6

An Agent Based Intrusion Detection System with Internal Security

Rafael Páez
Pontificia Universidad Javeriana, Bogotá
Colombia

1. Introduction

Intrusion Detection Systems (IDS) are software or hardware products that automate the analysis process of traffic on a network or a host. and they are a complementary security tool in computer networks; it can be deployed at different points depending on the application, host or network segment to be monitored. Accordingly to its location, the IDS must be parameterized in a different way, for example, an IDS located in a Demilitarized Zone (DMZ) must be more flexible than an IDS located inside the internal network to reduce false alarms or to avoid system overload, allowing intrusions without generating an alarm. Likewise the IDS can receive different kinds of attacks if it is located in a DMZ or in the intranet zone.

Due to the increasing rate of attacks, Intrusion Detection Systems has become a complementary and mandatory security tool to any organization, and in addition it is useful to perform forensic analysis procedures in order to complement the IDS use. An IDS performs passive monitoring and captures information to be analyzed subsequently, it can launch an alarm to a server or send an email warning about possible intrusions but it cannot modify its environment, otherwise it is named Intrusion Prevention System (IPS). An IPS responds in real time if an intrusion is detected, the IPS takes an action modifying its environment; it could modify the firewall by closing a suspicious connection, or reconfigure the router, etc.

In the last two decades many research studies about technologies, architectures, methodologies and technologies have been proposed in order to increase the IDS effectiveness. One of them is the agent technology. Agents offer many advantages to IDS like scalability, independence, solution to complex tasks, reduction of network traffic, etc. For these reasons, agents are appropriate but they have inherent security drawbacks and they must be tackled. There are four risk scenarios: agent against agent, agent against platform, others against agent and platform against agent. The most difficult problem to face is the last one because the platform can be accessed by the agent code and it could eventually modify it. The internal security of a system is treated in few research works and it is a critical situation because it is a barrier for attackers, and one of their first challenges is to cheat or attack defence systems.

In previous studies (Páez et al., 2005), many IDS architectures based on agents were analyzed, and it was possible to conclude that it was necessary to propose techniques to protect internally an agent based IDS, by securing its different entities. The new IDS's

architecture proposed is named Laocoonte and it is a work on progress, using an open framework developed in the Pontificia Universidad Javeriana named BESA (*Behaviour-oriented, Event-driven and Social-based agent framework*) (González et al., 2003).
BESA is a platform developed in Java in order to take advantage of its portability. BESA allows the deployment of a Multi agent Systems (MAS), its environment is object oriented and offers multithreading and transparency in communications among entities. The abstract model of BESA is based on three fundamentals concepts: Behaviour-Oriented, Event-Driven and Social-Based. Moreover BESA uses a set of containers located in each host on the system, and each container contains several agents performing specific functions. Likewise, each agent has a set of guards; each guard has two associated procedures. The first one is executed automatically by the firing mechanism to verify a boolean condition. The second is executed by behaviour when the guard has fired. The code implementing behaviour is generic and can be written independently of the agent specificity. Behaviour receives an event, and depending on its type executes a program. In this way, the internal structure of each agent can be constructed.

2. Important

An IDS is a software or hardware tool which allows to detect and warn about an attack or intrusion from authorized or non authorized users to the system which is being protected. The intrusion can be from inside or outside the protected system and it can be intentional or accidental. An intrusion is an unauthorized or non wanted activity that puts in risk any security service like confidentiality, integrity and/or availability of the information or computer resource.

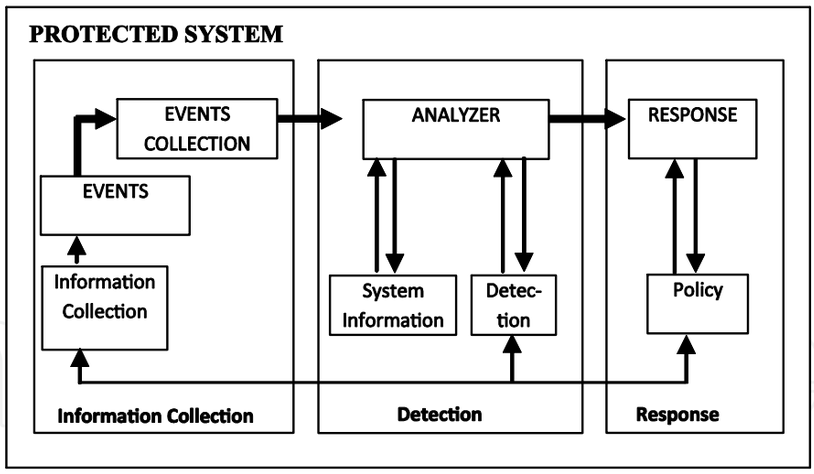


Fig. 1. Basic IDS architecture

The basic architecture of an IDS consists of three main modules: The information collection module, detection module and Response module (Páez et al., 2006). The information collection module contains: an event generator, an event collection sub-module and a reference to detect an intrusion (Database of attacks, behaviour profile, etc.). Fig. 1 illustrates the basic architecture of an IDS.
The event generator sub module can be the operating system, the network or an application. The events generator sub module sends the information to the event collection sub module, and then it sends the traffic to the detection module.

The detection module has an analyzer or sensor which decides if actually there is an intrusion. Finally, the detection module sends the suspicious traffic to the response module which acts accordingly, based on a policy database.

There are many ways to classify an IDS [Debar et al., 1999]. Debar et al classify IDS´s accordingly to the detection method, the behaviour on detection, the audit source location and usage frequency.

The main concern of researchers in Intrusion Detection Systems is to avoid false negative and positive events because they are common drawbacks to IDS's, a false positive occurs when the IDS detects an intrusion and reacts accordingly to its configuration but actually is not malicious traffic, and a false positive occurs when the IDS does not alert about a real intrusion. Both factors affect negatively the IDS but maybe the second one is the most dangerous because it would allow an attack, but the first one would reduce the IDS' reliability and distracts network administrators from real issues. Thus, an IDS must to minimize false negative and positive events and keep updated the system. Moreover, a well configured IDS is also another resource to perform forensic analysis through its log files, and depending on its location it can offer many information about inside or outside attacks. An IDS is a complementary security tool; in neither case, it would replace other security tools like firewalls, antivirus, etc.

Moreover false positive and negatives events, there are many other problems regarding IDS's such as evasion, in this case an intruder may try to avoid detection by fragmenting communication packets, in this way, the signature database would not match what the IDS expects to see. Fragmented packets are put back together into its original form and the attack is performed without being detected by the IDS. Another problem to solve is related to cryptographic techniques; because a network based IDS cannot access raw data and it could not detect a possible attack or intrusion. Finally, depending on the used technology other problems could arise inside the IDS.

