Intrusion Detection in Computer Networks by a Modular Ensemble of One-Class Classifiers

Giorgio Giacinto, Roberto Perdisci, Mauro Del Rio, and Fabio Roli
Department of Electrical and Electronic Engineering - University of Cagliari, Italy
Piazza d'Armi - 09123 Cagliari, Italy
{giacinto,roberto.perdisci,mdrio,roli}@diee.unica.it

Abstract

Since the early days of research on Intrusion Detection, anomaly-based approaches have been proposed to detect intrusion attempts. Attacks are detected as anomalies when compared to a model of normal (legitimate) events. Anomaly-based approaches typically produce a relatively large number of false alarms compared to signature-based IDS. However, anomaly-based IDS are able to detect never-before-seen attacks. As new types of attacks are generated at an increasing pace and the process of signature generation is slow, it turns out that signature-based IDS can be easily evaded by new attacks. The ability of anomaly-based IDS to detect attacks never observed in the wild has stirred up a renewed interest in anomaly detection. In particular, recent work focused on *unsupervised* or *unlabeled* anomaly detection, due to the fact that it is very hard and expensive to obtain a labeled dataset containing only *pure* normal events.

The unlabeled approaches proposed so far for network IDS focused on modeling the normal network traffic considered as a whole. As network traffic related to different protocols or services exhibits different characteristics, this paper proposes an unlabeled Network Anomaly IDS based on a modular Multiple Classifier System (MCS). Each module is designed to model a particular group of similar protocols or network services. The use of a modular MCS allows the designer to choose a different model and decision threshold for different (groups of) network services. This also allows the designer to tune the false alarm rate and detection rate produced by each module to optimize the overall performance of the ensemble. Experimental results on the KDD-Cup 1999 dataset show that the proposed anomaly IDS achieves high attack detection rate and low false alarm rate at the same time.

1 Introduction

Intrusion Detection Systems (IDS) are tools placed inside computer networks that analyze the network traffic (referred as network-based IDS) or audit data recorded by network hosts (referred as host-based IDS). IDS look for known or potential threats and raise an alarm whenever an intrusion attempt is detected. Two main approaches to intrusion detection are currently used, namely *misuse* detection and *anomaly* detection [20, 23]. Misuse detectors are based on models of known attacks, i.e., a detailed description of attack patterns. Most misuse-based detectors model the attack patterns by a set of rules often called *attack signatures*. This approach allows the IDS to precisely detect intrusive events that perfectly match the signatures. As precise signatures can only be written to describe well known attacks, the effectiveness of signature-based IDS is strictly related to the extent to which the IDS are updated with the signatures that describe the latest attacks. Conversely, most anomaly-based detectors are based on statistical models of *normal* or

legitimate[1] events, i.e., a statistical profile of what constitutes normal network activities. The main assumption at the base of anomaly detection is the fact that manifestations of intrusive activities usually deviates from legitimate activities and then can be detected comparing them to the model of normal events.

Anomaly-based detection has been the first approach to be developed, in account of its theoretical ability to detect intrusions regardless of the specific system vulnerability and the type of attack [5]. Even though anomaly-based approaches are promising, they usually produce a relatively high number of false alarms, due to the difficulties in modeling the normal events. For this reason, signature-based detectors are usually chosen to be deployed in many organizations, instead, thanks to their ability to reliably detect known attacks while producing a relatively low number of false alarms. Nevertheless, attackers are constantly developing new attack tools designed to *evade* signature-based IDS. For example, techniques based on metamorphism and polymorphism are used to generate instances of the same attack that look syntactically different from each other, yet retaining the same semantic and therefore the same effect on the victim [29]. In principle this problem could be solved by designing *vulnerability-specific* signatures [30] that capture the "root-cause" of an attack, thus allowing the IDS to detect all the attacks that try to exploit the same vulnerability. This type of signatures usually works well when implemented as part of host-based IDS. However, it is difficult to implement vulnerability-specific signatures for network-based IDS due to computational complexity problems related to the high volume of events to be analyzed. Attempts to accomplish this task using approximate solutions usually cause the IDS to produce a high number of false positives, i.e., legitimate events that match an attack signature [23]. Pattern recognition techniques for misuse-based IDS have also been explored [4, 7, 9, 10]. The main motivation in using pattern recognition techniques to develop misuse-based detectors is their generalization ability, which may support the recognition of new "variants" of known attacks that cannot be reliably detected by signature-based IDS. In order to apply these techniques a dataset containing examples of attack patters as well as normal traffic is needed. However, it is very difficult and expensive to create such a dataset and previous works in this field funded by DARPA and developed by the MIT Lincoln Laboratory group [18] have been largely criticized [21, 19].

The reasons mentioned above motivate the renewed interest in network-based anomaly detection techniques, and, in particular, in *unsupervised* or *unlabeled* anomaly detection. Because it is very hard and expensive to obtain a labeled dataset, clustering and outlier detection techniques are applied on completely unlabeled traffic samples extracted from a real network. Alternatively, when a (small) set of labeled data samples is available in addition to unlabeled data, semi-supervised techniques may be used [2]. However, semi-supervised techniques are not guaranteed to provide better solutions than unsupervised techniques, because semi-supervised techniques require the labeled data to be representative and accurate [2]. Unfortunately, in the intrusion detection field it is hard to meet these requirements. For this reason, in this paper we propose an unsupervised method, where the only *a priori* knowledge about the data is represented by the following assumptions: i) the extracted dataset contains two classes of data, normal and anomalous traffic; ii) the numerosity of the anomalous traffic class is by far less than the numerosity of the normal traffic class. The first assumption comes from the consideration that pure legitimate traffic can only be generated by simulations in an isolated environment. However, this simulation process cannot reproduce traffic patterns of a real network [19]. In turn, an anomaly detector derived from such an artificial dataset of normal traffic may easily produce a large number of false alarms during the operational phase, because real network traffic is in general different from the one artificially generated. On the other hand, it is very hard, time-consuming and expensive to produce a dataset of pure legitimate traffic

[1]The terms *normal* and *legitimate* are used as synonyms throughout the paper.

by thoroughly "cleaning" real network traffic traces. As a result, traffic traces collected from real networks are not guaranteed to be attack free. The second assumption is due to the fact that in general the majority of the network traffic is legitimate [22]. Moreover, it is possible to use existing signature-based IDS to eliminate the know attacks from the collected traffic, thus further reducing the percentage of attack events in the dataset.

Recently proposed anomaly detectors use clustering and outlier detection techniques on unlabeled data [22, 8, 17]. These techniques try both to cope with the possible presence of outliers attack patterns in the training data and to detect anomalous network events during the operational phase. They are based on the common assumption that in a suitable feature space attack traffic is statistically different from normal traffic [5, 12]. Reported results show that the proposed approaches can be used to design effective anomaly detectors based on unlabeled traffic traces.

However, the unlabeled techniques proposed so far in the literature are based on "monolithic" approaches, i.e., one model of normal traffic is designed that covers all the protocols and services offered by the network. Besides, different models of normal traffic have been proposed and compared, but their combination has not been explored. This paper proposes an unlabeled approach for Network Anomaly IDS based on a modular Multiple Classifier System (MCS), whereby

i) A module is designed for each group of protocols and services, so that they fit the characteristics of the normal traffic related to that specific group of services [16, 31].

ii) Each module can be implemented by using an individual classifier as well as a combination of different classifiers.

The proposed modular architecture allows the designer to choose the rejection threshold of each module, so that the overall attack detection rate can be optimized given a desired total false alarm rate for the ensemble.

Experiments have been carried out on the KDD-Cup 1999 Dataset, available at the KDD-UCI repository. Even though this dataset has been criticized and cannot be deemed accurate to test the effectiveness of IDS in real scenarios, it still has the capability to allow researchers to compare different techniques on a common dataset [21]. Experimental results show that the anomaly IDS proposed in this paper achieves high attack detection rate and low false alarm rate at the same time. Besides, the proposed IDS is able to detect attacks that were included in the training set as "noise" mixed with normal traffic, as well as "new" attacks.

