INTERNAL SECURITY ON AN IDS BASED ON AGENTS

Rafael Páez, Mery Yolima Uribe, Miguel Torres

Pontificia Universidad Javeriana, Bogotá, Colombia

paez-r@javeriana.edu.co
mery.uribe@javeriana.edu.co
metorres@javeriana.edu.co

ABSTRACT

An Intrusion Detection System (IDS) can monitor different events that may occur in a determined network or host, and which affect any network security service (confidentiality, integrity, availability). Because of this, an IDS must be flexible and it must detect and trace each alert without affecting the system´s performance. On the other hand, agents ina Multi-Agent system have inherent security problems due to their mobility; that's why we propose some techniques in order to provide internal security for the agents belonging to the system. The deployed IDS works with a multiagent platform and each component inside the infrastructure is verified using security techniques in order to provide integrity. Likewise, the agents can specialize in order to carry out specific jobs, for example monitoring TCP, UDP traffic, etc. The IDS can work without interfering in the system's performance. In this article we present a hierarchical IDS deployment with internal security on a multiagent system, using a platform named BESA with its processes, functions and results.

KEYWORDS

Mobile Agents, Multi-Agent Systems, Mobile Code, Security Techniques, Intrusion Detection System.

1 INTRODUCTION

The LAOCOONTE system corresponds to the implementation of a secure agent-based architecture proposed by Páez et.al. [4]. This project contextualizes the different advantages of using this architecture based on a multi-agent system and its convenient implementation of attack correlation. Likewise shows the corresponding use of internal security procedures to define specific methods for checking the identification of agents entities and platform, that can eventually attack or be victims of an attack in any of the scenarios previously identified.
Security in multi-agents systems is a subject that has generated large amounts of research and development, following the Bell-Lapadula theorem (Basic Security Theorem, BST) that states that a system is secure if its initial state is secure and if each state transition preserves security [1]. This paper proposes that a mobile agent system is secure if each component of the infrastructure is secure and their relationships with other entities are security preserving. In this way, the idea is to provide internal security to agents with security techniques based on a hierarchical infrastructure to control a set of agents, using basic math operations in order to not affect the system's performance. The results were obtained using as test environment an IDS over a multi-agent system with its processes, functions and results.

In addition to implementing the proposed security tools in an agent-based IDS, another objective of this paper is to prove its implementation viability, in any agent platform that handles sensitive information or which requires to ensure the integrity of agents belonging to the system, independent of the platform and the type of service provided by such a system, and without affecting significantly the system's performance.

2 BACKGROUND

2.1 LAOCOONTE project

LAOCOONTE´s objective is to demonstrate the validity of the proposed security mechanisms, testing them in an environment based on agents and as a case of study, using an IDS with the common features of such tools, but minimizing the risks of using mobile agents. Some of these attacks are: agent against agent, agent against platform, others entities against a platform and platform against agents. The last scenario, is the more difficult to solve because of when an agent comes into a platform, it has access to the agent's code. For this reason, the security mechanisms presented can be adapted to any system based on agents.

2.2 Intrusion Detection Systems

An IDS monitors different events that may occur in a determined network or host. Such events could affect any network security service (confidentiality, integrity, availability). The IDS must be flexible and must be able to detect and trace each alert, in such a way that does not affect the system's performance. A multi-agent system is a deployment option for an IDS, because it could allow the execution of intrusion detection tasks in the system, and also the agents can specialize themselves in order to carry out specific jobs, among other advantages [3].

The deployed solution is focused on attacks coming from the platform against a resident or itinerant agent, because it is one of the most difficult problems to solve in the agent programming paradigm.

2.3 Hierarchical IDS based on agents

In this context, a hierarchical scenario has a central agent controlling a set of subordinated agents, and each central agent is in charge of blocking, stopping, eliminating or cutting the connection with any malicious entity. It is important because the jobs can be distributed among agents and in this way it is possible to avoid bottlenecks.

There are four attack scenarios in an agent environment: agent against agent, agent against platform, other entities against a platform and platform against agents, where the last one, is the more difficult to solve because when an agent comes into a platform, it has access to the agent's code and the collected responses from other hosts. Because of this, the security mechanisms presented in this paper can be adapted to any system based on agents.

In the deployed system the proposed security mechanisms are implemented and take into account the principal functionalities and properties of an IDS, some of them are: Execution continues without constant supervision, its operation does not involve regular interaction with a system's manager or administrator, it doesn't produce overhead on the system or equipment, it records incidents depending on their level of relevance, records the sources and destinations of possible attacks on network segments and hosts and checks the identity of agents and platforms, using computing security techniques.

These objectives focus on addressing the security proposal in an agent platform without interfering with its performance. The project was developed following this set of activities: identification and specification of requirements for an IDS, IDS architecture design, and finally, the deployment of a prototype of an agent-based IDS with the BESA [17] platform. The following sections describe each activity.

