Towards a new approach for intrusion detection with intelligent agents

Houda Labiod, Karima Boudaoud, and Jacques Labetoulle
Eurécom Institute
Corporate and Mobile Communication Laboratories
2229, route des Crêtes
BP 193, 06904 Sophia-Antipolis
Email: {boudaoud,labiod, labetoulle}@eurecom.fr
Phone: (33) 4 93 00 26 38 / (33) 4 93 00 29 06
Fax: (33) 4 93 00 26 27

Abstract

In this paper, we focus on one critical issue in security management that is intrusion detection. Intrusion detection requirements and concepts are reviewed. Some existing systems are described. Their advantages and limitations are illustrated. Drawbacks of existing intrusion detection systems involve the necessity of designing a new generation of self-adaptive systems. In fact, mainly, self-control, flexibility, adaptability, autonomy and distribution are the main features to be addressed in a suitable architecture that fulfills these requirements. In this context, we propose a new approach based on intelligent agent technique. Therefore, the introduction of a multi-agent system in an intrusion detection system is proposed as a means of implementation of adaptive and autonomous decision features embedded in agents distributed over intrusion detection related entities. A new multi-agent intrusion detection architecture called MAIDA is described. To bear out the feasibility of the multi-agent approach, two specific security attacks (doorknob rattling and IP spoofing) are explored within the platform, that we choose to use to develop our multi-agent system architecture, which is named Development and Implementation of the Multi-Agents systems (DIMA).

Keywords

Network security management, intrusion detection, network attacks, distributed management, intelligent agent theory, multi-agent system technique.

Introduction

Huge use of networks and computer systems has come at the price of weakening data and network security, which can not be overlooked. Data and networks are distributed, making themselves more vulnerable to attacks. In addition, most networks and systems have some vulnerabilities that can be exploited by unauthorized users and permit them to gain unauthorized access to sensitive data. As these networks are used by a large number of organizations, network security becomes an important and major issue that must be considered carefully. Many existing systems may be designed to prevent specific types of attacks, but other methods to gain unauthorized access may still be possible [1]. So, protecting a network or system against all possible attacks and having a completely-secure network is, in practice, irrealistic. Due to all efforts made into the existing architecture of "open" communication networks, it is not possible to deploy new, secure and possibly "closed" networks [1]. So, it is necessary to detect security violations when they happen.
The existing solutions are very complex and costly. What needed is a flexible, adaptable and affordable security solution, which provides greater autonomy. Therefore, it is required to review the way in which standard intrusion detection is designed and performed in order to identify and to alleviate its weakness. In this context, we propose a new approach based on intelligent agent technique, which reveals itself as a suitable candidate to make a balance between security requirements and system flexibility and adaptability in the case of the network intrusion detection.

Actually, intelligent agent technology is viewed as one of the fastest growing areas of research and new application development in telecommunications. The Distributed Artificial Intelligence (DAI) concept [2] consists of a group of individual agents that have distributed environments. Each agent cooperates and communicates with other agents. Combined knowledge and experience of the agent with the information coming from neighboring agents permits the agent to make the best (optimum in some sense) decision. In this paper, we suggest to improve network security by integrating DAI technology based on multi-agent system (MAS) technique in Intrusion Detection Systems (IDS). This technique seems so promising to embed adaptive features in network entities. These entities become more intelligent, capable of making various decisions with autonomy to detect attacks and to overcome their bad effects. In our work, the term intelligence is used in the sense that network security entities should provide deliberation capabilities, exhibit behavior autonomy, adaptability, interaction, communication and co-operation in order to reach some intrusion detection goals. Then, we built a new architecture called MAIDA (Multi Agent Intrusion Detection Architecture). It is used to provide a flexible integration of multi-agent technique in a classical network to enhance its protection level against inherent attacks.

This paper includes four main parts. The first one is devoted to a state of the art on intrusion detection and the second one to the intelligent agent technology and multi-agent systems. This part gives also the reasons for choising such an orientation. The third part exposes our proposal for a global architecture technique for intrusion detection based on multi-agent system technique. The fourth part describes, on one hand, the multi-agent platform DIMA (Development and Implementation of the Multi-Agents systems) used to implement our new architecture and our agent model. On the other hand, some technical aspects related to MAIDA implementation are also discussed.

I. STATE OF ART ON INTRUSION DETECTION SYSTEMS

I.1 Introduction

Since the computer exists, the networks did not cease growing and evolving. Indeed, networks and computer systems became essential tools in the evolution of the majority of companies. Thus, computer systems and networks started to touch various fields such as bank, insurance, medicine,military field,... Increase in interconnection of these various systems and networks, made them accessible by a diversified population of users which does not cease increasing. These users, known or not, do not have always good intentions when they access to these networks. Actually, they can try to access to critical information to read, modify or destroy it, or simply to attack and disturb the system. Since these networks seem being targets of potential attacks, making them safe become an important issue that can not be ignored.
The security of networks and systems can be performed, either in a preventive or curative way. In the preventive approach, it concerns the protection of both data and resources against any unauthorized or abusive access. However, to prevent against all security violations appears to be an unrealistic task. It is practically impossible to have a completely secure network and to protect it against all possible attacks. The curative approach seems then a good way to assure the security of networks and systems, since it would consist in trying to detect these attacks when they occur. Caused damages are then repaired. To evaluate and ensure the security needs for a company, three aspects must be considered:

- Security services (authentication, access control, confidentiality, integrity of data, non-repudiation),
- security attacks,
- and finally security mechanisms.

In this paper, we focus on intrusion detection. So we start by presenting the security attacks and after we will study the intrusion detection, which corresponds to the curative approach.

I.2 Security attacks and intrusions

A security attack or intrusion can be defined like any action or set of actions which can violate the security of a system or a network and tries to compromise the confidentiality, the integrity or the availability of a resource or a service [3][4][5][6].
Because of the significant number of possible attacks, we initially start by classifying them then we present some of them.

I.2.1 Classification of the attacks

In spite of the diversity of security attacks, we find in the literature five main classifications.

a. **First classification**
Because of the various forms of intrusion, they can be gathered in two main classes [3]:

- **Misuse intrusions**: are known and well defined attacks, which generally exploit known vulnerabilities of the target system,
- **Anomaly intrusions**: They are unknown attacks, which are observed as deviations from the normal profile of a system. They are detected when an abnormal behavior of the system is observed with comparison to its normal profile.

b. **Second classification**
The various attacks can be classified in two categories [7]:

- **External attacks**: They are generated from the outside by a hacker who is trying to access a network to have information, have fun, or trying any kind of denial of service attack.

- **Internal attacks**: They are generated by internal users. These users abuse of their rights and privileges to do unauthorized activities and to obtain unauthorized access.

c. **Third classification**

It is the most traditional classification, it contains four types of attacks [3]. Thus an attack can compromise:

- the **confidentiality** of information by breaking private rules,
- the **integrity**, by altering the data,
- the **authenticity** of the data,
- or finally the **availability**, by making a system or a computer network unavailable. This case is related to denial of service attacks.

d. **Fourth classification**

It corresponds to the classification of Stallings [8], which identifies two types of attacks:

- **passive attacks**: They violate the confidentiality. They are not easily detectable because they do not imply any deterioration of information. We distinguish two types:
  - *delivery of confidential messages*: electronic mail, transferred file ;
  - *traffic analysis* : which determines the elements of a communication (identity of communicating hosts, frequency and length of the exchanged messages,...).
- **active attacks**: They involve a modification of the data or creation of incorrect data. Therefore, they compromise the integrity, the authenticity and the availability. We find four types of active attacks:
  - *masquerade*: It happens when an entity pretends to be a different one,
  - *replay*: It is a retransmission of captured messages during a communication for illegitimate purposes,
  - *modification of messages*,
  - *denial of service*.

e. **Fifth classification**

Security attacks can also be classified in terms of:

- **Network attacks**: They aim to prevent the users from using a network connection, to make unavailable a machine or a service. They also monitor the network traffic in order to analyze it and collect relevant information.
- **System attacks**: They compromise the system by modifying or erasing critical files such as the password file.
- **Web attacks**: An example of these attacks is the modification of the site's web page by a hacker in order to discredit it or simply to make it ridiculous.

I.2.2 Description of attacks

In the literature, we find a lot of kinds of attacks but, in this paper we limit ourselves to a short description of the most important ones, related to network and system attacks

I.2.2.1 Network attacks

- **IP spoofing** : consists of sending packets with a faked IP source address. The server should believe that the packets come from another host, probably a host that is allowed to establish connections with the attacked host, if the real one is not allowed [9].
- **TCP SYN flooding or SYN attack**: its objective is to make a TCP service offered on a machine unavailable [10][11].
- **ICMP flooding or Smurf attack**: ICMP (Internet Control Message Protocol) packets (usually ping requests but other type of requests are also possible) can be used to flood a network and bring it down. Requests must be sent at a high rate to many host destinations.

Concurrent answers generate many collisions on local area networks and fill routers queues [12].

- **Doorknob Rattling**: it results in repetitive attempts to log in several hosts with any userid/password combination in order to obtain an access to an account.
- **Sniffing**: it is the observation and analysis of network traffic in order to obtain relevant information (such as IP addresses and host functionalities) to perform other attacks.

I.2.2.2 System attacks

- **Virus and logical bomb**: a virus is a program, which is written in order to destroy a system (erasing for example disks and relevant files); whereas a logical bomb is a program that only destroy activities when they are launched.
- **Worms**: it is an autonomous agent able to be propagated without the use of a specified program or an action of a person. Only host resources are used in order to attack other hosts. A famous example is the worm produced on internet in November 1988 [13].
- **Trojan horse**: is a program, which hides himself in another program. When the victim host launches this program, it launches also the hidden Trojan horse. For example, it can allow access to the system to particular people or even to everyone.
- **Trap door**: it is an entrance point in a system, which passes through existing safety measures. Mostly, it is a hidden program, which is often activated by an event or a normal action and of which the goal is to weaken a protection system or even to make it ineffective.
- **Cracking of passwords**: the most traditional way to crack passwords is to use a program called "cracker " which uses a dictionary of words and proper names. It operates on the encrypted passwords, which are saved in a particular file (such as : / etc/passwd).

I.3 Detection of intrusion

Intrusion detection is a general term which refers to the capacity of a computer system to determine automatically, by analyzing security-relevant events, that a violation of security occurs or occurred in the past [14][15]. Intrusion detection requires that a great number of security-relevant events are collected and recorded in order to be analyzed. The role of an intrusion detection system (IDS) is to monitor system activities to detect malicious actions and to identify unauthorized and abusive uses [16]. It offers a defense when system vulnerabilities are exploited without requiring expensive equipments. The IDS is executed in background. When it detects suspicious or illegal activities, it notifies the security administrator [7].

In this section, we describe some characteristics of IDS. Then we present the various methods of intrusion detection. Lastly, after having identifying criteria of a good IDS, we briefly present some existing IDS.

I.3.1 Charateristics of intrusion detection systems

When designing an IDS several parameters are to be taken into account:

- the **source of data** which will be collected in order to be analyzed,
- the **topology** of the target system to be analyzed, which can either be a stand-alone computer system or a distributed one,
- and **mode of analysis**, which can be either real time, or in batch mode.

a. Source of the data

For detecting intrusive activities, an IDS can use two kinds of data:

- **audit file data**,
- **network traffic data**.

