Towards A Multiagent-Based Distributed Intrusion Detection System using Data Mining Approaches

Imen Brahmi[1], Sadok Ben Yahia[1], Hamed Aouadi[2], and Pascal Poncelet[3]

1 Faculty of Sciences of Tunis, Tunisia
sadok.benyahia@fst.rnu.tn
2 ISLAIB, Beja, Tunisia
Hamed_aouadi@yahoo.fr
3 LIRMM UMR CNRS 5506,161 Rue Ada, 34392 Montpellier Cedex 5, France
poncelet@lirmm.fr

Abstract. *The system that monitors the events occurring in a computer system or a network and analyzes the events for sign of intrusions is known as Intrusion Detection System (IDS). The IDS need to be accurate, adaptive, and extensible. Although many established techniques and commercial products exist, their effectiveness leaves room for improvement. A great deal of research has been carried out on intrusion detection in a distributed environment to palliate the drawbacks of centralized approaches. However, distributed IDS suffer from a number of drawbacks e.g., high rates of false positives, low efficiency, etc. In this paper, we propose a distributed IDS that integrates the desirable features provided by the multi-agent methodology with the high accuracy of data mining techniques. The proposed system relies on a set of intelligent agents that collect and analyze the network connections, and data mining techniques are shown to be useful to detect the intrusions. Carried out experiments showed superior performance of our distributed IDS compared to the centralized one.*

Keywords: Intrusion Detection System; Multi-agents; Misuse Detection; Anomaly Detection; Data Mining Techniques

1 Introduction

Since the cost of information processing and Internet accessibility is dropping, more and more organizations are becoming vulnerable to a wide variety of cyber threats. Therefore, it has become increasingly important to make our information systems, especially those used for critical functions such as military and commercial purpose, resistant to and tolerant of such attacks. *Intrusion Detection Systems* (IDS) are an integral part of any security package of a modern networked information system. An IDS detects intrusions by monitoring a network or system and analyzing an audit stream collected from the network or system to look for clues of malicious behavior.

Basically, two main IDS can be distinguished: *anomaly detection* and *misuse detection* [9]. Indeed, misuse detection systems [9] use patterns of well known attacks or weak spots of the system to match and identify known intrusions. However, misuse detection techniques, in general, are not effective against novel attacks that have no already matched rules or patterns [9]. On the contrary, the anomaly detection systems flag activities, that significantly deviate from the established normal usage profiles as abnormal or in other words as intrusions. Anomaly detection techniques have been shown to be effective against unknown or novel attacks since no prior knowledge about specific intrusions is required. Nevertheless, the main moan that can be addressed to the anomaly detection systems is that they tend to generate more false alarms than do misuse detection systems, *i.e.*, an anomaly can be simply a new normal behavior [32].

The conventional approaches to intrusion detection involving a central unit[1] to monitor an entire system have several drawbacks [14]. To palliate them, the dedicated research witnessed a wealthy number of works heading towards a distributed[2] framework of monitors that carry out local detection and provide information to perform global detection of intrusions [14].

Nevertheless, current distributed IDS also suffer from some drawbacks [19]. In fact, most commercial IDS are built in a hierarchical architecture: a tree structure with a control system at the top; information aggregation units at the internal nodes; and sensor units at the leaf nodes. Within these systems, the local intrusion detection components look for local intrusions and transmit their analysis results to the upper levels of the hierarchy. The components at the upper levels scrutinize the refined data from multiple lower level components and seek to establish a global view of the system state. Such IDS are not truly distributed systems, because of the centralized data analysis performed at the higher levels of the hierarchy [14]. Moreover, these systems suffer from the problem of single point of failure. In addition, most IDS detect attacks by analyzing information from a single host, or a single network interface, at many locations throughout the network. Thus, the designed feature of communication and cooperation between an IDS components are badly missing. This fact hampers the capability to efficiently detect large-scale distributed attacks [14].

In the past twenty years, two of most prominent, dynamic and exciting research areas: multi-agent systems [42] and data mining [12] have emerged and developed separately. On the one hand, IDS needed to be able to resist attacks on themselves, fault tolerant, highly adaptable and configurable [29]. Given these characteristics, the multi-agent technology seemed to be an appropriate alternative for developing IDS [18]. Indeed, a distributed IDS based on multi-agent technology can effectively improve the detection accuracy, the detection speed, and enhance the system's own security [19]. Several IDS based on the multi-agent

[1] An IDS is considered as centralized whenever the analysis of data is performed at a fixed number of locations, independent of how many hosts are being monitored [39].

[2] An IDS is considered as distributed whenever the analysis of the data is performed in a number of locations proportional to the number of hosts that are being monitored [39].

methodology were developed [4, 22, 19, 29, 39]. On the other hand, along with the increasing complexity of networks, protecting a system against new and complex attacks, while keeping an automatic and adaptive framework, is a thriving issue within the intrusion detection domain. One answer to the problem could rely on data mining techniques [6, 28]. Following this trend, a wide variety of data mining techniques have been successfully applied into the intrusion detection domain [3, 8, 20, 24, 37].

In this paper, we investigate another way of tackling the aforementioned problems. Thus, we introduce a new distributed IDS, called MAD-IDS (*Multi-Agent using Data mining based Intrusion Detection System*). MAD-IDS is based on the integration of the multi-agents technology and the data mining techniques. In this respect, our proposed system uses a set of agents that can be applied to a number of tasks, namely: data capturing, detecting the known and unknown attack categories, extracting the set of signatures-based rules and ultimately alerting the administrator. Moreover, MAD-IDS efficiently merge both anomaly and misuse detection strategies by exploiting the advantages of the one to palliate the limitations of the other and vice versa. The performance of the IDS can be improved by combing anomaly and misuse analysis [10, 17, 33]. The main thrust of our approach stands in the use of an unsupervised anomaly detection technique based on the clustering algorithm. Furthermore, given the very high volume of connections observed per unit time, the association rule mining technique is shown to be useful in enabling a security analyst to understand and characterize emerging threats. The obtained model is more effective since the integration of multi-agent technology and data mining techniques makes an IDS more autonomous and efficient. Through extensive carried out experiments on intrusion detection benchmark datasets and real life network traffic, we show the effectiveness of our proposal in terms of (*i*) the detection delay; (*ii*) the scalability-related criteria such as network bandwidth and system response time; and (*iii*) the detection ability.

The remaining of the paper is organized as follows. Section 2 sheds light on the related work that focuses on the integration of the multi-agent systems with the data mining techniques. We introduce our new distributed intrusion detection system based on the multi-agent technology in Section 3. We also relate the encouraging results of the carried out experiments in Section 4. Finally, Section 5 concludes and points out avenues of future work.