2.4 Requirements for an IDS

The development focus of this project was oriented mainly on: different kinds of attacks, types of attack detection, standards and models of IDS and access control models.

In the first place, for intrusion detection methods, we specified two types: by signature recognition and estimation of probability (anomaly detection). Given the nature of the system and the benefits of signature recognition techniques that lies in the ease of upgrade the large number of signatures that are stored in the IDS' database [5], compared with the estimation of probability, which is not so accurate, and therefore, it's not as popular as signature recognition [6], we decided to handle the signature recognition method for detection in the proposed IDS system.

For IDS standards specification, we found that those standards depend on the implemented application. The first one is CIDF (Common Intrusion Detection Framework) [7], which specifies the creation of interfaces (APIs) and protocols that enable communication between different IDS. CIDF uses CISL (Common Intrusion Specification Language), as a communication language. The second one is CIDSS (Common Intrusion Detection Signatures) [8, 9], which defines a common XML data format for storing signatures from different IDS'. For the IDS architecture CIDF was used, making a change in the language used to identify attacks; instead of using CISL, CIDSS was used, because it is XML based, allowing flexibility in implementation and deployment. In addition, CIDSS is based on signature recognition, so, this language can be adapted to the project, and allows defining attacks that are invisible to the IDS most of the time, such as port scanning [7].

The kind of attack can be assigned to each particular rule; these classes are defined based on the type signatures in Snort Sourcefire [10,11] and, in order to work with this kind of signatures, we used its classification of attacks given in its signature field called "ClassType" .

On the other hand, the IDS must implement an access control model. This kind of model is a mechanism that allows a clear security policy. Based on different models, such as: Discretionary and Mandatory Access Control (DAC and MAC) models, as Bell-Lapadula [13], Clark-Wilson and Ferraiolo and Kuhn [12], Access Control based on roles (RBAC) [14,15], Chinese wall [16]. For the proposed specifications, it was determined that the models supported by it will be: The Chinese wall and Ferraiolo and Kuhn [12, 16], because these models allow flexibility to the agent monitor located in the highest level of the hierarchical architecture.

According to the IDS' classification [5], this system is classified as an anomaly detection system, hybrid, distributed and active, which focuses on verifying the integrity of different kinds of agents (Transceiver, Receiver and / or Monitor) at run time. These agents belong to the hierarchical multi-agent system.

In this way, if a host is malicious, in order to trace an attack or intrusion, the most important aspects to consider are:

- Agent location: The Collectors, which are located in different containers, each container can be located on a single different machine within the network. The Transceivers, which are given by the volume of flow of information processed. At least there will be a transceiver in each container in order to control each PC with it. The Itinerants, which are mobile agents – between containers and the monitors, the location of these agents depend on the physical characteristics and network design, because a monitor must meet the needs of the transceiver in order to change, depending on traffic and the number of registered agents to handle.

- Communication among agents: the Collectors are only able to communicate with Transceiver agents. The transceivers receive the information filtered by Collector agents and perform correlation functions of information. Finally, sends its results to the Monitor agent.
- Itinerants communicate with Transceiver and Monitor agents.
- Monitors receive information from the Itinerant and Transceiver agents and take the corresponding decisions, such as: sending an alarm, modifying the environment or cutting the connection with the malicious host.
- Internal Security of the system: Because of the IDS is developed as a multi-agent system, it must take into account factors that affect the identity of the agents, in order to provide system integrity and data integrity. The deployed technique consists in a check of a mark on an unreliable platform through matrices that correspond to the fingerprinting of each agent's identity, allowing reviewing the proper performance of the system in a determined period of time [4].
- Multiagent Platform: To the project's development, the multiagent platform selected was BESA [17]. BESA is a platform developed at Pontificia Universidad Javeriana with several features such as management of mobile agents and static directory of agents (white pages) and services (yellow pages), homeland security, among other features. BESA is based on the handling of containers which offer "a performance space where agents live" [17]. These containers work together and allow the identification, location and migration of agents transparently.
- Access: agents of the hierarchical IDS is shown in Figure 1, where T refers to a transceiver, I to an itinerant agent and An to collector agents.
- This communication schema is very important to keep the consistency and integrity of the system and that of the network where the IDS is located.

