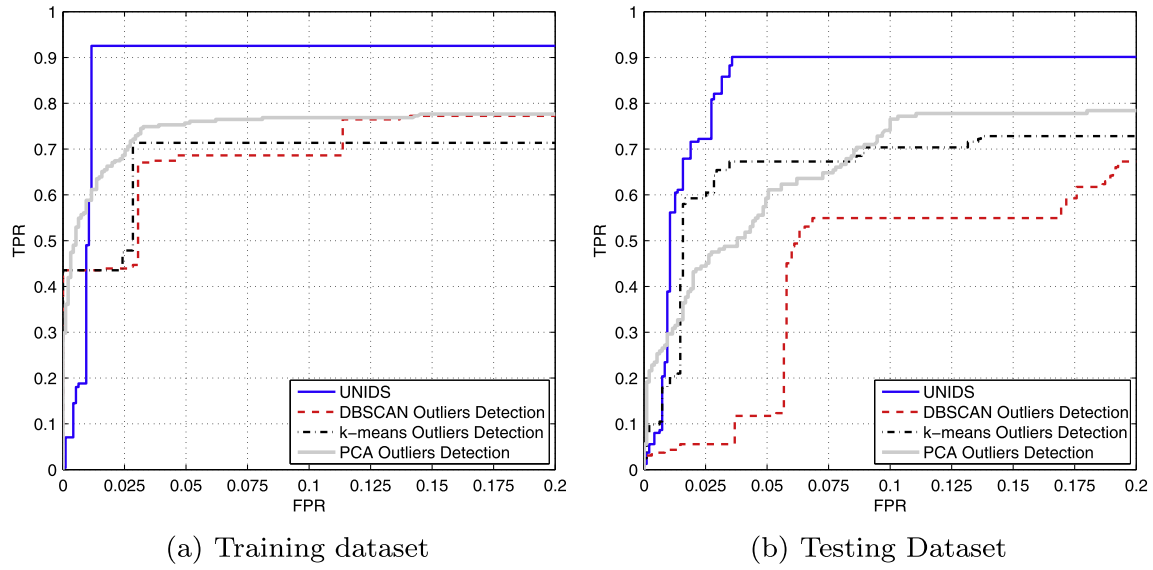



**Fig. 7.** True Positives Rate vs false alarms in KDD99.

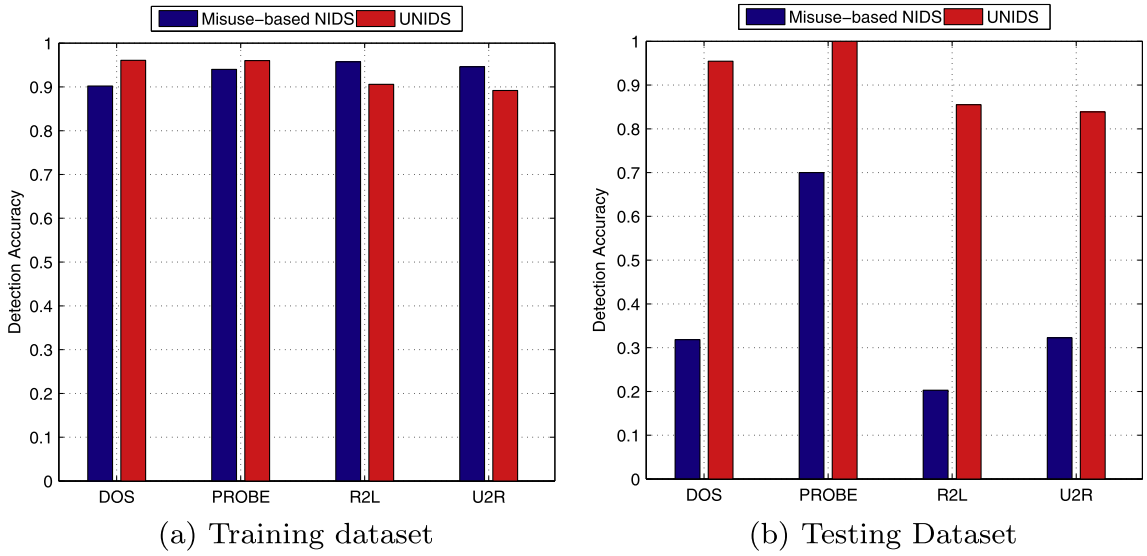
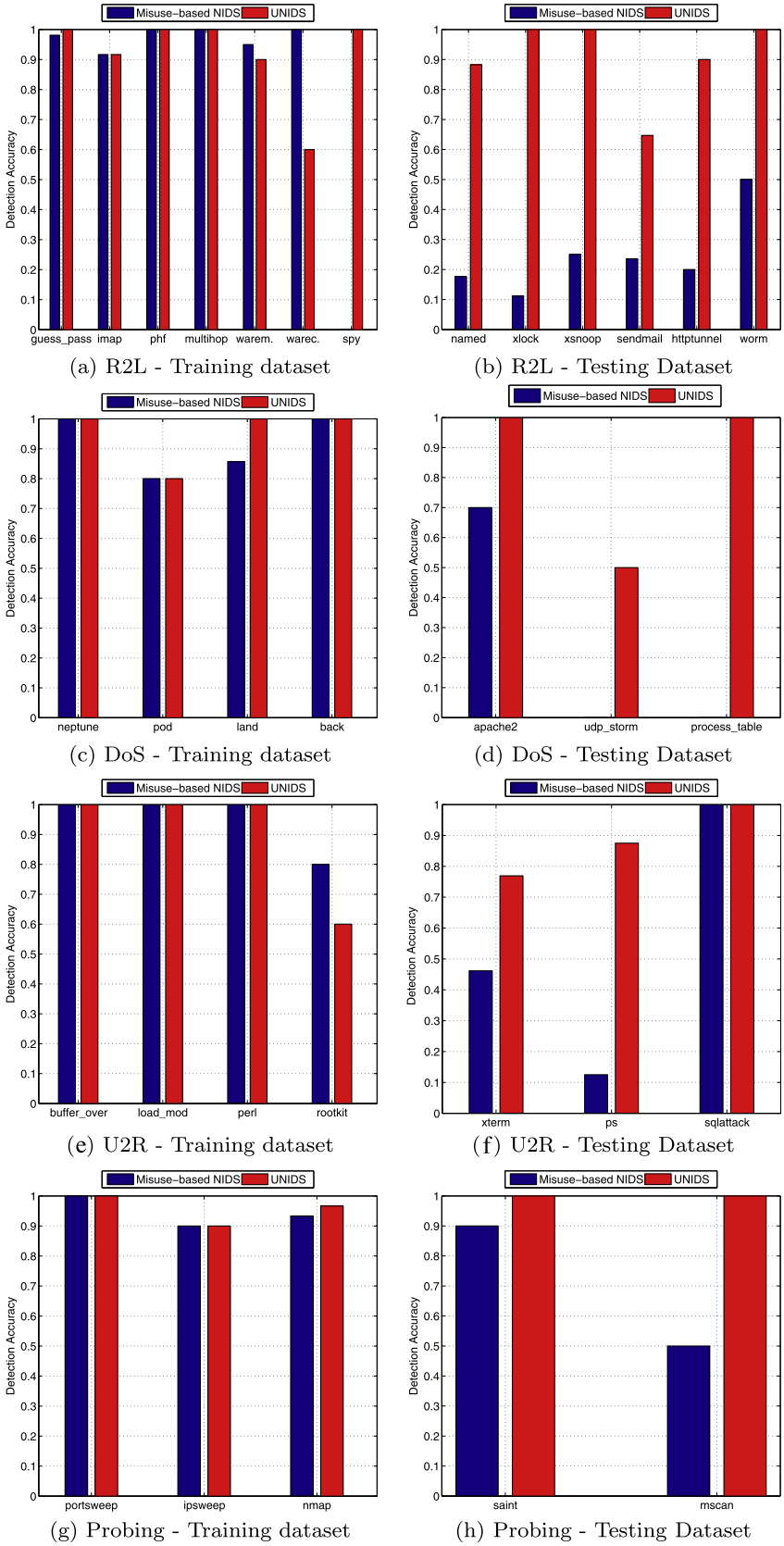



**Fig. 8.** UNIDS vs Misuse-based NIDS in KDD99.

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.

Fig. 10 depicts the CT of the EA4RO 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. Fig. 10 (a) compares the CT obtained when clustering the complete feature space $\mathbf{X}$, referred to as CT($\mathbf{X}$), against the CT obtained with EA4RO, 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 EA4RO for a given value of $m$ and $M$, we proceed as follows: first, we separately cluster each of the $N$ sub-spaces $\mathbf{X}_i$, and take the worst-case of the obtained clustering time as a representative measure of the CT in a single sub-space, i.e., $\mathrm{CT}(\mathbf{X}_{\mathrm{SSCwc}}) = \max_i \mathrm{CT}(\mathbf{X}_i)$. Then, if $N \leqslant M$, we have enough slices to completely parallelize the algorithm, and the total CT corresponds to the worst-case, $\mathrm{CT}(\mathbf{X}_{\mathrm{SSCwc}})$. 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 $\mathrm{CT}(\mathbf{X}_{\mathrm{SSCwc}})$, where % represents integer division. The first interesting observation from Fig. 10 (a) regards the increase of CT($\mathbf{X}$) when $m$ increases, going from about 8 s for $m = 2$ to more than 200 s 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 *probably* be used to analyze large volumes of traffic using many traffic descriptors in a reasonable time scale as compared