The paper is organized as follows. Related work is presented in Section 2. Section 3 describes the proposed MCS architecture. In particular, a heuristic for choosing the rejection threshold of each module is proposed. The implemented detection algorithms and the combining techniques are described in Section 4. Section 5 illustrates the experimental results on the KDD-Cup 1999 dataset and compares them to the results obtained by applying other approaches reported in the literature. Conclusions are drawn in Section 6.

2 Related Work

One of the first works on unsupervised network anomaly detection was presented in [22], where a variant of the single-linkage clustering algorithm is used to discover outliers in the training dataset. Once the normal patterns are separated from outlier patterns, the clusters of normal data are used to construct a supervised detection model. In [8], Eskin et al. presented a geometric framework to perform anomaly detection. The patterns are mapped from a feature space F to a new feature space F' and anomalies are detected by searching for patterns that lie in sparse regions of F'. Three different classification technique are used, a clustering algorithm, a k-NN algorithm and a

SVM-based one-class classifier. Experiments are performed on the KDD-UCI dataset, both on the portion containing network traffic, and on the portion containing sequences of system-calls.

In [32], Yang et al. proposed an anomaly detection measure called EWIND and an unsupervised pattern learning algorithm that uses EWIND to find outliers in the data, whereas Leung et al. [17] presented a new density-based and grid-based clustering algorithm to perform unsupervised network anomaly detection. In [33] a two-layer IDS is presented. The first layer uses a clustering algorithm to "compress" the information extracted for network packets, whereas the second layer implements an anomaly detection algorithm to classify the patterns received from the first layer.

In [31], Wang et al. proposed a payload-based anomaly IDS. A model is trained for each different service running on different server hosts in the protected network. For each packet, the frequency of the byte values in the payload (i.e., the data portion of the packet) is measured and a simple Gaussian model is trained. The detection of anomalous packets is performed by using a *simplified* Mahalanobis distance between the packets and the model of normal traffic.

Recently the paradigm of Multiple Classifier Systems (MCS) has been proposed for intrusion detection [9, 10, 3]. In [9, 10] classifiers trained on different feature subsets are combined to attain better performance than those attained by classifiers trained on the large feature space made of the union of subsets. A different approach is proposed in [3] where a serial combination of classifiers is proposed. Network traffic is serially processed by different classifiers. At each stage a classifier may either decide for one attack class or send the pattern to another stage, which is trained on difficult cases. Reported results show that MCS improve the performance of IDS based on statistical pattern recognition techniques.

The work presented in this paper was mainly inspired by [8] and [9]. Our work is new in that we propose a service-specific approach to address the *unlabeled anomaly detection* problem by applying multiple one-class classification techniques, which are often referred to also as outlier detection techniques. In particular, an heuristic to tune the false alarm rate produced by each anomaly detection module is proposed so that, given a fixed tolerable false alarm rate for the IDS, the overall detection rate is optimized. Besides, the combination of one-class classifiers is a new and not completely explored research area. A heuristic approach to combine multiple one-class classifiers' output was proposed by Tax et al. in [25]. Nevertheless, the proposed heuristic may present some problems, in particular when density-based and "distance-based" one-class classifiers are combined. We propose a new heuristic to map the output of two different "distance-based" one-class classifiers to a class-conditional probability, which aims at overcoming the mentioned problem.

3 Modular MCS Architecture

3.1 Problem Definition

The traffic over a TCP/IP network consists of packets related to communications between hosts. The exchange of packets between hosts usually fits in the *client-server* paradigm, whereby a client host requests some information offered by a service running on a server host. The set of packets related to the communication established between the client and (the service running on) the server forms a *connection*. Each connection can be viewed as a pattern to be classified and the network-based anomaly detection problem can be formulated as follows [10]:

Given the information about connections between pairs of hosts, assign each connection to the class of either normal or anomalous traffic.

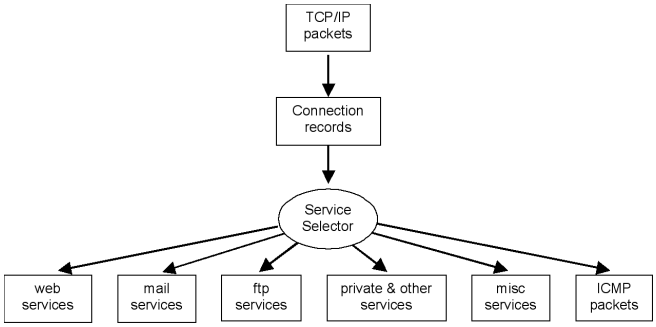

Figure 1: Modular architecture

3.2 Modular Architecture

As mentioned in Section 3.1, each connection is related to a particular service. Different services are characterized by different peculiarities, e.g., the traffic related to the HTTP service is different from the traffic related to the SMTP service. Besides, as different services involve different software applications, attacks launched against different services manifest different characteristics. We propose to divide the network services into m groups, each one containing a number of "similar" services [9]. Therefore, m modules are used, each one modeling the normal traffic related to one group of services. The intuitive advantage given by the modular approach is supported by the results in [16], where Lee et al. used information theory to measure the "complexity" of the classification task. The subdivision of services into groups turns into a decrease of the entropy of each subset of data, which in general coincides to the ability to construct a more precise model of the normal traffic. An example of how the services can be grouped is shown in Figure 1, where the groupings refer to the network from which the KDD-Cup 1999 dataset was derived (see Section 5). In Figure 1, a "miscellaneous" group is used to aggregate different services that are rarely used in the computer network at hand. It is worth noting that the number of groups and the type of services in each group depend on the network to be protected, as different networks may provide different services with different characteristics.

3.3 Overall vs. Service-Specific False Alarm Rate

Anomaly detection requires setting an acceptance threshold t, so that a traffic pattern $\mathbf{x}$ is labelled as anomalous if its similarity $s(\mathbf{x}, M)$ to the normal model M is less then t. The similarity measure s depends on the particular technique chosen to implement the model of normal traffic M. As we use different modules (i.e., different models) for different services, a method to tune the acceptance threshold for each module is necessary. In order to solve this task, we propose an heuristic approach whereby given a fixed tolerable false alarm rate for the IDS, the overall detection rate is optimized.

Let m be the number of service-specific modules of the IDS; FAR be the overall tolerable false alarm rate; FAR_i be the false alarm rate related to the i-th module; t_i be the acceptance threshold for the i-th module; $P(M_i) = n_i/n$ be the prior distribution of the patterns related to the i-th group of services (i.e., the module) M_i in the training data, where n_i is the number of patterns related to the services for which the module M_i is responsible and n is the total number of patterns in the training dataset. Accordingly, FAR is defined as

$$FAR = \sum_{i=1}^{m} P(M_i) \cdot FAR_i \tag{1}$$

Given a fixed value of the tolerable false alarm rate FAR for the IDS, there are many possible ways to "distribute" the overall FAR on the m modules. A value of FAR_i has to be chosen for each module M_i so that Equation (1) is satisfied. Once a FAR_i has been set for each module M_i, the thresholds t_i can be chosen accordingly. As a first choice, we could set $FAR_i = FAR$ for each module M_i. This choice satisfies Equation (1) and appears to be reasonable, given that no service is seemingly penalized. Nevertheless, this choice presents two drawbacks. One drawback is related to the actual number of false positives generated by each service. As the number of false positives is proportional to $P(M_i)$, the group of services (i.e., the module) accounting for the largest portion of the traffic produces a number of false alarms that is by far larger than the one produced by poorly represented services (i.e., those services which are rarely, or not so often used in the network). This behavior is not adequate as the modules of the IDS that produce an overwhelming number of false alarms could be "turned off" by the network administrator. The other drawback is related to the relation between FAR_i and the detection rate of the i-th service, DR_i. We observed experimentally that for a fixed value of FAR_i, the corresponding value of DR_i strongly dipends on $P(M_i)$. In particular, the larger $P(M_i)$ the larger DR_i. This effect can be explained as follows. Small values of $P(M_i)$ are related to services rarely used in the network, whereby a smaller training set for M_i can be extracted and the corresponding classifier(s) in general will not be able to adequately model the normal traffic.