2 Scrutiny of the related work

In recent years and within the intrusion detection domain, an increasingly evident trend has emerged. The trend stands within the crossroads of multi-agent systems and data mining. Its development has reached to the level as a new and promising research area. In this respect, various distributed intrusion detection architectures using the multi-agent design methodology and the data mining techniques have been developed.

One of the well known examples of applying a combination of multi-agents methodology and the data mining techniques is JAM (*Java Agents for Meta-learning*) [40]. The original idea behind JAM was the use of data mining techniques to correlate knowledge to build better models for fraud and intrusion detection. Thus, JAM uses association rules and frequent episode mining algorithms to determine the relationships between the different fields in audit trails. Moreover, a meta-learning classifier learns the signatures of attacks. For instance, JAM seemed to be shared between multiple organizations. However, it suffers from serious shortcomings that include the increase demand for runtime system resources and the inability to combine multiple models computed over distributed datasets with different schemas.

With the aim to improve the JAM's approach, Helmer et *al.* introduced in [16] a distributed IDS that integrates the data mining techniques, as well as mobile and static agents. Indeed, static agents collect the information and transform it in a common format. Besides, the mobile agents travel between monitored systems in a network, to obtain the information from the static agents, classify, correlate and report it to a user interface and a database via mediators. Finally, data mining agents use machine learning approaches towards an automated discovery of concise rules from system logs and audit data, to facilitate building, monitoring, and analyzing intrusions on distributed systems. In contrary to JAM, the proposed IDS focuses on data mining within an organization. It uses a different data representation that provides a single signature for each process.

Motivated by the issue that most of the existing commercial network-based IDS products are signature-based but not adaptive, Lui et *al.* [25] proposed an adaptive system using various data mining techniques and agent architecture. The data mining approaches include: the clustering, association and sequential rule extraction algorithms. These algorithms are used to accurately capture the actual behavior of the network traffic, and the mined portfolio is useful for differentiating "normal" and "attack" traffics.

Zhang et *al.* [44] combined the advantages of agent-based distributed analysis and clustering-based intrusion detection technique. The proposed approach focuses on the use of the clustering algorithm: (*i*) to select the candidate anomalies at the level of IDS agents; and (*ii*) to choose the true attack at the central IDS. However, the results of the clustering algorithm depend on the choice of the values of the cluster width. Whenever the latter is an unappropriate value, then the algorithm might label some intrusions as "normal", and some normal ones as "intrusion".

Moreover, Rehák et *al.* [35] introduced a prototype of agent-based IDS designed for deployment on high-speed backbone networks. The main contribution of the system, called CAMNEP, is the integration of several anomaly detection techniques by means of collective trust modeling within a group of collaborative detection agents, each featuring a specific detection algorithm. These algorithms maintain a model of expected traffic on the network and compare it with real traffic to identify the discrepancies that are identified as possible attacks. How-

ever, CAMNEP is not able to detect attacks that consist of few packets, *e.g.*, *Buffer Overflow* attack.

Palomo et *al.* [30] proposed an approach that integrates the Self-Organization Map (SOM) clustering approach within a multi-agents system. Indeed, the approach describes a multi-agents system with capabilities to analyze and discover knowledge gathered from distributed agents. These enhanced capabilities are obtained through a dynamic SOM and a multi-agents based communication system. The central administrator agent dynamically obtains information about the intrusions from the distributed agents and maintains a knowledge pool using a proposed growing SOM.

Recently, Shyu and Sainani [38] proposed a data mining assisted multi-agent-based IDS, called DMAS-IDS. DMAS-IDS employs three layers, called *Host*, *Classification* and *Manager layers*. Each one of these layers comprises agents capable of communicating the obtained results with each other. The agents are facilitated with a supervised classification algorithm called the *Collateral Representative Subspace Projection Modeling* (C-RSPM). However, C-RSPM is an ineffective classification algorithm with datasets possessing a very high number of classes, especially when the classes' distributions are very similar. In addition, the results of C-RSPM rely on a labeled training data instances which is not suitable for an on-line IDS.

In this paper, we propose a new distributed IDS, called MAD-IDS, which integrates the data mining techniques and the multi-agent technology. Unlike most of the surveyed works, where only one strategy is used for the detection of various attacks; MAD-IDS exploits the advantages of both misuse and anomaly detection strategies by merging them to remedy the corresponding disadvantages. Particularly, MAD-IDS focuses on the use of: (*i*) an unsupervised anomaly-based clustering technique to detect the unknown intrusions; and (*ii*) an association rule mining algorithm to summarize the network connections that are classified as anomalous.

3 The MAD-IDS system

The distributed structure of MAD-IDS is composed of different cooperative, communicant and collaborative agents for collecting and analyzing massive amounts of network traffic, called respectively: *Sniffer, Filter, Misuse Detection, Anomaly Detection, Rule Mining* and *Reporter Agent*. Figure 1 sketches the overall architecture of MAD-IDS.

The processing steps of MAD-IDS can be summarized as follows:

1. The Sniffer Agent is the first agent that connects to the network and gathers the packets;
2. The gathered packets are send to the Filter Agent which filters them;
3. The Misuse Detection Agent analyzes the collected and filtered packets, to detect the network connections that correspond to attacks for which signatures are available, and then to remove them from further analysis;

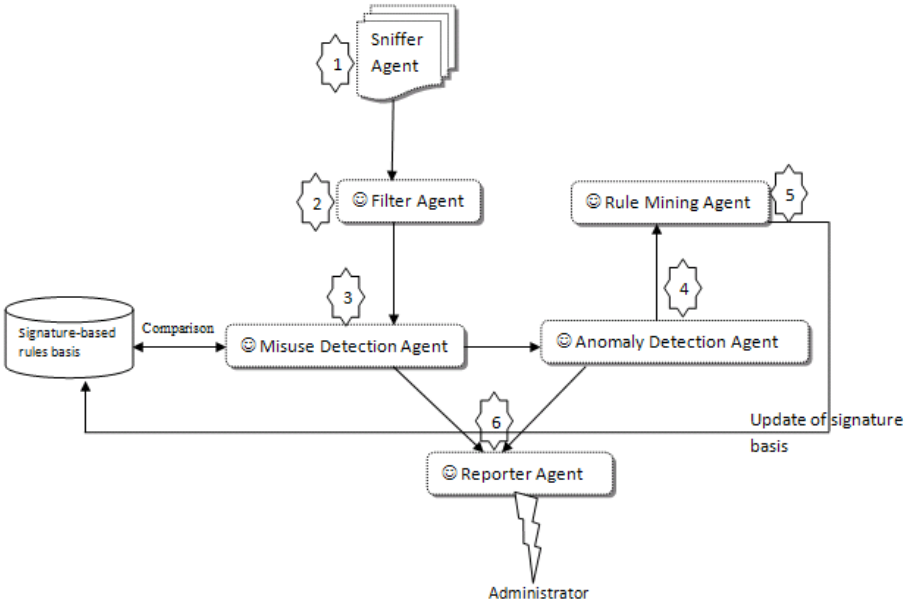

Fig. 1. The architecture of MAD-IDS

4. The remaining network packets are fed into an Anomaly Detection Agent, which uses the clustering technique to detect the abnormal connections;
5. The Rule Mining Agent aims at providing a concise representation of the network traffic. Typically, it summarizes the network connections that are selected as anomalous by the Anomaly Detection Agent. The obtained set of generic association rules can be periodically fed to the Misuse Detection Agent to update its signature database allowing the detection of known attacks;
6. Finally, the Reporter Agent generates reports and logs.