We have focused our work in an IDS based on autonomous and mobile agents. Agents have many characteristics appropriates to IDS (Balasubramaniyan et al., 1998) because they can perform simple tasks individually without supervision (i.e. traffic monitoring; event correlation of results), they can solve complex tasks such as detect an intrusion or attack. Agents can be classified according to their mobility, they can be static or mobile; according to their behaviour, they can be deliberative or reactive and according to their attributes, they can be autonomous, cooperative or intelligent. By combining these properties, they can be classified as: collaborative agents, interface agents, information or internet agents, hybrids or heterogeneous agents.

On the other hand, mobile agents have inherent security drawbacks due to their nature, because of this its use has not been widely spread in industrial environments. These security problems can be summarized in the following four scenarios (Fig. 2): agent against agent, agent against platform, others against platform and platform against agent (Jansen, 2000). These scenarios have common attacks and they are: masquerade, denial of service and unauthorized access; A masquerade attack occurs when an entity (platform, agent, etc.) claims the identity of another entity in order to impersonate it and obtain access to services and resources; A denial of service attack occurs when an entity consumes an excessive amount of computing resources; this attack can be launched intentionally or unintentionally. The denial of service can occur on a resource (i.e. printer) or on information; An unauthorized access attack occurs when an entity access resources' information to which it has not granted permission. Additionally, attacks from agent to agent scenario can suffer repudiation attacks, this occurs when an entity which have participated in a transaction or communication, later it

denies its participation. In the case of attacks from others entities to agent, exists the copy and replay attack, it means that a malicious entity intercepts an agent or agent's message and then tries to copy the agent or message in order to clone or retransmit it. Finally, in the platform to agent scenario, the eavesdropping attack is a passive attack and occurs when an entity, in this case the platform, intercepts and monitors secret communications.

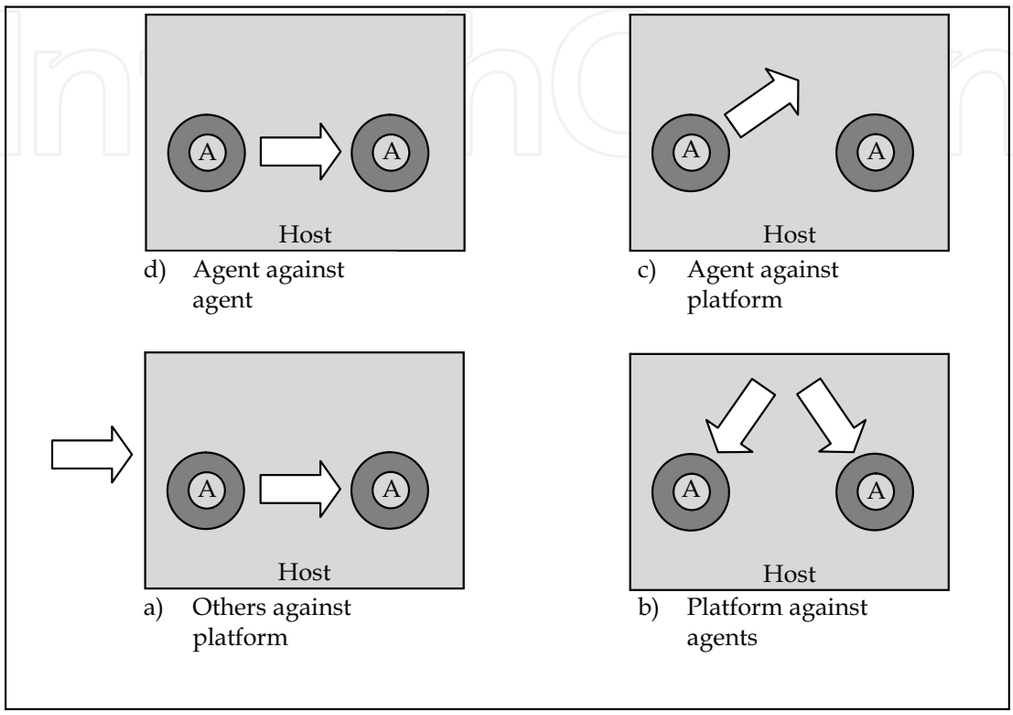


Fig. 2. Different attacks in an agent environment

There are some proposals to protect both agents and platform (Jansen, 2000) (Collberg, 2002) (Knoll et al., 2001), but many of them have general purpose to any agent based systems. Some of these proposals are not viable to all systems, because they add overload to the system or use cryptographic techniques which have significant drawbacks in response time. The most difficult problem to solve is the attack from a platform against agents. This is because the platform has access to the data, code and results of the agents located on it. In this way, if a host is malicious, it can perform an active or passive attack. In the case of a passive attack, the host obtains secret information as electronic money, private keys, certificates or secrets that the agent uses for its own security requests. On the other hand, to perform an active attack, the host would be able to corrupt or to modify the code or the state of the agents. A malicious host can also carry out a combination of passive and active attacks, for example, by analyzing the operation of the agent and applying reverse engineering to introduce subtle changes, so the agent shows malicious behaviour and reports false results. Some authors consider two phases in a Mobile Agents System (MAS), transferring phase and running phase (Xiaobin, 2004). Thus, an agent could be attacked either when it is moving or when it is located in a malicious host performing its job. In an IDS based on mobile agents the threats are transferred, but they must be treated in a particular way. It is important to provide internal security in order to avoid or detect an attack but without adding overload to the system and using simple techniques which do not consume excessive response time.

We have centred our research in the worst scenario: attacks from platform against agent. Several IDS based on agent architectures have been analyzed in order to investigate about its protection mechanisms, in order to provide internal security, because all of them must face the same issues. Among them are: IDA [Asaka et al., 1999], MA-IDS [Shao-Chun et al., 2003], JAM [Salvatore et al., 1997] and Sparta system [Kruegel et al., 2001] and others systems based on agents.

2.1 IDA´s architecture

The IDA´s (Intrusion Detection Agent system) architecture (Fig. 3) uses a sensor agent that resides at a node in search of an MLSI (Marks Left by Suspected Intruder) from the system's log; and upon discovery, notifies the Manager who dispatches a tracing agent to the host where the MLSI was detected. This agent activates an information-gathering agent. The information-gathering agent collects, in a independent way, information related to the MLSI on the target system. The information-gathering agent returns to the Manager with its results and logs them on a bulletin board. The tracing agent moves to the next targeted system and activates a new information gathering agent. Meanwhile, the bulletin board integrates information collected about the intrusion, using the gathered information from several involved agents. So, bulletin board and message board are a common use area and can be accessed by tracing agents and information-gathering agents. The manager accumulates, analyzes and weighs the information entered on the bulletin board by the mobile agents and, if the weights exceed a threshold, it concludes that an intrusion has occurred and issues an alarm.