According to the source of data, we distinguish three types of IDS:

- **Host-based IDS**, which are designed to monitor a single host. They use their own host Operating System's audit trail as the main source of input for detecting intrusions.
- **Distributed Host-based IDS**, which are in charge of monitoring several hosts. They perform intrusion detection using Operating System's audit trail or information from multiple monitored hosts. This information is processed on a central site.
- **Network-based IDS**, which analyze traffic on a LAN to detect intrusive behavior.

b. Distributed vs Stand-alone computer systems topology

There is a significant difference between intrusion detection in a distributed environment and on stand-alone computer system [17]. In this latter, the audit process is performed on a single host whereas in the distributed one, it is done on all the hosts [18]. [17] provides a short description of the major differences between the two approaches:

- On a stand-alone computer system, the audit data collection is performed by a single audit mechanism, which makes the audit record format consistent. In a distributed system, the audit data collection is ensured by several audit mechanisms which require a comparison of the audit records of the various components and a coordination of the analysis of the different hosts.
- In a distributed environment, the data collection and analysis, can be done using a central repository, or distributed repositories with several analyzers. In this last case, coordination between the various analyzers is necessary for producing system-wide audit information.
- The heterogeneity of a distributed network multiplies the vulnerabilities of the various systems, contrary to a stand-alone computer system.
- Collected information is more significant in a distributed environment which implies the use of sophisticated algorithms and large archiving systems.
- In a distributed system, communication protocols are used to support the audit data of the various components and the messages exchanged between the multiple analyzers. In order to preserve data integrity and confidentiality, making safe the exchanges is needed.

c. Batch mode vs Real time analysis

In batch mode intrusion detection, the analysis of audit data occurs some time after the data has been collected, either during low CPU periods or on a different system altogether [17]. The disadvantage of this analysis is that it detects the attacks only after the damages are caused. The batch mode intrusion detection can be very useful in environments where periodic summaries of suspicious uses are sufficient [17].

In real time intrusion detection, the audit data must be analyzed as soon as they are created.

This mode can appear crucial for critical systems in order to identify suspicious user behaviors when they occur and to detect instantaneously any security violation. However, the analysis in real time, can generate hardware and software performance problems. Indeed, a high reliability, a large storage capacity and a high-speed hardware, become key issues [17]. Moreover, to make real-time audit data analysis viable, the delay between the moment when an audit transaction occurs and the moment when it is written on the disk must be reduced considerably.

I.3.2 Intrusion detection methods

Intrusion detection is based on two principal approaches: behavior-based approach named anomaly detection approach and knowledge-based approach called also misuse detection approach. These two approaches are presented in the following sections.

I.3.2.1 Behavior-based intrusion detection approach

Intrusion detection based on this approach permits the detection of unknown intrusions and thus does not require any a priori knowledge on intrusions. This approach is based on the fact that an intruder does not behave in same manner as a regular user. Contrary to the user, who

has a normal behavior, the intruder has an abnormal behavior. Thus, all the intrusive activities are inevitably abnormal [19].
For [20], anomaly detection tries to quantify the usual behavior considered to be acceptable and considers irregular behaviors as potential intrusions.
For [21], abnormal behavior detection is perceived as a problem of classification which can be valued in a binary way. System measurements are then used to decide that the state of the system is normal or abnormal.
This approach consists in establishing normal behavior profile for user activity and observing significant deviations of actual user activity with respect to the established normal pattern [22]. Thus, such a system of detection learns the normal behavior profiles and reports the anomalies to a human operator or to a part of the system for a detailed analysis [23].
In the context of this approach, we can find the system that has been designed by [21]. Its goal is to learn a user profile and to use it to detect an abnormal behavior. The behavior-based detection is done in two phases:

- A learning phase: system learns the normal behavior from a subject (user or system). Thus, it creates "the normal profile" of a user based on collected data.
- A detection phase: system examines the current audit trail or network information. It compares the collected information with the profile in order to check possible intrusive activities. An alarm is raised in the case of significant differences.

To formalize the normal behavior of a subject (user or system), there are several approaches. They are divided into two approaches: modelisation and prediction.

1. Modelisation
Normal behavior can be modeled using a statistical model or an expert system.

- **Statistical model**: It is the most used. It generates a model of normal behavior of a user or a system. It consists in using statistical measurements to model a profile of behavior and to detect intrusive behaviors. These measurements include CPU time used and the number of connections during a certain period of time. Each of these values is associated with a threshold or an interval that indicates an abnormal activity. This approach is based on two techniques [17][18]:
  - *Threshold detection*: each occurence of specific events are recorded. When the number of events exceeds a fixed threshold, the presence of an intruder is detected and an alarm is raised.
  - *Statistical profile-based anomaly detection*: this approach uses statistical measurements to identify the expected behavior of a user or group of users [18] and to build a user profile considered as normal. This is made during the learning phase. During detection phase, the variation between the current profile and the expected profile is evaluated continuously. Suspection of intrusive behaviors is then detected when the deviation between the current profile and the expected profile is significant.
- **Expert systems**: the main difference between an expert system and a statistical model is that this last one uses statistical formulas to identify behaviors in the audit data whereas the expert system uses a set of rules to represent these behaviors [17][18]. The main disadvantage of an expert system is that its base of rules is not easy to create, and to maintain.

There is another model, called **Denning model** [24], which is independent of system, application environment and of inrusion types. It is a generic model. It is composed of six elements: subjects, objects, audit records, profiles, anomaly records and activity rules. In this model, a profile is characterized in terms of a statistical metric and model. A metric is a random variable representing a quantitative measure accumulated over a period. The period

can be a fixed interval of time (minute, hour, day, week, etc.) or the time between two audit-related events (login and logout, connection and disconnection, etc) [24]. There are three types of metrics: event counter, interval timer and resources measure. Given a metric for a random variable x and n observations x_1, ..., x_n , the goal of a statistical model of x is to determine whether a new observation x_{n+1} is abnormal with respect to the previous observations [24]. The statistical models suggested by Denning are: operational model, standard deviation/average model, covariances model, Markov process model and temporal model.

2. Prediction

This approach tries to predict future events based on events that have already occurred [19]. It is based on the assumption that the sequences of events are not random. If a certain amount of the current data does not agree with the predicted data, an alarm is raised. The most significant approaches are:

- **Predictive pattern generation**: It predicts the most probable patterns based on previously observed patterns. During the learning phase, it determines the time-based rules that characterize the normal behavior of users. In the detection phase, a deviation is detected if the observed sequence of events matches the left hand side of a rule but the following events deviate significantly from those predicted by the rule [20].
- **Neural Networks**: The idea of this approach is to try to predict the behavior of users. The neural network constitutes the normal profile of a user by learning a set of representative command sequences of this user. In the detection phase, it tries to predict the next command, after each executed command by a user, by taking account the N previous commands. If the real command deviates from that predicted command then an alarm is raised.

I.3.2.2 Knowledge-based intrusion detection approach

This approach consists in detecting intrusions exploiting well-known system vulnerabilities. It is based on the fact that any known attack produces a specific trace in the audit trail or in the network data. It needs *a priori* information about the attacks it is able to detect. An alarm is raised when the trace of an attack is detected in the current audit trail or network data. This approach works as follows:

1. Attacks scenarios are collected,
2. These scenarios are translated into system-dependent models (such as rules or patterns),
3. These models are recorded in a database,
4. The system-dependent audit trail or the network information are translated and compared with the models within the data base,
5. If the audit trail contains a sequence that is similar to an attack model, an alarm is raised.

This approach uses three methods:

- **Rule-based method** (expert systems): It translates attacks, we want to detect, into rules. Also known system vulnerabilities are translated into rules. The current data are compared with the defined rules; if a rule matches, an alarm is raised. The construction of these rules depends entirely on the expertise of the security officer.
- **Signature-based method**: It translates attacks into signatures. The current data is compared with these signatures and if there is a match, an alarm is sent. There are three techniques:
  - *Pattern matching*: It encodes known intrusion signatures as patterns that are matched against the audit data [25][26][27]. It attempts to match incoming events to the patterns representing attack scenarios. [28] considers each event of a scenario attack as a letter taken in an alphabet, which represents the set of the auditable events. Then

considers the audit file like a principal character string, in which it is necessary to locate the attack scenarios that are seen like sub-chain of this chain.

- *State transition analysis*: In this approach, the monitored system is represented as a state transition diagram [19]. The attacks are then represented as specific sequence of state transitions. When the data is analyzed, a new state is added. If a specific sequence of states is observed, an alarm is generated.
- *Reasoning-based model*: It is a variation on misuse intrusion detection that combines models of misuse with evidential reasoning to support conclusions about the occurence of a misuse. The system contains a database of attack scenarios, each of which comprises a sequence of behaviors making up the attack. At any given time, the system is considering a subset of these attack scenarios as likely ones under which the system might be currently under attack. According to those scenarios, a set of behaviors to check is established and an attempt is made to verify those scenarios by seeking the occurence of the given set of behaviors in the audit trail. The process is repeated several times and some scenarios become more and more probable while others are dropped. The evidential reasoning process permits updating the likelihood of occurence of attacks in the active subset [25].
- *Keystroke monitoring*: This approach uses user keystrokes to determine the occurence of an attack. The primary means is to search specific keystroke sequences that indicate an attack.

- **Genetic algorithms**: They are used in order to optimize the search of attack scenarios in audit files [28][29]. This approach, owing to its good balance between exploration/exploitation, permits to obtain in a reasonable processing time, the subset of the potentially attacks present in the audit trail [28].

I.3.3 Desirable features of an intrusion detection system

In [3][4][6][7], the following characteristics are identified as desirable for an IDS:

- **continual running** of the system without human supervision,
- **fault tolerance**: it must be able to recover when the system crashes without having to rebuild its knowledge base during the restarting phase,
- **resilience to subversion** : it must be able to monitor itself in order to detect if there is an attacker that tries to compromise it,
- **efficiency**: it must impose a low overhead on the system where it is running in order to do not slow down the monitored system,
- **observation of deviations** from normal behavior,
- **scalability** : it must be able to scale to larger systems while providing correct results,
- **adaptability**: it must be able to adapt to changes that occur on the monitored system (e.g., installation of new applications, addition of new ressouces),
- **graceful degradation of service**: when some IDS elements stop working for any reason, it should be affected as little as possible,
- **dynamic reconfiguration**: it must be able to do the modifications without restarting all the IDS.

Moreover, [22] identifies only four fundamental needs for an IDS:

- **power and convenience**: the system must use a powerful model to represent and detect attacks.
- **reusability**: it must be able to be reused in different environments with a minimum adaptation effort.
- **efficiency**: it must have an efficient detection engine to detect complex attack scenarios, preferably in real time.

- **adaptability**: it must be easily customizable in order to meet specific security policies.

I.3.4 Architecture of global intrusion detection system

A global IDS is not a single generic intrusion detection tool that can deal with all possible attacks. A global IDS, is a system composed of several individual IDS (IIDS), each of them dedicated to specific monitoring tasks in order to answer to special intrusion types. Fuctionally, a global IDS can be decomposed in three layers: correlation layer, IIDS layer and detection schemes layer.