Each of these agents is individually described in the following subsections.

3.1 The Sniffer Agent (SA)

A sniffer is a device that is able to intercept and log traffic passing over a network. It allows the capture of each packet and, if needed, it analyzes its content. The traffic can be IP, IPX, or AppleTalk network packets. In general, the sniffing can be used to: *i*) analyze network problems; *ii*) detect network intrusion attempts; and *iii*) documenting regulatory compliance through logging all perimeter and endpoint traffic; etc.

Firstly, the Sniffer Agent (SA) captures the incoming packets by reading them from the network card in the machine and caches them in the memory at the interval of every 5 seconds. The benefits of this kind of agents include: *i*) the

cloning and the distribution throughout the network; and *ii*) the duplication in order to lighten the network charge. Finally, the captured packets will be the input of the next agent, *i.e.*, the Filter Agent.

3.2 The Filter Agent (FA)

A distributed IDS must undertake to analyze a huge volumes of events collected from different sources around the network. Consequently, the FA filters the packets already captured by the SA. It will treat these crude packets by achieving the following tasks:

- Distinguish the various fields of the packets collected in crude such as destination address and the protocol;
- Sort the packets by the category of packets (TCP, UDP, ICMP, etc.) concerned by a specific kind of intrusion.

The FA performs its tasks as a pretreatment phase, which precedes the analysis phase carried out by the following agent.

3.3 The Misuse Detection Agent (MDA)

This kind of agents analyzes the packets captured by the Sniffer Agent and then pre-processed by the FA. In fact, MDA searches for attack signatures[3] in these packets. Hence, if there is a similarity between the filtered packets and attacks signatures, then the agent raises an alert to the Reporter Agent, and then removes these anomalous packets from further analysis.

Within MAD-IDS, the MDA detects known attacks in the network connections, using a rule set based on the intruder signatures as well as an efficient pattern-matching using the RETE algorithm [13]. Attack signatures are a specific rule set provided by the Rule Mining Agent. Thus, each rule has the following structure:

1. *Rule antecedent part*: Contains the basic information about the rule, including:
   a Protocol: The protocol used by the packet being analyzed.
   b Source information: IP address and port of the source computer that originated the packet. It is also possible to ignore these addresses to apply the rule on all packets irrespective of the IP address or the port number;
   c Destination information: IP address and port of the destination computer in the packet.
2. *Rule conclusion part*: Indicates the alert that will be generated whenever the rule antecedent conditions are met.

[3] An attack signature is a known attack method that exploits the system vulnerabilities and causes security problem.

Example 1. Let consider the rule: {*Protocol=TCP, Src_IP=206.163.37.95, Dst_Port = 139*}⇒{*Attack*}. This rule will generate an alert, each time a packet using the IP source {206.163.37.95} and the destination port {139} is detected for any IP destination and for any source port.

Even thought, the known attacks are detected, it remains nevertheless the problem of the new attacks detection. In this respect, to protect a system against new and complex attacks, while building an automatic IDS, researchers are increasingly looking at using data mining techniques for anomaly detection [7, 32–34]. In particular, the clustering technique can be considered as one of the most important unsupervised learning algorithms [5, 7, 15, 45].

Thus, we introduce an anomaly detection agent based on the clustering technique. The agent is described in the following.

3.4 The Anomaly Detection Agent (ADA)

The Anomaly Detection Agent provides the combination of distributed IDS with the anomaly-based clustering technique.

Clustering is a widely used data mining technique [11, 26] which groups similar items, to obtain meaningful groups/clusters of data items in a data set. These clusters represent the dominant modes of behavior of the data objects determined using a similarity measure. A data analyst can get a high level understanding of the characteristics of the data set by analyzing the clusters.

Clustering provides an effective solution to discover the expected and unexpected modes of behavior and to obtain a high level understanding of the network traffic [7]. The main advantage that clustering affords is the ability to learn and detect intrusions in the audit data, while not requiring the system administrator to provide explicit descriptions of various attack classes/types [32]. The idea behind this technique is that the amount of normal connection data is usually overwhelmingly larger than the number of intrusions [15, 32]. Whenever this assumption holds, the anomalies and attacks can be detected based on cluster sizes, *i.e*, large clusters correspond to normal data, and the rest of the data points, which are outliers, correspond to attacks [15, 32].

Indeed, the ADA detects the abnormal packets using an unsupervised anomaly-based clustering algorithm that we introduce, called $\mathcal{AD}$-$\mathcal{C}$LUST (*Anomaly Detection-based Clustering*) [5]. $\mathcal{AD}$-$\mathcal{C}$LUST combines two prominent categories of clustering, namely: distance-based as well as density-based (*e.g.* K-MEANS [26] and DBSCAN [11], respectively). It exploits the advantages of one to palliate the limitations of the other and vice versa. On the one hand, K-MEANS is fast, easy to implement and not memory greedy. However , it is easily affected by the noise[4] and suffers from a greater time complexity, which becomes an extremely important factor within intrusion detection due to the very large dataset sizes [34]. Moreover, the *number of clusters dependency* and the *degeneracy* constitute the

[4] A noisy data point does not belong to any cluster properly [11].

drawbacks that hamper the use of K-MEANS for anomaly detection [15]. On the other hand, while the density-based clustering can solve the problem of the number of clusters dependency and handles noise well, it is not suitable as a stand-alone tool for intrusion detection. Indeed, the density-based clustering produces the suboptimal partitions of the dataset and puts lots of instances into noises which are dropped out, which makes the detection rates and the false positif rates worse [45]. Thus, the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm enables DBSCAN to efficiently handle very large data and improves the quality of the K-MEANS algorithm by removing the noisy data and solving the problem of the number of clusters dependency.

The processing steps of our algorithm $\mathcal{AD}$-$\mathcal{C}$LUST can be summarized as follows [5]:

1. Extraction of the density-based clusters that are considered as candidate initial cluster centers. The density-based clustering is used as a preprocessing step for the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm;
2. Compute the Euclidean distance between the candidate cluster center and the instance that will be assigned to the closest cluster. For an instance x_i and a cluster center z_i, the Euclidean distance is defined as:

$$distance(x_i, z_i) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2} \tag{1}$$

3. The size of a neighborhood of instances is specified by an input parameter. We use k' parameter to distinguish it from the k parameter used by the K-MEANS algorithm. Hence, k' specifies the minimal number of instances in a neighborhood and controls the granularity of the final clusters of the clustering-based density. If k' is set to a large value, then a few large clusters are found. To reduce the number of candidate clusters k' to the expected number k, we can iteratively merge the two most similar clusters. Otherwise, if k' is set too small, then many small clusters will be generated. The clusters will be split, new clusters will be created to replace the empty ones and the instances will be re-assigned to existing centers. This iteration will continue until there is no empty cluster. Consequently, the outliers[5] of clusters will be removed to form new clusters, in which instances are more similar to each other. In this way, the value of initial cluster centers k will be determined automatically by splitting or merging clusters;
4. Within the detecting phase, the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm performs the detection of intrusions. Thus, for each novel instance I the algorithm proceeds as follows:
   (a) Compute the Euclidean distance and find the cluster with the shortest distance to I.

[5] An outlier is defined as an object where an instance p of the dataset is further than distance D from the object, where p and D are parameters specified by the users.

(b) Classify I by the category of the closest cluster. Clearly, if the distance between I and the cluster of "normal" is the shortest one, then I will be a normal instance. Otherwise, I is an intrusion.

The pseudo-code is shown by Algorithm 1 and the used notations are summarized in Table 1.

Table 1. List of used notations in the $\mathcal{AD}$-$\mathcal{C}$lust algorithm.

$\mathcal{D}$	: The network dataset which contains n packets $\{p_1, ..., p_n\}$.
ε	: The Euclidean neighborhood radius.
η	: The minimum number of neighbors required in ε.neighborhood to form a cluster.
N	: The set of points in ε.neighborhood of p_i
C	: The set of clusters that outputs the algorithm.
Z	: The final cluster centers.
k	: The number of clusters to be found.
C'	: The clusters initially found by density based clustering algorithm.
Z'	: The initial cluster centers.
k'	: The number of clusters initially found by density based clustering algorithm.

Firstly, the algorithm starts by the use of the density-based clustering algorithm to produce a set of initial candidate clusters (*cf.* lines 2-14). Thus, it determines the neighborhood of each instance. If the neighborhood of instances is smaller than η, then a new cluster is created. Otherwise, the current instance is assigned to noise. The resulted clusters are considered as candidate initial cluster centers. The algorithm calculates the distance between the instances and the centers of the candidate clusters. Next, each instance will be assigned to the closest cluster (*cf.* lines 15-17). In addition, $\mathcal{AD}$-$\mathcal{C}$LUST splits or merges the clusters to automatically determine the value of initial cluster centers k (*cf.* lines 18-26). Finally, a new set of initial cluster centers is obtained and used in the K-MEANS algorithm to classify the instances into 'normal' clusters and 'abnormal' clusters. (*cf.* lines 27-28).

Once the anomaly network connections are selected, the Rule Mining Agent is ready to carry out its task, which is described in the following subsection.

3.5 The Rule Mining Agent (RMA)

Even thought improving the detection rate and reducing the false alarm rate for attacks is an objective of paramount importance for an IDS, this task alone is not sufficient if the rate of network traffic is very high. For example, suppose a security analyst examines the output of an anomaly detector every 10 minutes during which two million connections are established. Even if a hundred of these connections are reported as anomalous, it is unfeasible for the analyst to examine

Algorithm 1: The $\mathcal{AD}$-$\mathcal{C}$LUST algorithm

Input: $\mathcal{D}$: A network dataset.
Output: C: The set of classified clusters.

```
1  Begin
2    C' :=0 ;
3    k' :=0 ;
4    Foreach p_i ∈ D do
5      Compute the number of points N in the neighborhood of p_i defined
       with ε;
6      If N < η then
7        Mark p_i as noise ;
8      Else
9        Add p_i to cluster C' ;
10       C' := C' + 1 ;
11       k' := k' + 1 ;
12       return (N) ;
13     // Now we obtain C' clusters based on density.
14   Foreach cluster C' do
15     Find the cluster centers z'_j by computing the mean ;
16     Assign each instance p_i to the closest cluster such that C'(p_i) =
       min_distance(p_i, z'_j);
17   repeat
18     If k' > k then
19       Select two clusters based on density and join them;
20       Find the new cluster center z'_j ;
21       // Finally we will have k centers
22     Else
23       Select a cluster based on density and split it using the K-MEANS
         clustering algorithm ;
24       Find the new cluster center z'_j ;
25       //Finally we will have k centers ;
26   until achieving k clusters;
27   Z :=Z';
28   C := K-MEANS(k, Z, D);
29 End
```

them in 10 minutes. Thus, there is a need for an approach whereby anomalous connections can be summarized into patterns that capture the essence of the anomalous behavior. In addition to making the human analyst's task manageable, this also had the advantage of providing templates from which signatures of novel attacks can be built for augmenting the database of signature-based IDSs.

In this respect, the RMA uses the association rule mining to achieve summarization of detected attacks. The formalization of the association rule mining problem was initially introduced by Agrawal et *al.* [1]. Given a set of records, the

objective of mining association rules is to extract all rules of the form $\mathcal{X} \Rightarrow \mathcal{Y}$ that satisfy a user-specified minimum support and minimum confidence thresholds. In fact, $\mathcal{X}$ is the premise of the rule and $\mathcal{Y}$ is its conclusion.

In the intrusion detection context, the association rule mining have been found to be valuable for analyzing a network traffic data [24, 41, 43]. Often times, the number of anomalous connections flagged by an IDS can be very large, thus requiring analysts to spend a large amount of time interpreting and analyzing each connection. By applying the association rule mining techniques, analysts can obtain a high-level summary of anomalous connections [7].

Generally, standard association rule mining technique relies on a user-specified minimum support threshold to eliminate patterns that occur infrequently in the data. For a network intrusion data, the proportion of network traffic that corresponds to an attack is considerably smaller than that of normal traffic [7]. As a result, we should set the *minsupp*[6] value to a very low value to detect patterns involving the attack class [23]. However, this fact will degrade the performance of the association rule mining algorithm considerably and produces an overwhelming large number of rules for the normal class.

Although the association rules can detect sets of features that occur frequently in the network traffic data, the number of mined rules can be quite large, depending on the value of *minsupp*, which affects the speed of IDS and hampers its whole performance [7]. Some of these rules are redundant since they contains patters that correspond to the subsets of other patterns.

Example 2. Given two association rules:

1. R_1: *Protocol=TCP, Dst_Port=8888* $\Rightarrow$ *Attack*
2. R_2: *Dst_Port=8888* $\Rightarrow$ *Attack*

The first association rule is more descriptive than the second one. The premise of R_2, *Dst_Port=8888*, is a subset of the premise of R_1. If the support of these two rules is close, then R_2 is considered as redundant.

As consequence, to generate association rules without redundancy, we apply the *generic association rule basis*, which provides a concise representation of association rules introduced in [31]. Particularly, we apply the *Informative Generic Basis* ($\mathcal{IGB}$) [2]. In addition to the elimination of redundancy, the application of the $\mathcal{IGB}$ basis during an intrusion detection process provides the increase of the overall coverage of detectable attacks.

Thus, once the anomalous connections are detected by the ADA, then the RMA is ready to mine the generic association rule set using $\mathcal{IGB}$. The benefit is the reduction of information overloading for the security analysts. The extracted rule set may be a candidate signature-based rule for addition to the signature basis used by the MDA. This means that the database of signatures is updated regularly in order to ensure an adequate protection.

[6] *minsupp* is the minimum support threshold pre-defined by the user.

3.6 The Reporter Agent (RA)

The MDA and the ADA report their findings to the RA which transmits them to the administrator. Whenever an intrusion is detected, it will send an alert to the system administrator. This alert can be a message on the screen or a message to a centralized machine or an alert file.

4 Experimental results

In order to assess the overall performance of MAD-IDS in a realistic scenario, a prototype of the proposed architecture was implemented using Sun's Java Development Kit 1.4.1, the well known platform JADE 3.7, the Eclipse and the JPCAP 0.7.

JADE (*Java Agent DEvelopment Framework*)[7] is a software Framework, which simplifies the implementation of multi-agent systems. The agent platform can be distributed by moving agents from one machine to another one.

In addition, JPCAP (*Java library for CAPturing and sending network Packets*)[8] is an open source library for capturing and sending network packets.

All experiments were carried out on equivalent machines equipped with a 3GHz Pentium IV and 2GB of main memory running under Linux Fedora Core 6.

Through the carried out experiments, we have a twofold aim: first, we focus on the assessment of the $\mathcal{AD}$-$\mathcal{C}$LUST detection ability. Second, we have to stress on evaluating the overall performance of the MAD-IDS system.

4.1 $\mathcal{AD}$-$\mathcal{C}$lust performances assessment

To assess the performance of the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm, we choose to compare $\mathcal{AD}$-$\mathcal{C}$LUST *vs.* K-MEANS and DBSCAN algorithms respectively. During experiments, we partly use the common benchmark, namely, the KDD99 dataset [9]. The dataset provides a standard corpus for evaluating intrusion detection algorithms. It also introduced more insider attacks. The attacks can roughly split into four main categories:

1. *DoS* (*Denial of Service*);
2. *R2L* (*Remote to User*);
3. *U2R* (*Unauthorized Access to Root*);
4. *Probing* (*Surveillance and Probing*).

Indeed, Table 2 shows the distributions of record types in training and testing datasets, used during our experiments. The first row shows the numbers of normal network packets, while the second row gives the distributions of the network attacks.

[7] Available at: http://jade.tilab.com
[8] Available at: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/
[9] Available at: http ://www.ll.mit.edu/IST/ideval/data/data_index.html

Table 2. The considered KDD99 datasets.

Record Type	Training Set	Testing Set
Normal	897277	1260592
Intrusion	116743	180436
Total	1014020	1441028

Generally, to evaluate the detection ability of the intrusion detection algorithms, two interesting metrics are usually of use [34]: the *Detection Rate* (DR) and the *False Positive Rate* (FR).

1. The DR equals the number of correctly detected intrusions divided by the total number of intrusions in the dataset;
2. The FR equals the total number of normal instances that were incorrectly considered as attacks divided by the total number of normal instances in the dataset.

The value of the DR is expected to be as large as possible, while the value of the FR is expected to be as small as possible.

Table 3 sketches the DR and FR for the considered datasets, using, respectively the K-MEANS, the DBSCAN and the $\mathcal{AD}$-$\mathcal{C}$LUST algorithms.

Table 3. The DR and FR of $\mathcal{AD}$-$\mathcal{C}$LUST *vs.* K-MEANS and DBSCAN.

Algorithm	k	DR (%)	FR (%)
K-means	5	26.29	0.32
	23	57.52	3.09
	300	87.98	8.90
	600	97.19	12.50
	700	99.11	15.92
DBSCAN	621	94.64	24.58
AD-Clust	621	**99.12**	**1.41**

Indeed, we test K-MEANS using different numbers of the initial cluster centers k. According to Table 3, we remak that both DR and FR increase as far as the number of clusters increases. However, this relationship is not suitable, since within an effective anomaly detection algorithm the FR should decreases along with the increase of both number of clusters and DR [15]. In addition, whenever the values of k are equal to 5, 23 and 300, the DR value is low. Otherwise, when the value of k is equal to 600, the DR increases to 97%. The increase of DR is due to the partition optimal of K-MEANS. Thus, within the K-MEANS algorithm, we have to try different values of k in order to find the optimal one.

In addition, Table 3 shows that the DR of $\mathcal{AD}$-$\mathcal{C}$LUST is 99.12% and is greeter than that of DBSCAN. The FR of $\mathcal{AD}$-$\mathcal{C}$LUST is only 1.41% and is by far lower

than that of DBSCAN. The bad results of DBSCAN can be explained by the suboptimal partitions of the dataset and the consideration of several instances as noises which are dropped out.

Finally, Table 3 shows that the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm automatically partitions the same data set into 621 clusters. Moreover, it allows the reduction of FR along with the increase of DR.

During the following experiments, we set the value of k of K-MEANS at the optimal value, *i.e.* 621.

The rates DR and FR change in relation to each other. Whenever the DR is the highest, the FR is the lowest, and vice versa. Consequently, these two metrics can be of use to plot a ROC curve (*Receiver Operating Characteristic*) [27]. Figure 2 (Right) compares the ROC curve of $\mathcal{AD}$-$\mathcal{C}$LUST *vs.* those of K-MEANS and DBSCAN. The ROC curve assesses the accuracy of the intrusion detection algorithm [27]. Thus, we conclude that $\mathcal{AD}$-$\mathcal{C}$LUST is more accurate than K-MEANS and DBSCAN.

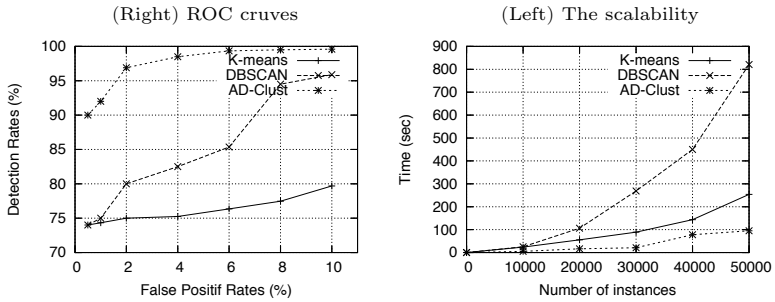

Fig. 2. Accuracy and scalability of $\mathcal{AD}$-$\mathcal{C}$LUST *vs.* K-MEANS and DBSCAN.

In addition, we tested the scalability of $\mathcal{AD}$-$\mathcal{C}$LUST, DBSCAN and K-MEANS. Figure 2 (Left) plots the running time against different numbers of data points. These results show that the running time of the three algorithms linearly increases with the number of instances. Thus, we can conclude that $\mathcal{AD}$-$\mathcal{C}$LUST is faster than both DBSCAN and K-MEANS.

4.2 Overall Performance of the MAD-IDS System

In the following, the performance of the entire MAD-IDS architecture will be evaluated. In this respect, we used machines that were connected via a switch, thus forming a switched network. For a realistic testing environment, attacks needed to be interjected into a volume of network traffic. Consequently, we simulated attacks using the well known tool *Metasploit*[10] version 3.5.1. Metasploit is both a penetration testing system and a development platform for creating

[10] Available at: http://www.metasploit.com/

security tools. It is used by the security researchers world-wide to test an IDS. The description of the eight different attack types used in the evaluation is shown in Table 4.

Table 4. The simulated attacks at a glance.

Attack Name	Description
attack1: DoS Smurf	ICMP echo reply flood, caused by an ICMP echo packet with spoofed address (of victim) sent to a network broadcast address.
attack2: Backdoor Back Office	A remote administration tool that allows almost complete control over a computer by the remote attacker.
attack3: SPYWARE-PUT Hijacker	The spyware Hijacker is a type of malware that can be installed on computers, and which collects small pieces of information about users without their knowledge.
attack4: Nmap TCP Scan	Scans many ports to determine available services on a single host using UDP packets.
attack5: Finger User	Allows an attacker to disrupt a network using the redirection capability in the finger daemon.
attack6: RPC Linux Statd Overflow	Buffer overflow vulnerability exists making it possible for malformed requests by an attacker to be devised giving root privileges.
attack7: DNS Zone Transfer	DNS server provides information for all DNS resource records registered with DNS server that can be used by attackers to better understand a network.
attack8: HTTP IIS Unicode	An attacker could send a specially crafted URL containing Unicode characters to access files and folders on the Web server with the privileges of the user account.

We have conducted evaluations in terms of (*i*) the detection delay; (*ii*) the scalability-related criteria such as network bandwidth and system response time; and (*iii*) the detection ability. During the evaluations, we compare the results of the MAD-IDS system *vs.* that of Snort. Snort is an open source network IDS. It is able to perform the analysis of network traffic in a real-time using a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods [36].

Multi-agents *vs.* centralized IDS Firstly, we answer to the question: why the realization of the multi-agents IDS is advantageous ? Indeed, Figure 3 (Right) plots the detection delay against the number of packets, using the MAD-IDS and Snort systems. Clearly, the results show that the detection delay of both

systems linearly increases with the number of packets. Thus, the figure highlights that our proposed system MAD-IDS is faster than the system SNORT.

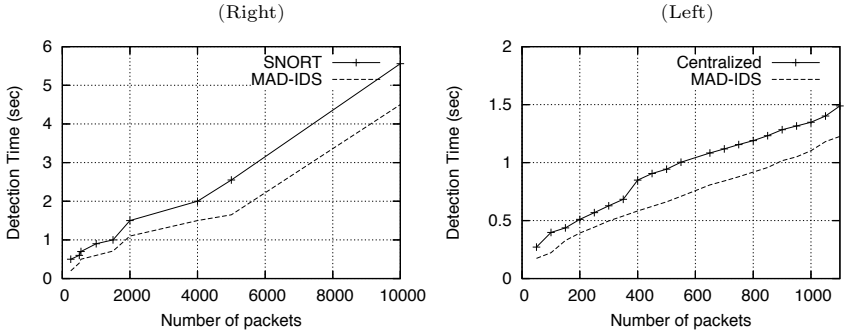

Fig. 3. Multi-agents *vs.* centralized IDS in terms of detection delay.

Moreover, we also implemented a centralized IDS with local sensor that forward filtered data to a central analysis node. Figure 3 (Left) shows the detection time needed, using the MAD-IDS system *vs.* that of need by a centralized one. Thus, our proposed distributed IDS is much faster than the centralized one. This can be explained by the fact that agents operate directly on the host, where an action has to be taken, their response is faster than systems where actions were taken by the central coordinator.

Bandwidth consumption and response time It is known that two of the most important elements of network performance are bandwidth and latency. On the one hand, the bandwidth is the transmission capacity of the network, usually measured in bits per second.

On the other hand, the network latency is the amount of time it takes for a packet to travel from the source to the destination. We use latency to describe the amount of time it takes once an attack takes place till it gets resolved.

As depicted in Figure 4 (Right), the maximum bandwidth consumed by MAD-IDS is 0.06 Mbits/sec, which is very low as well. This makes our proposed system low cost which is definitely a desirable feature for any distributed system.

Figure 4 (Left) illustrates the network latency, *i.e.*, the response times required by MAD-IDS with respect to the attack types. According to this figure, the detection of all attack types, on average, result in very similar and low response time. This is definitely a desirable feature.

In conclusion, it is clear from the obtained results that in our proposed architecture, the performance of the system will not deteriorate too much with the increase in the number of attacks, which is justified by its low bandwidth consumption and quick response time behavior. Also, in case of more machines are connected to the network, the MAD-IDS system still withstand the load and

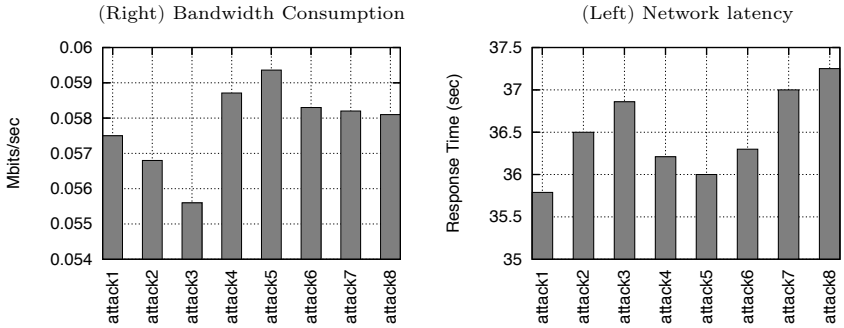

Fig. 4. The bandwidth Consumption and the network latency of MAD-IDS.

swiftly deliver the results.