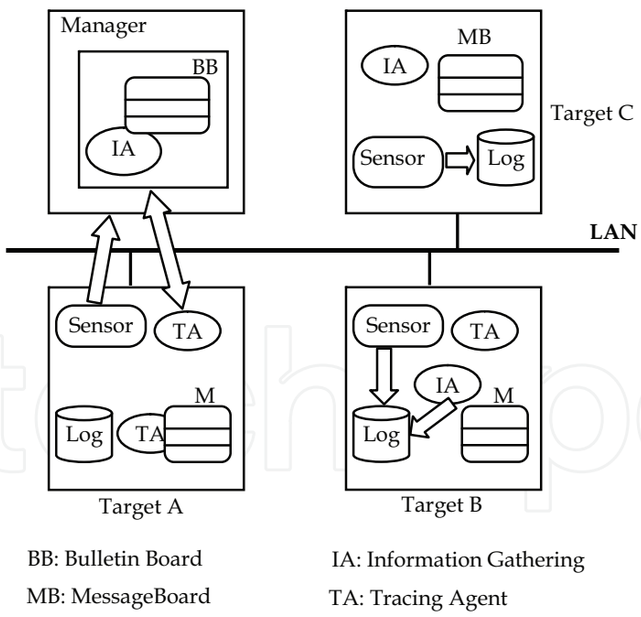


Fig. 3. The IDA´s architecture

2.2 MAID's architecture

The MAID's (Mobile Agents Intrusion Detection System) architecture is a distributed IDS which includes a Manager, Assistant Monitor Agent, Response Monitor Agent and a Host Monitoring Agent. There is a Monitor Agent (MA) in each host. If an agent detects an intrusion, it reports it

directly to the Manager. The host Monitor Agent can request aid to the manager. If a Manager receives an Assistant's request, it will send an Assistant MA to patrol the network in order to gather information, and thus to determine if some suspicious activities in different hosts can be combined to carry out a distributed intrusion. Finally, the manager analyzes the gathered information and if it detects a plausible distribution intrusion it will dispatch a Response MA. The manager is a central point of correlation and therefore, if it is located by any attacker, the system would be in a dangerous situation. The mobile agents (Assistant MA and the Response MA) are encrypted using a symmetric key algorithm with a one-time session key. Then, this session key is encrypted using a public key algorithm, this turns the MAIDS's runtime environment slow. The MAID's architecture is shown in the Fig. 4.

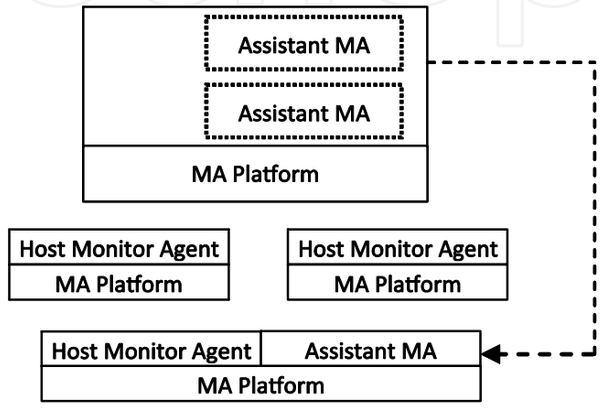


Fig. 4. MAID's architecture

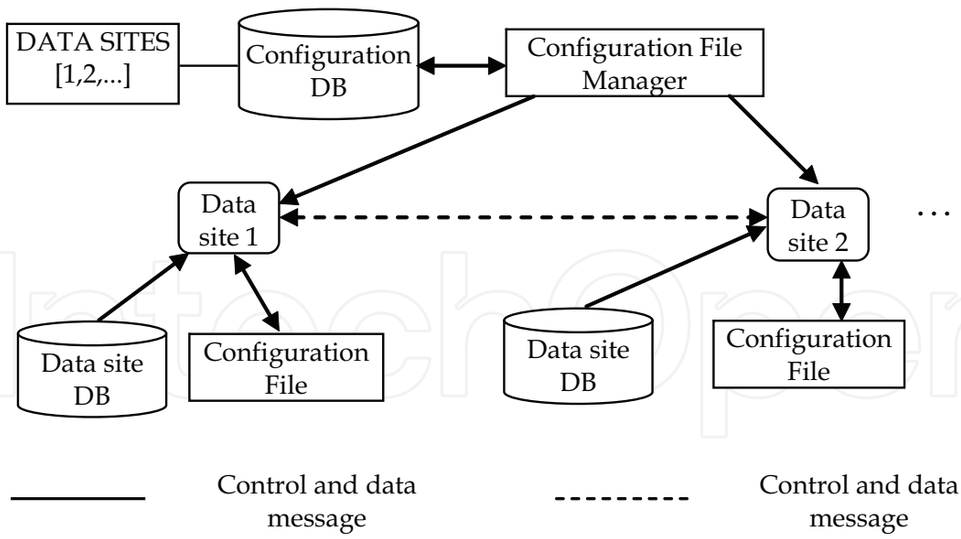


Fig. 5. JAM's architecture

2.3 JAM's architecture

The JAM's (Java Agents for Meta-learning) architecture (Fig. 5) applies learning and meta-learning agents to distributed database sites and the system's configuration is maintained by a Configuration File Manager (CFM), which is a server that provides information about the

participating data sites and log events, and the CFM is in charge of keeping the state of the system up-to-date.

2.4 Sparta's architecture

The Sparta (Security Policy Adaptation Reinforced Through Agents) system uses correlation mechanisms among agents in order to identify and relate suspect events in a distributed manner. Three different types of users can be recognized by a host in this system: Administrators, regular users and owners of agents. Moreover, each node belonging to the system, has a Loacal even generator (sensor), event storage component, mobile agent platform and agent launch and query unit (optional). Sparta can detect network intrusion in a dynamic environment using a pattern language (Event Definition Language-EDL) to express intrusions in a declarative way. Each host has at least a sensor (local event generator), a storage component and the mobile agent platform installed (Fig. 6). Sensors monitor interesting occurrences on the network or at the host itself and this information is stored in a local database for later retrieval by agents; the mobile agent subsystem provides an execution environment, communication layer, clock synchronization and a directory service.

The Sparta's architecture provides mechanisms to protect agents and platforms implementing a Public Key Infrastructure (PKI). Agents are encrypted and signed during its transportation and authenticated on arrival.

The use of a public key introduces delay on the system, which is not a desirable attribute on an IDS.

We have analyzed others proposals [Yonggang Chu et al., 2005], [Zaman & Carray, 2009] but none provide internal security or there is no information is available. So, we propose complementary techniques in order to avoid and/or prevent an intrusion or attack providing internal security to an agent based IDS. Few authors treat the internal security of an IDS and it is in our opinion an important factor because it is one of the first barriers that an intruder would find when trying to access a network. So, the IDS become a susceptible objective to be attacked.