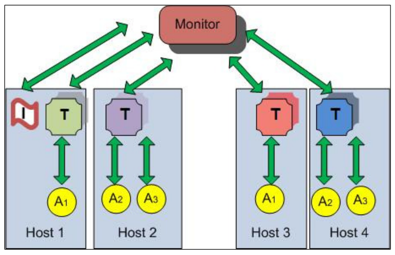


Figure 1. IDS Communication

2.5 IDS architecture design

The IDS's architecture (Figure 2) specifies the principal elements about interaction, communication and performance, like:

- Matrix of marks: Marks of Transceiver and Itinerant agents to verify their identity. They are handled by the Monitor agent.
- New attack signatures: signatures of distributed attacks which are recorded in the system and managed by the Monitor agent (XML file).
- Attack signatures: Snort1 type attack signatures for network traffic. They are handled by the Collector agent (XML files).
- Collector packages: a tool that collects information of a specific device in a host (such as JPCAP [18]).

Below, will be explained every detail.

[1] http://www.snort.org/

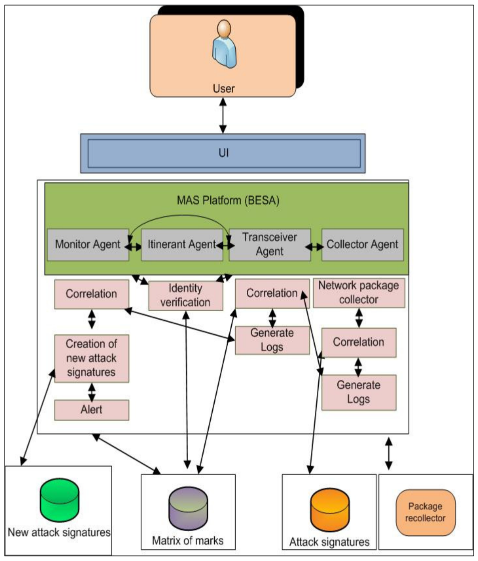
2.5.1 Data flow

Data in the architecture depends on each internal process. In the first place, the Collector agent performs a network scan where it is located, and checks the host against the database of attack signatures for possible matches. These possible attacks are sent to the respective Transceiver agent.

Then, the Transceiver agent checks the packages and correlates the events. Likewise, it finds new behaviors sent by Collector agents, to verify if there is some relationship (in source and destination IP addresses, ports or content) between them and find a previously defined pattern. If this agent finds one of these patterns, the Transceiver agent sends the results to the corresponding Monitor agent. Then, the Monitor agent performs the correlation among events that are sent to it, and if it finds some connection, it generates a new signature with these findings.

Figure 2. IDS's Architecture

Checking the internal identity of each Itinerant and Transceiver agent, the Monitor agent ensures the integrity of the different entities which belongs to the agent-based IDS, in our case, by the



implementation of the matrix of marks and the use of hash functions [4].

The matrix of embedded marks in the Transceiver and Monitor agents, allows a check of the marks used by the Itinerant agent, which identifies any changes on the marks, detecting any unexpected event. The use of hash functions ensures the integrity of each agent (Transceiver or Monitor), by associating a matrix of marks for each one in the system. In this way, each Monitor agent issues a fingerprint mark (matrix of marks) to each Transceiver agent and keeps a copy for itself in order to compare the math operations requested by others and then compare the result against the response given by the agent that is being verified.

It also generates a Cooperative Itinerant Agent, who is responsible of traveling through the network segment in order to send coordinates of the matrix at the request of the Monitor agent, and the corresponding Transceiver agent's response. This Transceiver agent's response is calculated to make simple math operations on these coordinates in its respective matrix; finally,

the Monitor agent performs the same calculations, keeping the timestamp in order to confirm the response.
On the other hand, each Transceiver agent is responsible for verifying the Itinerant agent's identity through their unique signature to allow the exchange of information with the host where the Transceiver is residing. This can ensure that each Transceiver agent keeps its integrity, and that it is running at the time of the request, and that its execution has not been stopped by an unauthorized entity.

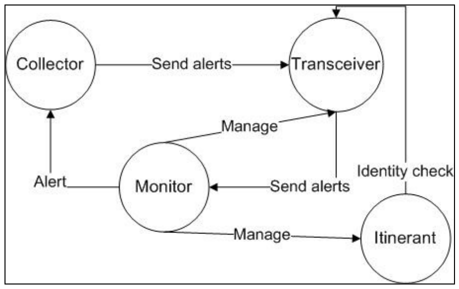


Figure 3. Multi-agent system

The multi-agent system, manages agents and their relationships (as shown in Figure 3), fullfilling the activities that where specified previously. These relationships are associated with the hierarchy and communication acts that are defined among agents and that are admitted between each type of agent.

2.5.2 Event Correlation

In an Intrusion Detection System like Snort, despite the high levels of detection and registered anomalies, there are many cases of false positive alerts [19], this means, identifying an attack when actually it is not a threat to the system. Several works address these issues in order to improve the function of the IDS and offer appropriate services based on the nature of attacks and threats [19,20].

Because most of these attacks are distributed and they have different forms of access and recognition, the correlation of events is important in the performance of an IDS, even though there is no consensus on how to perform this correlation, some authors propose matching up and compare information such as source and destination IP addresses, source port, destination and time between occurrences of events, in order to trace the threats or intrusions.