Detection ability According to Figure 5 (Right), we can see that the false alarm rates of our adaptive system is significantly lower compared to that of Snort. For instance, adaptive mechanisms used by the agents can change normal profiles correspondingly, enabling MAD-IDS to better suit the environment. Consequently, false alarms can be reduced correspondingly.

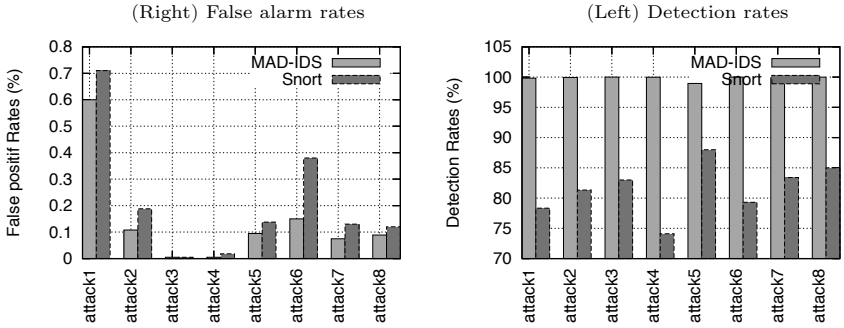

Fig. 5. Detection ability of MAD-IDS *vs.* Snort.

This result is confirmed by Figure 5 (Left) that shows that the detection rates of MAD-IDS is by far better than Snort configuration.

Knowing that a main challenge of existing IDSs is to decrease the false alarm rates, the main benefit of our adaptive system is to lower the false alarm rate, while maintaining a good detection rate.

5 Conclusion

In this paper, a novel distributed IDS, called MAD-IDS was introduced. MAD-IDS incorporates the desirable features of the multi-agents methodology with the highly accurate data mining techniques. On the one hand, the multi-agent system is efficient for enhancing security, flexibility and cooperative detective ability of distributed IDS. On the other hand, the MAD-IDS system uses the clustering and the association rules techniques to analyze large network packets more intelligently and automatically. In fact, the anomaly-based clustering technique, particularly the $\mathcal{AD}$-$\mathcal{C}$LUST algorithm, will try to automatically determine which data instances fall into the normal class and which ones are intrusions. The $\mathcal{AD}$-$\mathcal{C}$LUST algorithm was able to detect a large number of intrusions while keeping the false rate reasonably low. As consequence, our system addresses the specific limitations described by the central approaches and the monolithic systems. It performs a distributed search and analysis using multi-agent system, eliminating then the single point of failure drawbacks. In addition, MAD-IDS guarantees that the system can be extended in a straightforward manner. For example, new task agents can be added after the MAD-IDS is initially deployed, without requiring any changes to the existing set of agents. The experimental results showed that MAD-IDS is an efficient system that yields promising results, indicating a high accuracy, a good scalability and a low response time.

Future work include the following issues:

- First, the network data is temporal (*i.e.*, *streaming*) in nature, and the development of algorithms for mining data streams is necessary for building real-time IDS;
- Second, the low frequency of computer attacks requires modification of standard data mining algorithms for their detection;
- Finally, new approaches have been suggested using ontologies as a way to represent and understand the attacks domain knowledge, expressing the IDS much more in terms of their domain and performing intelligent reasoning [21]. Thus, as future work, we need to integrate ontology within MAD-IDS. This fact can provide agents with a common interpretation of the environment signatures which are matched through the data structure based on the MDA.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the International Conference on Management of Data, Washington, D.C.*, pages 207–216, 1993.
2. S. Ben Yahia, G. Gasmi, and E. Mephu Nguifo. A New Generic Basis of Factual and Implicative Association Rules. *Intelligent Data Analysis*, 13(4):633–656, 2009.
3. Y. Bouzida and F. Cuppens. Detecting Known and Novel Network Intrusion. In *Proceedings of the 21st IFIP International Conference on Information Security, Karlstad, Sweden*, pages 258–270, 2006.

4. I. Brahmi, S. Ben Yahia, and P. Poncelet. A Snort-Based Mobile Agent For A Distributed Intrusion Detection System. In *Proceedings of the International Conference on Security and Cryptography, Seville, Spain*, 2011. To appear.
5. I. Brahmi, S. Ben Yahia, and P. Poncelet. $\mathcal{AD}$-$\mathcal{C}$LUST: Dtection des Anomalies Basée sur le Clustering. In *Atelier Clustering Incrémental et Méthodes de Détection de Nouveauté en conjonction avec 11ème Conférence Francophone d'Extraction et de Gestion de Connaissances EGC 2011, Brest, France*, pages 27–41, 2011.
6. A. Chalak, R. Bhosale, and N. D. Harale. Effective data mining techniques for intrusion detection and prevention system. In *Proceedings of the International Conference on Advanced Computing, Communication and Networks'11, Chandugari, India*, pages 1130–1134, 2011.
7. V. Chandola, E. Eilertson, L. Ertoz, G. Simon, and V. Kumar. Data Mining for Cyber Security. In A. Singhal, editor, *Data Warehousing and Data Mining Techniques for Computer Security*, pages 83–103. Springer, 2006.
8. D. Christine, J. Hyun Ik, and Z. Wenjun. A New Data-Mining Based Approach for Network Intrusion Detection. In *Proceedings of the 7th Annual Conference on Communication Networks and Services Research, Moncton, New Brunswick, Canada*, pages 372–377, 2009.
9. H. Debar, M. Dacier, and A. Wespi. Towards a Taxonomy of Intrusion-Detection Systems. *Computer Networks*, 31:805–822, 1999.
10. O. Depren, M. Topallar, E. Anarim, and M.K. Ciliz. An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks. *Expert System with Applications*, 29:713–722, 2005.
11. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, Oregon*, pages 226–231, 1996.
12. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD Process of Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, 39(11):27–34, 1996.
13. C. Forgy. RETE: A Fast Algorithm for the many Pattern/many Object Pattern match Problem. *Artificial Intelligence*, 19(1):17–37, 1982.
14. R. Gopalakrishna and E.H. Spafford. A Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, Davis, CA, USA*, 2001.
15. Y. Guan, A. Ghorbani, and N. Belacel. Y-MEANS: A Clustering Method for Intrusion Detection. In *Proceedings of Canadian Conference on Electrical and Computer Engineering; Montréal, Québec, Canada*, pages 1083–1086, 2003.
16. G. Helmer, J.S.K. Wong, V.G. Honavar, and L. Miller. Automated Discovery of Concise Predictive Rules for Intrusion Detection. *Journal of Systems and Software*, 60(3):165–175, 2002.
17. T. Helmy. Adaptive Ensemble Multi-Agent Based Intrusion Detection Model. In K. Ragab, T. Helmy, and A. E. Hassanien, editors, *Developing Advanced Web Services through P2P Computing and Autonomous Agents: Trends and Innovations*, pages 36–48. IGI Global, 2010.
18. Á. Herrero and E. Corchado. Multiagent Systems for Network Intrusion Detection: A Review. In *Proceedings on the Computational International in Security for Information Systems, AISC 63, Berlin, Heidelberg*, pages 143–154, 2009.