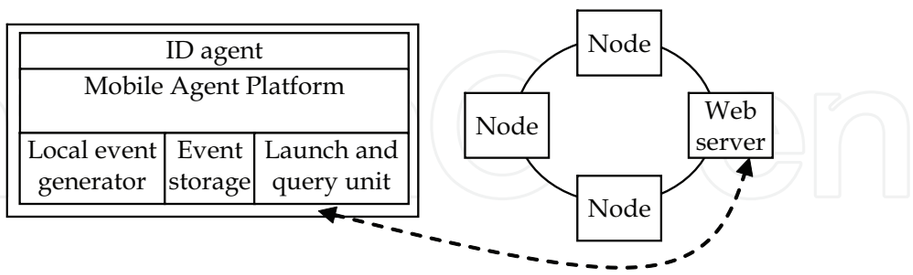


Fig. 6. SPARTA's architecture

Our proposed system is named Laocoonte. Laocoonte is a distributed and hierarchic IDS based on AAFID project [Balasubramaniyan et al., 1998] but we have made modifications to the architecture in order to increase the internal security level. We have added Cooperative Itinerant Agents (CIA) as an intermediary agent to request for a particular mark, all entities are mobile inside a network segment. Moreover we use different techniques to prove the integrity on central point of failure, such as transceivers or monitors. Laocoonte uses autonomous and mobile agents but its main characteristic is its internal security.

3. Laocoonte's architecture

Laocoonte is a hierarchical IDS with three main levels. Each level has different entities performing a particular job in a cooperative manner, but the entities which belong to the same level cannot communicate directly among them. Each entity is an agent but with different characteristics and duties. Fig. 7 depicts Laocoonte's architecture. In the lower lever are located collector agents, in the middle level are located the transceivers and in the higher level are located the monitors. Finally, there is an interface which is not considered as component of the architecture but it is necessary for the system administrator in order to set different parameters on each entity to configure the system.

On each host, there are one or more collector agents, monitoring the information which travels on the network or host. Each collector agent is specialized on a particular kind of traffic (TCP, UDP, ICMP). Collector agents determine if a determined traffic is malicious based on a database of attacks. When it occurs, the collector agent sends this information to the transceiver. So the only two functions performed by a collector agent are: monitoring and report its findings.

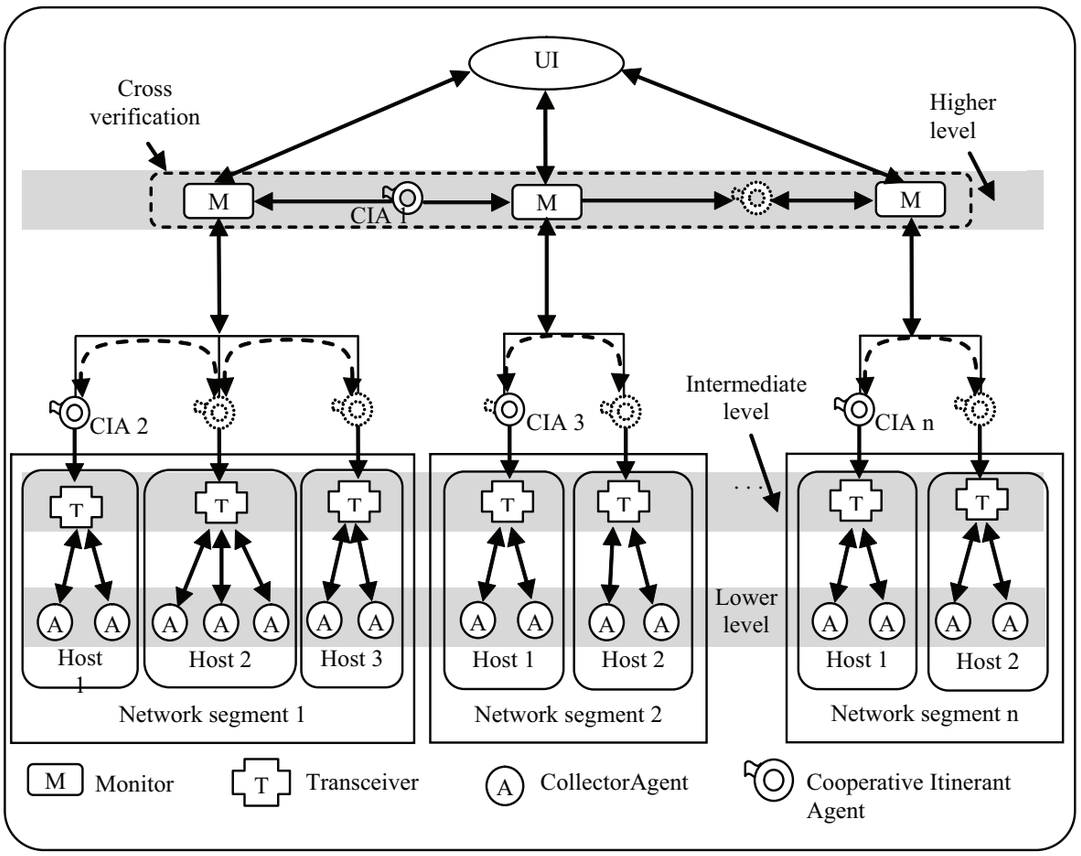


Fig. 7. Laocoonte's architecture

Transceivers are agents located in the intermediate level, and they must take the findings of the collector agents and perform correlation of events in order to identify a possible attack or intrusion in the host on which it is located. There is only a transceiver for each host and it can detect an intrusion or attack in a local way. Moreover, a transceiver can perform control

functions on collector agents (create, stop... etc.). If the transceiver identifies an attack, it reports this information to the corresponding monitor.
Monitors are agents located in the upper level of the infrastructure, and they are in charge of controlling the transceivers. When a Monitor creates a transceiver, sends it to a host and waits for results. The Monitor collects the information from its transceivers and performs high correlation functions, in this way it can detect a distributed attack because the information comes from different network segments. A Monitor could depend of another Monitor; it is because in large distributed systems a monitor could receive large amounts of information. Thus, to avoid any overload it is necessary to create another level.
Cooperative Itinerant Agents (CIA) are mobile entities and they are created by a Monitor. CIA agents are part of the internal security scheme. It must ask for a particular mark to a transceiver located in each host and sends the response to the monitor, and then it continues its itinerary inside a network segment. There is an exception in the upper level because monitors do not have another superior entity performing control activities. In this case, the CIA agent must travel among different network segments to ask for the mark to each monitor.
We have discussed the general IDS architecture. Next we will present and explain the internal security scheme.

