Agent IDS based on Misuse Approach

F. A. Barika[1] & N. El Kadhi[2] & K. Ghédira[1]
1: Higher Institute of Management, LI3, Tunisia
farah.barika, khaled.ghedira@isg.rnu.tn
2: LERIA, Epitech, Paris
ECCE Dept. Chairman Ahlia University Bahrein
nelkadhi@ahliauniversity.edu.bh

***Abstract*— Most current IDS are generally centralized and suffer from significant limitations when used in high speed networks, especially when they face distributed attacks. This paper shows that the use of mobile agents has practical advantages for intrusion detection. For this purpose we carried out a comparative experimental study of some IDS, showing their limits and then we propose an implementation of a new MAFIDS (Mobile Agent for Intrusion Detection System) model focusing on misuse approach. The performance of MAFIDS is investigated in terms of detection delay, false alarm and detection rate by comparing it to a centralized IDS over real traffic and a set of simulated attacks.**

***Index Terms*— Mobile Agents, Intrusions Detection System, Misuse Approach, Detection Delay, False Alarm, Detection Rate.**

I. INTRODUCTION

Over the years computer systems have successfully evolved from centralized monolithic computing devices supporting static applications, into a distributed computing called Network. Nevertheless, as our systems are becoming more open, and so are the associated security threats and vulnerabilities [1]. Thus, a key challenge is to provide the computer systems with mechanisms to overcome this attacks and threats. Security is crucial to the success of active networking; especially when the current network is characterized by its dynamic nature and its increasing distribution. Traditional network relies on the security mechanisms and policies deployed on the underlying operating system. Nevertheless, these measures are insufficient, specially, for those systems the design and implementation of which present security vulnerabilities [2]. Among all security issues, intrusion is the most critical and widespread. An intrusion occurs when an attacker takes advantage of security vulnerabilities and thus violates the confidentiality, integrity or availability of the objects on the network. In other words, an intrusion is informally defined as a deliberate attempt to violate the security policy [3]. The taxonomy of the techniques against intrusions is as follow [4] :

- Prevention: design, implement and configure the system as correctly as possible,
- Dissuasion: devaluate the system by camouflage or overestimate his protection,
- Detection: analyze the log file searching an intrusion signature or an abnormal behavior,
- Deflection: make the intruder believe that his intrusion is a success, while being diverted to a controlled environment,
- Correction: react when the intrusion takes place.

The field of automated computer security intrusion detection is currently some twenty years old, the result of which lead to Intrusion Detection system (IDS). The goal of IDS is to analyze the event stream on the network and identify manifestations of attacks. Commercial solutions are, generally centralized and suffer from significant limitations when used in high speed networks. It is too tedious to detect a malicious action through the network, especially when we should consider multiple distributed events and even simultaneous. Most of the current IDS assume that the environment is static, whereas, in reality, the environment is dynamic and unpredictable. So, to detect without mistake an intrusion and to make the appropriate decision at a favorable time we need a cooperation of different sensors. It is advisable to consider mobile agents as a challenge to intrusion detection. Among the most remarkable characteristics of the agents we find the cooperation to carry out a global objective. So the goal of our research work is to develop a distributed intrusion detection system using mobile agents.

For this purpose Our paper is organized as follow. We present in the first section a background of IDS, point out the limitations of the centralized IDS. We select a set of opensource commonly used IDS, investigate their features theoretically and experimentally and we distinguish their limits. After showing the comparison tables we argue the usefulness of mobile agent in the detection process, present the architecture of our IDS, its implementation and its behavior in front of the simulated intrusions.

II. IDS BACKGROUND

IDS play an important role in achieving survivability of information system and preserve its safety from the attacks [5]. An IDS can be classified as network-based (NIDS) or host-based (HIDS) [6]. The major qualitative difference between these two categories is that NIDS base their analysis on information obtained by monitoring the stream of data in the network to which the hosts

Manuscript received January 1, 2009; accepted March 1, 2009.

are connected, while the HIDS perform their analysis on information collected at a single host by the audit trails. The HIDS act above the network layer making it unable to detect some kinds of attacks [7] while NIDS infer their decision from low-level network packets traveling among hosts [8].

An IDS can also be classified into two categories, according to the approach used in analyzing network events: those based on behavioral approach, and, those based on misuse approach.

- Behavior approach: it relies on models of the normal behavior of a computer system [9]. Behavior profiles may be focused on the users, the applications or the network. In this approach, to detect abnormal activity patterns, the predefined profile patterns are compared with the actual ones in use. The detected patterns will be considered as intrusions.
- Misuse approach: relies on a set of attack descriptions, also called attack signatures [10]. These descriptions are matched to the stream of audit data, attempting to verify that the defined signature is occurring.

Both behavioral and misuse approaches present advantages and disadvantages. An IDS based on misuse approach can detect only those attacks that have been defined. Behavioral approach able us to detect attacks that are unknown in advance; this advantage causes a large number of false positives occurred when an IDS alerts an event that is not an intrusion [5]. Commercial IDS products such as NetRanger [11] and RealSecure [12] work on misuse approach.

A. IDS requirements

In [13], the authors have defined a set of desirable characteristics for an IDS along two themes : functional and performance requirements. In the following section, we summarize some of these characteristics.

- Functional requirements
  - The IDS must continuously monitor and report intrusion,
  - The IDS should have a very low false alarm rate,
  - The IDS must provide enough information to repair the system in the case of intrusion detection,
  - The IDS must detect and react to distributed and coordinated attacks. Coordinated attacks against a network will be able to marshal greater forces and launch many more and varied attacks against a single target.
  - The IDS should be adaptive to network topology and configuration changes.
- Performance requirements
  - Intrusion should be detected in real-time and reported immediately to minimize the damage to the network,
  - The IDS must be scalable to be able to handle the additional computational and communication load.

B. IDS limitations

The most common IDS shortcomings are :

- High number of false positives,
- Lack of efficiency : usually, when an IDS is faced to a huge number of events in the network, it slows down a system or drop network packets that it don't have time to process,
- Vulnerability to be attacked : many IDS have hierarchical structures. This gives the opportunity to the attackers to harm the IDS by cutting off a control branch or even tacking out the root command.

Apart from these shortcomings, we will be interested especially in the following limitations :

- Many of the existing network- and host-based IDSs perform data collection and analysis centrally using a monolithic architecture [14]. Indeed, the collection of data achieved by a single host and even the analysis is performed by a single module. Moreover the centralize detection scheme suffers from number of problems :
  - A central analyzer presents a favorable target to the attackers. If an intruder manages to neutralize it, the entire network becomes without protection,
  - In case of high network load leading to excessive data traffic, the system suffers from scalability problems. A single analyzer unit limits the network size,
  - It is difficult to make modifications to the sensor stations,
  - Given that the collection of network data is performed in a host other than the one to which the data is destined give the possibility to the attackers to make insertion and evasion attacks [15].

Notice that Intrusions can consist of several steps that occur at different hosts, and consequently cannot be detected by a single sensor. The cooperation of different sensors becomes a necessity for the identification of distributed intrusions. Thus using agent technology for IDS can potentially overcome a number of the shortcomings mentioned above.

III. COMPARING IDS

We establish a comparative study of four open source IDSs (SNORT, Prelude-IDS, Tamandua, Firestorm). These IDS are commonly used. Through a set of simulated intrusion tests, we explore their general features and limitations, and we show their weakness to distributed attacks.