19. W. Huang, Y. An, and W. Du. A Multi-Agent-Based Distributed Intrusion Detection System. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering, Chengdu, Sichuan province, China*, pages 141–143, 2010.
20. L.-F Iren, M.-P. Francisco, M.-G. F. José, L.-F. Rogelio, G.-M.-A. J. Antonio, and M.-J. Diego. Intrusion Detection Method Using Neural Networks Based on the Reduction of Characteristics. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks, Salamanca, Spain*, pages 1296–1303, 2009.
21. G. A. Isaza, A. G. Castillo, and N. D. Duque. An Intrusion Detection and Prevention Model Based on Intelligent Multi-Agent Systems, Signatures and Reaction Rules Ontologies. In *Proceedings of the 7th International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAMS'09, Salamanca, Spain*, pages 237–245, 2009.
22. G. Kolaczek and K. Juszczyszyn. Attack Pattern Analysis Framework for Multiagent Intrusion Detection System. *International Journal of Computational Intelligence Systems*, 1(3), 2008.
23. W. Lee. A D*ata* M*ining* F*ramework for* C*onstructing* F*eatures and* M*odels for* I*ntrusion* D*etection* S*ystems*. Phd thesis, Columbia University, New York, NY, USA, 1999.
24. T. R. Li and W. M. Pan. Intrusion Detection System Based on New Association Rule Mining Model. In *Proceedings of the International Conference on Granular Computing; Beijing, China*, pages 512–515, 2005.
25. C.-L. Lui, T.-C. Fu, and T.-Y. Cheung. Agent-Based Network Intrusion Detection System Using Data Mining Approaches. In *Proceedings of the 3rd International Conference on Information Technology and Applications, Sydney, Australia*, pages 131–136, 2005.
26. J.B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley*, pages 281–297, 1967.
27. R. A. Maxion and R. R. Roberts. Proper Use of ROC Curves in Intrusion/Anomaly Detection. Technical report series cs-tr-871, School of Computing Science, University of Newcastle upon Tyne, 2004.
28. R. G. Mohammed and A. M. Awadelkarim. Design and Implementation of a Data Mining-Based Network Intrusion Detection Scheme. *Asian Journal of Information Technology*, 10(4):136–141, 2011.
29. E. Mosqueira-Rey, B. Guijarro-Berdias A. Alonso-Betanzos, D. Alonso-Ros, and J. Lago-Pieiro. A Snort-based Agent for a JADE Multi-agent Intrusion Detection System. *International Journal of Intelligent Information and Database Systems*, 3(1):107–121, 2009.
30. E. J. Palomo, E. Domínguez, R. M. Luque, and J. Mu noz. A Self-Organized Multiagent System for Intrusion Detection. In *Proceedings of the 4th International Workshop on Agents and Data Mining Interaction, Budapest, Hungary*, pages 84–94, 2009.
31. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Journal of Information Systems*, 24(1):25–46, 1999.
32. A. Patcha and J.M. Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Computer Networks*, 51:3448–3470, 2007.

33. S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas. Modeling Intrusion Detection System Using Hybrid Intelligent Systems. *Journal of Network Computer Applications*, 30:114–132, 2007.
34. L. Portnoy, E. Eskin, and W. S. J. Stolfo. Intrusion Detection with Unlabeled Data using Clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001), Philadelphia, PA*, 2001.
35. M. Rehák, M. Pechoucek, P. Celeda, J. Novotny, and P. Minarik. CAMNEP: Agent-Based Network Intrusion Detection System. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, Estoril,Portugal*, pages 133–136, 2008.
36. M. Roesch. SNORT - Lightweight Intrusion Detection System for Networks. In *Proceedings of of the 13th USENIX Conference on System Administration (LISA'99), Seattle, Washington*, pages 229–238, 1999.
37. J. Shun and H.A. Malki. Network Intrusion Detection System Using Neural Networks. In *Proceedings of the 4th International Conference on Natural Computation (ICNC'08), Jinan, China*, pages 242–246, 2008.
38. M.-L. Shyu and V. Sainani. A Multiagent-based Intrusion Detection System with the Support of Multi-Class Supervised Classification. In D*ata* M*ining and* M*ulti-agent* I*ntegration*, pages 127–142. Springer-Verlag, 2009.
39. E.H. Spafford and D. Zamboni. Intrusion Detection Using Autonomous Agents. *The International Journal of Computer and Telecommunications Networking*, 34(4):547–570, 2000.
40. S. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan, and P.K. Chan. JAM: Java Agents for Meta-Learning over Distributed Databases. In *Ptoceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, California*, pages 74–81, 1997.
41. F.S. Tsai. Network Intrusion Detection Using Association Rules. *International Journal of Recent Trends in Engineering*, 2(2):202–204, 2009.
42. M. Wooldridge. A*n* I*ntroduction to* M*ultiAgent* S*ystems* - S*econd* E*dition*. John Wiley and Sons, 2009.
43. W. Xuren, H. Famei, and X. Rongsheng. Modeling Intrusion Detection System by Discovering Association Rule in Rough Set Theory Framework. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation, Sydney, Australia,*, pages 24–29, 2006.
44. Y.F. Zhang, Z.Y. Xiong, and X.Q. Wang. Distributed Intrusion Detection Based on Clustering. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou*, pages 2379–2383, 2005.
45. Z. Zhao, S. Guo, Q. Xu, and T. Ban. G-MEANS: A Clustering Algorithm for Intrusion Detection. In *Processing of the 15th International Conference on Advances in Neuro-Information, Auckland, New Zealand*, pages 563–570, 2008.