4. Laocoonte's security scheme

The objective of this proposal is to ensure the integrity of a mobile agent which could be attacked by an untrusted platform. The proposed technique, consist on the inclusion of a marks matrix inside the code of the entities which are going to be verified (transceivers and monitors), and also store a copy of the verifying entities. The matrix is composed by a set number of cells which can be determined by the protected hosts' processing capabilities. In each cell of the matrix there is a prime number (sub-mark). The Monitor which will behave as a verifier, informs the cooperative agent which are the coordinates that are to be verified from the matrix; in this way what is really requested is not a mark itself, but the result of an operation between primer numbers (the sub-marks) which are located in the given coordinates. The sub-marks involved in the operation are different every time a request is generated, and every time a set of coordinates is used, the position in the matrix gets blocked from being used again in the subsequent processes. The result is then masked though a module function and a randomly generated coefficient value generated by the verified entity. when the cooperative agent warns the Monitor about its transfer to a host, the Monitor sends the coordinates which will be used in the operations, the CIA agent serves as an intermediary and sends the petition to the entity that is going to be verified (transceiver or monitor), then the entity returns the product of the requested sub-marks.
The reason to use module functions is to mask the result of the operation performed on the given sub-marks. Also the prime number product increments security by means that if the result is captured by an attacker, it would not be trivial to find the values, and then protecting the information that is going to be received by the verifying entity, in this case the Monitor which controls the verified entity. The Monitor knows the module that has been applied to the verified entity, and then it would only have to repeat the product of the two sub-marks and then apply the correspondent module, and finally compare the results.
The Monitor's request is also performed using randomly generated prime numbers as coefficients from a previously established module. The result of the product of such given

prime numbers and the random variable are in fact the coordinates of the sub-masks in the matrix. In this way if the result is intercepted, the attacker would not be able to identify the requested coordinates.

The procedure since the creation of the entity (monitor or transceiver), through the verification of it is as follows:

1. The Monitor generates the transceiver.
2. The Monitor assigns the prime number matrix; it inserts it as a mark to the transceiver and keeps a copy of the matrix and its contents.
3. The Monitor generates a prime number x, which will be used as module for the function (1). The transceiver also stores this value.
4. The Monitor sends the transceiver to the correspondent host.
5. When the CIA agent announces the host's change, the Monitor generates the random numbers and solves function (1). In this way it sends the masked coordinates of the sub-marks matrix.
6. The CIA agent requests the result of function (2) to the verified entity (the transceiver), giving it the masked coordinates generated by the Monitor.
7. The transceiver gets the module x of the given coordinates. With this information it can obtain the values of the sub-marks in those coordinates.
8. The transceiver generates t as a multiplicative factor and solves function (2) with the given sub-marks. then it send the results to the CIA agent
9. The CIA agent resends the results to the Monitor.
10. The Monitor receives the result from the CIA agent and applies the module y having S1. Then solves the product of the sub-marks and having then S2.
11. The Monitor then compares S1 and S2. If the values match, then there is a high probability that the transceiver has not been compromised.

And now we proceed to present the mathematical definition of the variables used in Laocoonte's proposal.

Being M a matrix of dimensions fxc, where f means rows and c columns, and the set represented by (f_i, c_j) indicates a cell in the matrix where $(i, j) = (0..f, 0..c)$. In each cell a randomly generated prime number is stored by the corresponding Monitor. Each prime number is a sub-mark of the main mark (the matrix). The verifying entity (the monitor which controls the CIA Agent and the transceiver) will request a set of sub-marks in order to test the integrity of the agent (transceiver or monitor). The requested sub-marks will be different each time the request is generated (because of the actual blocking of previously used coordinates). when the CIA agent arrives to a host, the Monitor sends the coordinates of the sub-marks that wants to verify. The CIA agent issues a request to the transceiver, and when it gets an answer it resends it to the Monitor. The Monitor uses the information on the request in function (1):

$$f_1(x) = \{(x * p_1 + f_1), (x * p_2 + c_1), (x * p_3 + f_2), (x * p_4 + c_2), \ldots, (x * p_k + f_n), (x * p_n + c_n)\} \quad (1)$$

Where x is an integer greater or equal to zero. The value of x corresponds to a fixated value, randomly generated by the Monitor and known by the transceiver. This value represents the module used by the Monitor to mask the values. The values p_1, p_2, p_3, p_k, p_n are integer random variables generated by the monitor, and the variables $f_1, c_1, f_2, c_2,$ correspond to the two sets of coordinates of the sub-marks in the matrix generated randomly by the Monitor

every time the cooperative agent issues a request of marks to the transceiver. In some cases the set of requested sub-marks can be more than two. This coordinates are given to the transceiver and it calculates the correspondent module to get the sub-marks from the matrix. Knowing the coordinates the sub-marks values w_1, w_2 are multiplied in function (2):

$$f_2(y) = \{(y * t) + (w_1 * w_2)\} \tag{2}$$

Where y corresponds to a fixed number which represents the previously assigned module. This model is known by the Monitor and it is used to mask the sub-marks; in this way there will not be inferred information about the given values of the sub-marks. Parameter t is a randomly generated integer by the transceiver every time it uses function (2). The agent receives this parameter and sends it back to the Monitor. The Monitor then applies the