- The *correlation layer* interacts with the administrator to alert him about potential attacks. This layer analyses reports received from other layers and performs correlations between these various reports in order to decide if a real intrusion occured.
- The *IIDS layer* is constitued of several IIDSs. Each of them implements a different algorithm with respect to what kind of detection is aimed to be addressed. In fact, it implements a specific detection scheme that represents local decisions rules. When an attack is detected, IIDSs send an alarm to the correlation layer.
- The *detection schemes* layer contains all detection schemes. Detection scheme is performed following a detection logic implemented in IIDS.

A global IDS allows to deal with the problem of the great variety of detection schemes. It permits also to distribute processing time for auditing log files to the various IIDSs.
Such global intrusion detection schemes can either be built according to a centralized or distributed architecture.

- In a *centralized system*, the function of the correlation layer is ensured by a central manager, which generally is installed in a specifically dedicated host. It communicates with IIDS to launche detection requests on IIDS and performs correlation tasks based on the different alarm reports received from IIDSs.
- In *a distributed architecture*, the correlation layer does not contain a centralized correlation module, but, it is implemented as part of each IIDS. These IIDS can communicate between each other to perform correlation tasks and can report alarms to the administrator, when an intrusion is detected.

I.3.5 Some existing intrusion detection systems

In this section, we briefly present some intrusion detection systems [7][30].

I.3.5.1 IDES and NIDES

IDES (Intrusion-Detection Expert System) was developed by SRI International. It represents the reference model for a great number of IDS. It was designed to supervise only one host and it treats only the audit data recordeded in the audit file. IDES is independent of the monitored system. Indeed, it runs on a dedicated host, connected to the system by a network. Its goal is to detect security violations in real time. IDES is based on both statistical approach and expert system [7][17][30][31][32]. Thus, it consists of two components:

- *statistical anomaly detector*: permits the detection of atypical behaviors by using statistical methods of Denning's model,
- *expert system*: detects attacks using a base of rules containing information about known attack scenarios, known system vulnerabilities and site-specific security policies [30].
-

NIDES (Next- Generation IDES) [7][30][33], is an extended version of IDES. It performs the intrusion detection on several hosts (i.e. distributed environment) using their audit trails. There is no analysis of the network traffic. It uses the same algorithms as IDES.

I.3.5.2 NADIR

NADIR (Network Anomaly System Detection) is a rule-based expert system, which was developped for the Los Alamos National Laboratory's Integrated Computing Network (ICN) [7][17][30][31][32]. This network is divided into four partitions; each partition is defined to process at a different security level [30]. Special nodes, called *service nodes*, link these partitions. NADIR uses audit records coming from these nodes to perform the analysis of network activity. The main disadvantage of NADIR is that it uses a non-standard network protocol. So, it can not be easily implemented in an heterogeneous environment.

I.3.5.3 DIDS

DIDS (Distributed Intrusion System Detection) is a project developed jointly by UC Davis, Lawrence Livermore National Laboratory, Haystack Laboratory and the US Air Force [7][30][32]. It was designed to monitor a local area network (LAN). It is a distributed host-based intrusion detection system. It is built around three components:

- a *DIDS director*: is responsible of analyzing all data received from the two other components and detecting possible attacks;
- a *LAN monitor*: it monitors all traffic on a LAN segment and reports to the DIDS director unauthorized or suspicious activities on the network ;
- a serie of *Host Monitors*: each of them monitors a single host. It collects audit data and analyze them. The relevant information is then transmitted to the DIDS director.

The centralized nature of DIDS represents a major disadvantage in the case of WANs where communications with the *DIDS Director* may congest the network.

I.3.5.4 CSM

CSM (Cooperating Security Manager) was developed at Texas A&M University. It was designed to perform intrusion detection in a distributed network environment. More specificaly, CSM aims detecting suspicious activities without the use of an established centralized director [7][32]. In fact, each CSM performs intrusion on its own system and communicates with the other CSM in order to detect cooperatively intrusive activity. This cooperation allows CSMs to handle some kinds of attacks such as *Doorknob Rattling* attack in a proactive manner, instead of reactive. CSM is composed of six elements:

- a *Local Intrusion Detection Component* (IDS): it performs intrusion detection for the local host;
- a *Security Manager* (SECMGR): it co-ordinates the distributed detection intrusion between CSMs;
- an *Intruder Handling* component (IH): its role is to take actions when an intruder is detected ;
- a *Graphical User Interface* (GUI): it permits the security administrators to interact with individual CSMs;
- a *Command Monitor* (CMNDMON): it captures commands executed by users and send them to the IDS ;
- a *TCP Communication* (TCPCOM): it permits TCP communications between CSMs.

CSM presents a disadvantage because it cannot be easily implemented to another system. In fact, the action-based intrusion detection module is system-specific.

I.3.5.5 GrIDS

GrIDS (Graph-Based Intrusion Detection System) was designed to detect large-scale automated attacks on networked systems [34]. It collects the data related to the hosts activities and the network traffic between these hosts. All this information is then gathered in graphs of activities, which represent the causal structure of network activities. The nodes of an activity graph correspond to hosts constituting the network, while the edges represent the network

activity between these hosts. During the detection phase, GrIDS analyzes the characteristics of the graphs, compares them with known patterns of intrusive activities, and if there are similarities, an alarm is raised.

I.3.5.6 AAFID

AAFID (Autonomous Agent for Intrusion Detection) deals with the problem of intrusion detection with another view [3][4][5][6]. Instead of designing a monolithic system of intrusion detection, it proposes a distributed architecture, where several small independent processes operate in a co-operative manner to perform the monitoring of the target system. Its principal advantages are: efficiency, fault tolerance, scalability, and adaptabilty. It is composed of three essential components:

- *agents*: are independently-running entities which monitor certain aspects of a host and report abnormal or suspicious behaviors to the suitable *transceiver*. On a host, there can be one or more agents. The agents do not have any authority to generate alarms and cannot communicate between them;
- *transceivers*: represent the external communications interface of a host. For each monitored host, there must be a transceiver running on it. It has two significant roles:
  - a *control role*, where it must launch and stop the agents running on its host, keep track of these agents and respond to commands sent by its monitor;
  - and a *data processing role*, where it must receive and analyze reports generated by the agents, and send the processing results either to other agents or to a monitor ;
- *monitors*: they represent the highest-level entity. They have also, like the transceivers, a control role and a data processing role. The difference between the two entities is that a monitor can control agents that are running on different hosts whereas transceivers can control only local agents. In their control role, monitors can receive instructions of other monitors and control transceivers and other monitors. In their data processing role, monitors perform high level correlations on information received from the transceivers that they control. So, they detect events that imply different hosts. The various monitors can communicate with user interface and thus provide an access point to AAFID.

I.3.6 Other systems and tools for intrusion detection

In addition to the six systems, which we have described above, others systems have been also developed. [7], [28] and [30] present a short description of various tools. Below, we cite a non-exhaustive list of some of these tools:

- ASAX [7][22][28][35],
- ComputerWATCH(Watch Computer To that Trail Analysis Tool) of ATT [7][30][31],
- DISCOVERY developed by TRW [7][30][31],
- EMERALD(Event Monitoring Enabling Response to Anomalous Live Disturbances) [7] developed by International SRI,
- GASSATA [7][28],
- HAYSTACK, developed by the Haystack laboratories on behalf of Los Alamos National Laboratory [7][22][30][31],
- HYPERVIEW [7][28],
- IDIOT [7],
- ISOA(Information Security Officier' S Assisting) [17][30][31],
- MIDAS(Multix Intrusion Detection and Alerting System) [7][17][22][30][31],
- NSM was conceived at the University of California-Davis [7][30][31],
- USTAT [17][18].

II. OVERVIEW OF INTELLIGENT AGENT TECHNOLOGY

From its nature, the network environment varies considerably. Aiming at providing a high level of security in networks, the use of appropriate and adaptive security tools seems necessary to apply in future networks and in particular in high speed networks. Intrusion detection tools are considered as the most important ones. Consequently, the components of the dorsal architecture of the network (switches, nodes, …) will be provided with complex capacities producing a certain form of "intelligence". They will be able to take various decisions with autonomy and will have adaptive behaviors to deal with potential, unpredictable and severe security attacks.
Intelligent agent technology is a growing area of research and new application development in telecommunications. Recently, the technology of multi-agent systems (MAS) knows a great success. MAS is primarily based on the concept of intelligent agents. They exhibit substantial competencies of reasoning and representation of the world in which they evolve. They also apply to reactive capacities in order to satisfy real-time constraints. Indeed, year by year, the industrial developments increase and this technology becomes maturer. The idea of applying such a new technique, coming from artificial intelligence (AI), field, appears interesting to explore. In fact, this technology offers mainly well-adapted characteristics (autonomy, adaptability…) for netwoks. Therefore, after having highlighted the main requirements of intrusion detection systems in the previous section, the intelligent agent concept can be seen as an appropriate approach to fulfill these requirements.

In this section, we present some basic concepts: definitions of the term agent, its characteristics, the various categories of agent architectures. Then, we focus on the concept of MAS since it is in major concern of the study undertaken in this work.

II.1 Basic Concepts

Distributed Artificial intelligence (DAI) was born, at the beginning of the Seventies, to find solutions to specific AI problems. Traditional AI concentrates *intelligence* within a single system. This involves some difficulties because of the need for integrating, within a same base of knowledge, expertise, competencies and knowledge of different individuals who, in reality, communicate and collaborate in the realization of a common goal [36]. The purpose of DAI is to extend the AI field in order to *distribute the intelligence* among several agents not subject to a centralized control. These agents constitute a society in which each one has a certain autonomy and where all work in complex co-operation modes, conflicts and competition to achieve a global objective.
Initially, on the basis of historical interest of the researchers in the field of DAI [Davis (1980/82), Fehling (1983), Smith (1985), Gasser (1987), Sridharan (1987)], three sub-fields were defined:

1. Distributed Problem Solving (DPS): it considers the distribution of a particular problem between various agents. This field is characterized by a set of agents (modules of knowledge, sources of knowledge...) which interact to solve a main problem [37].
2. Multi-agents systems that allow modeling the behavior of a group of entities, which are, more or less, expert according to their functionalities. These entities are called autonomous intelligent agents that coordinate their knowledge, goals, skills and plans jointly to take actions or to solve problems. The agent in a multi-agent system may be working towards a single global goal, or towards separate individual goals that interact.

3. Parallel AI (PAI): research in this field is concerned with developing parallel computer architectures, languages and algorithms for AI. The goal of PAI is to solve performance problems of AI systems. This branch, gradually, left DAI domain.

Thus, we can say that the goal of the DAI is the development of a society of intelligent agents that cooperate to solve complex problems.
Without loss of generality, the term agent is used to indicate an autonomous and intelligent entity [38] [39]. Several definitions has been given to clarify the vague concept of *intelligence*; however it must be strongly noted that there is no single, unifying definition of agenthood in the literature. Even if, there are several widely accepted concepts that characterise agent systems, we give below some principal definitions to the agent concept.