2.5.3 Related Work

Some researchers have focused in multi-agent systems and intrusion detection systems to address different objectives. For instance, to adapt the application to secure a specific kind of network, such as a mobile [6] or wireless network [7], or to maintain security in specific applications such as a web services based application [28], or to offer security for robust networked or distributed systems [29]. The principal solutions in those works are: adaptability to external changes, minimize the computational cost, defense from attacks effectively from insider or outsider threats, and improve task execution flow and improve the process of message passing between agents in specific kinds of networks.

IDS' System architectures, agent platforms, agent's organization and individual implementations are very different and unique for each case, but there is not a research work where a transversal problem, such as internal security, is faced.

Most research works rely on the external security to the networks where the systems are being deployed, but internal security is not a part of them. This security hole in multi-agent systems is covered by this project, because it provides a solution for external and internal security issues.

3 IMPLEMENTATION OF A PROTOTYPE OF AN AGENT-BASED IDS WITH THE BESA PLATFORM

3.1 Proposed algorithm

Based on CALM - Compromise and Attack Level Monitor [19, 21], which is an algorithm that correlates events and generates new attacks which don't have a predetermined pattern, the following actions were identified as a solution to correlate events for the system (Figure 3).

- Collector agent: Captures network traffic information, compares this information with Snort type attack signatures for possible attacks, notifies the Transceiver agent when a Collector agent finds an anomaly. This package is sent to the Transceiver agent too.
- Tools: Snort signatures in XML format, Boyer and Moore's algorithm [22] and JDOM (Extract XML files' data) [23].
- Transceiver agent: It receives anomalies from Collector agent, verifies matches (destination and source IP, destination and source port) among alerts received within a time period of 2 minutes and sends possible attacks and the packets involved to the Monitor agent.
- Monitor agent: It receives possible attack signatures, verifies matches between alerts, comparing the sent packages of Transceiver agents, verifies matches among alerts and the System Alerts file (if the Monitor agent has written on it at that time), in order to find the same attack behavior in different hosts: issues an attack alert to the respective hosts, verifying the order of the attacks in the same host: issues an attack alert and intruder alert to the respective hosts. Behavior isolated: issues an alert to the respective hosts. If the agent finds a new behavior, it keeps it stored as a new attack and if an attack is related with any existing behavior, this new one is added to the older.
- It saves the new attack signatures in a XML file according to a behavior. This XML file will have the following syntax:

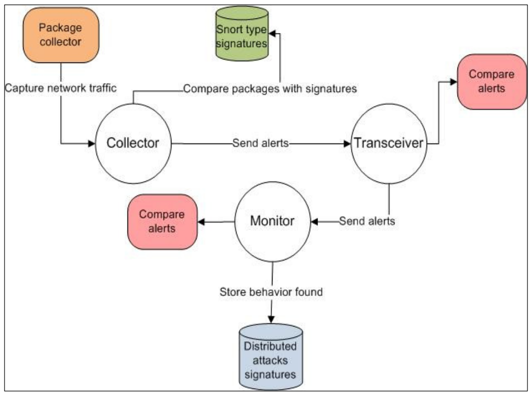
```
<?XML version="1.0" encoding="UTF-8"?>
    <Directives>Signatures distributed Attacks<Directive>
            <Directive>
                    <Signature> </Signature>
            </Directive>
    </Directives>
```

- Tools: JDOM: Extracts and stores data on XML signatures files [23].

In order to test the implementation, both internal security and detection attacks are implemented according to the specifications of the BESA platform. The base pattern of BESA is the agent's behavior, so the principal actions of each agent depend on these behaviors. In order to detect attacks, the guards concept defined in BESA was used. Each guard has an associated event and has an asynchronic mechanism, if the event occurs the guard is triggered and the event is processed.

Having all those technical MAS and IDS aspects, it is necessary to map them into agent's activities.

Figure 4. Event Correlation



For LACOCOONTE's execution of, both internal security and detection attacks are implemented accordingly to the specifications of the BESA platform. The base pattern of BESA is de agent's behavior, so the principal actions of each agent depend of these behaviors.

In the first place, to detect attacks, we use the guards in the involved agents:

- Monitor Agent:
  - GuardCreateTrans: Create Transceiver agents, which depend on a Monitor agent.
  - GuardRemoveTrans: Eliminates the Transceiver agent when it is necessary according to identity verification of the agent.
  - GuardSendMon: Receives alerts generated by Transceiver agent.
  - GuardMonCheck: in charge of performing correlation among alerts sent by the Transceiver agent. As this correlation progresses, we generated new attack signatures depending on behaviors found by these agents in order to verify distributed attacks.
  - GuardCheckTransceivers: Checks Transceiver agent state.