A. SNORT Features

SNORT is an open source NIDS. It is able to perform the analysis of network traffic in a real-time using a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods [16]. In fact, Snort is primarily a rule-based IDS,

however input plug-ins are present to detect anomalies in protocol headers [17]. This NIDS is usually used to detect various attacks such as port scan, buffer overflow, web applications attacks and virus attacks.

SNORT has three basic components :

- Packet capture: Sniff and collect network event,
- Rule matching: comparative analysis between the collected event and the attack signature,
- Output: the generated result from the rule matching.

An example rule is :

```
alert tcp $EXTERNAL_NET any->$HOME_NET
21 (msg:"FTP passwd
attempt" flags:A+; content:"passwd";)
```

The rule header consists of the action keyword alert and everything else before the left parenthesis, and the parenthesized list contains the rule options. This rule matches TCP packets from any external source IP address and port, to port 21 on the local network, containing the string passwd and having at least the ACK flag set. Whenever a matching packet is found, an alert is generated with the text "FTP passwd attempt".

SNORT has three different modes:

- Sniffer mode: read all network packets and display them into its interface,
- Audit mode: save the network event on the log file,
- Detection intrusion mode: analyze the network traffic, compare it with the intrusion signatures and perform actions accordingly.

B. Prelude-IDS Features

Prelude is a hybrid IDS framework [19]. It performs either on a host (HIDS) or on a network (NIDS). Prelude has a modular and distributed architecture which allows installing the needed components at a specific node of the network, such that, no network overload is occurred. Prelude is an IDS based on the misuse detection. It relays on the attack signature base of SNORT to analyze the network event in a real time or in a differed time, by reading the log file. Prelude uses the regular expression language PERL (Practical Extraction and Report Language) which allows writing a specific rules.

C. Tamandua Features

Tamandua is an open source NIDS which is based on the misuse approach [20]. Tamandua uses distributed sensors and a centralized console. The sniffer deployed by this NIDS can collect packets from different connections and performs the defragmentation process. Tamandua can import SNORT rule sets. In case of intrusion, the IDS can perform a pro-active reaction such as disabling an IP address. In fact, the code of the instructions used includes the RFC standard in order to support various protocols (IP, UDP, ICMP, TCP). Besides, Tamandua has other features such as a human readable signatures and session-based network analysis.

D. Firestorm Features

Firestorm [51] is a hierarchical NIDS based on misuse approach. It is fully pluggable and hence extremely flexible.

The architecture of Firestorm is composed of four components :

- The probe: sniff the network events, analysis them using the SNORT signatures and save the alerts in an extensible log format.
- The extended log: presents a new format for the alert data transport,
- The Stormwall: supervises the alerts and performs tasks when a new extended log appears. The probe notifies to Stormwall the modification of the log file,
- The consol: permits to the user to search, sort, filter, link and extract data from the probe.

Firestorm presents the following characteristics :

- Comprehensive snort rule support,
- Preprocessors to allow supplementary detection modes (anomaly or behavior approach),
- Full IP defragmentation.

IV. Intrusion Simulations

After the above overview, and in order to compare the four previously described IDS, we propose to perform a set of intrusion tests aiming at evaluating the following aspects :

- Design of the system architecture and the network,
- Vulnerability of components,
- Configuration of equipments,
- Administration procedures of the detection process.

We use the NMAP [18] tool to simulate intrusions and compare the selected IDS. We perform our tests on a network with four Linux hosts.

A. The NMAP Port Scanning

The port scanning is used in order to observe the state of a host or a network. This kind of intrusion allows attacker to gather information about the opened ports (listening) which are a potential transmission routes. These information help the intruder to infiltrate the system.

The port scanning has three variants: open scan, half-open scan and stealth scan.

- Open Scan We consider two modes :
  - TCP connect :

    ```
    nmap -sT target
    ```

    It opens a connection in a complete, classic and legitimate manner. If the port is opened then it will accept the connection. Otherwise the connection will be refused (closed port). This technique is fast, so that many parallel scans can be performed. Besides, it does not require a root privilege. If the result of the connect function is equal to 0 then the relative port is open, else it is closed.
  - Reverse ident :

TABLE I.
HALF CONNECTION (OPENED PORT)

attacker	→	SYN	→	target
attacker	←	SYN | ACK	←	target
attacker	→	RST	→	target

`nmap -I target`

This mode permits to identify the characteristics and the behaviors of the process which listens to the port of the target. The attacker can use the identification protocol which allows him to identify the name of the owner of any process which uses the TCP. The daemon relative to this protocol is called "identd". If the port is open, NMAP sends an "identd request" to the port 113 reserved to the host target.

- half-open Scan
  - TCP SYN Scan:

    `nmap -sS target`

    This mode consists on sending a SYN to the target in order to initiate a connection. Then the target will send an ACK (a valid sequence number) which means that the port is opened. In this case the attacker send immediately a RST for canceling the connection, as shown in Table I. The half-open scan is usually used to achieve the TCP SYN Flooding.
- Stealth Scan Using this mode may avoid certain IDS [4]. The stealth scan, which looks like unexpected network traffic, can pass through the firewall and routers.

  This kind of intrusion takes advantage from TCP/IP vulnerability. In practice, a closed port which receives an IP packet, sends a RST as a response. So, there is no response the attacker concludes that the port is opened. It is difficult to log the stealth scan given that there is no connection completely opened.
  - Stealth FIN Scan

    `nmap -sF target`

    The attacker sends a packet with an active FIN flag to the target. The interpretation of the response is the same as above. Nevertheless, the attacker can have further information like the kind of the OS. In fact, Windows systematically sends a RST whether the port is opened or not.
  - Fragmentation

    `nmap -s[F|X|N] -f target`

    It consists on the fragmentation of the IP packets, especially when the firewall, hosts and the network do not perform defragmentation.
- Other Scans The attacker can use the ICMP packet to know if the host is available or not. Indeed, he can send an ICMP ECHO request (ICMP type 8) to the target, and then if he receives ICMP ECHO reply (ICMP type 0), he concludes that the host is active, in other cases the target is disposed. This technique is called *ping sweep*.

TABLE II.
COMMON FEATURES IDS COMPARISON.

	SNORT	Prelude	Tamandua	Firestorm
Type	N	H/N	H/N	N
OS	L	L	L	L
UP	Yes	Yes	Yes	Yes
GUI	R	R	R	R
Doc.	Yes	Yes	No	Yes
Port.	Yes	Yes	No	Yes
Op.S	Yes	Yes	Yes	No
Instal	Yes	Yes	Yes	Yes
TS	S	D	D	S

Type : Host based Intrusion Detection System (H)/Network based Intrusion Detection System (N)
OS : Operating system (L: Linux by default)
UP : Updating Facilities
TS : Type of signatures (Rules(R)/simples filters (SF)/simples scripts (SS))
GUI : Graphical User Interface
Doc. : Documentation
Port. : Portability
Open.S : Open Source
Instal : Installation (Static (Monolithic) (S), Dynamic (Distributed) (D))

TABLE III.
FUNCTIONAL FEATURES IDS COMPARISON.