II.1.1 The Concept of agent

Until now, there is no an internationally accepted definition of an intelligent agent concept [40]. So, each individual or group working in this area seems to have his own definition.
J. Ferber [41][42] released a common minimal definition which is roughly the following one.
Definition
"An agent is viewed as a computational or physical entity situated in an environment (either real or virtual) which is able to act in the environment, to perceive and partially to represent its environment and to communicate with other agents. It is also driven by internal tendencies (goals, beliefs,..) and has an autonomous behavior which is the consequence of its perception, its representation and its interactions with the environment and with the agents".
Shoham (1993), Cohen and Levesque (1988), Wooldridge and Jennings (1995) define an agent in terms of mental state, which makes reference to the concepts of beliefs, knowledge, engagements with respect to itself and to other agents [43].
The agents are able to act and not only to reason as in the case of classical AI systems. Moreover, *communication* and *behaviors* are two very significant concepts in DAI. They express the fact, on one hand, that the agents are not pure 'arguers' but what they do is essential and on the other hand, that it is necessary to these agents to communicate in order to cooperate and coordinate their actions.
This new concept is used in different domains and possesses various meanings depending on the context of its application. However, it can be described by a set of properties including:

- **Autonomy**: is the ability of an agent to operate without direct humans intervention or other agents and to have some kind of control based on its internal state and/or external environment.
- **Socialability (or Ability to Communicate)**: is the capability of an agent to integrate itself in a large environment populated by a society of agents with which the agent has to exchange messages to achieve purposeful actions. This property is satisfied even when systems have to share their knowledge and mental attitudes (beliefs, goals, desires, etc.).
- **Proactivity**: is the ability of an agent to anticipate situations and change its course of action. It is a relevant property, which occurs in network and system management in order to avoid disastrous effects on global performance. Indeed, proactive agents are capable of exhibiting goal-direct behaviors by taking some initiative [44].
- **Reactivity**: this kind of behavior means that the agent reacts in real-time to changes that occur in its environment.
- **Adaptability**: is the ability of an agent to modify its behavior over time to fulfill its problem-solving goals.
- **Intelligence**: the term "Intelligence" means that the agent is able to exhibit a certain level of intelligence priority, ranging from predefined actions (planning) up to self learning (define new actions). It includes:

- **Reasoning:** An agent can exhibit the ability to infer and extrapolate based on current knowledge and experiences in a rational and a repoductible way.
- **Ability to Plan:** An agent can synthesize and choose between different courses of action intended to achieve its goals.
- **Learning and Adaptation:** An agent may be able to accumulate knowledge based on past experience, and consequently modify its behavior in responses to new situations

Two characteristics hold our attention: adaptability and intelligence. The agent is regarded as an adaptive entity able to control its aptitudes (communicative, behavioral...) according to the environment in which it evolves [43]. This characteristic is fundamental, in particular in the fields where the environment is dynamic; as it is the case of the network environments. As regards to the intelligence, the task is more difficult because a brief definition of this term remains a source of great conflict debates. From AI's point of view, the agents are viewed as intelligent if they gather emotional/mental capacities and rely on concepts such as: knowledge, beliefs, intentions and obligations [45][46]. Nevertheless, we adhere to another definition of intelligent agent that postulates that the agent has its own field of expertise and is provided with a capacity of interaction with other agents.

Other attributes of agents:

- **Mobility:** Agents can have the ability to move from one location to another while preserving their internal state. However the mobility property should not be confused with mobile agent frameworks that just provide transport mechanisms that let computation hop from site to site over the network [47].
- **Security, safety, veracity:** It constitutes the most important drawback of today's agent paradigm.
  - **Security:** Security is important so that one can be confident while interaction with an agent, having therefore guarantees that the agent has not been corrupted by any virus, false beliefs and nonsense knowledge.
  - **Safety:** Safety is like security but at an intentional level, therefore, it is much more difficult to specify and verify. Safety means that agents must behave according to behaviour specifications. For example, agents are not permitted to create undesired goals, perform forbidden tasks or generate useless information.
  - **Veracity:** In a distributed environment populated by agents, most of the high-level information exchange will be done through the agents. Veracity does not refer only to the likelihood that an agent would be lying, but to the assumption that an agent will not knowingly communicate false information

II.1.2 Definition of MAS

"Multi-agent systems, as a sub-domain of DAI, are viewed as computational systems in which several autonomous and intelligent agents interact and work together to perform a set of tasks and to satisfy a set of goals" [48].

The interest of a MAS resides, indeed, in the concept of collective action and its capacity to articulate the individual with the collective, via the cognitive structure of agents [49].

II.2 Classification of agents

The intelligence level of agents varies accordingly to three coexistent tendencies in the DAI field; this leads to the first classification of MAS. This taxonomy reflects the existence of three schools: *cognitive*, *reactive* and *hybrid*.

II.2.1 " Cognitive " School

It is extensively represented in DAI because it was originally used to permit communication between classical expert systems. These systems include a small number of complex agents comparable to expert systems. A cognitive agent is able to find a solution for a complex problem while communicating with other agents and interacting with its knowledge base. Its main features include a high reasoning capacity, data processing, perception, learning, control, communication and expertise per activity domain. Each agent can be broken up in the following way [42][43][50]:

- A complex behavior: It is a behavior that corresponds to various treatments and actions. It is defined either by numerical algorithms, symbolic systems, production rules, logical expressions or more simply by a truth table. The agent behavior represents its expertise and is described as intentional goals and explicit plans to achieve these goals.
- A local memory: It allows saving its observations on the environment and information related to its expertise field.
- Knowledge of the environment: This is materialized by information about its environment and a list of agents with which it can communicate.
- Mechanisms of communication: It gathers communication protocols between the agent and its collaborators and the way in which it will take into account the received messages.

These agents are qualified as agents with strong granularity (Corse grain). The analogies are primarily social, co-operation problems are similar to those of small groups of individuals who must coordinate and negociate to solve their conflicts [42].

The highly accepted model architecture for cognitive agent is called the BDI (Beliefs Desires Intentions) model [51][52]. The basic idea of the BDI approach is to describe the internal processing state of an agent by means of a set of mental atttitudes and to define a control architecture which rationally selects its course of action, based on a representation of the mental attitudes. The mental categories are beliefs, desires and intentions. In more practical BDI-approaches such as [52][53], these have been extended with further notions such as goals, plans and commitments. Informally, these concepts are described as follows:

- **Beliefs**: express exceptations of an agent about the current state of the environment and about the likelihood of a course of action achieving certain effects. Beliefs are modeled using possible-environments semantics, where a set of possible environments is associated with each situation, denoting the environment the agent believes to be possible [54][55][56].
- **Desires**: are abstract notions that specify preferences about future environment states or courses of action. An important feature of desire is that an agent is allowed to have inconsistent desires, and that it does not have to believe that its desires are achievable[57].
- **Goals**: the weak definition of desires as above enforces an additional step of selecting a consistent subset of desires that an agent might pursue, i.e., goals that denote the option the agent currently has. However, there is not yet any commitment to specific courses of action. The notion of commitment describes the transition from goals to intentions. Additionally, it is often required that the agent believes its goals to be achievable.
- **Intentions**: since an agent is resource-bounded, it cannot pursue all its goals or options at once. Even if the set of goals is consistent, it is often necessary to select a certain goal (or a set of goals) to commit to. This process is called the construction of intention. Thus, a set of selected goals and their state of processing describe the current intentions of an agent.
- **Plans**: intentions are partial plans of action that the agent is committed to execute to achieve its goals. Thus, intentions are structured into larger plans, as the agent's intentions are part of the plans it has currently adopted.

- **Commitments**: are classes of goals that an agent has selected to achieve unconditionnaly; thus they are not agent desires or preferences.
- **Capabilities**: represent the agent ability (skills) to develop determined tasks by using appropriate sensors and effectors [47].
- **Preferences**: are either an alternative mental attitude to desire or a complementary one.

II.2.2 " Reactive " School

In reactive MAS, it is not necessary that all agents are intelligent to reach a global intelligent behavior. A reactive agent reacts quickly in the case of a simple problem that does not require complex reasoning. Thereby, system intelligence emerges from interactions between a great number of this type of agents. A reactive agent is characterized by the following properties:

- No explicit representation of the environment and other agents,
- No saving of the events produced in the past,
- Simplistic Behavior: A reactive agent does not deliberate to decide actions to undertake. On the other hand, it has stimulus / response capacities to react by simple reflexes to the changes of its environment,
- Communication via the environment: It has only one communication protocol and a communication language. Communication is generally unintentional, but the presence of agents, stimuli which they produce and traces that they leave in their environment, are signals which can be interpreted by other agents. Here, the agents have a low granularity (fine grain).

II.2.3 " Hybrid " School

Hybrid agents combine reactive and cognitive capacities, which enable them to adapt their behavior in real-time to the environment evolution.
Consequently, an hybrid agent owns:

- some reflex (reactive attitude) to solve repetitive problems,
- some reasoning capacities (cognitive attitude) to deal with complex system situations.

Most of proposed architectures rely on a modular decomposition of the agent in a series of modules: cognitive, reactive and a module of supervision which deals with the management of the coherence of the agent [57]. Layering is powerful means for structuring functionalities and control. Thus, it is a valuable tool for system design supporting several desired properties such as reactivity, reasoning, cooperation and adaptability.

Discussion

The existing multi-agent systems are in general classified in two main categories: cognitive systems and reactive systems. The reactive systems exhibit simplistic behaviors. They are based on event-reaction mechanisms to solve quickly problems of average complexity; for this reason, they do not require a complex reasoning faculty. So, they have neither an explicit knowledge of the environment nor action plans enabling them to anticipate some steps. Reactive agents taken individually are not powerful. On the other hand, their force as well as the intelligence of the global system arises from the interactions within the community of agents able to adapt to their environment.
Contrarily to reactive systems, cognitive systems are more sophisticated thanks to their reasoning capacity. Hence, they are more intelligent capable of solving complex problems. They are mainly based on knowledge and interactions with other agents to control an explicit global view of the environment. Capacity of anticipation and planning, absent in the reactive case, allow agents to optimize their behavior and to carry out only truly necessary actions.

In hybrid systems, we apply a category of agents that own complex cognitive capacities and simple reactive aptitudes. This kind of system is more adapted to complex systems where the environment is very dynamic.
Obviously, following the majority of taxonomies, the one we mention above should not be taken as a fixed standard but rather as a reference mark.

II.3 MAS Models

MAS are modeled according to two criteria [50]:

1. Type of communication between agents
The interactions can take place according to various modes:
- Information sharing: This architecture primarily relates to evolved agents as cognitive agents. All these agents work on a space that includes all the elements needed to the resolution of a problem. Blackboard systems are an example.
- Sending of messages: This method is used indifferently by cognitive or reactive agents. It consists in exchanging messages in an asynchronous way. Therefore, we notice a distribution of knowledge, partial results and methods used to lead to a result. Among these systems, we distinguish those based on actors.

2. Mode of distribution of knowledge and tasks

Three approaches are distinguished:
- Hierarchical modular approach: This type of architecture is founded on the definition of a number of modules that must cooperate. Each module is designed as an expert system having its own base of knowledge and communicates with other modules to achieve its function.
- Expertise in competition: Each agent has a base of knowledge that reflects a specific point of view of the problem to be solved. Conflicts should then be managed. These latter systems are called multi-expert systems.
- Simulation of universe: We define a population of agents with behavior's rules that lead the system to solve a problem. The structure of the expert system corresponds to a self-monitored system.

Of course, we can find all the combinations and the possible intermediaries between these various models.

II.4 Agent languages

Within the world of agent's technology, we must choose three kinds of languages: programming languages, communication languages and knowledge representation languages.

1. Programming languages: allow the development of agents accordingly to the selected architecture (reactive, cognitive or hybrid). Some of these languages satisfy AOP (Agent Oriented Programming) principles such as AGENT0 and LALO. Others concern OOP (Object Oriented Programming) such those proposed by ABE from IBM for C++ or Java, ILOG,…

2. Communication languages: provide agent interactions. KQML (Knowledge Query Manipulation Language) is intended to be a high-level language to be used by knowledge-based systems to share knowledge at run time.

3. Knowledge representation languages: are used by agents to represent their internal state and their knowledge about the environment. There are some specific languages which are used to represent the shared knowledge. KIF (Knowledge Interchange Format) is one of the proposed languages; it is a computer oriented language for the interchange of knowledge among disparate programs (agents written by different programmers at different times in different languages). When an agent needs to communicate with another agent, it maps its internal data structures into KIF.

III. MAIDA: A NEW ARCHITECTURE FOR INTRUSION DETECTION

We have highlighted in section I, the main requirements for intrusion detection. So, in our architecture, we propose to add appropriate functionalities to make network entities more autonomous by performing local intrusion detection tasks. The key characteristics of our architecture include autonomy, adaptability, efficiency and distribution to make the network intrusion detection more flexible and less costly in term of maintenance. In our proposed approach, we define a new architecture, called MAIDA, which supports security management activities. It is based on a multi-agent system architecture (see Figure 1). It is viewed as a collection of autonomous and intelligent agents located in specific network entities named intrusion detection hosts (IDH). These agents cooperate and communicate in order to perform intrusion detection tasks efficiently and achieve consequently better performance.
In fact, by giving more autonomy to agents in the control of the overall intrusion detection, the task of administration becomes easier. Security officers do not have to concern about all the security problems. They interact with the agent using security policies. Security policies tell the agents which behavior they should exhibit when attacks occur. Hence, communications between agents permit to collect information. This information permits the agents to identify attacks that can not be detected if it is not shared. Giving more autonomy to the agent permits the system to react in real-time to attacks and to take necessary actions to avoid severe consequences of attacks.
This section describes the functional architecture of the proposed multi-agent system, the agent architecture that constitutes the MAS and the agent architecture that we have used to develop this multi-agent system.

III.1 The MAIDA Functional Architecture

In the functional architecture of MAIDA (see Figure 1), we distinguish two types of agents: *Manager agent and Slave agent*.

- The *Manager Agent* (MA) manages the global security of a network. It manages the security policies specified by the security officer. Its role is to manage and control *Slave Agents* (SA). These agents report pertinent analysis to the MA. The role of the latter is then to perform another analysis on suspicious events in order to confirm or not the detection of an attack. It can also ask for another data processing, delegate then new monitoring tasks or specify new attacks to detect to the SAs. The MA is also responsible of distributing the hosts to each SA.
- The *Slave agent* (SA) manages the security of a domain, which is constituted by a set of hosts. It has specific functions as analyzing monitored events and detecting specific behaviors. In fact, the *manager agent* specifies to the *slave agent* the goals that must be reached (detecting specific intrusive activities, monitoring specific hosts or tasks, etc.).

In this hierarchical multi-agent model, the *manager agent* has the ability to control specified agents and to analyze data, whereas, the *slave agents* perform intrusion detection tasks for specified domains. MA interacts with SAs by sending goals, delegating specific monitoring/detection tasks and receiving pertinent reports and alarms. SAs communicate and exchange their knowledge and analysis for detecting intrusive activities in a co-operative manner.

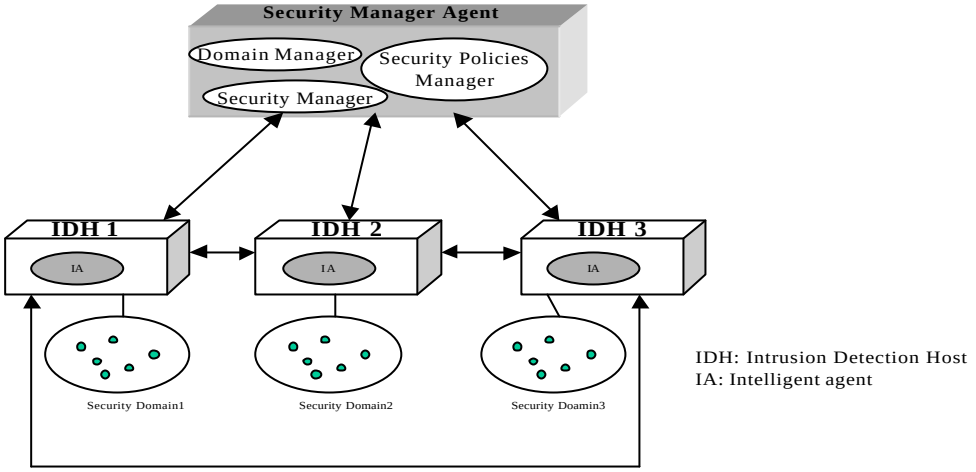

Figure 1: MAIDA Functional Architecture

Which agent model should be adopted? Will be reactive, cognitive or hybrid?

The reactive agent is straightforward to support all the complex functionalities to carry out adaptive security control. Indeed, the latter must handle information, which reflects the current security state of the network (security-relevant events, security attacks to detect, security policie, etc.), must keep trace of the last experiments and must know about the global state of the system. Moreover, they must be able to react quickly to events that indicate an abnormal state of the system such as a great congestion of the network due to denial of service attacks. In order to offer an acceptable security state of the system, the hybrid model is the most appropriate. Thanks to the combination of reactive and cognitive characteristics, it will be flexible, effective and extensible.

For the multi-agent architecture presented in this paper, the hybrid model is adopted for each agent.

III.2. Hybrid Agent Model

In this section, we describe the architecture of our hybrid agent (see Figure 2).

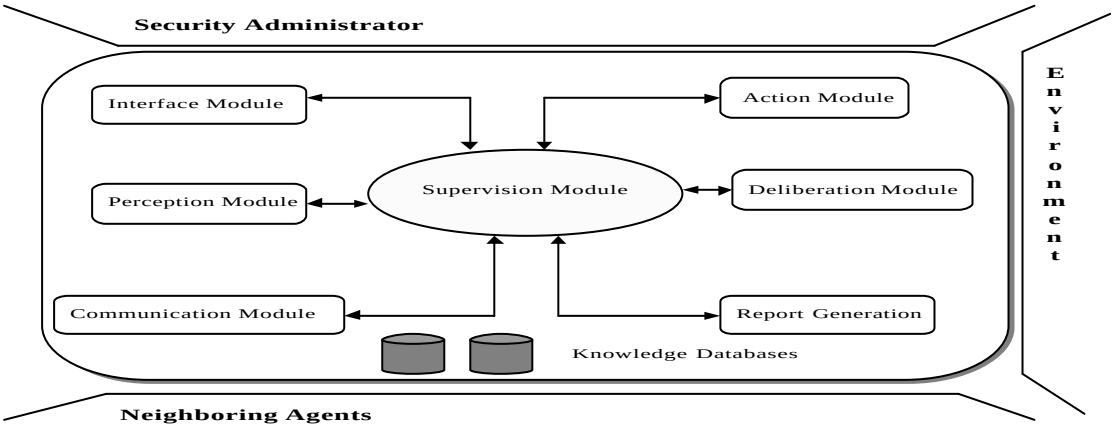

Figure 2: Hybrid agent functional model

Our hybrid agent reference model is composed of seven modules: perception, deliberation, communication, action, interface and report. The supervisor entity coordinates tasks of the different modules.

An automaton founded on the concepts of state, transition and action (Figure 2) carries out the implementation of each agent. Each module is represented by a state and each transition between the states is activated following the arrival from an event. We observe clearly that the supervision module is the central entity of the automaton. When an event arrives, it activates the transition and put the agent in the suitable state.

- A **perception module**: It gathers all security-relevant events produced in the agent environment. It is used to scan the environment of the agent to react in real-time to the asynchronous events. The new perceived data are sent automatically to the reasoning module under the authorization of the supervision module. These data can concern unauthorized access, authentification failure, access in out of hours activity, etc.
- A **communication module**: It allows agents to communicate their analysis, decisions and knowledge. It fulfills two functions: reception of messages coming from the other agents and sending of messages to the concerned agents. This module allows agents to communicate their decisions and their experiment. It permits an agent to communicate pertinent information and analysis conclusions to inform others agents about a suspicious behavior or intrusive activities. In the case of the IP spoofing attack, the agent that detects a suspicious source address should warn others agents to warn of a potentiel intruder. These informed agents could then monitor all packets coming from this source address in order to detect a potentiel attack.
- An **action module**: It takes appropriate actions when an intruder is detected.
- A **report generation**: It establishes reports on detected attacks to be sent to the security administrator.
- A **deliberation module**: It enables agent intelligence and autonomy. The hybrid agent should be able to reason and extrapolate by relying on built knowledge and experience in a rational way. Decisions of the agent depend on the security environment status, the neighboring system evolution and its mental attitudes.
  Thanks to this module, the characteristic of autonomy is assured. The agent is rationally based on its base of knowledge and its experiment to find the adapted answers. In the context of security management it uses the BDI architecture in order to perform its intrusion detection tasks. It must uses its beliefs resulting from the data provided by the perception module for reaching its specified goals.
- An **interface module**: interacts with the security administrator receiving administrator requests/specifications, delivering reports, sending alarms when an attack is detected and asking for additional information or confirmation when necessary. For example, the security officer can specify new attacks to detect or ask for the behavior of specific users.
- A **supervision module**: coordinates interactions between the different modules using a finite state automaton. The modules of perception, communication and reasoning constitute the behavior level of the agent whereas the module of supervision represents the meta-behavior level. It is an entity that coordinates the operation of all the modules. The implementation of this module in the entities of MAIDA is based on the use of ATN (Automated Transistion Network).

IV. MAIDA PLATFORM DESIGN WITH DIMA ARCHITECTURE

In this section, we describe the platform that we choose to use to develop our multi-agent system architecture MAIDA and we give some details about its implementation aspects.

IV.1 DIMA

To concretize the various research disciplines in the multi-agent area, several agent architectures have been proposed addressing different key features an agent should have.

Most of these architectures describe the functional components of an agent and how these components work together. Some of these architectures are described in [58]. To implement our platform, we have opted to use a generic, modular and open architecture called DIMA. In attempt to define a modular and generic architecture, which owns the main properties of an agent [44], DIMA proposes the extension of the single behavior of an active object into a set of behaviors [43]. This generic platform has been realized at LIP6 laboratory of the University of Paris VI [43][59]. The technical environment combines object-oriented programming and production rules. An agent is considered as a pro-active entity. It does not simply act in response to the received messages from other agents. For example, it interacts with its environment and deliberates to determine the most appropriate action. The Smalltalk-80 language has been used in the first version of DIMA, applying several existing frameworks such as Smalltalk discrete event simulation package, RPC-Talk routines, Actalk and Neopus [43] but currently it exists a new java version.
Aiming at investigating the feasibility of multi-agent approach, we restrict ourseleves to use a simplest DIMA environment that is minimal but sufficient. It includes the agent architecture and the discrete-event simulator framework (DES). In the following section, we describe these two components [43].