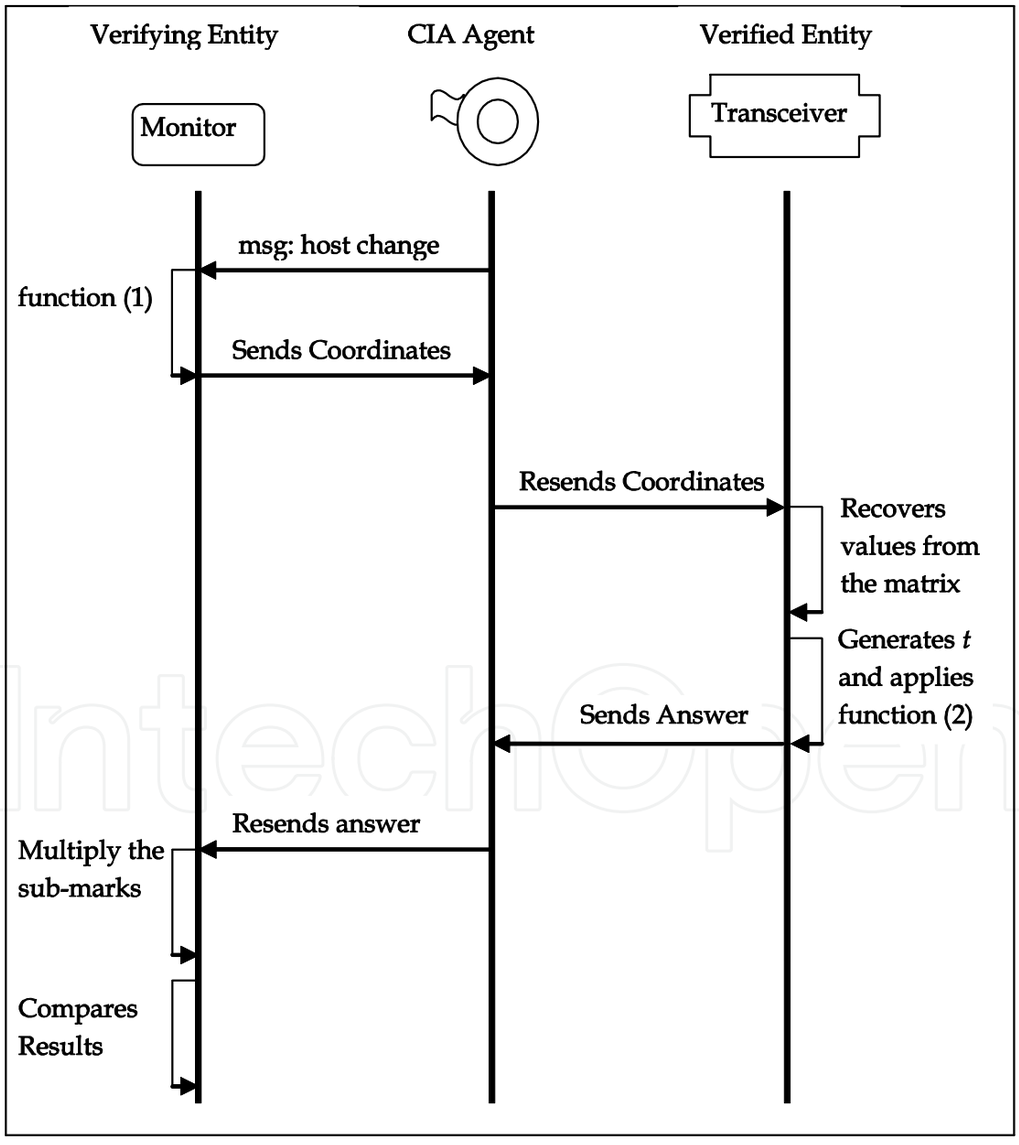


Fig. 8. Sequences diagram among entities for marks' verification

accorded module function and compares it to the given value obtained of applying the same module to the product of the requested values. If the result of the module function does not match, it means that the transceiver has been compromised and it must be considered as malicious. Otherwise, there is a high probability that the integrity of the agent has not been compromised.

Fig. 8 presents a sequence diagram which depicts the processes performed by each entity. The proposed integrity verification process of a transceiver or a monitor can be considered as a Zero-Knowledge Proof or ZKP. The concept was introduced in 1985 for systems verification identity [U. Fiege et al., 1985], but it can be applied to cryptographic applications. A ZKP is an interactive protocol between two parts, which gives a test, a tester (verified entity, in our case, a transceiver or a monitor) and the verifier of the test (the monitor). The Verified entity must convince the verifier that it knows the answer to a given theorem without revealing any kind of additional information [Yang & Liu, 2005]. This means, an entity probes its identity by showing it knows a secret, without giving the secret away, this is particularly useful in the case the messages are being intercepted, since the attacker will not receive any information related to the secret. Another important characteristic is that the messages are always different and randomly generated in order that the attacker could not memorize a concrete sequence.

4.1 Generating the matrix

In the following section we will describe the process used by the Monitor to generate the sub-mark matrix and the actual verification of the sub-marks. After generation the transceiver, the Monitor creates a matrix of size f by c which will be used as a mark. this matrix must be stored in the Monitor and in the transceiver and it must be unique for each transceiver. For the example we are going to present, the matrix will be of size 4 by 4 (16 sub-marks) and two sub-marks will be requested. It also will be generated a module (x) which will remain constant during the process to be used on function (1). Several additional random values will be generated to multiply the module and to pick the two sets of coordinates from the matrix.

For simplicity, small values have been chosen for the given example. It is also possible for a CIA agent to request more than two sets of coordinates (sub-marks) to verify. In this case the Monitor must provide the different sets of coordinates; also the transceiver must get such values in order to replace them in function (2).

The set of modules and the values of the sub-marks in the matrix are:

1. Sub-marks Matrix:

	0	1	2	3
0	68813	79687	36599	98663
1	59879	16993	98689	36997
2	79657	11383	35729	21991
3	78643	41299	86323	59693

Table 1. Transceiver Sub-marks

The Monitor generates the transceiver, the sub-marks matrix, the prime numbers in the matrix (Table 1), module x and y. After this the Monitor sends the transceiver to the corresponding host. These given values remain constant and will be used later in the verification process.

2. The values given to the modules in this example are (Table 2):

Variable	Value	Description
x	17	Module of function (1)
y	53	Module of function (2)

Table 2. Fixed Values

3. The Monitor's randomly generated values, used to issue a sub-mark request to the transceiver (for this example) are (Table 3):

Variable	Value	Description
P_1	37	Random values that multiplies the module of function (1).
P_2	13	
p_3	7	
p_4	23	
t	53	Random value, generated by the transceiver which multiplies the module in function (2).
f_1	0	Raw coordinate of the first sub-mark (w1) to be verified. f1 < m (m is the number of rows in the matrix)
c_1	2	Column coordinate of the first sub-mark (w1) to be verified. c1 < n (n is the number of columns in the matrix)
f_2	2	Raw coordinate of the first sub-mark (w2) to be verified. f2 < m (m is the number of rows in the matrix)
c_2	3	Column coordinate of the first sub-mark (w2) to be verified. c2 < n (n is the number of columns in the matrix)

Table 3. Randomly Generated Values

The sub-marks in the coordinates $f_1, c_1 = (0,2)$ y $f_2, c_2 = (2,3)$; correspond to the values 36599 and 21991 in the sub-marks matrix.

The Monitor generates the random values $(p_1, p_2, p_3, p_4, f_1, c_1, c_2)$ and applies function (1):

$$f_1(x) = \{(x * p_1 + f_1), (x * p_2 + c_1), (x * p_3 + f_2), (x * p_4 + c_2)\}$$

$$f_1(x) = \{(17 * 37 + 0), (17 * 13 + 2), (17 * 7 + 2), (17 * 23 + 3)\}$$

$$f_1(x) = \{629, 223, 121, 394\}$$

The Monitor gives the result to the CIA agent, and the CIA agent resends it to the transceiver. The transceiver uses those values and applies the corresponding module in order to obtain the coordinates of the requested sub-marks:

$$C = \{(629) mod_x, (223) mod_x, (121) mod_x, (394) mod_x\}$$

$$C = \{(629)mod_{17}, (223)mod_{17}, (121)mod_{17}, (394)mod_{17}\}$$

$$C = \{0,2,2,3\}$$

The values of the sub-marks in the given coordinates (w_1, w_2) are respectively (36599 y 21991). The transceiver uses function (2). For this example the values will be y = 53 y t = 3571, y is generated by the Monitor and t is generated by the transceiver.

$$f_2(y) = \{(y * t) + (w_1 * w_2))\}$$

$$f_2(y) = \{(53 * 3571) + (36599 * 21991))\}$$

$$f_2(y) = \{805037872\}$$

The CIA agent receives the answer from the transceiver and resends it to the Monitor. The Monitor uses the value of y and obtains the module of the given value:

$$S_1 = \{f(2)mod_y\}$$

$$S_1 = \{(805037872)mod_{53}\}$$

$$S_1 = \{43\}$$

Then it gets the module of the product of the two requested sub-marks:

$$S_2 = \{(36599 * 21991)mod_y\}$$

$$S_2 = \{(804848609)mod_{53}\}$$

$$S_2 = \{43\}$$

After getting S_2, it verifies if the result is equal to the module of the product of the two sub-marks (36599, 21991), this is if $S_1 = S_2$ then it means that there is a high probability that the transceiver has not been compromised.

