A SNORT-BASED MOBILE AGENT FOR A DISTRIBUTED INTRUSION DETECTION SYSTEM

Imen Brahmi[1], Sadok Ben Yahia[1] and Pascal Poncelet[2]
[1]*Faculty of Sciences of Tunis, Tunisia*
[2]*LIRMM Montpellier, France*
imen.brahmi@gmail.com, sadok.benyahia@fst.rnu.tn, Pascal.Poncelet@lirmm.fr

Keywords: Misuse Detection; Intrusion Detection System; Mobiles Agents; SNORT Rules.

Abstract: Due to the rapid growth of the network application, new kinds of network attacks are endlessly emerging. Thus, it is of paramount importance to protect the networks from attackers. Consequently, the Intrusion Detection Systems (IDS) are quickly becoming a popular requirement in building a network security infrastructure. Most existing and commercial IDS are generally centralized and suffer from a number of drawbacks, *e.g.*, high rates of false positives, low efficiency, etc, especially when they face distributed attacks. In this paper, we introduce a novel mobile agent-based intrusion detection system focusing on the misuse detection approach, called DIDMAS (*Distributed Intrusion Detection using Mobile Agents and Snort*). DIDMAS takes advantages of the mobile agent paradigm to implement an efficient distributed system, as well as the integration of existing techniques, *i.e.*, the well-known IDS SNORT. Carried out experiments showed that our proposed system presents better performance as well as a good scalability compared to the pioneer known centralized IDS SNORT system over real traffic and a set of simulated attacks.

1 INTRODUCTION

With the rapid growth of Internet, the security-relevant incidents have shown an exponential increase. In addition, cracking technology has evolved into complex approach such as coordinated attacks and cooperative attacks (Li et al., 2004). Under these circumstances, software tools, that can automatically detect a variety of intrusions, are of a compelling need. Standing as a gatekeeper of network, *Intrusion Detection Systems* (IDS) must have the ability to detect and defend intrusions more proactively in short period.

Intrusion detection techniques are largely classified into two areas: *misuse detection* and *anomaly detection* (Denning, 1987). Indeed, misuse detection systems (Denning, 1987) use patterns of well known attacks or weak spots of the system to match and identify known intrusions. As example of such systems, we cite the network IDS SNORT (Roesch, 1999). SNORT is configured using a database of *signatures* which characterize network packets that are potentially malicious. Using this database, SNORT monitors a network connection and logs all occurrences of network packets that match any of the configured signatures. However, misuse detection techniques, in general, are not effective against novel attacks that have no already matched rules or patterns (Denning, 1987). On the contrary, the anomaly detection systems flag activities, that significantly deviate from the established normal usage profiles as abnormal or in other words as intrusions. Anomaly detection techniques have been shown to be effective against unknown or novel attacks, since no prior knowledge about specific intrusions is required. Nevertheless, the main moan that can be addressed to the anomaly detection systems is that they tend to generate more false alarms than do misuse detection systems, *i.e.*, an anomaly can be simply a new normal behavior (Denning, 1987).

IDS are also traditionally classified as either *network-based* or *host-based* (Denning, 1987). Network-based systems monitor the network traffic and inspect packet transmissions for suspicious behavior (Denning, 1987). Host-based systems operate on single hosts, and operate on low-level system data, such as patterns of system calls, file access, or process usage (Denning, 1987). They can monitor for suspicious behavior, or they can scan configurations to detect potential vulnerabilities.

Most of the current IDS use centralized architectures made of individual host and network monitors along with a centralized controller component (Spafford and Zamboni, 2000). The network monitors send intrusion data to the centralized controller component that performs analysis of the information it receives from each of the monitors (Spafford and Zamboni, 2000). Worth of mention that conventional approaches to intrusion detection, involving a central unit to monitor an entire system, have several drawbacks (Jansen et al., 1999; Spafford and Zamboni, 2000). Indeed, the procession of all the information at a single host implies a limit on the size of the network that can be monitored. Likewise, the additions of new hosts increase significantly the load on the centralized controller. Consequently, the centralized IDS suffers from scalability problems (Spafford and Zamboni, 2000). In addition, the communication with the central components can overload parts of the network. Thus, the designed feature of communication and cooperation between a centralized IDS components are badly missing. This fact hampers the capability to efficiently detect large-scale distributed attacks (Spafford and Zamboni, 2000).