- Transceiver Agent:
  - GuardCreateReco: Creates Collector agents according to the protocol type needed (tcp, udp, icmp, ip).
  - GuardSendTrans: Receives the packages and alerts of Collector agents.
  - GuardCheckTime: Checks alerts generated by Collector agents and correlate them to find threads by a given host.

- Collector Agent:
  - GuardCapture: Allows collecting data from NIC (depending on the device number detected) and performs the correlation with the snort type signatures

In second place, for internal security there are the following guards:

- Monitor Agent:
  - GuardCreateItinerant: Creates an Itinerant agent.
  - GuardAlertHash: If the hash verification is negative, the agent creates an alert and eliminates the Transceiver agent.

- GuardNoticeMove: Itinerant agent notifies the Monitor agent in order that it moves itself from the current host to another host, and then the Monitor agent sends the coordinates to perform a hash check in the new host.
- GuardSendPosition: Defines the positions for the hash check for Transceiver and Itinerant agents.
- GuardRespF2: Verifies hash function one data and performs the respective calculations to send the response to the hash function two.
- GuardRequestT: Receives the Transceiver agent's request from other container.

- Transceiver Agent:
  - GuardSendMark: Sends the mark for verification of the agent's identity with the Itinerant agent.
  - GuardSaveCoord: Receives the matrix' coordinates to perform verification of the identity (hash).
  - GuardCheckHash: Checks that the mark corresponds to the one received by the Itinerant agent.

- Itinerant Agent:
  - GuardSendHash: Sends a hash to the Transceiver agent for identity verification (hash).
  - GuardF2: Sends hash function two data for its verification with the Monitor Agent.
  - GuardSaveCoord: Sends coordinates given by the Monitor agent to the Transceiver agent in order to verify the hash.
  - GuardReceiveRespHash: Receives hash verification response from the Monitor agent. If this response is correct, it sends a permission request to the Monitor so it can move to another host.

These guards are implemented for each agent and provide certain functions for the IDS (as shown in Figure 5).

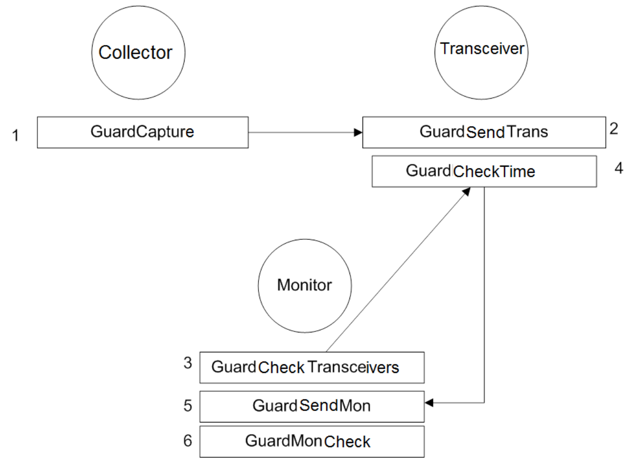


Figure 5. Events correlations – Agents guards

3.2 Results

The developed IDS on the BESA platform was launched to verify the treatment of external attacks and internal security. For this, we used a network with 12 hosts with the following distribution:

- Four BESA containers.
- Monitor Agents: 1.
- Transceiver Agents: 12 (one in each host).
- Collector Agents: 48 (four for each host (TCP, ICMP, UDP e IP)).
- Itinerant Agent: 1.

We used three scenarios, the first one without attacks, the second one performing scan operations on the hosts and the third one, accessing them. These attacks were performed using BackTrack [24] (Penetration Testing Distribution), an OS for ethical hacking. The principal measures on the scenarios were as follows (Table 1):

Table 1. Measures

Measure	**Average**	**Units**
Throughput	0,0247	Bytes/sec
Average packet size	184,7936	Bytes
Time between arrival of packets	53,5535	msec
Propagation Time	0,0316	seconds
Response time of the principal container	0,0144	seconds
Time that Itinerant agent takes to move from container to container	1,4810	seconds
Time to verify the hash	0,15418	seconds

Results such as time between arrival of packets, response time of the principal container and propagation time will increase as the attacks are increasing, because of the number of network packages generated by the attacks. Likewise, the Itinerant agent took more time for doing its normal work when the host was attacked.

Figure 6 shows the time between arrival of packets without attacks and Figure 7 shows it but with attacks. The time in the first case is shorter than in the second case because of the attacks.

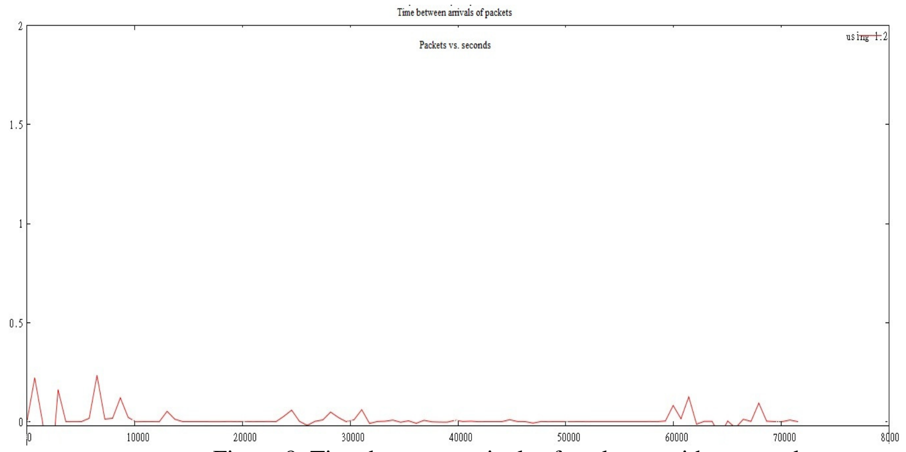


Figure 8. Time between arrivals of packets - without attacks

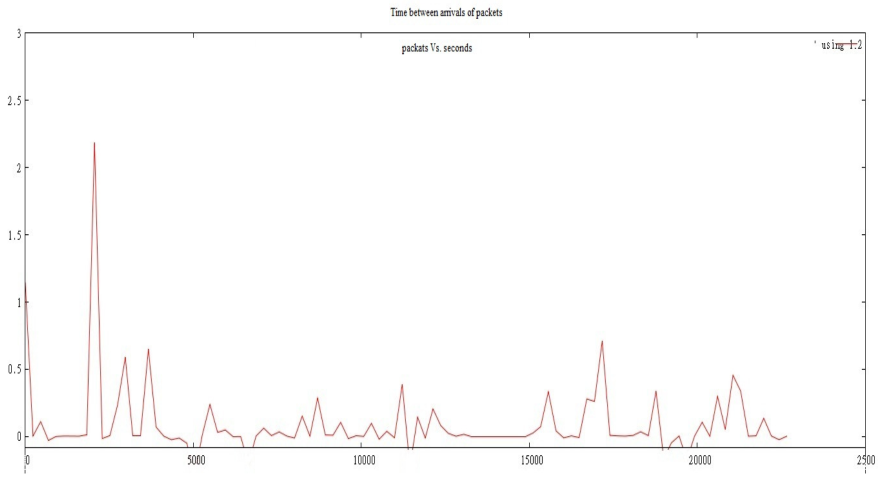


Figure9. Time between arrivals of packets - with attacks

Likewise, the hash verification, showed important differences between these two scenarios, without and with attacks. Figures 10 and 11 shows that the system can perform more hash verifications without attacks than when it is under attack.

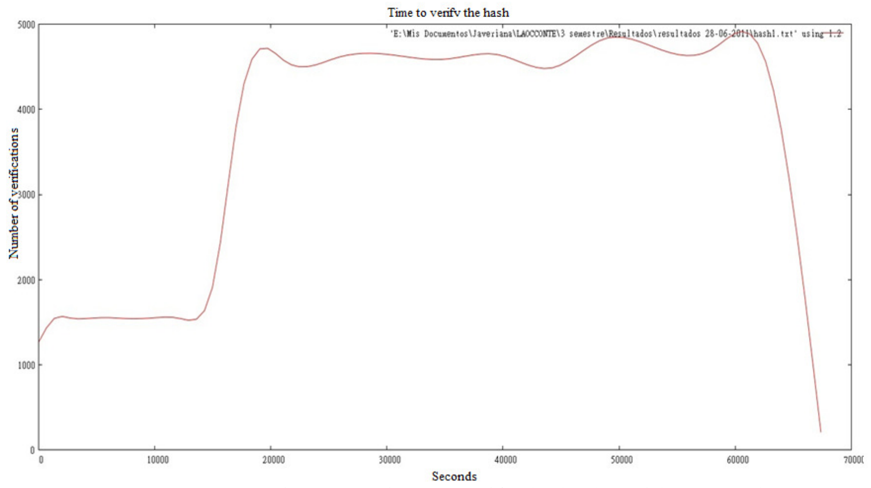


Figure10. Time to verify the hash - without attacks

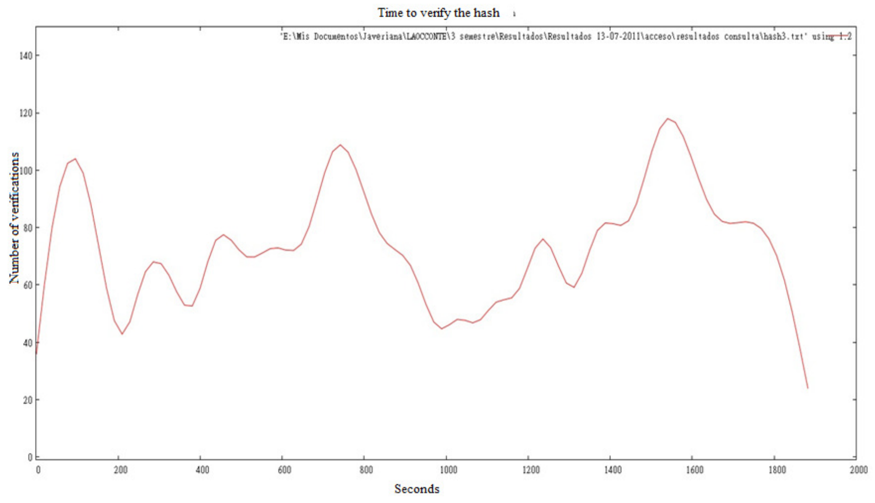


Figure11. Time to verify the hash - with attacks

Based on [25], the average time that the Itinerant agent takes to move from one container to another container and back is given by (Equation 1.):

$$t_m(L_1, L_n) = \left\{\left[3\partial(L_1, L_n) + \frac{B_{code} + B_{data} + B_{state}}{\tau(L_1, L_n)}\right] + t_{vm} + t_i + t_{va}\right\} * n$$

Where:

n: Number of nodes, B_{code}: Size of agent's code, B_{data}: Size of agent's data, B_{state}: Size of agent's state, t_{va}: Verification matrix time of the agent, ∂: network delay, τ throughput, t_{vm}: Average time of mark's verification and t_i: Report incidence time.

But with the obtained results, it is possible to deduce that is important to have in mind another variable: the hash verification time of the Itinerant agent (t_h), so, the new equation could be (Equation 2.):

$$t_m(L_1, L_n) = \left\{\left[3\partial(L_1, L_n) + \frac{B_{code} + B_{data} + B_{state}}{\tau(L_1, L_n)}\right] + t_{vm} + t_h + t_i + t_{va}\right\} * n$$

With this equation, the average time that the Itinerant agent takes to move from container to container and back in the implemented system is:

Table2. Results

Measure	**Value**	**Units**
n	12	Nodes
$\mathbf{B_{code}}$	13	KBytes
$\mathbf{B_{data}}$	582	Bytes
$\mathbf{B_{state}}$	3072	Bytes
$\mathbf{t_{va}}$	0,15418	Seconds
$\mathbf{t_{vm}}$	0,05	Seconds
$\mathbf{t_h}$	0,15418	Seconds
$\mathbf{t_i}$	0,02	Seconds
δ	0,04	Seconds

With this equation, the average time that the Itinerant agent takes to move from container to container and back in the implemented system is:

$$T_m(L_1,L_n)=6.757 \text{ seconds}$$

All these results show that the Intrusion Detection System does not overload the network, making it suitable to be deployed on any agent platform working on any kind of network. In this way, it was possible to prove the validity of the proposed techniques to protect agents without forgetting the Intrusion Detection System objectives. In this way the security techniques employed in the proposed architecture can be used in any agent platform.

4 CONCLUSIONS

An IDS based on a multi-agent system, can take advantage of characteristics such as independence of tasks, specialization of agents, management of platforms, among others, to have a system that offers benefits in performance, allowing the execution of more specialized functions depending on the type of agent, without interfering with the actions of others agents or with the system's normal activity.
Regarding of the treatment of new alerts, each event is isolated depending on the network protocol, and because of the proposed agent hierarchy; a better process can be developed to specify and clarify possible attacks (both local and distributed) within the system.

In this project, in order to monitor its environment, the IDS ensures the identity of each of the inside agents (Monitors and Transceivers), by offering a higher level of security for the IDS and to the system that it is being monitored. Likewise, to ensure information integrity during its operation, the IDS allows an effective operation of the system; because the proposed security techniques do not affect the system's performance.

The proposal of a matrix of marks provides an effective way to monitor the agent´s integrity in any agent based system using simple math operations, combining it with hash functions to determine if an agent was attacked or if a determined host suffered an intrusion. In this way, it is possible to assure that the proposed techniques are suitable to be deployed in any agent platform, providing not only agent's integrity but confidentiality and avoiding replay attacks or attacks from a malicious host against mobile agents.

REFERENCES

[1] L. J. LaPadula and D. E. Bell. Secure computer system: A mathematical model. Technical Report ESD-TR-278, vol.2, The Mitre Corp., 1973.

[1] R. Páez, M. Torres. "Laocoonte: An Agent Based Intrusion Detection System". International symposium on Collaborative Technologies and systems, COLSEC09, ISBN: 978-1-4244-4585-1 May 18-22, 2009. Baltimore, Maryland-EEUU.

[3] H.S. Nwana., Software Agents: An Overview, Knowledge Engineering Review, 11(3),1996, 1-40.

[4] R. Paez, J. Tomas-Buliart, J. Forne, M. Soriano (2008). "Securing Agents against Malicious Host in an Intrusion Detection System". LNCS, 5141:94-105. ISSN: 0302-9743

[5] Ghorbani Ali A., Lu Wei, Tavallaee Mahbod; ISBN 978-0-387-88770-8, e-ISBN 978-0-387-88771-5, DOI 10.1007/978-0-387-88771-5. Springer New York Dordrecht Heidelberg, London. 2010.

[6] Frederick and K. Kent, "Network intrusion detection signatures,". Available: http://www.securityfocus.com/infocus/1524

[7] I. DARPA, "The common intrusion detection framework (CIDF)". 1999. Available: http://gost.isi.edu/cidf/

[8] A. Wierzbicki, J. Kalinski, and T. Kruszona, "Common Intrusion Detection Signatures, standard (CIDSS)". 2008. Available: http://tools.ietf.org/html/draft-wierzbicki-cidss-05

[9] "CIDSS: XML Schema" Available: http://xml.coverpages.org/appSecurity.html#cidss

[10] Sourcefire, "Snort". 2010. Available: http://www.snort.org/

[11] Alfon, “Sistemas de detección de intrusos y SNORT”. 2008. Available: http://seguridadyredes.nireblog.com/post/2008/01/23/sistemas-de-deteccion-de-intrusos-y-snort-ii-creacion-de-reglas-ii-opciones-de-las-reglas
[12] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli, Role Based Access Control, 2003.Available: http://books.google.com.co/books?id=48AeIhQLWckC&printsec=frontcover&source=gbs_slider_thumb#v=onepage&q&f=false
[13] RedHat, “The Bell-La Padula model (BLP)”. 2008. Available: http://docs.redhat.com/docs/es-ES/index.html
[14] V. GUARD, “Control de acceso basado en roles”. Available: http://www.visual-guard.com/download/VisualGuard-Detailed-Features-SP.pdf
[15] Sánchez, M.; Jiménez, B.; Gutiérrez, F. L., Paderewski, P; Isla, J. L. "Modelo de control de acceso en un sistema colaborativo". Actas VII Congreso Internacional de Interacción Persona-Ordenador. pp: 227 - 237 (2006)
[16] E. Ruiz, A. Rivera, D. Quintero, and E. Hernández., “Seguridad y protección en sistemas operativos - políticas de seguridad”. 2009. Available: http://www.seguridadso.netai.net/index.php?option=com_content&view=article&id=53&Itemid=65
[17] E. Gonzalez, C. Bustacara, J. P. Garzon, M. Torres, and D. Ahogado, Desarrollo de aplicaciones basadas en sistemas multiagentes, editorial Javeriana, Ed., 2007, no. 978-958-683-871-4.
[18] K. Fujii, “Jpcap.”. Available: http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/
[19] A. Párrizas, “Propuesta de una arquitectura de sistemas de detección de intrusos con correlación,” Ph.D. dissertation, Universidad de Valencia, Escola Técnica Superior de Enginyeria, Valencia, 2005.
[20] L. Coppolino, S. DAntonio, M. Esposito, and L. Romano, “Exploiting diversity and correlation to improve the performance of intrusion detection systems”. 2009.
[21] F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer, “A comprehensive approach to intrusion detection alert correlation”. IEEE Transactions on dependable and secure computing, 2004: pp. 146-169.
[22] Steeb, Willi-Hans, Solms Fritz. “C++ programming with applications in administration, finance, and statistics”. World Scientific Publishing Co, Pte. Ltd. ISBN: 981-02-4066-X. 2000.
[23] J. Hunter, “Jdom”. Available: http://www.jdom.org
[24] http://www.backtrack-linux.org/
[25] R. Páez, C. Satizábal, and J. Forné, “A performance model to cooperative itinerant agents (CIA): A Security Scheme to IDS,” in Second International Conference on Availability, Reliability and Security. IEEE Computer Society, 2007.
[26] JADE, Java Agent Development Framework. Available: http://jade.tilab.com/

Authors

Rafael V. Páez PhD. works at Pontificia Universidad Javeriana and is one of the head of in the research group SIDRe in distributed systems and networks area.

Mery Yolima Uribe Rios MSc is a systems and computing engineer and magister, how works and researches in networks, agents systems, data security, digital television, security and grid computing.

Miguel E. Torres M. Systems Engineer from National University of Colombia (1999), Master of Science in Computer Science, Mississippi State University (2003), Associate Professor at the systems Engineering Department at Pontificia Univesidad Javeriana and member of the ISTAR research group.