For the example we have used a four rows by four columns matrix, in this case the number of possible combinations of a couple of marks which can be obtained is given by (3):

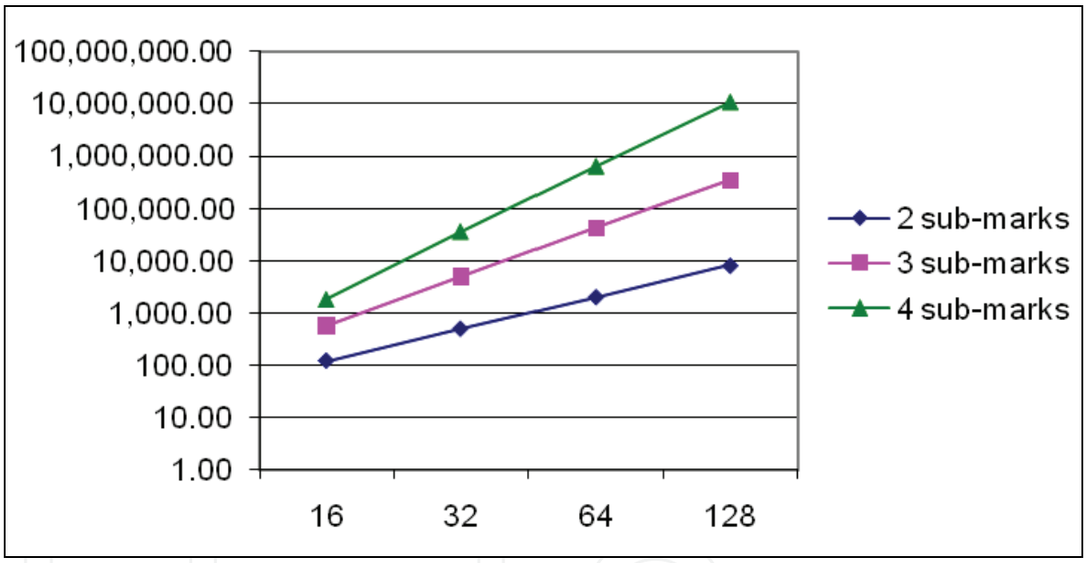
$$\frac{n!}{w!(n-w)!} = \frac{16!}{2!(16-2)!} = 120 \tag{3}$$

Each time a CIA agent issues a reuest for sub-marks, the monitor randomly picks the cells coordinates. With a 16 cell matrix there are 120 possibilities of obtaining the same pair of sub-marks, thus the probability that a set of sub-marks are repeated on a request is 1/120. If the number of requested marks is greater than two, the probability that an attacker could guess the sub-marks is smaller. If the matrix grows (more rows and columns), the probability gets closer to 0. Table 4 presents the different sets of combinations that can be achieved for matrixes size 16, 32, 64 y 128, by choosing 2, 3 or 4 sub-marks.

Cells	2 *sub-marks*	3 *sub-marks*	4 *sub-marks*
16	120	560	1.820
32	496	4.960	35.960
64	2.016	41.664	635.376
128	8.128	341.376	10.668.000

Table 4. Possible Combinations by Number of cells in the matrix and sub-marks to be verified

Table 4 shows that the probability to find a set of 4 sub-marks from a 128 cells matrix is minimal (1/10'668.000). Accordingly to the amount of requested sub-marks and the length of the prime numbers (the actual sub-marks) special care must be taken because the returned value of function (2) may be too large and generate a stack overflow. The next graphic is the representation of previous values on the table, presenting a logarithmic projection.



Graphic 1. Number of cells and sub-marks to be proved

5. The BESA framework

BESA (Behavior-oriented, Event-driven, and Social-based agent framework) is a Multi-Agent Development Framework, which allows multithreading, and is FIPA (Foundation for Intelligent Physical Agents) compliant; this allows interoperability with other platforms. BESA is based on three main concepts: Behavior-oriented, Event-driven, and Social-based, behavior allows and agent to encapsulate the needed characteristics that will guarantee the fulfillment of a well defined purpose, by forming cooperative societies and giving as a result agents composed by a set of concurrent behaviors; Event-driven means that the agent must react accordingly to its knowledge to a change in the environment. Finally the social-based concept, allows that different entities interact forming microsocieties. The way in which internal cooperation is achieved is by the implementation of a social mediator agent which

also works as an external representative. BESA is structured in three different levels: the agent level, the social level and the system level.
Agent level: at this level the behavior of the agents is specified as well as the events of the environment. An agent is composed by a channel, a set of behaviors and a state. Each agent uses mutual exclusion synchronization, in this way it avoids concurrency issues.
Social level: At this level the mediator agent is incorporated in order to form the actual abstraction of the intermediate levels. The mediator agent is a facilitator of the cooperative work between agents and acts as a receiver o thrower of events.
System level: At this level the life cycle of the agents is specified and managed.

6. Laocoonte's implementation on BESA

An agent based system implemented in BESA is a distributed system composed by one or many containers which are instantiated on a physical or virtual machine. Each container is processing space in itself, with a local administrator which supports the communication among agents and provides yellow and white pages services. The white pages service is used by the administrator to register the instantiation of an agent in the container. The yellow pages service is used by the contained agents to publish its own services. When there is a change in the agent's life cycle, the white and yellow pages services are updated and a copy mechanism is triggered in order to maintain the consistency among local administrators. The system includes a single FIPA compatible communication port, which communicates and receives queries from other systems using FIPA ACL (Agent Communication Language).
BESA is formed by container instances, each one belongs to the same group and they share the same IP address and a multicast port. Each event can reach any agent located on any container of the group, and notify the group members if some change has happened in the life cycle of the container or the agent. In this way, in order to implement a Laocoonte's collector agent, it must be located inside a container. The collector agent can monitor any kind of traffic (TCP. UDP, ICMP), but the agent must be specialized in only one kind. The collector agent filters and identifies suspicious events, then sending this information to its correspondent transceiver. Theses collector agents are located in the bottom of the architecture, one or several of them can be located on a container verifying the same o different kinds of traffic.
Transceivers are located in the middle layer of the architecture and they must be located inside a container. It can only exist one in each container. The transceiver has the local view of what is happening in the system, meaning that the transceiver is capable of identifying an intrusion in the host in which it is located. The transceiver must execute information correlation functions, based on the information gathered from the collector agents which in turn are managed by the collector agent (create, activate, sleep or delete).
The Monitor is an agent located on the highest level of the architecture; it is also located in a container. There is only one Monitor for each network segment and it controls all the transceivers in the network segment. The Monitor also executes correlation functions, but to a higher level than the transceivers, by filtering the information the Monitor gathers from the transceiver in its network segment. A Monitor can detect if an distributed attack or intrusion are being performed on the system. Eventually some Monitors could depend on other monitors in order to give some scalability to the system. At the root level (the highest level of the architecture), there must be at least two Monitors, in order that they could verify each other.