As accuracy is the essential requirement for an IDS, its extensibility and adaptability are also critical in today's network computing environment. Considering the growth of the network, it is necessary that the IDS be able to resist attacks on themselves and also needed to be fault tolerant, highly adaptable and configurable (Mosqueira-Rey et al., 2009). Given these characteristics, agent-based technology seemed to be an appropriate alternative for developing IDS (Mosqueira-Rey et al., 2009). In this respect, the agent is a software entity that operates continuously and autonomously in a particular environment, and is able to carry out activities in a flexible and intelligent manner (Herrero and Corchado, 2009). Therefore, the agent-based technology can improve the means of applying detection techniques (Mo et al., 2009). For example, the agents could be deployed at different user computers to collect data as well as they could provide an interface to user application systems for smooth integration. Consequently, the agents were applied within an IDS could provide a good mechanism for implementation of intrusion detection on network-based application systems (Mo et al., 2009).

Particularly, the deficiency of the centralized IDS naturally leads us to the idea of using the mobile agent technology (Jansen et al., 1999). Indeed, the mobile agents reduce the network bandwidth consumption by moving the data analysis to the location of the intrusion data. Besides, they support heterogeneous platforms, and offer a lot of flexibility in creating a distributed IDS. Whenever, an intruder tries to disable the single point in a network, *i.e* the central analyzer. If the latter is disabled, the entire network becomes without protection (Ktata et al., 2009). In this case, a mobile agent-based IDS allows to palliate the drawback of central point of failure, since there is no central station. In addition, the implementation of the mobile agents within languages such as JAVA provides mobile agent with system and platform independence and considerable security features, which are a necessity in IDS (Mo et al., 2009).

In this paper, we investigate another way of tackling the issues within the centralized IDS. Thus, we introduce a novel distributed IDS, called *Distributed Intrusion Detection using Mobile Agents and Snort* (DIDMAS). DIDMAS focuses on the use of the above-mentioned beneficial features offered by the mobile agent technology for detecting intrusion in network-based application systems. It provides an option for setting up a misuse distributed network IDS by integrating a component of the well-known network IDS SNORT. The proposed system permits the data collection, the filtration, the detection of known intrusions using a database of signature-based rules and the response. The specific objectives of DIDMAS are as follows:

- ***(i)*** A new mechanism was designed for acquiring data about user action from client machines or from access control module in server applications. It provides a distributed IDS that reduces the congestion within the network;
- ***(ii)*** Current IDS include many sensors distributed over the network and a centralized management station. These systems cause many bottlenecks and they suffer from the problem of single point of failure;

DIDMAS exploits the benefits of employing mobile agents such as reduced network bandwidth, increased flexibility and ability to operate within heterogeneous environments. Through extensive carried out experiments on real life network traffic, we show the effectiveness of our proposal in terms of (*i*) the scalability-related criteria such as network bandwidth and system response time; and (*ii*) the IDS performance.

The remaining of the paper is organized as follows. Section 2 presents the basics of mobile agents. We list the advantages of mobile agent-based IDS in section 3. Section 4 sheds light on some related research in mobile agent-based IDSs. We introduce our new distributed intrusion detection system based on the mobile agent technology in Section 5. The interaction between the agents is thoroughly discussed in Section 6. We also relate the encouraging results

of the carried out experiments in Section 7. Finally, Section 8 concludes and points out avenues of future work.

2 THE MOBILE AGENTS

The software agents can be treated as mobile agents, as they are able to migrate from one computer to another one (Herrero and Corchado, 2009).

Useful characteristic of mobile agents are as follows (Jansen et al., 1999):

- **Autonomy**: the agents are independently running entities. Thus, they can operate without human control;
- **Mobility**: the agents are able to suspend processing on one platform and to move to another one where they resume execution. In particular, the mobility is the most important feature of the mobile agent for the following reasons (Outtagarts, 2009):
  - *Persistence*: Whenever a mobile agent is launched, it is no longer connected to its creator machine. It still works even in the failure of the machine that initiated them;
  - *Peer to Peer communication*: A failure in the paradigm of client/server is the inability of servers to communicate. The mobile agents are considered peer entities and, as such, can act as either client or server is like;
  - *Fault tolerance*: Within the client/server technology, the transaction state is generally divided between the client and the server. In the case where one server is down, then the client can resume the situation and re-synchronize with the server because the network connection is lost. However, since the mobile agent do not need to keep the connection permanently, in case of network failure it will continue to run on the node.
- **Rationality**: the agents embody the capacity to analyze and solve a problem in a rational manner;
- **Reactivity**: the agents perceive their environment and adapt their behavior in a dynamic way to match, as soon as possible, the new environment parameters;
- **Inferential capability**: the agents are able to share a set of knowledge in order to achieve a specific goal;
- **Pro-activeness**: the agents can decide to adapt their behavior to their environment;
- **Social ability**: the agents are able to meet and interact with other agents. The interaction and collaboration between agents is achieved by an *Agent Communication Language* (ACL).