(a) R2L - Training dataset (b) R2L - Testing Dataset

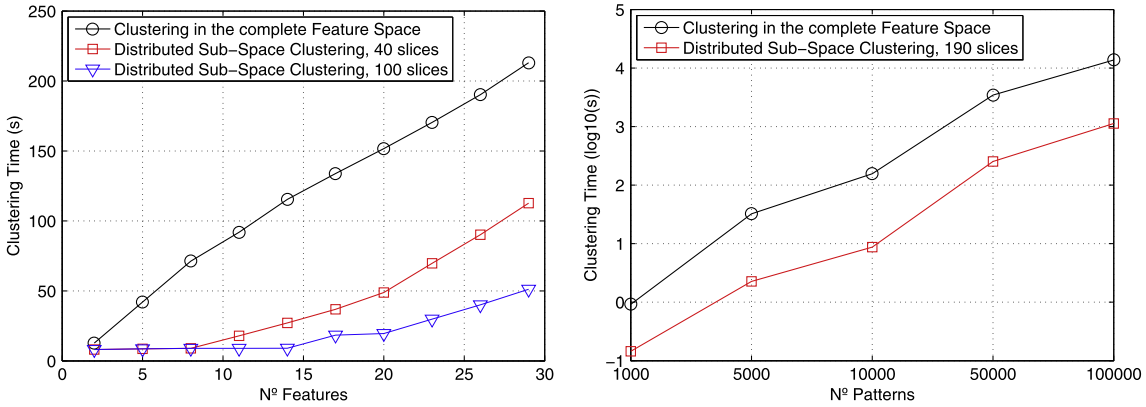
(c) DoS - Training dataset (d) DoS - Testing Dataset

(e) U2R - Training dataset (f) U2R - Testing Dataset

(g) Probing - Training dataset (h) Probing - Testing Dataset

**Fig. 9.** UNIDS vs Misuse-based NIDS – R2L, DoS, U2R, and probing attacks.

with the duration of time-slots. The use of the term *probably* emphasizes the fact that the simulation results presented in Fig. 10 do not take into account the additional overheads entailed in the deployment of the algorithms on a real parallel architec-




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

**Fig. 10.** Computational Time as a function of $n$ of features and $n$ of flows to analyze. The number of aggregated flows in (a) is $n$ = 10,000. The number of features and slices in (b) is $m = 20$ and $M = 190$ respectively.

ture, both at the hardware and at the system level (e.g., architecture of sampling components or pre-processing units and so forth).

Fig. 10 (b) compares CT($\mathbf{X}$) against CT($\mathbf{X}_{SSCwc}$) 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 EA4RO 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 EA4RO algorithm can analyze up to 50000 flows with a reasonable CT, about 4 min in this experience. In the evaluations with real traffic traces presented in Section 5, the number of aggregated flows in a time slot of $\Delta T = 20$ s rounds the 2500 flows, which represents a value of CT($\mathbf{X}_{SSCwc}$) $\approx 0.4$ s. For the $m = 9$ features that we have used ($N = 36$), and even without doing parallelization, the total CT is $N \times CT(\mathbf{X}_{SSCwc}) \approx 14.4$ s.

## 8. Conclusions

The Unsupervised Network Intrusion Detection System that we have presented has many interesting advantages w.r.t. previous proposals. It uses exclusively unlabeled data to detect network attacks, without assuming any kind of signature, particular model, or canonical data distribution. This allows to detect new previously unseen network attacks, outperforming current misuse-based NIDS.

Despite using ordinary clustering techniques to identify network attacks, our approach avoids the lack of robustness of general clustering methods, by combining the notions of Sub-Space Clustering, Density-based Clustering, and Multiple Evidence Accumulation to rank the degree of abnormality of traffic flows. We have verified the effectiveness of the proposed UNIDS system to detect both simulated and real network attacks of very different nature (DoS, DDoS, scans, worms, buffer overflows, etc.) in multiple traffic traces, all in a completely blind fashion, without assuming any particular traffic model, clustering parameters, or even clusters structure.

Additionally, we have shown detection results that outperform traditional approaches for intrusion detection and unsupervised outliers detection, providing a stronger evidence of the accuracy of this novel UNIDS to detect unknown network attacks.

Finally, and contrary to previous work on clustering and outliers analysis for detection of network attacks, we have conducted an evaluation on the computational time of the core algorithms. Results suggest that the application of UNIDS for unsupervised intrusions detection can be drastically improved regarding computational time of the analysis if the algorithms are deployed on a parallel computation architecture. Unfortunately, we do not have at the time of writing this paper an implementation of UNIDS on a real parallel computation architecture, which limits the practical applicability of drawn conclusions on computational time.

## 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, R. Hunt, A taxonomy of network and computer attacks, in: Computers and Security, vol. 24, no. 1, pp. 31–43, 2005.
[2] Symantec Enterprise Security, Symantec internet security threat report, Trends for 2010, Symantec reports, vol. 16, 2011.
[3] N. Wyler, Aggressive Network Self-Defense, Syngress-Elsevier, 2005. ISBN 978-1-931836-20-3.
[4] G. Cormode, S. Muthukrishnan, What is new: finding significant differences in network data streams, in: IEEE Transactions on Networking, vol. 13 no. 6, 2005, pp. 1219–1232.
[5] P. Casas, J. Mazel, P. Owezarski, UNADA: Unsupervised network anomaly detection using sub-space outliers ranking, in: IFIP Networking, 2011.
[6] G. Androulidakis, V. Chatzigiannakis, S. Papavassiliou, Network anomaly detection and classification via opportunistic sampling, IEEE Network 23 (1) (2009).
[7] R. Kemmerer, G. Vigna, Intrusion detection: a brief history and overview, IEEE Security & Privacy Magazine 35 (4) (2002).
[8] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, B. Stiller, An Overview of IP Flow-Based Intrusion Detection, IEEE Communications Surveys & Tutorials 12 (3) (2010).
[9] C. Kruegel, T. Toth, Using decision trees to improve signature-based intrusion detection, in: Proceedings RAID, 2003.
[10] J. Lee, H. Lee, S. Sohn, J. Ryu, T. Chung, Effective value of decision tree with KDD99 intrusion detection datasets for IDS, in: Proceedings ICACT, 2008.
[11] A. Lakhina, M. Crovella, C. Diot, Characterization of network-wide anomalies in traffic flows, in: Proceedings ACM IMC, 2004.
[12] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: Proceedings ACM SIGCOMM, 2004.
[13] A. Lakhina, M. Crovella, C. Diot, Mining anomalies using traffic feature distributions, in: Proceedings ACM SIGCOMM, 2005.
[14] D. Brauckhoff, K. Salamatian, M. May, Applying PCA for traffic anomaly detection: problems and solutions, in: Proceedings IEEE INFOCOM, 2009.
[15] H. Ringberg, A. Soule, J. Rexford, C. Diot, Sensitivity of PCA for traffic anomaly detection, in: Proceedings ACM SIGMETRICS, 2007.

[16] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, K. Cho, Extracting hidden anomalies using sketch and non gaussian multi-resolution statistical detection procedures, in: Proceedings ACM LSAD Workshop, 2007.
[17] A.K. Jain, Data clustering: 50 years beyond *K*-means, Pattern Recognition Letters 31 (8) (2010) 651–666.
[18] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Expl. Newsletter 6 (1) (2004) 90–105.
[19] L. Portnoy, E. Eskin, S. Stolfo, Intrusion detection with unlabeled data using clustering, in: Proceedings ACM DMSA Workshop, 2001.
[20] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data, Applications of Data Mining in Computer Security, Kluwer Publisher, 2002.
[21] K. Leung, C. Leckie, Unsupervised Anomaly detection in network intrusion detection using clustering, in: Proceedings ACSC, 2005.
[22] A. Fred, A.K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (6) (2005).
[23] M. Ester, P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings ACM SIGKDD, 1996.
[24] G. Fernandes, P. Owezarski, Automated classification of network traffic anomalies, in: Proceedings SecureComm, 2009.
[25] K. Cho, K. Mitsuya, A. Kato, Traffic data repository at the WIDE Project, in: USENIX Annual Technical Conference, 2000.
[26] The KDD Cup 1999 Dataset, at <kdd.ics.uci.edu/databases/kddcup99>.
[27] METROlogy for SECurity and QoS, at <laas.fr/METROSEC>.
[28] Bro intrusion detection system, at <www.bro-ids.org>.
[29] SNORT: an Open source network intrusion prevention and detection system, at <www.snort.org>.