	D. A.	IP D.	M. F.	InA	AF
SNORT	M	Yes	R	Yes	IDMEF
Prelude	M	Yes	R/D	Yes	IDMEF
Tamandua	M	Yes	R	Few	proprietary
Firestorm	M/B	Yes	R/D	Yes	extended log

D. A. : Detection approach (Behavior (B)/Misuse (M))
IP D. : IP defragmentation sensibility
M. F. : Mode of functionality (Real time (R)/Differed(D))
InA : Information about the detected attacks
AF : Alert format

  - ICMP ECHO Ping Sweep:

    `nmap -sP -PI target`

    The intruder can improve his attack by sending the ICMP ECHO to a broadcast server in order to scan the entire network.
  - UDP/ICMP Error

    `nmap -sU target`

    The aim of the attacker here is to identify the active UDP port. If he receives an ICMP port unreachable message then the port is closed.

B. Comparison Results and Discussion

An overall comparison of the general features of the selected IDS is shown in Table II.

Table III and Table IV show comparison results according to their functionality features and a set of particular characteristics, respectively. All the selected IDS are open source, rule-based and offering updating facilities. Only Prelude and Tamandua offer a dynamic installation and both host and network based intrusion detection. These characteristics make them able to deploy new sensors and provide flexibility to both application and network administrators. In fact, hybrid IDS offer an extensible toolkit which is able to adapt to the heterogeneousness

TABLE IV.
SPECIAL FEATURES IDS COMPARISON.

	A.B.	D.D.	C.C	Interdep
SNORT	No	No	No	No
Prelude	No	Yes	No	Yes
Tamandua	No	Yes	No	Yes
Firestorm	No	No	No	No

A. B. : Autonomous Behavior
D. D. : Distributed Detection
C. C. : Cooperative Component
I. C. : Interdependent Component

TABLE V.
IDS COMPARISON ACCORDING TO THE SIMULATED INTRUSIONS.

	SNORT	Prelude	Tamandua	Firestorm
TCP C.	1	1	1	1
R. I.	0	0	0	0
TCP S.	1	1	1	1
S. F.	1	1	0	0
F.	0	0	0	0
P. S.	1	0	0	0
UDP/ICMP E.	1	0	0	0

TCP C. : TCP Connect
R. I. : Reverse Ident
TCP S. : TCP SYN
S. F. : Stealth FIN
F. : Fragmentation
P. S. : Ping Sweep
UDP/ICMP E. : UDP/ICMP Error
0 : Not Detected
1 : Detected

and size of the network to monitor. The weakness of Tamandua is that it has no graphical user interface and no sufficient documentation. According to this set of common features, Prelude is better than the other IDS.

As for the functional characteristics of the chosen IDS (Table III), all of them perform IP defragmentation, thus offering protection, against evasion technique [21]. Only Firestorm offers protocol anomaly detection in addition to the misuse approach. Snort, Prelude and Firestorm give more information about the detected attacks than Tamandua. Moreover, we consider the alert format giving by each IDS as an important criterion. Generally, there was two specific contexts that express the needs for a standard alert format. In the first case, many organizations use many different IDS; and since each IDS uses its own alert format, the administrator needs to correlate all generated alerts. In the second case, Different sensors have been dispatched through the network. The original data format they process differs from one sensor to another. When a sensor raises an alert to the global IDS, it should output it in a standard format. The Intrusion Detection Message Exchange Format (IDMEF) [22] is a proposal of such a standard format. SNORT and Prelude alert into IDMEF.

We focus on a set of particular characteristics mentioned in Table IV. Generally, the attacker performs its attack step by step and tries to spread through the whole network. Consequently, it would be more advantageous to have an IDS with distributed components on the network. These components can be cooperative, autonomous or interdependent. Autonomous behavior (A. B.) stipulates that the components of the system act as independent and fully functional detection systems, i.e. they detect attacks independently without communication or interaction between them. Autonomous behavior requires special capabilities such as Self-Configuration and Self-healing [23]. One of the major problem faced by the IDS is that the attacker target it. So performing tasks autonomously makes the IDS more robust. Only Prelude and Tamandua has a distributed and interdependent component.

Our major interest is to compare these four IDS behaviors with respect to some commands performed by NMAP we presented previously. Tests were performed in a four host's network, using each IDS in its default configuration. Results are summarized in Table V. We conclude that SNORT detect almost the intrusions attempts performed by NMAP. Nevertheless, all the IDS

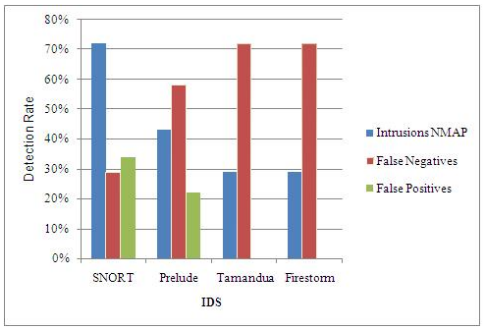

Figure 1. Comparison chart of the detection results

are enable to detect the fragmentation. Figure 1 shows the detection rate of each IDS as a result of the simulated intrusions. Snort captured approximately 72% of attacks. This is the best performance in terms of detection rate in comparison with the other IDS. However, Snort presents the higher rate of false positive (false alerts, 33%). In fact, this IDS processes one network packet at a time and matches this with the premises of a set of rules. Now, focusing on individual packets, leads to many false alerts because it misses the broader context of the network session of which the packet is only a small part. Tamandua and Firestorm present the same performance. Despite the fact that they have the lowest detection rate (29%), they did not raise any false alarms. An IDS which generates false positives is better than those which generate false negatives. Thus, we consider SNORT as the best IDS relatively to our set of IDS and to the simulated intrusions.

In addition to the fragmentation sensibility, SNORT turns out to be less efficient when facing multi packet attacks. Moreover, SNORT is efficient only for the attacks limited in one packet. The situation is worse when we consider distributed attacks such as the Man in the middle [24] or the DDOS (Distributed Denial of Service) [25].

Indeed, we have mentioned that Prelude has a distributed architecture. However, distributed intrusion detection systems are especially vulnerable to attacks since typically, each component resides at a static location and

components are connected together into a hierarchical structure. An attacker can disable such a system by taking out a node high in the hierarchy, amputating thus a portion of the distributed system.

In this paper, we propose a solution to this problem by using mobile agents, in order to distribute the detection process. It is hard to locate an intruder for a mobile agent. Moreover even if the host machine, which launched the agent, is eliminated from the network, the agent can still work. So as to ensure a better detection system our system will be more robust.

V. Distributed Intrusion Process

Detecting intrusion in distributed network from outside network segment as well as from inside is a difficult problem [26]. In many cases, an intruder achieves a set of stages to perform its attack. In each stage he can use a different node in the network. This technique has, especially, two consequences :

- Widening the range of the attack and controlling the major part of the network,
- Making hard detecting the intrusion.

A. Mobile agent usefulness

Agent Systems are used in various applications such as workflow, scheduling and optimization [27]. It is advisable to define, firstly, an agent. We refer to [28] :

- An agent is a physical or logical entity characterized by the following attributes :
  - **Autonomy :** agents are independently-running entities, they operate without the direct intervention of humans or others,
  - **Mobility :** agents are able of suspending processing on one platform and moving to another, where they resume execution of their code,
  - **Rationality :** agents embody the capacity to decompose and solve a problem in a rational manner,
  - **Reactivity :** agents perceive their environment and response in a timely fashion to changes that occur in it,
  - **Inferential capability :** agents are able to use prior knowledge of general goal in order to act on tasks,
  - **Pro-activeness :** agents can take the initiative to act and response to their environment,
  - **Social ability :** agents are able to meet and interact with other agents. The interaction and collaboration between agents is achieved by an agent communication language and may depend on an ontology to realize a common understanding of a situation.