3 ADVANTAGES OF USING MOBILE AGENTS IN INTRUSION DETECTION

The use of mobile agents for intrusion detection offers a new approach to the traditional IDS methodology (Jansen et al., 1999). With the new capability of mobility for intrusion detection, several advantages related to mobile agent usage are listed in literature (Jansen et al., 1999; Ktata et al., 2009). Some of them are listed as follows:

- **Delay caused by networks**: Whenever the hierarchical IDS's are used in a network, it results in slower response when an attack occurs. This fact is due to the central controller, which sends the information about the attack and the decision to be taken to particular host through the network. This may not always result in an immediate response against the attack, as the time taken for the information to reach the destination host might be too long. Consequently, the traditional hierarchical IDS are not successful in reducing the detection delay. On the contrary, whenever the mobile agents are used, the IDS can respond faster as they are directly dispatched from the central controller to the target host;
- **Minimizing the network traffic**: Traditional IDS employed different data collection mechanisms to collect data both at the host and the network level. Moreover, the central controller uses the collected data to track any intrusions. Generally, the data collected from different hosts is very huge. This results in increasing the network traffic and creating an overhead on the network. By employing the mobile agents, the load on the network can be reduced. The minimization of the load on the network can be explained by the efficient search mechanisms used by the mobile agents, which reduce the necessity for data traffic among several hosts;
- **Persistency**: Although the mobile agents operate autonomously and asynchronously, they are not prone to failure even if the machine, which hosted them, fails. This fact provides additional advantage of employing mobile agents within IDS. Whenever the central controller of the centralized machine fails, then the entire IDS is considered

to be down as there is no communication among other hosts;

- **Structure and platform independence**: The mobile agents can be used in IDS with a flexible structure. For example, one agent can be responsible for collecting the data in the network, the other agent can be used to detect and report anomalies while the rest of them can be used to take appropriate action. Due to this structure, the mobile agents find tremendous applications in IDS;
- **Dynamic nature**: The dynamic nature of mobile agents enables them to be moved around the network. Consequently, it is possible to also reconfigure the system during runtime. Moreover, the mobile agents can be cloned, dispatched or put to sleep when the network configuration has to be changed. Therefore, they can sense their execution environment and dynamically adapt to the situation;
- **Heterogeneous environment**: The mobile agents can be inter-operable on multiple platforms, since the virtual interpreter is installed on the host machine. Generally, the mobile agents are transport-layer independent and depend only on the execution environment. This feature enables the mobile agents to be used on several different platforms without compatibility problems;
- **Robust in nature**: 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. This robust behavior of mobile agents makes them more applicable within large environments where several agents and their interaction is needed for proper monitoring of the network;
- **Scalability**: By employing the distributed mobile agent-based IDS, however large the network grows, it could be easily handled. In this respect, the mobile agents have the capability to clone and distribute themselves to the new machines whenever they are added to the network.

4 THE MOBILE AGENT-BASED IDS

The applicability of the multi-agent technology and the mobile agents to intrusion detection has been explored in several approaches. One of the well known multi-agent-based IDS is the *Autonomous Agents For Intrusion Detection* (AAFID) (Spafford and Zamboni, 2000). Within this architecture, a set of agents monitors specific aspects of a machine requiring security. These agents send information to the transceivers, which in turn, amalgamate the information and re-send it to the monitors. The monitors process this information and decide whether or not an attack has occurred. To make the system more scalable, the monitors may be built in layers. Although this system was very innovative in its time, its main drawback is the extreme rigidity of its architecture, as this makes the introduction of new agents very complicated. Moreover, its hierarchy is such that if an attack manages to deactivate an agent in the upper layers, the entire system might be deactivated.

The key differentiating factor between the AAFID approach and the mobile agent-based approaches is the mobility of the agents participating in the IDS. Many proposed IDS have explored the advantages of the mobility aspect of mobile agents in the context of intrusion detection.