According to the considerations reported above, given a fixed FAR we propose to compute FAR_i as

$$FAR_i = \frac{1}{m \cdot P(M_i)} FAR \tag{2}$$

This choice satisfies Equation (1) and allows us to attain an higher overall detection rate DR than that attained by choosing a fixed value $FAR_i = FAR$ for each module.

In order to set an acceptance threshold t_i for the module M_i, to obtain the false alarm rate FAR_i computed as in Equation (2), we propose the following heuristic. Let us first note that for a given value of t_i, the fraction $p_{r_i}(t_i)$ of patterns rejected by M_i may contain both patterns related to attacks and false alarms. Let us denote with $p_{a_i}(t_i)$ the fraction of rejected attack patterns using the threshold t_i, and with $far_i(t_i)$ the related fraction of false alarms. It is easy to see that the following relation holds:

$$p_{r_i}(t_i) = p_{a_i}(t_i) + far_i(t_i) \tag{3}$$

We want to set t_i so that $far_i(t_i)$ is equal to the desired false alarm rate FAR_i (computed by using (2)). As for a given value of t_i the only measurable quantity in Equation (3) is the rejection rate $p_{r_i}(t_i)$, we need some hypothesis on $p_{a_i}(t_i)$ so that we can estimate $far_i(t_i) = p_{r_i}(t_i) - p_{a_i}(t_i)$, and therefore we can chose t_i in order to obtain $far_i(t_i) = FAR_i$. We propose to assume $p_{a_i}(t_i) = P_{a_i}$, where P_{a_i} is the expected attack probability for the i-th service[2]. In other words, we assume that for a given threshold value, the rejected patterns are made up of all the attacks related to that service contained in the training set, plus a certain number of normal patterns. Thus, having fixed the value of $p_{a_i}(t_i) = P_{a_i}$, we can tune t_i in order to obtain $far_i(t_i) = FAR_i$.

It is easy to see that the computed thresholds t_i (estimated according to the heuristic described above) produce the required overall FAR (see Equation (1)) only if the fraction of patterns rejected by each module actually contains all the attacks $n_i \cdot P_{a_i}$, where n_i is the total number of training

[2]In practice, if the network is already protected by "standard" security devices (e.g., firewall, signature-based IDS, etc.), we may be able to estimate P_{a_i} from historical data related to attacks to the network service i that occurred in the past.

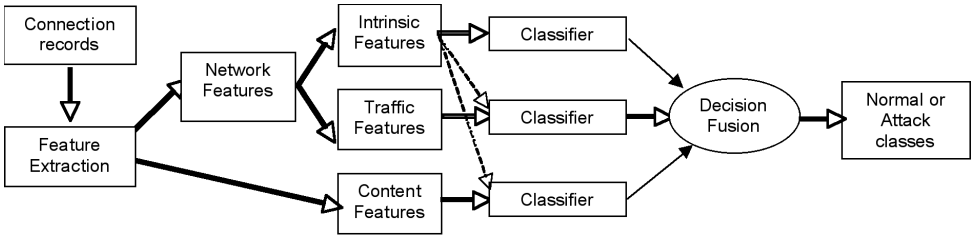

Figure 2: Feature subsets for service-specific MCS

patterns for the module M_i. If this is not the case and the rejection rate p_{r_i} includes just a portion of the attacks, a larger number of false alarms $far_i(t_i) > FAR_i$ will occur. However, if the training dataset is a good sample of the real network traffic, we expect most of the attacks will "look" different from normal traffic and will be likely deemed outliers and rejected by the model.

3.4 Service-Specific MCS

Lee et al. [15] proposed a framework for constructing the features used to describe the connections (the patterns). The derived set of features can be subdivided into two groups: i) features describing each single connection; ii) features related to statistical measures on "correlated" connections, namely different connections that have in common either the type of service they refer to or the destination host (i.e., the server host). The latter subset of features is usually referred as *traffic features*. On the other hand, the first group of features can be further subdivided into two subsets, namely *intrinsic features* and *content features*. The intrinsic features are extracted from the *headers* of the packets related to the connection, whereas the content features are extracted from the *payload* (i.e., the data portion of the packets). We call F the entire set of features and I, C and T the subsets of *intrinsic*, *content* and *traffic* features respectively, so that $F = I \cup C \cup T$.

As explained in Section 3.2, our IDS is subdivided into a number of modules. Each module implements a model of the normal traffic related to a group of services, so that a module can be viewed as a *service-specific* IDS. The problem of modeling the normal traffic for each module of the IDS can be formulated essentially in two different ways: i) a "monolithic" classifier can be trained using all the available features to describe a pattern; ii) subsets of features from the three groups described above can be used separately to train different classifiers whose outputs can be combined. Depending on the dimensionality of the feature space d and the size of the training set, one approach can outperform the other. In particular, a multiple classifier approach can be effective when the use of a "monolithic" classifier suffers from the "curse of dimensionality" problem, i.e. the training set n_i is too small with respect to d [6]. In this paper we propose to use, when needed, a MCS that consists of either two or three classifiers, depending on the module M_i we consider. When a two-classifiers MCS is used, the module is implemented by training two classifiers on two different features subsets, namely $I \cup C$ and $I \cup T$. On the other hand, when a three-classifiers MCS is used, the module is implemented by training a classifier on each single subset of features, namely one classifier is trained by using the subset I, one by using C and one by using T (see Figure 2).

4 Ensembles of Unsupervised Intrusion Detection Techniques

This section is divided into two parts. The first part introduces one-class classification concepts and outlines the techniques used to perform unlabeled intrusion detection, whereas the second part describes the techniques used to combine one-class classifiers.

4.1 One-Class Classification

One-class classification (also referred to as outlier detection) techniques are particularly useful in those two-class problems where one of the classes of objects is well-sampled, whereas the other one is severely undersampled due to the fact that it is too difficult or expensive to obtain a significant number of training patterns. The goal of one-class classification is to distinguish between a set of *target objects* and all the other possible objects, referred as *outliers* [25, 26]. A number of one-class classification techniques have been proposed in the literature. Following the categorization of one-class classifiers proposed by Tax [26], they can be subdivided into three groups, namely density methods, boundary methods and reconstruction methods.

We decided to use one classification method from each category to implement the service-specific MCS modules described in Section 3.4 in order to compare different approaches that showed good results in other applications. In particular, we chose the Parzen density estimation [6] from the density methods, the ν-SVC [24] from the boundary methods and the the k-means algorithm [11] from the reconstruction methods. These one-class classifiers exhibited good performance on a number of applications [26]. Besides, the output of the k-means and ν-SVC classifiers can be redefined as class-conditional probability density functions, so that they can be correctly combined with the output of the Parzen classifier (see Section 4.2). We also trained the clustering technique proposed by Eskin et al. [8] in order to compare the results of the combination of "standard" pattern recognition techniques with an algorithm tailored to the unlabeled intrusion detection problem.

4.1.1 Parzen Density Estimation

The Parzen-window approach [6] can be used to estimate the density of the target objects distribution

$$p(\mathbf{x}|\omega_t) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\,\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{4}$$

where n is the total number of training patterns belonging to the target class ω_t, $\mathbf{x}_i$ is the i-th training pattern, φ is a kernel function, h is the width of the Parzen-window and $p(\mathbf{x}|\omega_t)$ is the estimated class-conditional probability density distribution. When the Gaussian kernel

$$\varphi(x) = \frac{1}{(2\pi)^{\frac{d}{2}}}\exp\left(-\frac{1}{2}||\mathbf{x}||^2\right) \tag{5}$$

is used, $p(\mathbf{x}|\omega_t)$ can be written as

$$p(\mathbf{x}|\omega_t) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{(2\pi s)^{d/2}}\exp\left(-\frac{\|x-x_i\|^2}{2s}\right), \quad s = h^2. \tag{6}$$

and the one-class Parzen classifier can be obtained by simply setting a threshold θ whereby a pattern $\mathbf{z}$ is rejected (i.e., deemed an outlier) if $p(\mathbf{z}|\omega_t) < \theta$ [26].