Accordingly to the above attributes, we will argue, in this section, the use of mobile agent to improve the characteristics of the IDS, overcome the limitations described previously and evaluate their applicability to design an automated intrusion detection :

- *Reducing Network Load :* The actual IDS are facing one of the most pressing problems which is the processing of a tremendous amount of data over the network. Abstracted forms of these data are usually sent from all locations in the network to a central site to be processed, causing the increase of network load. Mobile agents offer the opportunity to overcome this problem by eliminating the need to this data transfer. Instead, the processing program (agent) will go to the data, given that the an agent is smaller in size than the network information. Furthermore, when an agent collects data related to the host on which it is running, we avoid the risk to be subject to the insertion and evasion attacks.
- *Overcoming Network Latency :* Mobile agents are able to dispatch from a host to carry out operations directly to the remote point of interest, thus agents can provide an appropriate respond faster than a hierarchical IDS that has to communicate with a central coordinator based elsewhere on the network.
- *Asynchronous Execution and Autonomy :* Agents can be stopped and started without disturbing the rest of the IDS. Notice that the mobile agents are able to continue to operate autonomously even if the host platform where it was created is not available or disconnected from the network. Mobile agent frameworks provide IDS the possibility to continue to work even if the failure of a central controller or a communication link was occurred; this fact allow mobile agents to provide Fault Tolerance characteristics.
- *Dynamic Adaption :* Mobile agents can be retracted, cloned, dispatched, killed or put to sleep as network's configuration, topology and traffic characteristics change over time. As the number of the node in the network increases, agents can be cloned and dispatched to these new computing elements.
- *Robust Behavior :* Mobile agents have the ability to react dynamically to insecurity conditions making easier to build robust distributed systems. Even if one of the agents fails, the other agents in the IDS can take up the tasks of the failed agent and continue the detection.
- *Scalability :* Distributed mobile agents IDS are one of several options that allow computational load and diagnostic tasks to be distributed throughout the network [13]. This improves scalability and holds up fault resistance behavior.

B. Related Work

The idea of distributing the intrusion detection system using a software agents is not entirely new. However, most of the related works emphasized static agents instead of mobiles ones. Applying mobile agent technology to IDS gives a result to only few research projects. In 1999, a project at The Information-Technology Promotion Agency (IPA) in Japan involves an Intrusion Detection Agent (IDA) System [29]. IDA is a classic host-based

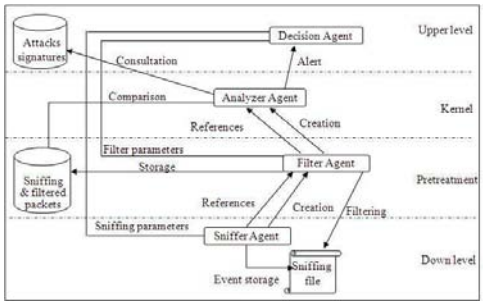

Figure 2. MA_IDS Architecture

system which relies on mobile agents mainly to trace intruders among the various hosts involved in an intrusion. In the same year, Micael [30] pursues a more ambitious aim where the entire system functionally with mobile agents. Nevertheless, only the architecture description has been presented and no details have followed so far.

In 2000, an IDS framework based on mobile agents has been described in [31]. Unfortunately, the detection is dealt with superficially. Globally, there have been some previous attempts to take advantage of agents in the field of intrusion detection, as for example [32]–[34]. It is worth mentioning the mobile-agents approach [35], [36]. Besides, there are other products such as Tritheme which is an IDS under LPG licence allowing the simultaneous use of HIDS and NIDS approaches distributing the different functions under agents scattered on the network under control [37].

In 2002, Trapathi and al. describes an IDS which are designed as mobile application that roam the network to detect attacks and track intruders [38].

The Skyrecon's StormShield [39] is a product complementary to firewall based on the behavior approach. StormShield treats in a coordinated way the potentially vulnerable aspects of a host: traffic network, operating system, applications.

MonALISA [40] is a distributed and dynamic system able to provide a complete control and an overall monitoring of a complex system. The architecture of MonALISA is based on autonomous entities capable of collecting, analyzing and processing data in distributed network.

VI. Our System: Architecture Overview

The distributed structure of our system consists of four levels, as shown in figure 2 : the down level, the pretreatment, the kernel and the upper level. We have four cooperatives, communicants and collaborative entities which are able to move from one station to another: Sniffer agent, Filter agent, Analyzer agent and Decision agent.

Every category of agent is assigned respectively to the levels cited previously.

A. The Sniffer agent

This kind of agent will be cloned and distributed throughout the network. This agent patrols the network, collects all the events occurred in the host to which it is related and storage the collected data in a sniffing file. The Sniffer agent can duplicate it self in order to lighten the network charge. On the down level, we are interested to collect all the events occurred through the network in real time. Sniffer are what is commonly called sensor [41].

B. The Filter Agent

Detecting intrusions in a distributed system turns out to be difficult. IDS must undertake to analyze a huge volumes of events. This task becomes more difficult especially when the events must be collected from distributed sources around the network. Intrusions seep in all levels of the distributed system; each level may require monitoring. So, to be able to determine whether an intrusion is taking place, we have to aggregate and merge events collected from various sources, which is among the set of tasks allocate to the Filter agent.

This agent performs its tasks in the context of the collected-events pretreatment phase, which precedes the analysis phase. The Filter agent access to the sniffing file which is modified by the Sniffer agent.

The Filter agent will treat these crude events by achieving the following tasks :

- Distinguish the various fields of the events collected in crude such as destination address and the protocol,
- Sort the events by the category of packet (TCP, IP, ...) concerned by a specific kind of intrusion.

C. The Analyzer Agent

This kind of agent processes and analyzes the events captured by the Sniffer agent and pre-processed by the Filter agents. We adopt the misuse approach, as mentioned previously. In fact, there are several techniques to perform this approach [42]; we focus on the pattern matching method which is used to scan for byte signatures in packets that may indicate an attack. If there are a similarity between the filtered packets and the attacks signatures then the Analyzer agent raises an alert to the Decision agent.

D. The Decision Agent

The administrator, depending on his need and requirement, can give some parameters relative to the full detection process. This parameters are saved on a configuration file which is consulted by the Decision agent in order to sort them by kind of treatment. In fact, we consider sniffing parameters such as the address of the monitoring hosts and filter parameters like the target protocol. Furthermore, the Analyzer agent report their findings to the Decision agent which transmits them to the administrator.

VII. Implementation

The obvious disadvantage of using mobile agent is the concern that they will introduce vulnerabilities into the network and the following security problems :

- Integrity: corruption of information,
- Denial of service: effecting availability of the host or the agent process,
- Secrecy: disclosure of information.

That's why our implementation is written using the IBM Aglet [43] platform and Sun's Java Development Kit. This choice is not arbitrary, it is made after a comparison study of nine agent platforms [44] during which we have been stressed on the security mechanisms deployed by each platforms to protect the agent and even the host.

Let us consider the most important security feature for our MA_IDS. In fact it is important to build our solution with an opened platform that allows agent migration, cloning agent and that support different communication protocols. These features are common to almost all agent platforms. What seemed to be so crucial for us is security features. We mainly focus on the following criteria :

- Agent authentication: Trusted and Untrusted Agent authentication systems,
- Resource Access Control: Implementing/or not a discretionary access control,
- Supporting encryption/decryption facilities: Using standard protocol such as SSL and supporting common used algorithm (DES, DEA, IDEA, RSA and so on),
- Code verifier (Agent action Verifier): Including a particular bytecode verifier or allowing a such ad on easily. We mainly plan to develop a specific Java bytecode analysis as in [45] for verifying applet security properties. An other alternative is to use a such verifier as introduced by Michiaki Tatsubori [46].

The open-source nature of the Aglets system, which have been made available through the SourceForge open-source initiative [47], allowed us to intercept the adequate security actions performed by Aglets and to log all the useful information such as the requested operation, its parameters, its outcome and the identity of the aglet. Mainly, Aglets supports the specific security mechanism that we required. The security model provided by Aglets supports the definition of security policies and describes how and where a secure Aglet system enforces these policies. The aglets are authenticated identities that are used to enforce the policies defined by authorities and to identify the host or the developer of the program. Aglets framework uses an asymmetric (public_private key pair) cryptography system to exchange private keys between hosts. These keys are useful to ensure agent when they are transferred over the network. Thus, the agent code is signed and can be authenticated before its execution, making the host platform protected. In Aglets, permission is defined as the capabilities of executing aglets by setting access restrictions and limits on resources consumption. An abstract syntax for permissions in Aglets is based on JDK policy [48] file definition.

Figure 3. The NMAP result

VIII. Tests and simulation attacks

We implement MA_IDS using Sun's Java Development Kit version 1.4.1 (Sun Microsystems, 2003), the framework Aglets Workbench 2.0.2., the Netbeans 3.4. and the Jpcap 0.01.16.

All the experiments were conducted on equivalent machines equipped with a Pentium Dual Core Processor running at 1.66GHz and 1.99 GB of main memory.

Our MA_IDS performs their tasks over any number of hosts in the network. Each host can receive any number of Sniffer agent that monitor all events occurring in it.

In a first phase, we test the communication model by sending a set of messages between our four agents classes. We also test the mobility of these agents by dispatching them and retracting over four hosts.

In a second phase we investigate and test four types of network attacks: Probe, R2L, U2R and DOS.

- Probe: Attacker tries to gather information on the target host.
- R2L (Remote to Local): Attacker does not have an account on the victim machine, hence tries to gain local access.
- U2R (User to Root): Attacker has local access to the victim machine and tries to gain super-user privileges.
- DOS (Denial of Service): Attacker tries to prevent legitimate users from using a service.

The examples of attacks of each type are SYN-scan (NMAP), NIMDA for both R2L and U2R, and SYN flood. In the following section we simulate these programs and show the performance of our Agent IDS in terms of false alarms and detection rate.

A. The NMAP Port Scanning

We have already mentioned this kind of attack in IV-A. The execution of the NMAP tool over our network gives us a set of useful information (figure 3) :

- 1702 filtered ports
- 139 TCP ports are opened, netbios-ssn
- 445 TCP ports are opened, microsoft-ds
- MAC Address: 00:03:0D:12:12:3C

The Sniffer agent based on the Jpcap library collects the network events using the CaptureTool class and saves them on a file (figure 4).

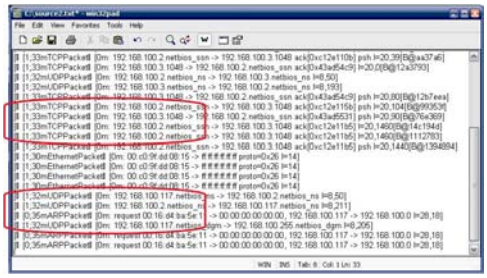

Figure 4. The Sniffing file

```
public void onCreation(Object o)
{   ct=new CaptureTool();
    addPersistencyListener(this);
}
```

The sniffer file contains various packets (TCP, UDP, ARP) including those relative to the execution of the NMAP. After a given number of collected event (Sniffing limit) the Sniffer agent creates the Filter. In fact, depending on the severity degree given by the administrator, the Decision agent fixes the Sniffing limit transmitted to the Sniffer agent as a parameter.

In the pretreatment phase, the Filter agent consults the sniffing file and performs a set of tasks explained in VI-B.

We create a filtered rule according to the SNORT signature used to detect the TCP ping NMAP presented below :

```
alert tcp $EXTERNAL_NET any->$HOME_NET
any (msg:"TCP PING NMAP";
flags:A,12;ack:0;
reference:arachnids,28;
classtype:attempted-recon;
sid:628;rev:2;)
```

This signature means that if an external TCP packet come in our network, from any port, with an active ACK flag, as the two reserved bits, and the sequence number is equal to 0, for any state of session, then it will be necessary to generate the alert relative to the TCP scan accomplished by NMAP.

This particular Nmap TCP Ping uses a TCP ACK with an ACK Number = 0. That's why we filtered this kind of packet only. Our filtered rule correlates the following two characteristics:

- The kind of the packet (TCP),
- An active ACK flag

The Filter agent considers only the packets which verify these two characteristics together.

He save the filtered events in a data base of sniffed and filtered packets (figure 5). The SNORT signature used to detect the TCP ping NMAP may lead to false positives. It is possible that other tools may also send a TCP ACK with an ACK number of Zero. We tested several times the nmap tool and we noticed a particular content in data fields of data of the filtered packets. We consider this content as new nmap's signature.

Figure 5. Table of filtered packets

That means that for all packets examined, the Analyzer agent will try to locate the pattern "B@d02b51".

The Analyzer agent needs two sources: the table of current filtered events and the signature of the target intrusion to perform the Snorts string matching algorithm, Boyer-Moore [54]. This is widely regarded as the providing the best average-case performance of any known algorithm [55]. It consist of comparing character by character data field of the event and of the signature. The algorithm scans the characters of the pattern from right to left beginning with the rightmost character. In case of similarity the Analyzer agent transmits the alert message to the Decision agent.

Thus, our agent system can detect the NMAP intrusion as well as SNORT.

B. The SYN flooding attack

A SYN flood is a type of Denial of Service attack. To realize this kind of intrusion the attacker tries to create a huge amount of connections in the SYN RECEIVED state until the backlog queue overflows. The SYN RECEIVED state is created when the victim host receives a connection request (a packet with SYN flag set) and allocates for it some memory resources. A SYN flood attack creates so many half-open connections that the system becomes overwhelmed and cannot handle incoming requests any more [49].

We simulate this attack using the HPING tool [50] which is able to send custom TCP/IP packets to network hosts. All experiments run on three machines:

- 172.16.0.41: Attack host,
- 172.16.6.220: Web client (IP address which we usurped),
- 172.16.6.40: Web server (the victim machine).

We usurp the IP address of the web client host and send a large number of SYN packets to the web server via this command :

```
hping -S -i u10 -p 80
-a 172.16.6.220 172.16.6.40
```

At the same time we disconnected the web client host. Thus we prevent the machine from answering the packets sent by the web server. Otherwise, it would send TCP RST packets which would stop the connection attempt. The web server machine waits for confirmation that never arrives. Hence the attack succeeds (figure 6).

Both agents, Sniffer and Filter, proceed in the same way as in the case of the NMAP attack. The Analyzer agent considers a specific rule to fulfill its processing.