IV.1.1 Agent's model

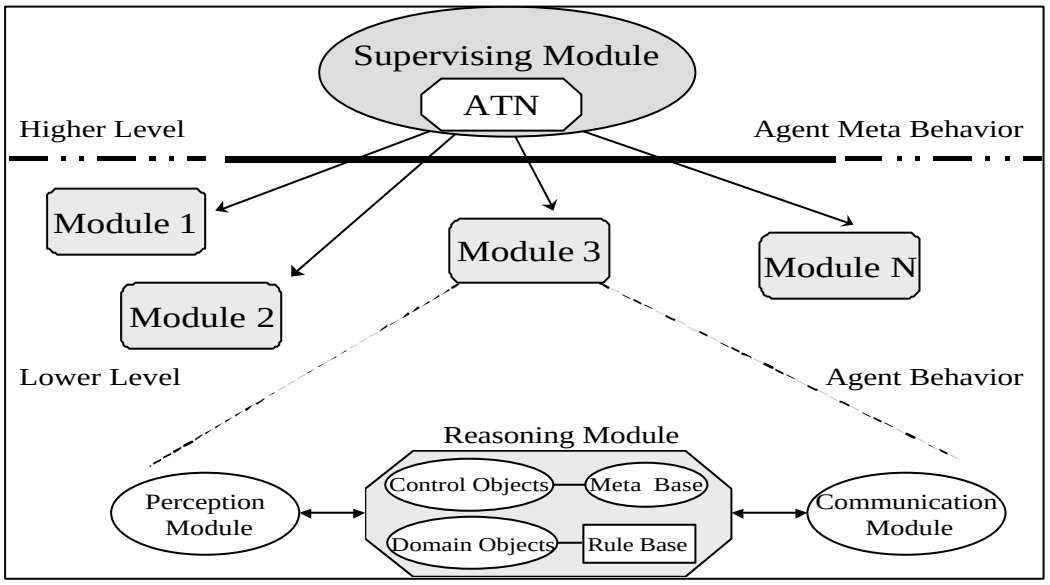

Figure 3: An Hybrid agent architecture

The DIMA architecture (see Figure 3) relies on two layers: a lower layer and a higher layer. The lower layer is made up of interactive modules, which represent the different concurrent agent behaviors such as communicating, deliberating and perceiving. They provide the agents with some properties described in [54] such as autonomy and cooperation. For example, the communication module manages the interaction between the agent and some other agents of the system. Therefore, it is very important to make the agent cooperative. The higher layer is made up of a supervision behavior representing the agent meta-behavior. This meta-behavior gives each agent the ability to reason about its own behavior.

IV.1.2 Agent Behaviors

To model intrusion detection, agents need to combine cognitive abilities (knowledge-based) to reason about complex situations, and reactive abilities (stimulus-response). So, an agent may have two kinds of behaviors: *reactive* and *cognitive* behaviors.

In this section, we give three examples of modules: the *perception module* (procedural behavior), the *deliberation module* (knowledge-based behavior) and the *communication module* (which can be either procedural or knowledge-based).

- The **perception module** manages the interactions between the agent and its environment by scaning all occured events. For example an agent percepts a list of suspicious events.
- The **deliberation module** represents beliefs, intentions and knowledge of the agent. It is responsible 1) for generating adequate responses to the messages transmitted by the *communication module*, or to the changes detected by the *perception module*, and 2) for achieving the agent goal(s). This goal can be the detection of a specific security attack.
- The **communication module** manages the interactions between the agent and the other agents of its group(s), no matter what machine they are running on. An agent may need some other information to refine its analysis. In this case, it asks other agents to provide the necessary information.

These three modules are suitable to our application domain. In fact, agents related to intrusion detection often own a deliberative behavior, and a communication behavior and/or a perception behavior. Moreover, the use of a modular approach facilitates the integration of new modules such as a learning module, an action module and a report generation module.

IV.1.3 Agent Meta-Behavior

Many systems have emphasized the need for explicit and separate representation of control or the reflexive aspect of meta-level architectures. Following this tradition of explicit and separate representation of control, DIMA proposes a meta-behavior in its agent architecture. This meta-behavior gives each agent the ability to make appropriate decisions about control or to adapt its behaviors over time to new circumstances. It provides the agent with a self-control mechanism to dynamically schedule its behaviors in accordance to its internal state and its world state. It relies on two fundamental notions: states and transitions, which naturally build up an Augmented Transition Network (ATN). Figure 4 gives an example of ATN that may describe the detection of specific security-relevant events. States represent decision points and are used to choose the next transition beyond the associated transitions. It represents a step of the global scheduling of the agent and links an input state with an output state. The conditions of transition test the occurrence of an asynchronous event (urgent message reception, new data,...). These asynchronous events often involve modifications on data. The actions of transition allow the management of the first layer behaviors (activate deliberation, terminate deliberation, activate communication, activate perception,...). The meta-behavior has to choose which behavior it should activate.

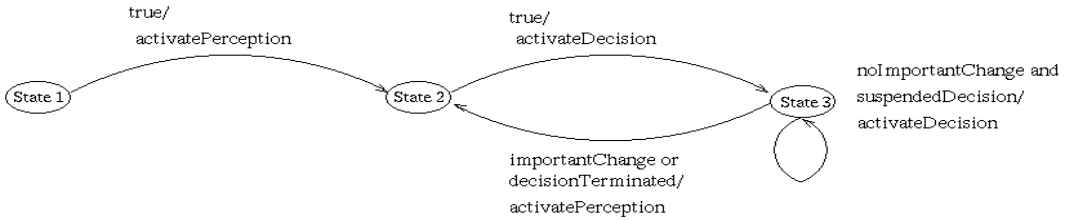

Figure 4: An example of a security agent ATN

At each state, the automaton-based meta-behavior evaluates the conditions of transition (representing new events) to select the most appropriate behavior. When these conditions are verified, the actions of the associated transition are executed and then the agent's state is modified. Moreover, the meta-behavior may evaluate and change its goal.

The meta-behavior represents the agent self-control mechanism. This self-control provides the agent with some kind of control over its behaviors and its internal state. It defines the agent pro-activity which is not restricted to message reception/sending. Therefore, it makes the

agent autonomous by allowing it to operate without direct intervention of humans or other agents.

IV.2 MAIDA architecture implementation

The hybrid agent model, as described in section III.2, is selected to be implemented in each intelligent agent of our MAIDA architecture. So, a set of such hybrid agents is installed in specific IDHs.

As depicted in Figure 6, deliberation, perception and communication modules are performed respectively by the deliberation, perception and communication modules of DIMA. The interface module, action module and report generation module are three new modules, defining three new behaviors, that must be integrated in the DIMA platform, in order to be implemented.

In our agent architecture, an ATN is associated to each module. In the ATN, we distinguish two main states: an initialization state 'INIT' and a supervision state 'WAIT' that manages the processing of each module. To illustrate ATN processing, we provide in figure 6 an ATN for a SA dedicated to the detection of both doorknob rattling and IP spoofing attacks.

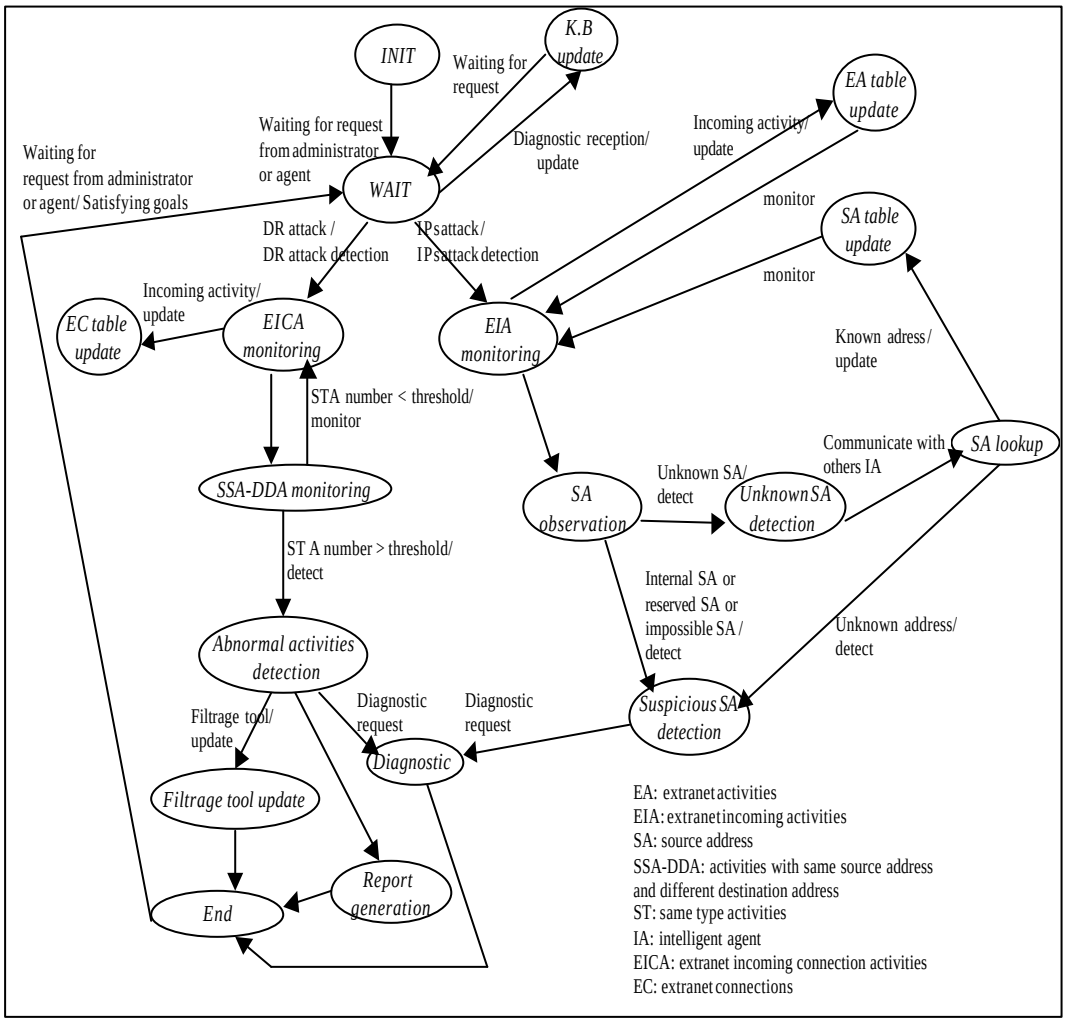

Figure 6: ATN of a slave agent

V. Intelligent Agent approach for Intrusion Detection

During these last years, computer systems and networks do not cease evolving, particularly in terms of number of users and offered services that are being more sophisticated. Systems and networks become then more complex, which make them more vulnerable to various kinds of complex security attacks. Consequently, security management of these systems and networks, particularly intrusion detection, require to be realized in a more intelligent and sophisticated way. Future networks are therefore evolving towards a new generation of intelligent networks. The introduction of a multi-agent system can be proposed as a means of modeling and implementing intelligent and suitable intrusion detection.