4.1.2 k-means

The k-means classifier is based on the well-known k-means clustering algorithm [11]. The algorithm identifies k clusters in the data by iteratively assigning each pattern to the nearest cluster. This algorithm can be used as a one-class classifier by clustering the training set and then computing the distance $d(\mathbf{z}, \omega_t)$ of a test pattern $\mathbf{z}$ from the target distribution ω_t as

$$d(\mathbf{z}, \omega_t) = \min_{i=1..k} ||\mathbf{z} - \mu_i|| \tag{7}$$

where μ_i represents the i-th cluster center. If the distance is larger than a threshold θ the pattern will be rejected [26]. It is hard to map the distance $d(\mathbf{z}, \omega_t)$ into a probability density distribution and thus the combination of the k-means one-class classifier with density-based classifiers (e.g., the Parzen classifier) may produce unreliable results, as will be explained in Section 4.2. In order to allow this algorithm to produce an output that can be interpreted as a probability density function, we propose to use all the k distances between the test pattern $\mathbf{z}$ and the centroids μ_i as follows

$$\begin{aligned} p(\mathbf{x}|\omega_t) &= \frac{1}{k} \sum_{i=1}^{k} \frac{1}{(2\pi s)^{d/2}} \exp\left(-\frac{\|\mathbf{x}-\mu_i\|^2}{2s}\right) \\ s &= \operatorname*{avg}_{i,j} ||\mu_i - \mu_j||, \quad i,j = 1,2,..,k \end{aligned} \tag{8}$$

In other words, we model the distribution of the target class by a mixture of k normal densities, each one centred on a centroid μ_i. An heuristic is used to compute s as the average distance between the k centroids. As for the one-class Parzen classifier, the one-class k-means classifier based on Equation (8) can be obtained by setting a threshold θ whereby a pattern $\mathbf{z}$ is rejected if $p(\mathbf{z}|\omega_t) < \theta$. It is worth noting that the same number of distances $||\mathbf{z} - \mu_i||$ have to be computed both in (7) and (8). Besides, the number of centroids is in general chosen to be low, therefore s can be efficiently computed. This means that the proposed probability density estimate does not add appreciable complexity to the classifier.

4.1.3 ν-SVC

The ν-SVC classifier was proposed by Schölkopf et al. in [24] and is inspired by the Support Vector Machine classifier proposed by Vapnik [28]. The one-class classification problem is formulated to find an hyperplane that separates a desired fraction of the training patterns from the origin of the feature space $\mathbb{F}$. This hyperplane cannot always be found in the original feature space, thus a mapping function $\Phi : \mathbb{F} \rightarrow \mathbb{F}'$, from $\mathbb{F}$ to a kernel space $\mathbb{F}'$, is used. In particular, it can be proven that when the Gaussian kernel

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2s}\right) \tag{9}$$

is used, it is always possible to find a hyperplane that solves the separation problem. The problem is formulated as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi,\rho} \left(\frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{\nu l}\sum_i \xi_i\right) \\ &\mathbf{w} \cdot \phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1,..,l \end{aligned} \tag{10}$$

where $\mathbf{w}$ is a vector orthogonal to the hyperplane, ν represents the fraction of training patterns that are allowed to be rejected (i.e., that are not separated from the origin by the hyperplane), $\mathbf{x}_i$ is the i-th training pattern, l is the total number of training patterns, $\xi = [\xi_1, .., \xi_l]^T$ is a vector of slack variables used to "penalize" the rejected patterns, ρ represents the margin, i.e., the distance of the hyperplane from the origin.

The solution of (10) brings to the decision function, for a generic test pattern $\mathbf{z}$, formulated as

$$f_{svc}(\mathbf{z}) = I\left(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) \geq \rho\right), \quad \sum_{i=1}^{l} \alpha_i = 1 \tag{11}$$

where I is the indicator function[3] and the parameters α_i and ρ are provided by the solution of (10). According to (11), a pattern $\mathbf{z}$ is either rejected if $f_{svc}(\mathbf{z}) = 0$ or accepted as target object if $f_{svc}(\mathbf{z}) = 1$. When the Gaussian kernel (9) is used, the output of the ν-SVC can be formulated in terms of a class conditional probability by

$$p(\mathbf{x}|\omega_t) = \frac{1}{(2\pi \cdot s)^{\frac{d}{2}}} \sum_{i=1}^{n} \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^{n} \alpha_i \frac{1}{(2\pi \cdot s)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2}\frac{||\mathbf{x}-\mathbf{x}_i||^2}{s}} \tag{12}$$

which respects the constraint $\int_{\mathbb{R}^d} p(\mathbf{x}|\omega_t) d\mathbf{x} = 1$.

It is worth noting that in general only a small number of coefficients α_i will be different from zero, thus $p(\mathbf{x}|\omega_t)$ can be efficiently computed. The training patterns $\mathbf{x}_i$ whereby the related $\alpha_i \neq 0$ represent the support vectors for the ν-SVC. The acceptance threshold can be rewritten as

$$\rho' = \frac{\rho}{(2\pi \cdot s)^{\frac{d}{2}}} \tag{13}$$

so that a pattern $\mathbf{z}$ will be considered an outlier if $p(\mathbf{z}|\omega_t) < \rho'$.

It is worth noting that Tax et. al [27] independently formulated a SVM-based one-class classifier whose solution is identical to the one of the ν-SVC when the Gaussian kernel is used.

4.2 Combining One-Class Classifiers

Traditional pattern classifiers can be combined by using many different combination rules and methods [14]. Among the combination rules, the *min*, *max*, *mean* and *product* rules [13] are some of the most commonly used. These combination rules can be easily applied when the output of the classifiers can be viewed as an *a posteriori* probability $P_i(\omega_j|\mathbf{x})$, where p_i refers to the output of the i-classifier, whereas ω_j is the j-class of objects. In case of a two-class problem, the *a posteriori* probability can be written as

$$P_i(\omega_j|\mathbf{x}) = \frac{p_i(\mathbf{x}|\omega_j)P(\omega_j)}{p_i(\mathbf{x})} = \frac{p_i(\mathbf{x}|\omega_j)P(\omega_j)}{p_i(\mathbf{x}|\omega_1)P(\omega_1) + p_i(\mathbf{x}|\omega_2)P(\omega_2)}, \quad j = 1, 2, \quad i = 1, .., L \tag{14}$$

where L is the number of classifiers. Unfortunately, in case of one-class classifiers in general it is not possible to reliably estimate the probability distribution of one of the two classes, namely the probability density of the outlier objects (i.e., one of the terms in the denominator in (14)). Tax et al. [25] proposed to consider the distribution of the outlier to be constant in a suitable region of the feature set, so that the *a posteriori* probability for the target class, for example, can be approximated as

$$P_i(\omega_t|\mathbf{x}) = \frac{p_i(\mathbf{x}|\omega_t)P(\omega_t)}{p_i(\mathbf{x}|\omega_t)P(\omega_t) + \theta_i \cdot P(\omega_o)}, \quad i = 1, .., L \tag{15}$$

where ω_t represents the target class, ω_o represent the outlier class and θ_i is the uniform density distribution assumed for the outlier patterns. Let's consider now the traditional *mean* combination rule. We need to compute

[3] $I(x) = 1$ if x is true, otherwise $I(x) = 0$

$$\begin{aligned}\mu(\omega_t|\mathbf{x}) &= \tfrac{1}{L}\textstyle\sum_{i=1}^{L} P_i(\omega_t|\mathbf{x}) \\ \mu(\omega_o|\mathbf{x}) &= \tfrac{1}{L}\textstyle\sum_{i=1}^{L} P_i(\omega_o|\mathbf{x})\end{aligned} \tag{16}$$

and the decision criterion is

$$\mu(\omega_t|\mathbf{x}) < \mu(\omega_o|\mathbf{x}) \;\Rightarrow\; \mathbf{x} \text{ is an outlier} \tag{17}$$