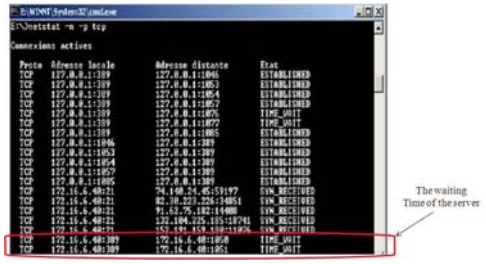

Figure 6. Result of the SYN flood attack

In fact, we borrow that of SNORT with some additions (threshold):

```
Alert tcp any any -> 172.16.6.40 80
(msg: "Alerte!...There was SYN
flood attack"; flow: stateless;
flags: S; threshold: type both,
track by_dst, count 100,seconds 20;
sid: 1000001;)
```

When the rule is triggered it will raise an alert and display a message. The protocol of the packets we are detecting is TCP and destination IP addresses and port number are defined. The source is defined on any IP address. The aim of this rule is looking for any host which is trying to make a SYN connection to the machine which is defined by its IP address (172.16.6.40) on ports 80. The threshold is a crucial parameter given that the purpose is to detect a big influx of demand of connections. Basically, if a machine attempts to make 100 or more connections within 20 seconds then the rule will fire.

C. The Nimda worm

Nimda is identified as U2R or R2L. Typically, this kind of attack exploit multiple vulnerabilities in different network services. For example, Nimda used many different propagation techniques to spread, can send it self out by e-mail, search open network shares, attempt to copy itself to unpatched or already vulnerable Microsoft IIS Web Server. This worm will create the Guest account and add this account to the ”Administer” group. This would enable the intruder to execute the arbitrary command in the local system. Nimda tries to propagate by looking for local shares using the NETBIOS protocol and creates a specially crafted riched20.dll file in any share folders that contains .doc or .eml documents. The rule used by Snort to describe the Nimda worm is:

```
alert tcp $EXTERNAL_NET any->$HOME_NET
139 (msg:"NETBIOS nimda
RICHED20.DLL";flow:to_server,established;
content:"R|00|I|00|C|00|H|00|E|00|
D|00|2|00|0"; reference:url,
www.fsecure.com/v-descs/nimda.shtml;
classtype:bad-unknown;sid:1295;rev:9;)
```

Thus, a signature for the Nimda worm would be the collection of the following parameters:

TABLE VI.
RELEVANT FEATURES (U2R)

Feature	Value
A23	≤ 64
A37	≤ 0.48
A35	≤ 0.91
A36	≤ 0.99
A29	> 0.32

- A packet belongs to a TCP connection from any port to the port 139 which is the port used by NETBIOS,
- The packet belongs to a TCP connection which is established and corresponds to a client request.
- The packet contains the sequence of bytes corresponding to the string ”RICHED20”.

Restricting the signature to the parameter ”packet on port 139” is too general and the derived rule will generate a lot of false alarms. To overcome this limit, we correlate the rule used by Snort with the results described in [56]. The authors use different machine-learning techniques such as Bayesian Classifiers and Decision Trees in the training process on the KDD (Knowledge Discovery in Databases) Cup 1999 data set [57] to learn normal and inconsistent patterns from the testing data and thus generate a set of rules that are able to detect U2R attack. We select relevant features from these rules:

- number of connections to the same host as the current connection in the past two seconds (A23),
- % of connections to the same service coming from different hosts (A37),
- % of different services on the current host (A35),
- % of connections to the current host having the same src port (A36),
- % of connections to the same service (A29),

According to [56], in the case of U2R attack, the selected features should have the values described in Table VI. The Filter agent computes all these values and send them to the Analyzer agent. The following query shows a function that computes the number of http connections to a given host during the last 2 seconds:

```
SELECT count(*) OVER (ORDER BY time_stamp
RANGE INTERVAL ’2’SECOND
PRECEDING) as http_count FROM
connection_event WHERE dest_host
=’ourhost’ AND service = ’http’;
```

Decision agent will receive an alert if the Analyzer Agent will verify these two rules:

- A23 $\leq$ 64, A37 $\leq$ 0.48, A35 $\leq$ 0.91, A36 $\leq$ 0.99, A29 $>$ 0.32,
- The rule used by SNORT to detect the Nimda attack.

D. Performance of Agent IDS

To evaluate the detection performance of our model, we present the false alarms and detection rates of the simulated attack in figure 7.

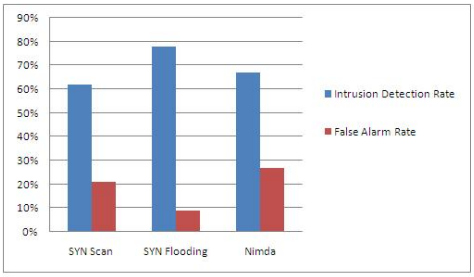

Figure 7. Agent IDS Performance

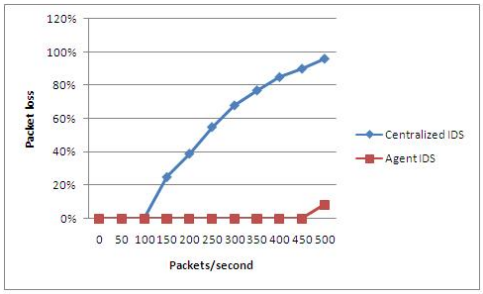

Figure 8. Percentage of dropped packets as the number of packets per second increases

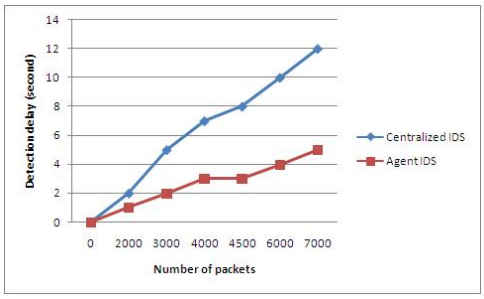

Figure 9. Agent IDS vs Centralized IDS in terms of detection delay

We have tested the Agent IDS system on real network data (18.15 millions of traffic packets), intermixed with 16417 attacks (228 U2R and 16189 R2L records) from the KDD Cup 1999.

A detection rate above 78% and a false alarm rate below 9% are achieved. The comparison of these results with those presented in Figure 1 shows that our system generates a fewer false positives than SNORT in case of probe attack. To achieve a total detection rate above 78% of DoS attacks, we have to tolerate 9% false alarms. The R2L attacks have the second best performance.

The implementation gives us the expected results. Our Agent IDS prototype we are testing detects the simulated attacks. This fact is intended as a proof of applying practically mobile agents to intrusion detection system. In this area, recent research works have been conducted. Most of them focus on the anomaly detection such as CAMNEP [52] and CIDS [34]. CAMNEP treats the problem of false positive by classic agent techniques, mobility is not mentioned. CAMNEP is not able to detect attacks consist of few packets, e. g. buffer overflow attack. CIDS has only one monitor agent responsible for collecting information from various data sources. Some data sources may generate a large amount of events in a short time, especially network traffic. In 2007, Singh and Sodhi [53] present a model based on a roaming agent as data collector and supervisor. The idea shares some similarities with our approach such as using Aglet framework. But the authors do not provide details about the implementation and attacks simulation.

IX. Agent IDS Vs a centralized IDS

The question is : why the realization of the system with mobile agents is advantageous?

We implement a centralized system with local sensor that forward filtered data to a central analysis node and compare it with Agent IDS.