Until now, components within networks related to security management had very simplistic behaviors via central entity (manager, director,….) and distributed entities among monitored hosts as in NIDES system or monitoring hosts like in DIDS system. For example, the overall task of a central manager consists in communicating with monitoring entities in order to collect and analyze their relevant report information. Based on this information, it determines whether intrusive activities are or have been taking place. In the case of detection of anomalies, it sends alarms and/or reports to the security officer. Besides, each of monitoring entities supervises locally their hosts. Hence, they can additionally communicate with other monitoring entities and exchange information about users when they move through the network. All these entities do not have knowledge on the neighboring hosts and systems. They delegate the task of a global management to being realized at a higher level. Obviously, an evident consequence results in increasing the response time of the system and probably flooding the network.

The described IDS, are not easily adaptable to an increasing complexity of interactions and user behaviors. These systems have rather static configuration. The complexity is enhanced particularly by a recent aspect, which is the mobility of users. On one hand, the nomadic characteristics of users and the corresponding needs in terms of network resources are unpredictable. On the other hand, it is obvious that security problems such as new vulnerabilities and new kind of intrusions that must be considered. Indeed, since a user, which is known in a static network, moves, the knowledge related to its activities and behaviors becomes different and varies continuously. This results in the necessity to access the new knowledge. A static classical intrusion detection system can not deal with new mobility aspects. A multi-agent system appears then more adapted and should offer many possibilities.

In order to adapt to applications requirements and non-predictive complex evolution of both network environments and security attacks, network entities must be provided with more complex functionalities. These components must include complex capacities as well as perception of internal and external environments, memorizing of relevant events produced in the past, learning, analysis of collected information, planning of steps and actions, exhibiting behavior autonomy, adaptability, etc. Moreover, in order to reach some intrusion detection goals, communication and co-operation is necessary. It is clear that all these functionalities must be carried out in a coherent way aiming at solving complex security problems to optimize intrusion detection tasks. The proposal of a new solution based on MAS techniques seems so promising and appropriate because of the similarity between intelligent agent features and network entities expected characteristics.

In this context, many works have been carried out and proved that an intelligent approach is well adapted to manage complex problems, particularly in the area of network management [47][48]. In our case, we propose to extend intelligent agent applications to security management domain, especially intrusion detection.

In this section, we will try to prove that the both intelligent and distributed nature of MAS can provide the required features.

V.1 Evolution of Security Requirements

Intrusion detection systems aim to detect security attacks and traditionally use methods based on expert system models, statistical model, neural networks, etc. These IDS are generally developed for well-defined networks and systems and are not adapted to dynamical environments. In fact, the parameters and content of used models are pre-specified. Then if an existing attack scenario changes or a new attack arises, it is difficult and complex to modify the IDS. Moreover, globally, the architecture of many existing IDS is based on a monolithic module. In this architecture, data analysis, which is collected by a single host or distributed hosts, is performed by a single module [4]. Without loss of generality, this monolithic approach present some disadvantages:

- It is a *single point failure* because if the host on which it is running is attacked, the security of the network is compromised.
- The *scalability* is limited because the analysis is performed in one single.

In some others systems such as CSM, data analysis can be performed without a central authority, which resolve some of the problems implied by the monolithic architecture. However, it still some others shortcoming that can be summarized as follows:

- IDS can not be *upgraded easily*. In fact, it is difficult to readapt existing systems with newer and better techniques of detection as they become available. To add new capabilities, a restart is needed.
- IDS are not *easily adaptable* to changes that occur in system and user behavior that varies considerably in time. These changes can be new resources or applications being added.

The dynamics and evolution of security requirements, due to the evolution of networks and systems, increase the need of having flexible and adaptable IDS in order to add new capabilities and reconfigure them easily. An IDS based on intelligent agent technology seems appropriate to meet the new security needs such as new intrusions to detect and new security policy to apply. In fact, from their flexible and adaptive nature and they ability of learning, they should fulfill these requirements:

- **Adaptability**: Security policies which are applied in an organization may change. So, the administrator should change or add new ones in order to modify and readapt intrusion detection tasks and monitoring functions. When a new security policy is added or modified, a set of new goals are derived and sent to the concerned intelligent agents. These agents will be capable then to reach these new goals.
- **Flexibility**: The monitored network can vary and change over time. In fact new services can be provided and new applications and resources can be added. So, the IDS to be developed must take into account these variations without having to restart the IDS. In fact, in an intelligent agent approach, the administrator via a set of new goals can specify these modifications to the intelligent agents. This does not need that the intelligent agents must be restarted because their behavior change according to the specified goals that must be reached.
- **Learning**: It is a fundamental characteristic to detect new attacks. In fact, attack scenarios are evolving continuously. So, it is very important to be able to learn new attacks in order to detect them when they happen. In our proposed approach, a new attack can be learned in two ways. The simplest one is done by the administrator that specifies in a set of goals how to detect new attacks. The IA can therefore modify its knowledge base. The second one, is the ability that has an agent to collect the knowledge based on previous experiences and consequently modify its behavior in response to new situations. In fact, when an intelligent agent monitors a list of suspicious events that do not match known attack scenario, it can detect that is an intrusive behavior using previous cases and

experiences and appropriate models (statistical,..) and techniques (genetic algorithms, neural networks,...). It can also communicate and co-operate with other agents to obtain information about this scenario. If it detect that is an intrusive behavior, it can add the new attack scenario in its knowledge base.

V.2 Intelligent Agent Properties and Efficient Intrusion Detection

In this section, we will discuss why certain characteristics are important for an efficient intrusion detection.

- **Distribution:** A common element in many network attacks is that a single user will often attempt to intrude upon multiple resources throughout a network [1]. These kinds of attacks are characterized by abnormal behaviors at different network elements (hosts, file servers, routers, etc.). For example, an intruder can try to attack the file server, to make a TCP service unavailable on a specific host, or to flood the network at a specific node. In each of the various network elements, the security-relevant events characterizing abnormal activities can be observed. Detecting attacks by a single system, running on a single component, is too complicated. So, it will be easier to distribute monitoring and processing tasks among a number of entities that can monitor the network at different points. This important aspect is provided by most existing approaches realized to deal with security attacks (NIDES, DIDS, CSM,.). For example, in NIDES and DIDS, data collection is assured by several entities but the analysis is performed by a centralized director, which communicates with host monitors. However, in CSM, there is no established central director but individual managers are responsible of making local intrusion detection.
- **Autonomy and Delegation:** In order to simplify intrusion detection, distribution of detection tasks among different entities is needed. However, this is not sufficient because distributed data collection can also cause congestion problems with excessive data traffic in the network between the various entities. Moreover, an attack that arises at a specific network element (host, server, etc.), is characterized by a set of security-relevant events that can be observed in this element. For example, an intruder that performs a "TCP Scans" attack, sends successive "Telnets" on each TCP ports of a specific host. It will be more judicious to let the entity monitoring this host and observing the sequence of "Telnets" sent to the various TCP ports, to detect any abnormal behavior. Thus, management entities must be autonomous in order to perform local analysis and detect intrusive behaviors that happen on the monitored hosts.
  The CSM and DIDS approaches have shown the necessity to use ***autonomous*** entities. They propose different techniques in the sense that the final decision in the DIDS system is taken by a centralized manager, whereas in the CSM some decision can be directly taken by the entity itself.
  Another aspect, related to autonomy, that must be considered is delegation function. In fact, the high level of dynamics in computer networks requires modifying, at any time, security management functions in order to adapt them to changes that occur in monitored networks. The model based on sharing delegation functionality among various management entities, allows to fulfill this requirement. Indeed, delegated management tasks, which are performed locally, can be modified dynamically, in a flexible way, at any time and in any place. For example, if an administrator wants to detect a new attack, he should send new monitoring and processing tasks to the various autonomous entities. Delegation of intrusion detection tasks among management entities seems necessary. This critical feature is not found in existing IDS.
- **Communication and co-operation**: Many security attacks are increasingly composed of co-ordinated attacks that are not easy to detect. In fact, the complexity of security attacks

makes them more difficult to detect by an individual entity. Because each entity has a local restricted view of the network, co-ordinated attacks that happen at different points of network can not be detected by the autonomous entity. For example, if an intruder performs a doorknob rattling attack, it tries few "logins" on each host. The autonomous entities, which monitor individually these hosts, can not detect this attack because of the low level of repeated "logins". In fact, the monitoring of local behavior, such as repeated "logins" is not sufficient in this case. In this kind of attacks, it will be necessary to correlate various analysis made, at different points of the network, by the various autonomous entities. So, a communication of these analyses and a co-operation between the various intrusion detection entities is needed in order to detect co-operatively co-ordinated intrusive behaviors.

The CSM system has shown the necessity of security manager co-operation in order to detect security attacks that can not be detected by individual CSMs. Each CSM detects local intrusions and cooperates with other CSMs by exchanging information in order to detect cooperatively intrusive activities.

- **Reactivity:** The aim of efficient intrusion detection is to react against an attack before serious damages can be caused. For example, in the case of an ICMP flooding attack, if an IDS waits until traffic congestion in the network is observed, it will be too late. So, it will be important to detect such an attack before it floods the network. The efficiency of an IDS can, then, be measured according to its swiftness in detecting an attack before it can cause luckless damages.

The six features (distribution, autonomy, delegation, communication, co-operation and reactivity) described above are considered as main requirements for detecting attacks more efficiently and responding to intrusions by reducing the detection time. Looking at these characteristics and the different properties of intelligent agents, it seems that an approach based on intelligent agents should be appropriate for detecting security attacks, particularly network attacks. CSM provides five features (distribution, autonomy, communication, co-operation and reactivity) except delegation function that is not provided. Moreover, CSM managers can not easily adapt their intrusion detection tasks to changes that can occur in networks, such as new security policies to apply. In addition, CSM managers do not have the ability of learning new attacks.

In order to illustrate how agent properties and proposed agent hybrid model is adapted for intrusion detection we will show how a doorknob rattling attack can be efficiently detected by an IDS based on intelligent agent technology.

In a doorknob rattling attack, the aim of an intruder is to gain access to insufficiently protected hosts by trying common user id/password (such as guest account) combinations on several hosts. The intruder tries only few logins on each host in order to hide its intrusive activities. In fact, this attack can not be detected by an individual IDS, which has not a lower "threshold of detection". This means if an individual IDS observe two failed logins, it can not consider these events as abnormal. For identifying intrusive behaviors, the number of failed logins must be higher. So a correlation between all failed logins that arise on the various hosts must be done. However, if each host monitoring entity reports each failed login to a centralized manager may swamp portions of the network, particularly in the case of large networks. We propose a solution with several intelligent entities that perform local intrusion detection tasks and co-operate for detecting this kind of attack. These several entities can be viewed as intelligent agents that are distributed throughout the network.

We consider two class of entities: a manager agent (MA) and a set of slave agents (SA). In order to detect this attack, we define two goals to reach G1 and G2.