If we assume $p_i(\mathbf{x}) \simeq p(\mathbf{x}), \forall i$, we can write

$$\mu(\omega_j|\mathbf{x}) = \tfrac{1}{L}\textstyle\sum_{i=1}^{L} \frac{p_i(\mathbf{x}|\omega_j)\cdot P(\omega_j)}{p(\mathbf{x})} = \frac{1}{L}\frac{P(\omega_j)}{p(\mathbf{x})}\sum_{i=1}^{L} p_i(\mathbf{x}|\omega_j) \tag{18}$$

where $j = t, o$ (i.e., (18) is applied to both the target and the outlier class). In this case we can compute

$$y_{avg}(\mathbf{x}) = \frac{1}{L}\sum_{i=1}^{L} p_i(\mathbf{x}|\omega_t) \tag{19}$$

$$\theta' = \frac{P(\omega_o)}{P(\omega_t)} \cdot \frac{1}{L}\sum_{i=1}^{L} \theta_i \tag{20}$$

and the decision criterion (17) becomes simply

$$y_{avg}(\mathbf{x}) < \theta' \;\Rightarrow\; \mathbf{x} \text{ is an outlier} \tag{21}$$

which means that we can combine the class-conditional probability density functions, instead of the *a posteriori* probabilities estimated by each classifier. The obtained $y_{avg}(\mathbf{x})$ can be used as a standard one-class classifier output and the threshold θ' can be independently tuned to attain the desired trade-off between false positives (i.e., target objects classified as outliers) and false negatives (i.e., outliers classified as belonging to the target class). This approach is (almost) exactly like the one proposed in [25] and [26] and can be extended to the *min*, *max* and *product* rules.

Another approach is to estimate $P_i(\omega_t|\mathbf{x})$ and $P_i(\omega_o|\mathbf{x})$ so that the decision criterion (17) can be used directly. For each one-class classifier i we have

$$\begin{aligned}P_i(\omega_t|\mathbf{x}) &= \tfrac{p_i(\mathbf{x}|\omega_t)P(\omega_t)}{p_i(\mathbf{x}|\omega_t)P(\omega_t)+\theta_i\cdot P(\omega_o)} \\ P_i(\omega_o|\mathbf{x}) &= \tfrac{\theta_i\cdot P(\omega_o)}{p_i(\mathbf{x}|\omega_t)P(\omega_t)+\theta_i\cdot P(\omega_o)}\end{aligned} \tag{22}$$

and, setting $\tau_i = \theta_i \cdot \frac{P(\omega_o)}{P(\omega_t)}$, the decision criterion for the classifier i can be written as

$$p_i(\mathbf{x}|\omega_t) < \tau_i \;\Rightarrow\; \mathbf{x} \text{ is an outlier} \tag{23}$$

It is worth noting that τ_i represents the decision threshold applied on the output of classifier i. According to (22) we can write

$$P_i(\omega_t|\mathbf{x}) = \frac{p_i(\mathbf{x}|\omega_t)}{p_i(\mathbf{x}|\omega_t) + \tau_i}, \quad i = 1, .., L \tag{24}$$

$$P_i(\omega_o|\mathbf{x}) = \frac{\tau_i}{p_i(\mathbf{x}|\omega_t) + \tau_i}, \quad i = 1, .., L \tag{25}$$

In practice, we can set the thresholds τ_i so that a given rejection rate is produced by each single one-class classifier. Once the thresholds $\tau_i,\ i = 1, .., L$, have been set, the posterior probabilities can be estimated using (24) and (25), and the rule (17) can be applied. This approach can be extended to the *min*, *max* and *product* rules by computing $\mu(\omega_t|\mathbf{x})$ and $\mu(\omega_o|\mathbf{x})$ according to the new rule and then applying (17).

As mentioned in Section 4.1, it is not possible to directly make use of the output of one-class classifiers that implement boundary or reconstruction methods in (19), (24) and (25). In order to solve this problem, Tax et al. [25] proposed an heuristic approach to map the output of "distance-based" classifiers to a probability estimate

$$\tilde{P}(\mathbf{x}|\omega_t) = exp\left(-\frac{\rho(\mathbf{x}|\omega_t)}{s}\right) \tag{26}$$

where $\rho(\mathbf{x}|\omega_t)$ is the output to be mapped (e.g., $\rho(\mathbf{x}|\omega_t) = \min_{i=1..k} ||\mathbf{x} - \mu_i||$, if the k-means classifier is considered). However, in general $\tilde{P}$ does not repsect the integral constraint for a density probability distribution, whereby

$$\int_{\mathbb{R}^d} \tilde{P}(\mathbf{x}|\omega_t)d\mathbf{x} \neq 1 \tag{27}$$

This fact may produce some problems, especially when the output of "distance-based" one-class classifiers is combined with density-based classifiers (e.g., the Parzen classifier described in Section 4.1.1), which respect the integral constraint by definition. On the other hand, the methods proposed in Section 4.1 to compute the output of the k-means and ν-SVC classifiers do not suffer from this problem and the decision criterion (21) can be used without further output transformations.

5 Experimental Results

Experiments were carried out on a subset of the DARPA 1998 dataset distributed as part of the UCI KDD Archive (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html). The DARPA 1998 dataset was created by the MIT Lincoln Laboratory group in the framework of the 1998 Intrusion Detection Evaluation Program (http://www.ll.mit.edu/IST/ideval). This dataset was obtained from the network traffic produced by simulating the computer network of an air-force base. It consists of seven weeks of traffic data for training purposes, and two weeks of data for testing. A subset of the traffic of the DARPA 1998 dataset has been mapped to a pattern recognition problem and distributed as the KDD-Cup 1999 dataset. The training set is made of 494,020 patterns, and the test set contains 311,029 patterns. Each pattern represents a network connection described by a 41-dimensional feature vector according to the set of features illustrated in Section 3. In particular, 9 features were of the intrinsic type, 13 features were of the content type, and the remaining 19 features were of the traffic type. Each pattern of the data set is labelled as belonging to one out of five classes, namely *normal traffic* and four different classes of attack: *Probe*, *Denial of Service* (DoS), *Remote to Local* (R2L), and *User to Root* (U2R). The attacks belonging to a certain attack class are designed to attain the same effect by exploiting different vulnerabilities of the computer network.

The DARPA dataset has been widely criticized [19, 21]. The main criticism is related to the fact that the traffic traces reported in the dataset are not representative of a real network scenario. In particular, it is worth noting that the prior probabilities of the attack classes included in the DARPA 1998 dataset (and thus in the KDD-Cup 1999) cannot be considered representative of the traffic in a real network. This fact has been clearly pointed out in a critique of the DARPA

	HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Normal	61,885 (96.55%)	4,172 (78.16%)	9,677 (99.83%)	1,288 (0.46%)	12,998 (81.27%)	7,257 (98.33%)
Attacks	2,208 (3.45%)	1,166 (21.84%)	16 (0.17%)	28,1250 (99.54%)	2,996 (18.73%)	123 (1.67%)

Table 1: Composition of the training set before the undersampling phase

	HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Normal	61,885 (99.83%)	4,172 (96.60%)	9,677 (99.83%)	1,288 (69.66%)	12,998 (96.09%)	7,257 (98.33%)
Attacks	106 (0.17%)	147 (3.40%)	16 (0.17%)	561 (30.34%)	529 (3.91%)	123 (1.67%)

Table 2: Composition of the training set after the undersampling phase

corpus of data by McHugh [21]. Although this dataset has been criticized, it is currently used by researchers because it is the only reference dataset that allows the designers to compare results obtained using different intrusion detection techniques.

In order to perform experiments with *unlabeled* intrusion detection techniques, we removed the labels of all the training patterns to simulate the *unlabeled* collection of network traffic. Besides, we relabeled the patterns of the test set as belonging either to the *normal traffic* class or to the "generic" *attack* class, thus discarding the different labels assigned to attack patterns related to different classes of attack. Nominal features have been converted into numerical values according to the procedure in [8]. Given a nominal feature with N possible distinct values, it is mapped to a vector of length N, i.e., the vector contains one coordinate for every possible value of the feature. When a particular value of the feature is mapped on the vector, the coordinate corresponding to the value of the feature is set to $1/N$, whereas the other coordinates (i.e., the ones corresponding to the other $N-1$ possible values for the feature) are set to zero.