Under 30 Mbps (250 packets/second), these two systems achieves almost the same performance in terms of false alarms and detection rate. We can see that centralized IDS cannot process all packets in rates higher than 30 Mbps so a significant percentage of packets is being lost (figure 8). The systems ability to process traffic at the maximum rate offered by the network with minimal packet loss is an important criterion for an IDS. Significant packet loss can leave a number of attacks undetected and degrades the overall effectiveness of the system. With the recent trend of high-speed networks, the capability of a centralized IDS can not meet the speeds demand, resulting in rising of false negatives. Agent IDS has proven itself to be capable of handling very high traffic. In such a design, the incoming network traffic is disseminated to a pool of agents, which process a fraction of the whole traffic, reducing the possibility of packet loss caused by overload. Agent IDS could support a load of up to 56 Mbps (450 packets/second) with zero traffic loss.

Moreover, we focus on a second important criterion for IDS: detection delay which is defined as the duration from the time the attack starts to the time epoch that the attack is detected. We generate a set of packets varied from 2000 to 7000. For each set we simulate the syn-scan attack and we calculate the detection delay. Figure 9 plots the measurement results. The detection delay is significantly reduced; Agent IDS is much faster than the centralized IDS. For example, in the case of 7000 packets, we observe that detection delay is reduced by 59% (12 second vs 5 second). This can be explained by the fact that mobile agents operate directly on the host, where an action has to be taken, their response is faster than systems where the actions are taken by central coordinator.

In fact, one of the most pressing problems facing current IDSs is the processing of the enormous amounts of data generated by the network traffic monitoring tools and host-based audit logs. IDSs typically process most of this data locally. Mobile agents offer an opportunity to reduce the network load by eliminating the need for this data transfer. Instead of transferring the data across the network, mobile agents can be dispatched to the machine on which the data resides, essentially moving the

computation to the data, instead of moving the data to the computation. It is obvious to see that the code-shipping versus data-shipping argument is only valid if, the mobile agents code and state that have to be transmitted are not larger than the amount of data that can be saved by the use of a mobile agent. That's why, in our Agent IDS, only Sniffer is not mobile. Its code is based on the Jpcap library. Its serialization phase spends much time.

Agent IDS does not only perform better in terms of effectiveness but also in terms of detection delay.

X. Conclusion and future work

As network attacks are becoming more and more alarming, exploiting systems faults and performing malicious actions the need to provide effective intrusion detection methods increases. Network-based, distributed attacks are especially difficult to detect and require coordination among different intrusion detection components or systems.

In 1999, Jansen and al. have said that the idea of mobile and autonomous components intuitively seems useful in intrusion detection and many other applications. However, it is difficult to realize the benefits of mobile agent technology in practice [13]. We propose a new MA_IDS model, based on misuse approach. The experiments emphasize the aim of applying agent to detect some kind of intrusions and compete others IDS. We take advantage of the multi-agent paradigm especially concerning reducing Network Load. Indeed, mobile agents offer the possibility to eliminate the need of transferring a huge amount of data to be analyzed. Hence the mobile agent will go to the data, given that an agent is smaller in size than the network information. In a future work, we will investigate, the behavior approach including new concepts in order to make our agent system more intelligent and exceed the actual performance; a special focus will be on the false alarms and timely identification of new attacks which are two of the biggest challenges to the effective use of network intrusion detection systems.

Acknowledgment

Thank goes to Kirmene Marzouk and Cecile Chatry.

References

[1] L. Me, Z. Marrakchi, C. Michel, H. Debar and F. Cuppens, La detection d'intrusion : les outils doivent cooperer, in *REE Journal*, pp. 50-55, No 5, May, 2001.

[2] R. H. Campbell, Z. Liu, M. D. Mickunas, P. Naldurg and S. Yi, Seraphim: An Active Security Architecture for Active Network, in *UIUCDCS-R-99-2167, UILU-ENG-99-1756, Urbana, IL 61801*, Nov. 1999.

[3] J. P. Anderson, Computer security threat monitoring and surveillance, *James P. Anderson Company*, Fort Washington, Pennsylvania, 1980.

[4] Dethy, *Examining port scan methods - Analysing Audible Techniques*, (2001). [Online]. Available: http: http://synnergy.net/downloads/papers/portscan.txt.

[5] G. Vigna, S. Eckmann and R. Kemmerer, Attack Languages, in *Proceedings of the IEEE Information Survivability Workshop*, USA, pp. 163-166, 2000.

[6] B. Mukherjee, T. L. Heberlein and K. N. Levitt, Network Intrusion Detection, in *IEEE Network*, June 1994.

[7] M. J. Ranum, Experiences Benchmarking Intrusion Detection Systems, in *NFR Security*, Dec. 2001.

[8] T. L. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood and D. Wolber, A Network Security Monitor, in *Proceedings of the Symposium on Research in Securirt and Privacy*, May 1990.

[9] D. E. Denning, An intrusion detection model, in *IEEE Transactions on software engeneering, SE-13 :222232*, 1987.

[10] S. Kumar, and E. Spafford, A Software Architecture to Support Misuse Intrusion Detection, *Department of Computer Sciences, Purdue University*, Mar. 1995.

[11] (2008, Mar.) CISCO. [Online]. Available: http: http://www.cisco.com.

[12] (2008, Mar.) RealSecure. [Online]. Available: http: http://www.iss.net.

[13] W. Jansen, P. Mell, T. Karygiannis and D. Marks, Applying mobile agents to intrusion detection and response, *NIST Interim Report - 6416*, Oct. 1999.

[14] J. S. Balasubramaniyam, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford and D. Zamboni, An Architecture for Intrusion Detection using Autonomous Agents, in *Proceedings of the 14th Annual Computer Security Applications Conference*, Dec. 1998.

[15] T. H. Ptacek and T. N. Newsham, Insertion, evasion, and denial of service: Eluding network intrusion detection, *Secure Network Inc.*, Jan. 1998.

[16] (2007, Oct.) SNORT. [Online]. Available: http: http://www.snort.org/.

[17] R. U. Rehman, *Intrusion Detection Systems with Snort Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*, Publishing as Prentice Hall PTR Upper Saddle River, New Jersey 07458, ISBN 0-13-140733-3, 2003.

[18] (2007, Nov.) NMAP. [Online]. Available: http: http://www.insecure.org/nmap.

[19] (2007, Oct.) Prelude-IDS. [Online]. Available: http: http://www.prelude-ids.org.

[20] (2007, Oct.) Tamandua. [Online]. Available: http: http://tamandua.axur.org.

[21] F. S. Rietta, Application Layer Intrusion Detection for SQL Injection, in *ACM SE06 1012*, Melbourne, Florida, USA, Mar. 2006.

[22] D. Curry and H. Debar, Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition, *Intrusion Detection Working Group*, 116 pages, Jan. 2003.

[23] F. Dressler, G. Mnz and G. Carle, Attack Detection using Cooperating Autonomous Detection Systems (CATS), *Wilhelm-Schickard-Institute of Computer Science, Computer Networks and Internet, University of Tbingen*, 2004.

[24] M. Eriksson, An Example of a Man-in-the-middle Attack Against Server Authenticated SSL-sessions, in *International Conference on Applied Cryptography and Network Security*, Oct. 2003.

[25] S. Specht and R. Lee, Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures, in *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*, pp. 543-550, Sep. 2004.

[26] G. Hulmer, J. S. K. Wong, V. Honavar, L. Miller and Y. Wang, Lightweight Agents for Intrusion Detection, in *Journal of Systems and Software 67 (03)*, pp. 109-122, 2003.

[27] K. Ghedira, MASC : une approche Multi-Agents de problémes de Statisfaction de Contraintes, 1993.

[28] Palmquis, Intelligent Agents in Computer and Network Management, 1998, (course paper). http://www.gslis.utexas.edu/ palmquis/courses.