One of the studies worth of mention was the MA-IDS architecture, proposed in (Li et al., 2004). The MA-IDS system employed mobile agents to coordinately process information from each monitored host. Its architecture includes the *Assistant* and the *Response* mobile agents. The Assistant mobile agent is dispatched by the manager component to gather information in the network. The mobile agents within the MA-IDS system are capable of evading attackers and resurrecting themselves if they are attacked. Moreover, the mobility of agents makes it possible that distributed intrusion can be detected by means of data correlation and cooperative detection. However, whenever the location of manager were found by attackers, the IDS becomes in a dangerous situation. Thus, MA-IDS suffers from the drawback of single point of failure.

The APHIDS architecture (*Agent-Based Programmable Hybrid Intrusion Detection System*), proposed by (Deeter et al., 2004), represents a variation of the existing mobile agent-based approaches (with some similarities to the MA-IDS system). It shares a common goal of exploiting the mobility of the agents to perform distributed correlation. It differs, however, in the mechanism by which these mobile agents are coordinated and combined. APHIDS provides a scripting capability that aims to automate evidence gathering tasks that system administrators would otherwise perform manually. It also attempts to utilize and integrate existing IDS technologies, *i.e.*, SNORT. In this respect, APHIDS allows to realize the scalability of mobile agent-based approaches, and addresses flexibility, extensibility, and delay limitations. Whereas, the virtualization and the serialization routines required for agent mobility cause performance

overhead that may be significant without proper design consideration.

Wang et *al*. proposed a distributed IDS, which includes: a *Manager* and a *Host monitor* (Wang et al., 2006). The components in such model are designed as mobile agents for the purpose of high adaptability and security of the system. It is claimed that the mobile agents of the proposed system can evade intrusion and recover by themselves if they suffer from intrusion. Nevertheless, the system uses a control center to carry out the major part of the intrusion detection. Consequently, if the location of this center is discovered, then the system collapses.

In (Singh and Sodhi, 2007), the authors presented a distributed IDS that comprises three different agents, called, respectively: *Roaming*, *Supervisor* and *Action* agents. In fact, a roaming agent moves to predefined host to collect data. The supervisor also acts as evaluator that takes the decision whether suspicious activity is detected at a particular host and alerts with the help of action agent.

Mo et *al*. implemented a misuse mobile agent-based IDS (Mo et al., 2009), which incorporates the SNORT system (Roesch, 1999). The architecture consists of three different components: (1) *an intrusion detection processor* (IDP), (2) *a mobile agent platform* (MAP), and (3) *distributed sensors or sniffer*. Indeed, the IDP is responsible for monitoring network segments. Beside, the MAP is responsible for accepting requests made by the IDP and generating mobile agents as well as sending them into the network to start sniffing activities within the local network, to stop it when necessary, and to send the collected data to the IDP for further analysis. Finally, the sniffer is responsible for gathering data. The mobile agents in this work are fully managed and network resources utilization is saved when there is no attack. However, the proposed system suffers from a high false positive rate, since many attacks could be missed.

Recently, Ye et *al*. (Ye et al., 2009) introduced a distributed IDS based on the mobile agent technology and the open source tool SNORT. The proposed system permits data collection, analysis and response on the supervisory node and the results are analyzed by mobile agent. Therefore, the central server has to take down the intrusion behavior and manage components, since most computation is distributed to the supervisory nodes. In fact, the proposed system palliates the drawback caused by the excessive flow within the system processing center.

Due to its usability and importance, detecting the distributed intrusions still a thriving and a compelling issue. In this respect, the main thrust of this paper is to propose a new distributed IDS called *Distributed Intrusion Detection using Mobile Agents and Snort*, DIDMAS, which is based on the mobile agents and integers an existing signature database. The main idea behind our approach is to address limitations of centralized current IDS systems by taking advantage of the mobile agent paradigm. Specifically, we address the following limitations of conventional centralized approaches:

- *The bandwidth scalability*. The bandwidth required to collect large, distributed network data sets from distributed sensors can pose a significant overhead cost, affecting network performance;
- *The processing scalability*. The processing capability of the centralized approach is limited by the computational power of a single analysis center, even though other resources may be available;
- *The analysis delay*. Within the centralized approach, the huge network traffic causes a long analysis delay. However, the long delays can hamper a timely and effective response.

5 THE DIDMAS SYSTEM

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

The processing steps of DIDMAS can be summarized as follows:

1. The Sniffer Agent is the first agent that connects to the network and gathers the packets. It has the capability to clone and distribute itself to the new machines whenever it connects to the network. It creates the Filter Agents and the Misuse Detection Agents and sends them toward the station to be analyzed;
2. The gathered packets are send to the Filter Agent which filters them;
3. The Misuse Detection Agent is a mobile agent. Based on a knowledge base containing the SNORT signatures, this latter analyzes the Filter Agent output. Whenever an attack is detected, then an alert is sent to the Reporter Agent;
4. Finally, the Reporter Agent is a stationary agent. It generates the reports and logs.

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

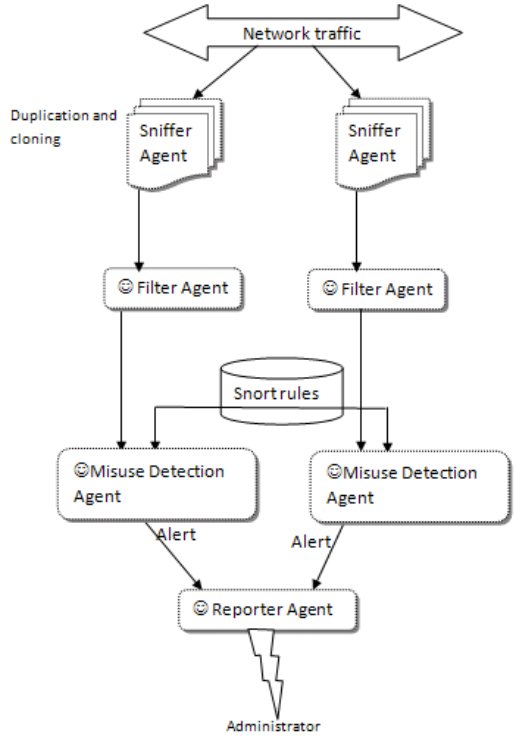

Figure 1: The architecture of DIDMAS

5.1 The Sniffer Agent

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

The Sniffer Agent is the first agent to work in the DIDMAS system. It captures the incoming packets by reading them from the network card in the machine and caches them in the memory at the interval of every 5 seconds. The benefits of this kind of agents include: *i*) the cloning and the distribution throughout the network; and *ii*) the duplication in order to lighten the network charge. Finally, the captured packets will be the input of the next agent, *i.e.*, the Filter Agent.

5.2 The Filter Agent

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

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

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

5.3 The Misuse Detection Agent

This kind of agent processes and analyzes the packets firstly captured by the Sniffer Agent and then preprocessed by the Filter Agent. In fact, it searches for intruder signatures in these packets. Hence, if there is a similarity between the filtered packets and intruder signatures, then an intrusion is detected. Consequently, the agent raises an alert to the Reporter Agent.

The knowledge of the Misuse Detection Agent is represented in rules that are based on intruder signatures. These rules are from the well-known signature-based IDS SNORT System (Roesch, 1999). Although the agent developed is based on the SNORT rules, it uses the efficient pattern-matching RETE algorithm (Forgy, 1982) and tailored to the JAVA language. The latter adapts RETE to an object-oriented interface and allows for more natural expression of rules with regard to domain objects.

SNORT (Roesch, 1999) is the foundation stone of open source network based IDS. It offers a lightweight, efficient and flexible intrusion detection engine with a complete intrusion signature rule set. It analyzes the packets that arrive to the network interface, trying to match their characteristics with those contained in the rules stored in its rule base. If a specific packet matches the premises of any rule, this rule is executed and a specific action is generated to give notice of this fact. Thus, each rule has the following structure:

1. *Rule header*: Contains the basic information about the rule, including:
   - **a Rule action**: The action that will be taken when rule conditions are met. The main actions are: alert (generate an alert), log (log the packet) and pass (ignore the packet).
   - **b Protocol**: The protocols used by the packet being analyzed. Currently, SNORT understands the following protocols: IP, TCP, ICMP and UDP.

c Source information: IP address and port of the source computer from where the packet originated. It is also possible to use the key word 'any' to apply the rule on all packets irrespective of the IP address or port number.

d Destination information: IP address and port of the destination computer in the packet. The keyword 'any' can be used again with the same meaning as before.

2. *Rule option*: Contains alert messages and information on the parts of the packet that should be inspected to determine if the rule action should be taken.

As an example, one possible SNORT rule could be:

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

This rule will generate an alert containing the message "*FTP passwd attempt*" each time a packet using the TCP is detected for any external source IP address and port, and port destination 21 on the local network.

Along our study, we use the signature database of SNORT version 2.3.2, which provides more than 2648 signatures. Those signatures are stored in a MySQL database.

5.4 The Reporter Agent

The Misuse Detection Agent reports its findings to the Reporter Agent which transmits them to the administrator. Whenever an intrusion is detected, it will send an alert to the system administrator. This alert can be a message on the screen or a message to a centralized machine or an alert file.

6 INTERACTION BETWEEN AGENTS

The agents use the ACL (*Agent Communication Language*) language to communicate. Moreover, the information transmitted among agents is sent as text messages and the process complies with the FIPA (*Foundation for Intelligent Physical Agents*)[1] protocols. Within the DIDMAS architecture, agents can communicate in two ways:

(*i*) First, called "*distant communication*". In this communication mode, the agent X needs information from agent Y deployed on another host. It must open one channel to communicate with it. Once the communication channel is created, they exchange messages.

(*ii*) Second, called "*local communication*". In this communication mode, the agent *X* which must receive information, will be deployed on the host where agent *Y* is. Thus, the exchanges are local and do not create large network load. As soon as an agent *X* has received the expected information, it can come back on the last host, remain on the same host (as *Y*) or move to a third host.

The DIDMAS uses several agent's group (sniffing, filtering, analyzing, reporting). Some of these agents need high communication, with rich information, and others just need to share a reduced amount of information.

Worth of mention is to highlight that our system DIDMAS improves sharing distributed resources. It performs their tasks over any number of hosts in the network. Each host can receive any number of agents that monitor all events occurring within it.

7 EXPERIMENTAL RESULTS

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

JADE (*Java Agent DEvelopment Framework*)[2] is a software Framework, which simplifies the implementation of multi-agent systems. The agent platform can be distributed by moving agents from one machine to another one.

In addition, JPCAP (*Java library for CAPturing and sending network Packets*)[3] is an open source library for capturing and sending network packets.

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

Through the carried out experiments, we have a twofold aim: first, we focus on the assessment of the scalability-related criteria such as network bandwidth and system response time. Second, we have to stress on evaluating the performance of our proposed system in terms of detection and false positive rates. During the evaluations, we compare the results of the DIDMAS system *vs*. that of SNORT.

[1] Available at: http://www.fipa.org

[2] Available at: http://jade.tilab.com

[3] Available at: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/

In this respect, we used machines that were connected via a switch, thus forming a switched network. For a realistic testing environment, attacks needed to be interjected into a volume of network traffic. Consequently, we simulated attacks using the well known tool *Metasploit*[4] version 3.5.1. Metasploit is both a penetration testing system and a development platform for creating security tools. It is used by the security researchers world-wide to test an IDS. The description of the eight different attack types used in the evaluation is shown in Table 1.

Table 1: The simulated attacks at a glance.

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

Bandwidth consumption and response time It is known that two of the most important elements of network performance are bandwidth and analysis delay. On the one hand, the bandwidth is the transmission capacity of the network, usually measured in bits per second. On the other hand, the analysis delay is the amount of time it takes for a packet to travel from the source to the destination. Additionally, we use the response time to describe the amount of time it takes once an attack takes place till it gets resolved.

As depicted in Figure 2, the maximum bandwidth consumed by DIDMAS is lower compared to that of SNORT. The reduction of the network bandwidth consumption is due to the use of the mobile agents. The latter move the data analysis to the location of the intrusion data. This makes our proposed system low cost which is definitely a desirable feature for any distributed system.

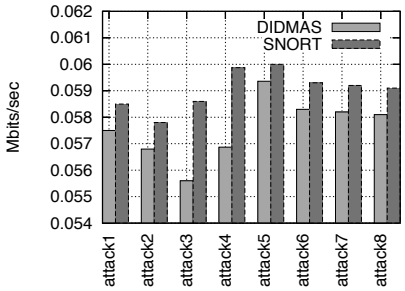

Figure 2: The bandwidth consumption of DIDMAS.

Figure 3 plots the detection delay against the number of packets, using the DIDMAS and SNORT systems.

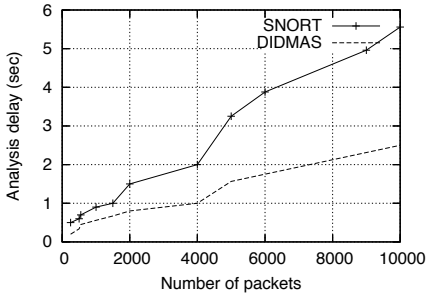

Figure 3: DIDMAS *vs*. SNORT in terms of the analysis delay.

Thus, our proposed distributed IDS is much faster than the SNORT system. This can be explained by the fact that the mobile agents operate directly on the host, where an action has to be taken, their response is faster than the system where actions were taken by the central controller, *i.e*., SNORT.

Figure 4 illustrates the response time required by DIDMAS with respect to the attack types.

According to this figure, the detection of all attack types, on average, result in lower response time compared to that of SNORT.

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

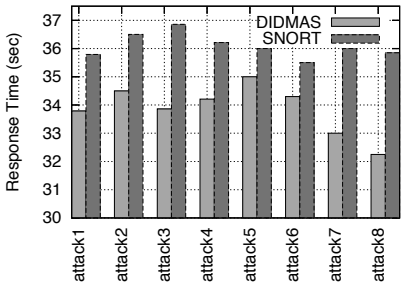

Figure 4: The response time of DIDMAS.

To sum up, it is clear from the obtained results that the performance of the DIDMAS will not deteriorate too much with the increase in the number of attacks, which is justified by its low bandwidth consumption and quick response time behavior. Also, in case of more machines are connected to the network, the DIDMAS system still withstand the load and swiftly deliver the results.

DIDMAS performances assessment In order to evaluate the performance of a IDS, two interesting metrics are usually of use (Eid et al., 2008): the *detection rate* and the *false positive rate*. Indeed, the detection rate is the number of correctly detected intrusions. On the contrary, the false positive rate is the total number of normal instances that were incorrectly considered as attacks. In this respect, the value of the detection rate is expected to be as large as possible, while the value of the false positive rate is expected to be as small as possible.

Figure 5 plots the false positive rates against the number of agents. We notice that all the tested attacks produced negligible false positive rates.

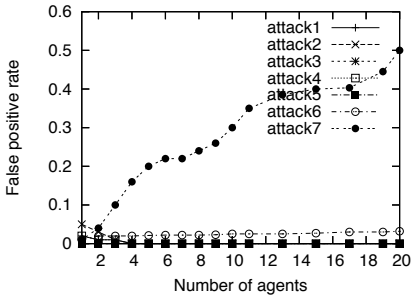

Figure 5: False alarm rates of DIDMAS *vs*. the number of agents.

However, the detection of attack7, *i.e*. *DNS Zone Transfer* attack, results in high false positive rate. This event indicates that an outside host requested a zone transfer from an internal DNS server, which can be legitimate traffic from a secondary DNS server, or an attacker gathering information about your domain, thus making the detection rules false positive prone.

According to Figure 6, we can remark that the false alarm rates of DIDMAS is significantly lower compared to that of SNORT.

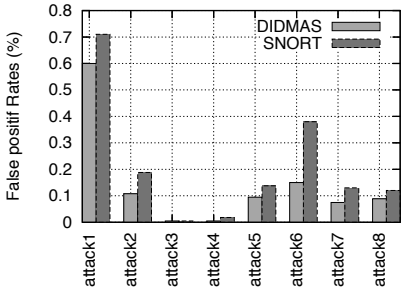

Figure 6: False alarm rates of DIDMAS *vs*. SNORT.

This result is confirmed by Figure 7 that shows that the detection rates of DIDMAS is by far better than SNORT configuration.

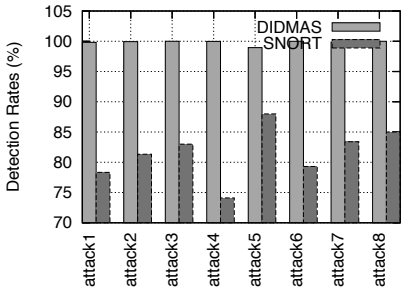

Figure 7: Detection rates of DIDMAS *vs*. SNORT.

Knowing that a main challenge of existing IDS is to decrease the false alarm rates (Eid et al., 2008), the main benefit of DIDMAS is to lower the false alarm rate, while maintaining a good detection rate.

8 CONCLUSIONS

In this paper, a novel distributed mobile agents IDS architecture, called DIDMAS was introduced, which incorporates existing IDS technologies. Indeed, the mobile agents technology is efficient for enhancing security, flexibility and cooperative detective ability of distributed IDS. In this respect, the DIDMAS system addresses the specific limitations described by the central approach. It shows the following advantages:

- *Performance scalability*: The mobile agents perform distributed search and analysis. Consequently, the analysis task related to each intrusion is inherently distributed, reducing the amount of

work done per host and removing a single host bottleneck;

- *Detection delay*: As we know, the integration of data from distributed detection stations takes a long time. Whenever a mobile agent can migrate to detection stations and analyzes data locally, it will significantly reduce the detection delay;
- *Extensibility*: DIDMAS guarantees that the system can be extended in a straightforward manner. For example, new rules, actions and task agents can be defined after the DIDMAS is initially deployed, without requiring any changes to the existing set of agents.

The carried out experimental results showed the effectiveness of the introduced approach and highlighted that DIDMAS realizes the scalability of mobile agent-based approaches, since it reduces the bandwidth consumption as well as the response time. Future issues for the present work mainly concern: (*i*) The consideration of the anomaly approach using the integration of data mining techniques (Helmer et al., 2003). (*ii*) Study of the security of the mobile agents themselves, which is one of the important issues that have to be addressed if system is to be deployed in the real environment. In this respect, a mobile agent itself can cause damage to a host or a host can do harm to the mobile agent (Jansen et al., 1999).

REFERENCES

Deeter, K., Singh, K., Wilson, S., Filipozzi, and Vuong, S. (2004). APHIDS: A Mobile Agent-based Programmable Hybrid Intrusion Detection System. In *Proceedings of the International Workshop on Mobility Aware Technologies and Applications MATA'04, Florianopolis, Bresil*, pages 244–253.

Denning, D. (1987). An Intrusion Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232.

Eid, M., Artail, H., Kayssi, A., and Chehab, A. (2008). LAMAIDS: A Lightweight Adaptive Mobile Agent-based Intrusion Detection System. *International Journal of Network Security*, 6(2):145–157.

Forgy, C. (1982). RETE: A Fast Algorithm for the many Pattern/many Object Pattern match Problem. *Artificial Intelligence*, 19(1):17–37.

Helmer, G., Wong, J., Honavar, V., Miller, L., and Wang, Y. (2003). Lightweight Agents for Intrusion Detection. *Journal of Systems and Software*, 67(2):109–122.

Herrero, A. and Corchado, E. (2009). Multiagent Systems for Network Intrusion Detection: A Review. In *Proceedings on the Computational International in Security for Information Systems, AISC 63, Berlin, Heidelberg*, pages 143–154.

Jansen, W., Mell, P., Karygiannis, T., and Marks, D. (1999). Applying Mobile Agents to Intrusion Detection and Response. Nist interim report - 6416, National Institute of Standards and Technology (NIST), Computer Security Division.

Ktata, F. B., Kadhi, N. E., and Ghèdira, K. (2009). Agent IDS based on Misuse Approach. *Journal of Software*, 4(6):495–507.

Li, C., Song, Q., and Zhang, C. (2004). MA-IDS Architecture for Distributed Intrusion Detection using Mobile Agents. In *Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004), Harbin, China*, pages 451–455.

Mo, X.-L., Wang, C.-D., and Wang, H.-B. (2009). A Distributed Intrusion Detection System Based on Mobile Agents. pages 1–5.

Mosqueira-Rey, E., Alonso-Betanzos, A., Guijarro-Berdinas, B., Alonso-Ríos, D., and Lago-Pineiro, J. (2009). A SNORT-based Agent for a JADE Multiagent Intrusion Detection System. *International Journal of Intelligent Information and Database Systems*, 3(1):107–121.

Outtagarts, A. (2009). Mobile Agent-Based Applications : A Survey. *International Journal of Computer Science and Network Security, IJCSNS*, 9(11):331–339.

Roesch, M. (1999). Snort Lightweight Intrusion Detection System for Networks. In *Proceedings of USENIX LISA'99*.

Singh, M. and Sodhi, S. (2007). Distributed Intrusion Detection using Aglet Mobile Agent Technology. In *Proceedings of National Conference on Challenges and Opportunities in Information Technology (COIT-2007) RIMT-IET, Mandi Gobindgarh*, pages 148–153.

Spafford, E. and Zamboni, D. (2000). Intrusion Detection Using Autonomous Agents. *The International Journal of Computer and Telecommunications Networking*, 34(4):547–570.

Wang, H., Wang, Z., Zhao, Q., Wang, G., Zheng, R., and Liu, D. (2006). Mobile Agents for Network Intrusion Resistance. In *Proceedings of the International Workshops on Advanced Web and Network Technologies, and Applications, APWeb 2006, Harbin , China*, pages 967–970.

Ye, X.-L., Zhang, Y.-C., Zhang, C.-L., Chen, C., and Huang, X.-Y. (2009). A Mobile Agent and Snort Based Distributed Intrusion Detection System. pages 281–285.