According to the description of the modular architecture presented in Section 3, we divided the traffic of the data set into six subsets, each one related to "similar" services: *HTTP*, containing the traffic related to the HTTP protocol; *FTP*, containing the traffic related to the control flow and data flow for the FTP protocol, and the traffic related to the TFTP protocol; *Mail*, containing the traffic related to the SMTP, POP2, POP3, NNTP, and IMAP4 protocols; *ICMP*, containing the traffic related to the ICMP protocol; *Private&Other*, containing the traffic related to TCP/UDP ports higher than 49,152; *Miscellaneous*, containing all the remaining traffic. For each module, the features taking a constant value for all patterns have been discarded, provided that these features have a constant value by "definition" for that service, and not by chance. For example, the intrinsic feature "protocol_type" is always constant and equal to the value "TCP" for the *http*, and *mail* services, thus for those services it can be discarded. As a result, for each module we used a subset of the 41 available features, namely: 29 features for the *HTTP* module; 34 features for the *FTP* module; 16 features for the *ICMP* module (in particular, the content features were discarded as they have no meaning for the ICMP traffic); 31 features for the *Mail* module; 37 features for the *Miscellaneous* module; 29 features for the *Private&Other* module.

5.1 Training Set Undersampling

As mentioned above, the prior probabilities of the attack classes in the training portion of the KDD-Cup 1999 dataset cannot be considered representative of the traffic in a real network. The analysis of the training set confirmed that it contained a large fraction of attacks compared to normal traffic patterns, as show in Table 1. This violates the first assumption behind unlabeled

	HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Normal	39,247 (95.04%)	1,170 (38.22%)	3,222 (25.50%)	380 (0.23%)	12,930 (16.35%)	3,644 (43.53%)
Attacks	2,050 (4.96%)	1,891 (61.78%)	9,412 (74.50%)	164,591 (99.77%)	66,145 (83.65%)	4,727 (56.47%)

Table 3: Composition of the test set

		HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
ν-SVC	F	**0.995**	0.894	**0.971**	0.862	**0.992**	0.987
ν-SVC - max rule	$I \cup C + I \cup T$	0.956	0.918	0.960	-	0.911	0.975
	$I + C + T$	0.807	0.566	0.956	**0.929**	0.918	0.939
ν-SVC - min rule	$I \cup C + I \cup T$	0.948	0.967	0.855	-	0.921	0.953
	$I + C + T$	0.773	**0.973**	0.954	0.913	0.904	0.944
ν-SVC - mean rule	$I \cup C + I \cup T$	0.952	0.962	0.970	-	0.957	0.965
	$I + C + T$	0.865	0.972	0.953	0.879	0.921	**0.988**
ν-SVC - product rule	$I \cup C + I \cup T$	0.951	0.961	0.857	-	0.919	0.963
	$I + C + T$	0.865	0.971	0.953	0.879	0.921	0.945

Table 4: Performance attained by the ν-SVC classifier on the six modules in terms of AUC. For each module, the best performance is reported in bold.

techniques, i.e., connections containing attacks should account for a small portion of the network traffic. As typical network traffic satisfies this assumption [22], we filtered the training set so that the selected data satisfied this assumption. To this end, for each service we retained all the normal connections, while we sampled the attack patterns so that they accounted for 1.5% of the total traffic. This sampling procedure is similar to the one performed by other researchers [22, 8]. Let us recall that each *attack class* is made up of connections related to a number of different *attack types*, each *attack type* designed to produce the same effect of all the attacks in the same class. For each type of attack, a different number of patterns is available because each attack type produces a different number of connections, and because of the simulations carried out during the DARPA programme. A number of techniques can be used to sample a set of data such that the resulting subset is representative of the whole data [1].

In the experiments reported in this paper we reduced the percentage of attacks by reducing the number of those attacks accounting for a number of connections larger than 973, which is 1% of the total normal connections. In particular we proceeded as follows:

a) 10 subsets, each one containing 101 patterns, are extracted randomly from each *attack type* (this "magic" number was chosen in order to attain a total percentage of attacks equal to 1.5%);

b) for each subset, we trained a ν-SVC classifier, and computed the error attained by using the remaining patterns of that attack type as a test set;

c) the subset with the smallest error is selected, as it can be considered representative of the entire set of available connections for that attack type.

Table 2 shows the composition of the training set obtained after the preprocessing phase. It can be observed that attacks are not distributed uniformly among different services. While the overall percentage of attacks has been reduced so that it is equal to 1.5% of all the training traffic, the percentages of attacks related to different services range from the 0.17% of the *HTTP* and *Mail* traffic, to the 30.34% of the *ICMP* traffic. The high percentage of attacks in the *ICMP* traffic can be explained by observing that the available training set contained a very small number of normal

		HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
k-means	F	**0.978**	0.820	0.899	0.736	0.918	0.955
k-means - max rule	$I \cup C + I \cup T$	0.864	0.874	0.926	-	0.917	**0.974**
	$I + C + T$	0.872	0.335	**0.930**	**0.913**	0.917	0.889
k-means - min rule	$I \cup C + I \cup T$	0.353	0.830	0.826	-	0.903	0.909
	$I + C + T$	0.814	**0.926**	0.630	0.750	0.907	0.284
k-means - mean rule	$I \cup C + I \cup T$	0.859	0.778	0.913	-	**0.965**	0.932
	$I + C + T$	0.961	0.850	0.929	0.740	0.920	0.947
k-means - product rule	$I \cup C + I \cup T$	0.858	0.777	0.913	-	**0.965**	0.932
	$I + C + T$	0.965	0.851	0.929	0.740	0.920	0.951

Table 5: Performance attained by the k-means classifier on the six modules in terms of AUC. For each module, the best performance is reported in bold.

		HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Parzen	F	0.977	0.878	0.932	0.743	0.921	**0.982**
Parzen - max rule	$I \cup C + I \cup T$	0.854	0.904	0.568	-	0.905	0.900
	$I + C + T$	0.858	0.368	0.581	**0.872**	0.903	0.909
Parzen - min rule	$I \cup C + I \cup T$	**0.987**	0.868	0.940	-	0.921	0.974
	$I + C + T$	0.982	0.914	0.940	0.704	0.864	0.698
Parzen - mean rule	$I \cup C + I \cup T$	0.854	0.904	0.828	-	**0.991**	0.900
	$I + C + T$	0.858	0.867	0.582	**0.872**	0.910	0.909
Parzen - product rule	$I \cup C + I \cup T$	0.857	0.913	0.839	-	0.977	0.906
	$I + C + T$	0.959	**0.924**	**0.941**	0.725	0.888	0.898

Table 6: Performance attained by the Parzen classifier on the six modules in terms of AUC. For each module, the best performance is reported in bold.

ICMP connections compared to attacks, so that the proposed reduction of the number of attack patterns left the *ICMP* traffic data unbalanced.

It is worth noting that the distribution of traffic reported in Table 2 was used to compute the prior probabilities related to the different modules of the IDS, according to the discussion in Section 3.3.

Table 3 shows the composition of the test set. As shown in the table, the test set contains a very large fraction of attacks, as it was designed to test the performance of IDS and not to be representative of a realistic network traffic. It is worth noting that we did not apply any changes on the test set.

5.2 Performace Evaluation

We divided the performance evaluation experiments into two phases. In the first phase, we evaluated the performance of one module of the IDS at a time. In particular, for each module the performance of a "monolithic" classifier is compared to the performance attained by combining classifiers trained on distinct feature subsets (see Section 3.4). In the second phase, the modules related to different services are combined, and the performance of the overall IDS is evaluated. Performance evaluation has been carried out by ROC curve analysis, i.e., by computing the detection rate as a function of the false alarm rate. Different ROC can be compared by computing the Area Under the Curve (AUC). AUC measures the average performance of the related classifier, so that the larger the value of AUC of a classifier the higher the performance [26]. It is worth noting that AUC usually measures the average performance of classifiers considering the entire range of variation of the false positive rate. For some ranges of the false alarm rate the classifier with the smallest AUC value may provide the highest detection rate. Therefore, it may be better to measure the AUC in the interval $[0, a]$, where $a < 1$ represents the maximum expected false positive rate. However, it is not always possible to know in advance the working point (or the set of possible working points) on the

		HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Cluster	F	0.967	**0.839**	0.891	0.739	0.847	**0.973**
Cluster - max rule	$I \cup C + I \cup T$	0.965	0.705	0.949	-	0.843	0.253
	$I + C + T$	0.740	0.478	0.949	**0.918**	0.390	0.141
Cluster - min rule	$I \cup C + I \cup T$	0.922	0.782	0.802	-	0.903	0.875
	$I + C + T$	0.970	0.809	0.814	0.856	0.848	0.936
Cluster - mean rule	$I \cup C + I \cup T$	0.932	0.829	0.962	-	**0.915**	0.876
	$I + C + T$	**0.983**	0.874	**0.970**	0.872	0.847	0.958
Cluster - product rule	$I \cup C + I \cup T$	0.924	0.802	0.802	-	0.903	0.875
	$I + C + T$	0.980	0.809	0.814	0.872	0.947	0.943

Table 7: Performance attained by the Clustering algorithm proposed in [8] on the six modules in terms of AUC. For each module, the best performance is reported in bold.

	HTTP	FTP	Mail	ICMP	Private & Others	Miscellaneous
Best	ν-SVC F $I + C + T$	ν-SVC min rule $I + C + T$	ν-SVC F	ν-SVC max rule $I + T$	Parzen min rule $I \cup C + I \cup T$	ν-SVC mean rule $I + C + T$

Table 8: Summary of the best results in terms of AUC attained for each module.

ROC curve that will be actually used during the operational phase. Moreover, in our application the overall false positive rate is "distributed" in different percentages on different modules in order to optimize the performance of the IDS (see Section 3.2). In these cases of unknown a the AUC measured in the interval $[0, 1]$ is a valuable indicator of the performance of the classifier.

5.2.1 Evaluation of Service-Specific Modules

The first phase of the performance evaluation consisted of three experiments for each of the six modules. The first experiment was designed to assess the performance of individual one-class classifiers, i.e., the ν-SVC, the k-means, and the Parzen classifier when the patterns are described by using the entire set of available features F. The performance of the clustering algorithm described in [8] have been also computed for comparison purposes. The second experiment was designed to assess the performance attained by combining classifiers trained on two distinct feature subsets, i.e. the subset of intrinsic and traffic features $I \cup T$, and the subset made of intrinsic and content features $I \cup C$ (see Section 3.4). In particular, each classifier has been trained using the two feature subsets, and then they have been combined by using four different combination rules, i.e. the *max* rule, the *min* rule, the *mean* rule, and the *product* rule. The third experiment was designed to assess the performance attained by combining classifiers trained on three distinct feature subsets, i.e. the intrinsic features I, the traffic features T, and the content features C (see Section 3.4) by using again four different combination rules.

When combining classifiers trained on different feature spaces, we used both the combination approaches described in Section 4.2. We noted that for the ν-SVC, the k-means, and the clustering algorithm proposed in [8], the best performance was obtained by estimating the posterior probabilities for the target class as in (24) and then comparing these probabilities to a varying threshold in order to compute the ROC curves. For the Parzen classifier, the combination of class conditional probabilities, using (19) and the decision criteria (21), produced the best results.

In the following, we discuss the results obtained by applying the best combination approach for each single classifier. Therefore, we present the results obtained by combining the posterior probabilities for the ν-SVC, the k-means, and the clustering algorithm, and the results obtained by combining the class conditional probabilities for the Parzen classifier.

Tables 4, 5, 6, and 7 summarize the performance results on the test set in terms of AUC, for the

Best Modules in terms of AUC		ν-SVC		Best ν-SVC modules in terms of AUC	
False Alarm Rate	**Detection Rate**	**False Alarm Rate**	**Detection rate**	**False Alarm Rate**	**Detection Rate**
0.87%	75.34%	0.91%	67.31%	0.88%	79.27%
2.10%	80.35%	2.06%	75.61%	2.07%	89.45%
2.64%	80.80%	2.65%	77.10%	2.66%	89.67%
4.00%	85.67%	3.20%	86.31%	3.28%	89.92%
5.49%	94.12%	4.51%	92.25%	4.82%	93.02%
6.86%	94.27%	6.72%	93.91%	6.49%	94.16%
8.25%	94.32%	8.09%	94.12%	8.05%	94.26%
10.44%	94.38%	9.62%	94.25%	9.49%	94.31%

Table 9: Results attained by the proposed three modular systems.

ν-SVC, the k-means, the Parzen classifier, and the clustering algorithm proposed in [8], respectively. For each algorithm, the parameters have been tuned on the training set. It is worth noting that in the case of the *ICMP* protocol only intrinsic and traffic features were available, thus only the third kind of experiment could be performed by combining two one-class classifiers trained on intrinsic and traffic features, respectively.

The obtained results are discussed in Section 5.3.

5.2.2 Evaluation of Overall IDS

In order to analyze the performance of the overall IDS, we built three systems:

1) An "optimal" system made up, for each module, of the classification techniques that provided the highest value of AUC, according to Table 8.

2) A system made up of one "monolithic" ν-SVC for each module. We chose to use ν-SVC classifiers because on average they provide better results than the other considered classifiers, as discussed in Section 5.3.

3) As in the second system, we chose to use ν-SVC classifiers. Then, for each module we chose between a "monolithic" versus a MCS approach, according to best performance results reported in Table 4. It is worth noting that for the *Miscellaneous* module the performance of the "monolithic" classifier is really close to the best performance result. Therefore, it is difficult to conclued which approach really performs better than the other. We chose to construct a system made up of one "monolithic" ν-SVC for the *HTTP*, *Mail*, *Miscellaneous* and *Private&Other* modules, and a MCS for the *FTP* and *ICMP* modules (we will further discuss the motivation for this choice in Section 5.3). For the *FTP* module we used an MCS constructed by using three ν-SVC classifiers, namely one trained on the subset of features I, one on the subset C and one on the subset T. For the *ICMP* module we constructed a MCS using two ν-SVC classifiers, namely one trained on the subset of features I and one on the subset T. In particular, for the *FTP* module, the *min* rule was used, whereas the *max* rule was used for the *ICMP* module.

In order to evaluate the performance of the three IDS systems, we computed some working points according to the heuristic proposed in Section 3.3. The attained results are reported in Table 9. The motivation for the choice of the three proposed IDS systems and the attained results are further discussed in Section 5.3.

5.3 Discussion

The results reported in Section 5.2.1 clearly show that the ν-SVC algorithm provides the highest AUC value for all services, when classifiers are trained using all the available features. The difference between the performance of ν-SVC and that of the other algorithms is very small in the case of the *HTTP* and the *Miscellaneous* traffic, while it is larger for the other services.

Tables 4, 5, 6, and 7 show that combining classifiers trained on distinct feature sets does not always improve performance, with respect to those attained by classifiers trained on the entire feature set. In particular, it can be seen that for the ν-SVC, k-means, and Parzen classifiers, the use of distinct feature sets clearly outperforms the use of the entire feature set F only for the *FTP*, and *ICMP* modules. In the case of the clustering algorithm, the use of distinct feature sets clearly outperforms the use of the entire feature set F only for the *Mail*, *ICMP*, and *Private&Others* modules. In all other cases the differences in performance are small, thus the superiority of one technique against the others cannot be concluded. Unfortunately, results show no regularity. For this reason, it is difficult to explain the behavior of different classifiers and combination rules on different modules. On the other hand, results clearly show that each module should be carefully and independently designed by making a decision about the classification algorithm to be used, and by choosing between an individual classification technique and the MCS approach.

Summing up, reported results allow us to conclude that the ν-SVC algorithm performs better than the other ones, on average. Further, it is easy to see that the combination of distinct feature representations usually provides significantly higher performance, with respect to just one classifier trained on the entire feature set, only for the *FTP* and *ICMP* modules. These observations have been used in Section 5.2.2, where three different overall IDS made up of six modules are described.