[29] M. Asaka, S. Okasawa, A. Taguchi and S. Goto, A Method of Tracing Intruders by Use of Mobile Agents, in *Proceedings of the 9th Annual Internetworking Conference (INET'99)*, San Jose, California, June 1999.
[30] J. D. De Queiroz, L. F. R. Da Costa Carmo and L. Pirmez, Micael: An Autonomous mobile agent system to protect new generation networked application, in *in 2nd Annual Workshop on Recent Advances in Intrsuion Detection*, Sep. 1999.
[31] M. C. Bernardes and E. Dos Santos Moreira, Implementation of an Intrusion Detection System based on Mobile Agents, in *In International Symposium on Software Engineering for Parallel and Distributed Systems*, pp. 158-164, June 2000.
[32] E. H. Spafford and D. Zamboni, Intrusion Detection Using Autonomous Agents, in *Computer Networks: The Int. Journal of Computer and Telecommunications Networking 34(4)*, pp. 547-570, 2000.
[33] I. M. Hegazy, T. Al-Arif, Z. T. Fayed and H. M. Faheem, A Multi-agent Based System for Intrusion Detection, in *IEEE Potentials 22(4)*, pp. 28-31, 2003.
[34] D. Dasgupta, F. Gonzalez, K. Yallapu, J. Gomez and R. Yarramsettii, CIDS: An agentbased intrusion detection system, in *Computers & Security 24(5)*, pp. 387-398, 2005.
[35] H. Q. Wang, Z. Q. Wang, Q. Zhao, G. F. Wang, R. J. Zheng D. X. Liu, Mobile Agents for Network Intrusion Resistance, in *APWeb 2006. LNCS, vol. 3842*, Springer, Heidelberg, pp. 965-970, 2006.
[36] K. Deeter, K. Singh, S. Wilson, L. Filipozzi and S. Vuong, APHIDS: A Mobile Agent-Based Programmable Hybrid Intrusion Detection System, in *Mobility Aware Technologies and Applications. LNCS, vol. 3284*, Springer, Heidelberg, pp. 244-253, 2004.
[37] (2007, Aug.) Tritheme Distributed and Hybrid Intrusion Detection and Response System. [Online]. Available: http: http://sourceforge.net/projects/tritheme/, 2001.
[38] A. Trapathi, T. Ahmed, S. Pathak, A. Pathak, M. Carney, M. Koka and P. Dokas, Active Monotoring of Network System using Mobile Agents, University of Minnesota, May 2002.
[39] (2007, Aug.) N. Daira, Strorshield presentation. [Online]. Available: http: http://www.skyrecon.com/, 2004.
[40] (2007, Aug.) MonALISA, MONitoring Agents using a Large Integrated Services Architecture. [Online]. Available: http: http://monalisa.cacr.caltech.edu/, 2005.
[41] A. Cardon, A distributed multiagent system for the self-evaluation of dialogs, in *Proceedings of the Joint JSAI 2001 Workshop on New Frontiers in Artificial Intelligence*, Springer-Verlag, pp. 43-50, 2001.
[42] F. A. Barika, Vers un IDS Intelligent base d'Agents Mobiles, (Institut Suprieur de Gestion de Tunis), July 2003.
[43] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Seconde Edition, ISBN 0-201-32582-9, Massachusetts, Addison Wesley, 1998, 225 p.
[44] N. EL Kadhi, F. A. Barika, E. Burstein and K. Ghedira, Toward Agent IDS : Agents Platforms Security Features Study, in *Proceedings of CSC 2003, Computer security Congress*, Mexique, Mar. 2003.
[45] N. EL Kadhi and P. Boury, Static analysis of Java Cryptographic Applets, in *Proceedings of ECOOP2001 Workshop on Java Formal Verification*, Budapest, June 2001.
[46] M. Tatsubori, An Extension Mechanism for the Java Language, (Titech), 1999. [Online]. Available: http: http://www.csg.is.titech.ac.jp/openjava/papers.
[47] The Aglets Software Development Kit. [Online]. Available: http: http://sourceforge.net/projects/aglets/, June 2002.
[48] L. Gong, Java Security Architecure, (JavaSoft), July 1997.
[49] M. Burdach, Hardening the TCP/IP stack to SYN attacks, in *SecurityFocus*, Sep. 2003.
[50] (2007, Nov.) HPING. [Online]. Available: http: http://www.hping.org.
[51] (2007, Oct.) Firestorm. [Online]. Available: http: http://www.scaramanga.co.uk/firestorm/.
[52] M. Rehk, M. Pechoucek, P. Celeda, J. Novotn, P. Minark, CAMNEP: agent-based network intrusion detection system. Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pp. 133-136, 2008.
[53] M. Singh, S S. Sodhi, Distributed Intrusion Detection using Aglet Mobile Agent Technology. Proceedings of National Conference on Challenges & Opportunities in Information Technology RIMT-IET. Gobindgarh, Mar. 2007.
[54] D. Gusfield, *Algorithms on Strings, Trees, and Sequences*, ISBN 0-521-58519-8, Cambridge University Press, 1997, 554 p.
[55] M. Fisk, G. Varghese, Applying Fast String Matching to Intrusion Detection, Los Alamos National Laboratory, University of California San Diego, 2004.
[56] Ben Amor, N., Benferhat, S., Elouedi Z., Rseaux baysiens nafs et arbres de dcision dans les systmes de dtection dintrusions, RSTI, VOL 25/2, pp.167-196, (2006) KDD Cup 1999:
[57] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Farah BARIKA KTATA received the Master Degree in Computer Science Applied to Management from the Higher Institute of Management of Tunis in July 2003. Since November 2004, her PhD thesis activities concern Multi-agent Systems and Computer Security. Since September 2006, she is an assistant in the department of computer science at the Higher Institute of Science and Technology. She is a member of LI3 (Laboratory of Intelligent Informatic science Ingineering) and the Tunisian Association of Artificial Intelligence (ATIA). Her current research interests include: Multi-Agents Systems and Information System Security.

Nabil EL KADHI Associate Professor, Computer Engineering department Chairman at AHLIA University. Dr Nabil has a PHD in computer science, Formal verification of cryptographic protocols. Hi used to be an associate professor at EPITECH Paris. Hi the director and creator of LERIA research lab. Dr Nabil has more than 35 international publications in international conferences and around 10 Journal publications. He is also a security consultant for e-payment solution. He has also worked for TASK euroopeen projet (VIP:Verified Internet protocol) at INRIA-Paris.

Khaled GHEDIRA obtained his Engineer Diploma in hydraulic from ENSEETHT (France) in 1983 and his Engineer Diploma in Computer Science from ENSIMAG (France) in 1986. He received the Master degree then the PHD thesis in Artificial Intelligence from ENSAE (France) in 1993. He also obtained the habilitation degree in Computer Science from the National School of Computer Studies of Tunis (ENSI) in 2001. Professor GHEDIRA was fellow research for 4 years from 1992 to 1996 at the I3A Institute in Switzerland. Director of ENSI from 2002 to 2008, he actually is the Head of LI3 (Laboratory of Intelligent Engineering of Compuer Science) and president of the Tunisian Association of Artificial Intelligence (ATIA). His current research interests include: Multi-Agents Systems, Constraint Problems Satisfaction, combinatorial optimization, scheduling problems and transport. He is author of about two hundred papers and member of several program committees relative to various conferences and journals.