- G1: < control failed logins rate for each userId;
  If threshold exceeds 4 *then* inform other agents;
  If another failed login is observed *then* alert MA>
- G2: < control failed logins rate for a suspected userId;
  If a failed login is observed *then* alert MA>

The MA delegates the goal G1 to all SAs. Each SA tries then to reach this goal by monitoring failed login attempts. If one SA detects that the threshold of logins for an userId is reached, it sends to MA and other SAs a suspicion belief B_{sus} <suspicionLevel, userId, numberOfFailedLogins>. B_{sus} concerns the suspected user. According to the goal G1, it also delegates to other agents a new goal G2, which is derived from G1. Therefore, SAs observe connection activities of the suspected user to reach the goal G2. If another failed login is observed by a specific SA, this latter alerts the MA, which sends an alarm to the security officer. The two goals G1 and G2 are different in the sense that G1 concerns monitoring of failed logins for all userId and G2 concerns monitoring of failed login of only one specific userId that is suspected by one SA.

VI. CONCLUSION

In this paper, we focus on one critical issue in security management that is intrusion detection. Intrusion detection requirements and concepts are reviewed. Some existing systems are described. Their advantages and limitations are illustrated. Drawbacks of existing intrusion detection systems involve the necessity of designing a new generation of self-adaptive systems. In fact, mainly, self-control, flexibility, adaptability, autonomy and distribution are the main features to be addressed in a suitable architecture that fulfills these requirements. In this context, we propose a new approach based on intelligent agent technique that reveals as an appropriate candidate to make a balance between security requirements, system flexibility and adaptability for intrusion detection.

The introduction of a multi-agent system in an intrusion detection system is proposed as a means of implementation of adaptive and autonomous decision features embedded in agents distributed over intrusion detection related entities. Thus, a new multi-agent architecture is outlined. In addition, a functional structure for hybrid agent is presented. A detailed description of the selected platform to implement our architecture is given.

The objective of demonstrating the feasibility of applying intelligent agents within DIMA platform has been illustrated through two specific security attacks: doorknob rattling and IP spoofing.

For further work, we intend to implement the multi-agent framework MAIDA related to specifications explained in this work based on DIMA platform. Obviously, we will integrate a large class of security attacks. Moreover, we will specify more precisely mental attitudes in terms of beliefs, goals and motivations used by the deliberation module of the agent to perform detection of network attacks**.**

REFERENCES

[1] C. Ko, D. A. Frincke, T. Goan, L. T. Heberlein, K. Levitt, B. Mukherjee, C. Wee, "Analysis of an Algorithm for Distributed Recognition and Accountability", 1st ACM Conference on Computer and Communication Security, pp. 154-164, 1993.

[2] L. Gasser, "An overview of DAI", Kluwer Academic Publisher, Boston 1992.

[3] M. Crosbie, E. H. Spafford, "Active Defense of a Computer System using Autonomous Agents", Technical report CSD-TR-95-008, Purdue University.

[4] M. Crosbie, E. H. Spafford, "Defending a Computer System using Autonomous Agents". Technical report CSD-TR-95-022, Computer Sciences Department, Purdue University.

[5] M. Crosbie, E. H. Spafford, "Applying Genetic Programming to Intrusion Detection", Technical report, Computer Sciences Department, Purdue University, 1996.

[6] . Balasubramaniyan, J. O. Garcia-Fernadez, D. Isacoff, E. H. Spafford, D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents", Technical report Coast-TR-98-05, Computer Sciences Department, Purdue University.

[7] K. Price, "Intrusion Detection Pages", 1998.Computer Sciences Computer, Purdue University, http://www.cs.purdue.edu/coast/intrusion-detection/ids.html.

[8] W.Stallings, "Network and internetwork security: principles and practice", Ed. Prentice-Hall, 1995

[9] CERT Advisory, CA-95.01, "IP Spoofing Attacks and Hijacked Terminal Connections", January 23, 1995.

[10] CERT Advisory, CA-96.21, "TCP SYN Flooding and IP Spoofing Attacks", Sep. 1996.

[11] C. L. Shuba, I. V. Krsul, M. G. Kuhn, E. G. Spafford, A. Sundaram, D. Zamboni, "Analysis of a Denial of Service Attack on TCP", Technical report Coast-TR-97-06, Computer Sciences Department, Purdue University.

[12] CERT Advisory, CA-98.01, "Smurf IP Denial-of-Service Attacks", January 1998.

[13] E. H. Spafford, "The Internet Worm Program: An Analysis", Technical report CSD-TR-823, Computer Sciences Department, Purdue University.

[14] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, Y. Yemini, "Decentralizing control and intelligence in network management", 4th International Symposium on Integrated Network Management, Volume 4, 1995.

[15] W. Ford, "Computer Communications Security: Principles, Standard Protocols and Techniques", Ed. Prentice Hall PTR, 1994.

[16] N. Puketza, M. Chung, R. A. Olsson, B. Mukherjee, "A Software Platform for Testing Intrusion Detection Systems", IEEE Software Journal, pp. 43-51, 1997.

[17] P. A. Porras, "A State Transition Analysis Tool for Intrusion Detection", Master thesis, University of California, 1992.

[18] K. Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX", Master thesis, University of California, 1992.

[19] A. Sundaram, "An Introduction to Intrusion Detection", Technical report, 1996. Department of Computer Sciences, Purdue University.

[20] S. Kumar, "Classification and Detection of Computer Intrusions", Technical report Coast-TR-95-08, Coast Laboratory, Purdue University.

[21] T. Lane, C. E. Brodley, "Sequence Matching and Learning in Anomaly Detection for Computer Security", Technical report Coast-TR-97-04, Purdue University.

[22] A. Mounji, "Rule-Based Distributed Intrusion Detection", PhD thesis, 1997.

[23] D. Samfat, "Architecture de Sécurité pour Réseaux Mobiles", PhD thesis, 1996.

[24] D. E. Denning, "An Intrusion-Detection Models", IEEE Transactions on Software Engineering, Volume SE-13, No. 2, February 1987.

[25] S. Kumar, E. G. Spafford, "An application of Pattern Matching in Intrusion Detection", Technical report CSD-TR-94-013, Computer Sciences Department, Purdue University.

[26] S. Kumar, E. G. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection", 17th National Computer Security Conference, pp. 11-21, Baltimore, 1994.

[27] S. Kumar, E. G. Spafford, "A Software Architecture to support Misuse Intrusion Detection", Technical report CSD-TR-95-009, Purdue University.

[28] L. Mé and V. Alanou, "Détection d'intrusions dans un système informatique",Valgo Journal, Volume 95, pp. 68-78, 1995.

[29] L. Mé, "Un algorithme génétique pour détecter des intrusions dans un système informatique: méthodes et outils", TSI Journal, volume 96(4), pages 429-450, 1996.

[30] L.T. Heberlein, B.Mukherjee, and K.N.Levitt, "Network Intrusion Detection", IEEE Network Journal, pp. 26-41, May/June 1994.

[31] V. H. Marshall, "Intrusion Detection in Computers", Trusted Information Systems Report, 1991.

[32] M.Gregory, B. White, E. A. Fisch, and U. W. Pooch, "Cooperating Security Managers: A Peer-Based Intrusion Detection System", IEEE Network Journal, pp. 20-23 January/February 1996.

[33] T. F. Lunt, D. Anderson, "Software requirements Specification: Next-Generation Intrusion Detection Expert Systems", Technical report, SRI International, 1993.

[34] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, S. Staniford-Chen, S. Templeton, K. Levitt, S. Walnum, C. Wee, and R. Yip, "GrIDS:A Graph-Based Intrusion Detection System for Large Networks", 19th National Information Systems Security Conference, 1996.

[35] Habra, B. Le Charlier, A. Mounji, I. Mathieu, "ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis", European Symposium on Research in Computer Security, Toulouse France, Springer-Verlag 1992.

[36] J. Erceau and J. Ferber, "L'intellligence Artificielle Distribuée", La Recherche, No. 23, 1991

[37] R. Smith and R. Davis, "Framework for Cooperation in Distributed Problem Solving", IEEE Transactions on Systems, Man and Cybernetics, SMC, pp.61-70.

[38] Y. Demazeau, J. P. Muller, "Decentralized Artificial Intelligence 1", Demazeau et Muller (Eds), Elseiver Science Publisher B. V., pp3-16, 1990.

[39] Y. Demazeau, J. P. Muller, "Decentralized Artificial Intelligence 2", Demazeau et Muller (Eds), Elseiver Science Publisher B. V., pp3-10, 1991.

[40] H. Nwana and M. Wooldridge. "Software Agent Technologies", BT Tech. Journal, 14(4) :68-78, 1996.

[41] J. Ferber and M. Ghallab, "Problématique des Univers Multi-Agents Intelligents", Journées Nationales du PRC IA, Toulouse, pp. 295-320, 1998.

[42] J. Ferber, "Les Systèmes Multi-Agents, Vers une intelligence collective", InterEditions 1995.

[43] Z. Guessoum, "Un environnement opérationnel de conception et de réalisation de systèmes multi-agents", PhD thesis, 1996.

[44] M. Wooldridge and N. R. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Engineering Review, 10(2) :115-152, 1995.

[45] Y. Shoham, "Agent Oriented Programming", Artificial Intelligence 60(1), pp.51-92,1993.

[46] M. Gondran, "Introduction à une théorie formelle de la communication", Bulletin de la Direction des Etudes et Recherche Série C, Informatique No. 3, pp. 37-62, 1989.

[47] R. F. Teixeira de Oliveira, "Gestion des Réseaux avec Connaissance des Besoins: Utilisation des Agents Logiciel", PhD thesis, 1998.

[48] K. Alagha, H. Labiod, "MA-WATM:A new Approach Towards an Adaptive Wireless ATM Network", ACM MONET Journal, Baltzer publishers, Vol. 4, No. 2,1999.

[49] J. Ferber, "Technologies Multi-Agent", Actes du Forum France Télécom Recherche, Mémento Technique, No. 8, pp. 63-79,1996.

[50] C. Bernon, "Conception et Evaluation d'une Plate-Forme pour le Placement Dynamique de Procesus Communicants", PhD thesis,Paul-Sabatier University, France, 1995.

[51] M. Braneci, "Protocoles de contrôle d"erreurs pour des nouvelles architectures des réseaux de télécommunications", PhD thesis, France, 1997

[52] A. S. Rao and M. P. Georgeff, "An abstract architecture for rational agents", 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), pp 439-449, October 1992.

[53] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Toward an architecture for resource-bounded agents", Technical report CSLI-87-104, Center for the Study of Language and Information, SRI and Stanford University, August 1987.

[54] M. J. Wooldridge, "On the Logical Modelling of Computational Multi-Agent Systems", PhD thesis, UMIST, Department of Computation, Manchester, UK, 1992.

[55] J. Y. Halpern and Y. Moses, "A guide to completeness and complexity for modal logics of knowledge and belief", Artificial Intelligence, 54:319-379, 1992.

[56] J. Hintikka, "Knowledge and Belief", Cornell University, Ithaca, New York, 1962.

[57] J. P. Müller, "The Design of Intelligent Agents – A layered Approach", Lecture Notes in artificial Intelligence, springer 1996.

[58] J. Müller, "The right architecture to do the right thing", Atal'99, Springer Verlag, pp. 1-18, 1998.

[59] Z. Guessoum, J. P. Briot, "From Concurrent Objects to Autonomous Agents", MAAMAW'97.