In order to compare the performance of the modular systems proposed in Section 5.2.2 to the approach used by Eskin et al. [8], we trained the clustering algorithm proposed in [8] and the ν-SVC on the entire training set obtained after subsampling. It is worth noting that this approach is the same used in [8]. Besides, our test set is the same as the one used in [8], and we also used an approach similar to the one proposed in [8] to adjust the training dataset. The performance results obtained on the test set are reported in Tables 10 and 11, respectively. It is easy to see that if the false alarm rate is set to 1%, the algorithms trained on the entire training set provide a detection rate near 18%, while the proposed modular approaches provide detection rates from 67% to 79% (see Table 9). As the effectiveness of IDS depends on the capability of providing high detection rates at small false alarms rates, the proposed modular approaches are very effective compared to the "monolithic" approaches. At 4% false alarm rate, the "monolithic" clustering algorithm provides better results than the modular approaches, in terms of detection rates. However, for higher false positive rates, the clustering algorithm does not provide performance improvements, whereas the proposed modular IDS reaches definitely better detection rates with respect to the ones obtained at low false positive rates. It is worth noting that, from a practical point of view, the working point of anomaly detectors are usually tuned to produce a low false alarm rate (e.g., equal to 1% or lower). Reported results clearly show that the proposed modular approach outperforms the "monolithic" approaches in the range of low false positive rates, due to its capability of allowing different false positive rates on different modules.

6 Conclusions

In this paper we have proposed a modular solution to the Unsupervised Anomaly Detection problem. In particular, one-class classifiers have been considered for designing each module. In addition, the combination of one-class classifiers trained on distinct features have been also considered. As the

Clustering	
False Alarm Rate	**Detection Rate**
1%	18.37%
2%	26.80%
3%	27.21%
4%	92.21%
5%	92.24%
6%	92.25%
7%	92.25%
8%	92.29%
9%	92.29%
10%	92.68%

Table 10: Results attained by applying the "monolithic" approach using the clustering algorithm proposed in [8].

ν-SVC	
False Alarm Rate	**Detection Rate**
1%	17.91%
2%	66.44%
3%	78.40%
4%	78.85%
5%	86.07%
6%	92.53%
7%	92.57%
8%	92.60%
9%	92.63%
10%	92.91%

Table 11: Results attained by applying the "monolithic" approach using the ν-SVC classifier.

combination of one-class classifier is a new and not completely explored research area, we proposed a technique aimed at mapping the outputs of one-class classifiers to density functions. The proposed framework allowed us to effectively combine the outputs of different classifiers using well-known fixed combination techniques developed for multi-class classifiers. We also proposed a heuristic designed to compute the false alarm rate for each module, given the maximum allowed overall false alarm rate. Attained results clearly show that subdividing the problem into different modules allows us to accurately model the traffic of each service, as for small values of false alarm rates the proposed systems attained high detection rates.

References

[1] Batista G., Prati R.C., Monard, M.C. *A study of the behavior of several methods for balancing machine learning training data.* ACM SIGKDD Explorations Newsletter, 6(1), 2004, pp. 20-29

[2] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, T. Huang. *Semi-supervised learning of classifiers: theory, algorithms and their applications to human-computer interaction.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(12), 2004, pp. 1553-1567.

[3] L. P. Cordella, A. Limongiello, C. Sansone. *Network Intrusion Detection by a Multi-stage Classification System.* In: Roli, Kittler, and Windeatt (Eds.): Multiple Classifier Systems, LNCS 3077, Springer, 2004, pp. 324-333

[4] H. Debar, M. Becker, D. Siboni. *A Neural Network Component for an Intrusion Detection System.* Proc. of the IEEE Symp. on Research in Security and Privacy, Oakland, CA, USA, 1992, pp. 240-250.

[5] D. E. Denning. *An Intrusion-Detection Model.* IEEE Transactions on Software Engineering, 13 (2), 1987, pages 222-232.

[6] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[7] C. Elkan. *Results of the KDD'99 Classifier Learning.* ACM SIGKDD Explorations 1, 2000, pp. 63-64.

[8] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo. *A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data*. In: D. Barbara and S. Jajodia (editors), Applications of Data Mining in Computer Security, Kluwer, 2002.

[9] G. Giacinto, F. Roli, L. Didaci. *A Modular Multiple Classifier System for the Detection of Intrusions in Computer Networks*. 4th Int. Workshop on Multiple Classifier Systems (MCS 2003), Guildford, United Kingdom, June 11-13 2003, T. Windeatt and F. Roli Eds., LNCS 2709, pp. 346-355.

[10] G. Giacinto, F. Roli, L. Didaci. *Fusion of multiple classifiers for intrusion detection in computer networks*. Pattern Recognition Letters, 24(12), 2003, pp. 1795-1803.

[11] A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[12] H. Javits, A. Valdes. *The NIDES statistical component: Description and justification*. SRI Anual Report A010, SRI International, Computer Science Laboratory, March 1993.

[13] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas. *On Combining Classifiers*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), 1998, pp. 226-229.

[14] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

[15] W. Lee, S. Stolfo. *A Framework for Constructing Features and Models for Intrusion Detection Systems*. ACM Transactions on Information and System Security, Volume 3, Number 4 (November 2000).

[16] W. Lee, D. Xiang. *Information-theoretic measures for anomaly detection*. IEEE Symposium on Security and Privacy, 2001.

[17] K. Leung, C. Leckie. *Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters*. In Proceedings of the 28th Australasian Computer Science Conference, ACSC2005.

[18] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, K. Das. *The 1999 DARPA Off-Line Intrusion Detection Evaluation*. Computer Networks, 34 (4), 2000, pages 579-595.

[19] M. V. Mahoney, P. K. Chan. *An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection*. 6th International Symposium on Recent Advances in Intrusion Detection, RAID 2003.

[20] J. McHugh, A. Christie, J. Allen. *Defending yourself: The role of intrusion detection systems*. IEEE Software (Sept./Oct. 2000), pages 42-51.

[21] J. McHugh. *Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory*. ACM Transactions on Information and System Security, Vol. 3, No. 4, November 2000, 262-294.

[22] L. Portnoy, E. Eskin, S. Stolfo. *Intrusion detection with unlabeled data using clustering*. Proceedings of ACM CSS Workshop on Data Mining Applied to Security, DMSA-2001.

[23] P.E. Proctor. *Practical intrusion detection handbook*. 2001, Upper Saddle River, NJ: Prentice Hall.

[24] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, RC Williamson. *Estimating the support of a high-dimensional distribution*. Neural Computation, Vol. 13, 2001, pages 1443-1471.

[25] D. M. J. Tax, R. P. W. Duin. *Combining One-Class Classifiers.* Proceedings of Multiple Classifier Systems, MCS 2001.

[26] D. M. J. Tax. *One-Class Classification, Concept Learning in the Absence of Counter Examples.* Ph.D. Thesis, 2001, Delft University of Technology, Delft, Netherland.

[27] D. M. J. Tax, R. P. W. Duin. *Support Vector Data Description.* Mchine Learning, Volume 54, Number 1, 2004, pages 45 - 66.

[28] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[29] G. Vigna, W. Robertson, D. Balzarotti. *Testing Network-based Intrusion Detection Signatures Using Mutant Exploits.* In the Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS). October 2004, Washington DC, USA.

[30] H. J. Wang, C. Guo, D. R. Simon, A. Zugenmaier. *Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits.* Proceedings of the ACM SIGCOMM Conference, 2004.

[31] K. Wang, S. J. Stolfo. *Anomalous Payload-based Network Intrusion Detection.* In Proceedings of the International Symposium on Recent Advances on Intrusion Detetion, RAID 2004.

[32] Y. Yang, F. Ma. *An Unsupervised Anomaly Detection Patterns Learning Algorithm.* Proceedings of ICCT 2003.

[33] S. Zanero, S. M. Savaresi. *Unsupervised learning techniques for an intrusion detection system.* Proceedings of the ACM symposium on Applied computing, 2004.