The CIA agents are generated by a Monitor and they have the mission of requesting the marks to each transceiver. The CIA agent is a mobile agent which is moving through network segments. When a CIA agent reaches a container, it must verify its identity requesting a hash function. This hash function corresponds to the sub-marks matrix of the transceiver; in this the container ensures that the CIA agent is not a malicious agent.
At the highest level of the architecture every Monitor must generate a CIA agent in order to verify other monitors at its same level, executing a cross verification.

7. Conclusions

Agents are appropriate entities to define an IDS, but they have inherent security risk due to its nature. These risks must be controlled in an efficient way in order to avoid interferences with the proper function of the system.
An IDS is a security tool which is one of the first barriers reached by an intruder, thus becoming a potential target of attack, it must also be configured in such a way that avoids the production of false positive and false negative alerts, it also must incorporate security techniques in order to avoid malfunctioning of the internal entities of the system.
With the proposed mark verification technique, using module arithmetic, the use of public key cryptography is avoided, reducing the integrity verification of the agents to simple mathematical operations. It also avoids overloading of the system and obtains acceptable response times for the IDS.
Accordingly to the proposed technique, if the sub-marks matrix grows, and the number of requested sub-marks also grows, the probability of being attacked is close to zero, making the deduction of requested or answered information by an attacker almost impossible.

8. References

Asaka, M., Okazawa, S., Taguchi, A., Goto, S. A Method of Tracing Intruders by Use of Mobile Agents. *INET99*, June 1999.

Balasubramaniyan, J.O. Garcia-Fernandez, D. Isaco, E. Spafford, D. Zamboni. An Architecture for Intrusion Detection using Autonomous Agents, *14th IEEE Computer Security Applications Conference ACSAC '98*. 1998.

C. Kruegel, C., Toth, T. Sparta – A mobile agent based intrusion detection system; In *Proceedings of the IFIP Conference on Network Security* (I-NetSec). (2001)

Collberg Christian S., Watermarking,Tamper-Proffing and Obfuscation-Tools for Software Protection, *IEEE Transactions on Software Engineering*. Vol 28,No,8,August 2002.

González E.; Avila J.A. y Bustacara C.J., 2003. BESA: Behavior-oriented, Event-Driven and Social-based Agent Framework. En: *PDPTA'03*, Las Vegas-USA, CSREA Press, vol. 3.

Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. Computer Networks: *The International Journal of Computer and Telecommunications Networking*, ISSN:1389-1286, Vol. 31, Pag. 805–822, April 1999.

J.S Balasubramaniyan, J.O. Garcia-Fernandez, D. Isacoff, E. Spafford, D. Zamboni, An Architecture for Intrusion Detection using Autonomous Agents, *Proceedings., 14th Annual Computer Security Applications Conference*, 1998, 13 – 24

Jansen W.A., Countermeasures for Mobile Agent Security, National Institute of Standards and Technology. *Computer Communications*, 23(17), 1667-1676, 2000

Knoll G. Suri and J.M. Bradshaw, Path-Based Security for Mobile Agents. *Proceedings 1st International Workshop Security of Mobile Multiagent Systems and 5th International Conf. Autonomous Agents*. ACM Press, New York, 2001, pp. 54-60.

Páez, R., Satizábal, C., Forné, J., Analysis of Intrusion Detection Systems Based on Autonomous Agents and their Security In: IPSI 2005 France, 2005, Carcassonne. *IPSI-2005 FRANCE International Conference on Advances in the Internet, Processing, System and Interdisciplinary Research*. 2005.

Páez, R. Satizábal C., Forné J. Cooperative Itinerant Agents (CIA): Security Scheme for Intrusion Detection Systems, *Proceedings of the International Conference on Internet Surveillance and Protection* (ICISP). ISBN:0-7695-2649-7. Pag. 26-32. 2006

Shao-Chun Zhong; Qing-Feng Song; Xiao-Chun Cheng; Yan Zhang. A safe mobile agent system for distributed intrusion detection. *Machine Learning and Cybernetics, 2003 International Conference on* , Volume: 4 , 2-5 Nov. 2003 Pages:2009 - 2014 Vol.4

Salvatore Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, and Dave W. Fan, JAM: Java Agents for Meta-Learning over Distributed Databases, The Third *International Conference on Knowledge Discovery & Data Mining* (David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, eds.), AAAI Press, August 1997.

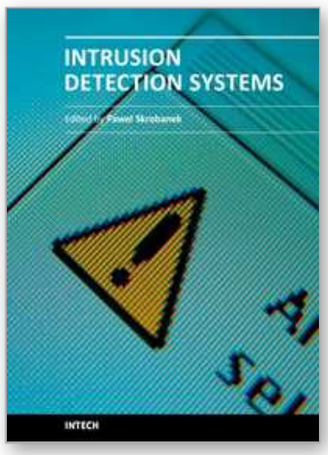
U. Fiege, A. Fiat, A. Shamir, Annual ACM Symposium on Theory of Computing archive *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ISBN:0-89791-221-7 New York, United States. Pages: 210 – 217, 1987.

Xiaobin Li, Aijuan Zhang, Jinfei Sun and Zhaolin Yin. (2004). The Research of Mobile Agent Security. In: *Grid and Cooperative Computing*. Springer Berlin / Heidelberg. Springer. Pag. 187-190. ISBN: 978-3-540-21993-4.

Yang, Z. and Liu, M., ZKp based identification protocol on conic curve in distributed environment, *The Fifth International Conference on Computer and Information Technology*, IEEE, pp. 690-694, 2005.

Yonggang Chu, Jun Li, Yixian Yang, The Architecture of the Large-scale Distributed Intrusion Detection System, *Parallel and Distributed Computing Applications and Technologies*, International Conference on, pp. 130-133, Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05), 2005.

Zaman, S.; Karray, F.; Collaborative architecture for distributed intrusion detection system. *IEEE Symposium on Computational Intelligence for Security and Defense Applications* (CISDA), 2009. ISBN: 978-1-4244-3763-4 Ottawa, Canada



Intrusion Detection Systems
Edited by Dr. Pawel Skrobanek

ISBN 978-953-307-167-1
Hard cover, 324 pages
Publisher InTech
Published online 22, March, 2011
Published in print edition March, 2011

The current structure of the chapters reflects the key aspects discussed in the papers but the papers themselves contain more additional interesting information: examples of a practical application and results obtained for existing networks as well as results of experiments confirming efficacy of a synergistic analysis of anomaly detection and signature detection, and application of interesting solutions, such as an analysis of the anomalies of user behaviors and many others.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Rafael Páez (2011). An Agent Based Intrusion Detection System with Internal Security, Intrusion Detection Systems, Dr. Pawel Skrobanek (Ed.), ISBN: 978-953-307-167-1, InTech, Available from: http://www.intechopen.com/books/intrusion-detection-systems/an-agent-based-intrusion-detection-system-with-internal-security

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen