Abstract

Self-organisation, as observed in natural systems, enables emergent behaviour, i.e., the realisation of complex collective tasks by simple entities without any central control. Such systems usually are more robust, and efficient, since they do not rely on hierarchical laws, and can evolve and adapt to new situations. Information systems also can benefit from self-organisation and emergent behaviour, e.g., detection of a network attack can be made more rapid, and its response more efficient if realised in a self-organised way rather than under centralised control. This paper presents a conceptual model for networking security, based on self-organisation of natural systems: an Intrusion Detection System (IRS), based on the immune system; and an Intrusion Response System (IRS), based on the self-organised behavior of insect colonies. This paper also describes the implementation part of the IDRS with JSeal-2, a pure Java mobile agent platform and presents the implementation results obtained from a chosen attack scenario.

Self-Organizing Architecture for Intrusion Detection and Response Systems

Noria Foukia
University of Geneva, rue Général Dufour 24, CH-1211 Geneva 4, Switzerland
Giovanna Di Marzo Serugendo
E-mail: Giovanna.Dimarzo@cui.unige.ch
University of Geneva, rue Général Dufour 24, CH-1211 Geneva 4, Switzerland

June 6, 2003

1 Introduction

In the last past years, a pleïade of products for intrusion detection have been deployed in the commercial sector as well as in the academic sector. All these systems propose a wide range of approaches to detect malicious activities. Of course, none of them is perfect in the sense that it cannot make the protected network completely impermeable with all the attacks. However, most of recent Intrusion Detection Systems (IDSs) are increasingly sophisticated. Their creators try to find new solutions to circumvent the new intrusive behaviors as well as to adapt them to the new network orientation due to growth of open networks such as the Internet. Among the weaknesses of these systems the character centralized of much of them is an undeniable lack of robustness. Notably, even if most of the existing IDSs use distributed data collection (host-based or network-based) many continues to perform central data analysis which limits their scalability. Besides, even if the IDS is dispatched in the network, its elements thus deployed remain for the majority static. With the means available to nowadays attackers, such as for example, automated intrusion tools, these static and distributed elements are easily reachable. Often, this does not always contribute to improve their reliability and resistance to attacks.

From one hand, this research work investigates the possibility to build an Intrusion Detection and Response System (IDRS) completely distributed whose entities are dynamic. Actually, the IDRS is incarned by a population of Mobile Agents (MAs) in perpetual movement whose furtivity makes the IDRS itself more reliable. Less inclined to protect itself against malicious attackers, the IDRS is more available to protect the network. Mobile Agents are lightweight entities that move autonomously from one location to another to perform their protection tasks and that can communicate with each other.

From the other hand, this paper intends to demonstrate that, in the case of network security and particularly in the case of intrusion detection and response, a self-organised behaviour mapped from natural systems, enables efficient attacks detection and source tracing. Indeed, self-organisation means the ability of a system to have complex collective tasks be realised by simple entites that alone could not be able to realise the global tasks. In addition, the realisation of the collective tasks occurs without central control, i.e., entities that take part to the system are autonomous and interact locally with each other as peers. Self-organisation enables nice features such as adaptation and evolution, with the important goal of ensuring robustness, e.g., survivability. Transposing these concepts in the field of computer science is thus appealing, particularly when coupled with the concept of mobile agents.
The paper is organized as it follows: section 2 describes the conceptual approach of the architecture based on natural life model. Section 3 summarizes the implementation of MAs for the IDRS. Section 4 validates the implementation through a port scan scenario and the corresponding results. Finally, section 5 draws the conclusion.

2 Conceptual Approach of the Architecture

We propose in this paper a distributed IDRS composed of two kinds of MAs population. An Intrusion Detection Agents (IDA) population taking its inspiration from Immune systems and an Intrusion Response Agents (IRA) population, taking its inspiration from social insects behavior. IDAs map the immune system defense mechanisms. The goal is to couple the efficiency found in collective behavior of immune system with the agents'mobility. Mobile agents travelling around the network work cooperatively to detect local and distributed suspicious patterns. IRAs map the collective behavior of a population of foraging ants, using mobile agents technology and an electronic version of pheromone. The goal is to trace the source where the alert was detected in order to perform the response task. The system to protect is a network of interconnected hosts which could be a LAN or an Intranet for instance. This Intranet is divided into several security domains depending on the topology as well as on security needs of the corresponding domain.

Schematically the IDRS architecture is as it follows:

- A population of IDAs incarnate the IDS and play a role equivalent to the one of immune defense cells of the body. IDAs perform a random walk through the network. To detect local[1] attacks IDAs responsible for a local domain have to be able to discriminate between normal and abnormal activity. A good activity is an authorized sequence of events previously established. From these sequences, the deviation between the listed authorized activity and current activity in the system is computed using Hamming distance as it was done in [1]. As explained in [2] there are two stages: initially, the learning stage to make the IDS know which are the good profiles of the applications to supervise, then the detection stage itself where the MAs placed in each domain and specialized in one program will compute deviation of system calls specific to this program. If the deviation exceeds a previously adjusted threshold, MAs launch an alert. Please refer to [2] to have more details of this part of the conceptual model.
- A population of Intrusion Response Agents (IRAs) incarnates the IRS and play a role equivalent to the one performed by the ants in social insects colonies to trace the source of food. In parallel to the IDA population each IRA of the IRS also performs a random walk through the network. As soon as an alert is launched, the IDA releases through the network a so called "electronic pheromone" which will be used by IRAs to trace the route back to the alert source. This pheromone is an electronic information that embeds all the essential features dedicated to the detected attack. The main fields are :
  - The Identifier of the IDA which detects the suspicious activity and builds the pheromone: *IdA*.
  - The Suspicion Index of the alert: *SI*. The proposed response scheme depends on a behavior-based IDS scheme depicted in [2]. In [2], IDAs dispatched throughout the network collect application-specific system calls and compute the deviation between these system calls and other safe system calls stored in a database. In other words, *SI* is the deviation between the safe authorized events stored in the database and the monitored events when the agent enters the node and detects an attack.
  - The number of Hops: *Hop*. It corresponds to the distance in term of the number of links through which the pheromonal information will be propagated from the initial node.
  - The pheromonal Gradient: *Gd*. Like ants with the source of food, IRAs have to find the better way to locate the source of an alert. For this purpose an electronic gradient field, *Gd*, is introduced in the pheromone. When the pheromone is diffused across the network, *Gd* diminishes hop by hop according to a strictly decreasing function. *Gd* is used in the opposite direction by other travelling IRAs to travel up to the source of the attack.
  - the date *t0*, which corresponds to the date when the attack is detected on the initial node and when the pheromone is built.

[1] Local in the sense of local in a security domain. In fact, anomalous patterns due to distributed attacks are also considered.

– The date *ti*, which corresponds to the date when the pheromone is deposited on each intermediate node i during the pheromone propagation.

This pheromone is diffused randomly in one direction by other agent population, namely the PheromonePropagators. This dissociation of roles is quite useful because it allows the different population of agents to work independently from each others in an asynchronous way. The pheromone is deposited by IDAs and the roaming IRAs will follow it as soon as they detect its existence. Self-organisation is achieved through this pheromonal information, which acts as a communication medium to alarm the good IRA population. As already said, the Intranet is logically divided into security domains and as the IDAs, the IRAs are also specialized in different tasks. Thus, the information carried by the pheromone is different according to the detected attack as well as its seriousness. This allows the IRAs to perform the adequate response. This is quite advantageous in term of performance because it avoids having inappropriate agents moving at a location where they will be useless. Moreover the pheromone will also evaporate after a certain laps of time. This laps of time is adjusted according to:

- the average time needed by an IRA to perform its task when visiting a node ; among all to read and interpret the pheromonal information. Also to choose the next node to visit.
- the number of IRAs that have already performed a response for the current pheromone.

Here also, this evaporation process contributes to limit IRA activity/movement as well as useless resources consumption.

3 Implementation Description

This section describes the implementation part dealing with our IDRS, using Sun Microsystems Java technology and Coco Software Engineering JSEAL-2 framework [?]. It offers an overview of the software architecture and selected highlights of the characteristics and specificities of the implementation environment. Figure 2 (Left) depicts the scheme of the global software architecture for the IRS and Figure 2 (Right) depicts the scheme of the global software architecture for the IDS.

3.1 Implementation of the Agents

The global architecture of the two components of the IDRS, namely the IDS and IRS, is based on the same class, the MobileAgent class but in each system, the MobileAgent class uses :

- A different algorithm to move;
- Different behaviors when it enters a node. In fact, it follows a sequence of actions which changes essentially according to the implemented agent (IDA or IRA).

The IDA follows the following sequence of actions:

- It moves randomly. When it enters a machine it probes the incoming events and computes the deviation: SI;
- If SI $\geq$ threshold then it launches an alert and builds a pheromone;
- Then it continues its random walk to detect a new deviation.

The IRA follows the following sequence of actions:

- It moves randomly and searches a pheromone (according to the search algorithm);
- If it finds a pheromone then it follows it until the source;
- If it follows a trace and reaches the source then it answers and inhibits the pheromone;
- Then it continues its random walk to discover another pheromone.

MobileAgent is an abstract class which has already an implementation of the MoveTo method. This class aquires also all necessary elements for its creation from the MobileHost class; indeed, the MobileHost creates the name of the MobileAgent, then creates the MobileAgent and opens a channel to allocate the name to the MobileAgent.

Besides, there are two other special agents: the PheromonePropagator as well as the Host-Seeker.

- Due to exclusive operations on the MobileHost (For instance the reading of the event files), if a MobileAgent wants to use the same exclusive operation as the one currently executed by another MobileAgent, this former MobileAgent has to wait. Thus, the implementation of a HostSeeker provides the IDA (respectively, the IRA) with an asynchronous way to obtain a travel destination. While the IDA (respectively, the IRA) is operating on the current host, its specially created HostSeeker visits the direct neighbors, selects the one where the visit frequency of IDAs (respectively, the IRAs) is the smallest and reports back. This frequency is calculated on the MobileHost, when the IDAs (respectively, the IRAs) request locks for exclusive operation.
- This implementation of a PheromonePropagator distributes initial pheromonal information through a number of hosts, following a randomly chosen path of nodes in the network. The pheromone gradient satisfies the monotonous decreasing requirement, as described in the conceptual model.

3.2 Implementation of the Artificial Pheromone

The pheromone is stored locally in the different nodes where it is deposited as an array of the different fields we mentioned in the conceptual model. The major advantage is to distribute the information through all the nodes of the network. The pheromone integrity is checked when it is written in the local file and the access of this local file by IRAs is under control. Moreover it is also duplicated in Oracle database (in test mode) which allows, for instance, the administrator system to have a good global view of the pheromone present in the network.

4 Experiments and results

This section describes the results obtained in a real network at the University of Geneva when performing a detection scenario as well as a response scenario. The combination of the mobile agent model with immune system and the social insect behavior is the driving idea behind this work. However, one cannot deliver a proof of concept unless it can be demonstrated in a real world scenario.

4.1 Detection of TCP Port Scans

The scenario that was chosen for implementation is not really an intrusion, but rather the prelude to it. At its very simplest, port scanning is exactly what it names implies. The scanner will test a number of ports of a TCP/IP node by sending packets of his choosing. This varies from ping requests that merely show whether a node is up, to protocol-incorrect sequences of TCP packets, the replies to which are usually implementation specific.
In order to detect the various types of port scans, one must analyze network traffic and interpolate common trends. An obvious signature to look for is a number of packets to different ports from the same source address. A variant of this signature consists on looking for the number of packets to different ports on different machines of the supervised network in a distributed port scan scenario.
Thus, IDAs behave as sentinels and access the required data, upon which the analysis is based, through raw TCP sockets on each host. Upon screening, the sentinels select only packets that fall into the port scanning scenario described above. These sentinels own a short-term memory to register the last events of the former visited nodes. At each visited node, each sentinel correlates its own short-memory events with those present in an historic file of the node. In fact, this historic file incarnates a long-term memory and is nothing else than the trace deposited in the current node by all the other sentinels. In this way, IDAs behave as simple sentinels interacting with one another through a common shared environment; here through the different network nodes. Thus, IDAs are collectively able to exhibit a complex global detection scenario, the distributed port scan, thanks to these local interaction.

4.1.1 Measured Results

- The port scan detection

  The IDS, as described in the previous section, behaves, from a summary point of view, in the following way. A given sentinel moves randomly between nodes that participate in the IDS. On each node it scans TCP network traffic and retains the source and time of suspicious packets. It stores it in memory and delivers alerts when the recorded data indicate short intervals between a large number of packets from the same source.

  The main variables of the sentinel's algorithm are what limit values it allows for these measures. The threshold for minimum average intervals between suspicious packets has an initial value and is adjusted downwards as more and more packets are detected. It also has a linear inertia factor that pushes it back to its initial state over long periods of inactivity. Thus, the threshold is a combined measure of both number of packets and intervals between them. Tests were conducted on various configurations to quantify these variables. Presented here are the results collected when executing the IDS on 15 virtual machines running Linux inside an IBM S/390 mainframe (Figure 3). The entire IDRS was implemented using Sun Microsystems Java technology and Coco Software Engineering JSeal-2 [4] framework. The latter is a mobile agent platform itself implemented in pure Java.

  At the beginning (Figure 4) the sentinel threshold value is low (red curve) because the average interval between packets measured by the sentinel is high (green curve). Gradually, during its random walk the sentinel correlates the different scan information deposited by other sentinels in the current visited node (memory plus historic information). In average as soon as the interval between packets becomes to low during a lapse of time (the green curve decreases), the sentinel's stress increases. And the red curve augments too. When curves meet, alert is given. The sentinel then escapes the node where the alerts was launched after generating pheromone and it sets the threshold to its initial value. Then, the response scenario can take place.

The port scan answer

After an alert, the corresponding pheromone is generated, awaiting the response mechanism to come into play. The response mechanism is straightforward in this case. IRAs trace back the source of the attack and switch to an aggressive mode which allows them to analyze more precisely the entering packets before reacting. They have the possibility depending on the service under scrutiny, to refer to a knowledge database of vulnerabilities. This database provides information on the more frequent as well as the new vulnerabilities. For instance, one well known exploit is wuftp buffer overflow. This exploit has been performed after the ftp service was scanned on the IBM virtual machines. The IRAs role was to stop the ftp service on the corresponding virtual machines. Thus, Table 1 shows the response frequency according to the number of hops in the case of an ftp port scan scenario.

4.2 IDRS self-organisation behavior

In consequence of the results obtained with the implementation and the demonstration scenario, the IDRS self-organized nature is highlighted. Indeed, the IDRS exhibits two self-control principles which are highlighted in [3] and mentioned below:

- Exploitation

  IDAs exploit the deposited information (average interval between packets) at each intermediate node thanks to the node historic. Besides IDAs influence the node historic and in this manner the other IDAs, thanks to their short-term memory. Thus, they evolve in an auto-catalytic processsus to do network control. In fact, their stress state depends on this auto-catalytic information exploitation. Figure 4 illustrates completely this effect; IDAs stress increases until the alert is launched. After the alert, the memory is cleaned and the stress decreases since there is no more effect.

IdA	*SI*	*Hop*	*Gd*	t_0	t_i

Figure 1: Electronic Pheromone

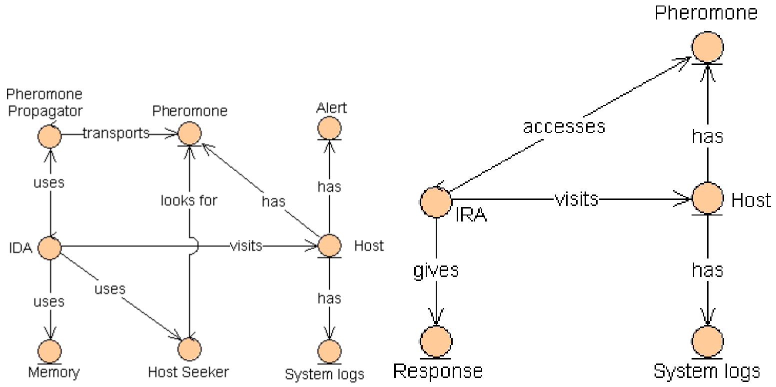

Figure 2: Left: IDS Global Software Architecture - Right: IRS Global Software Architecture

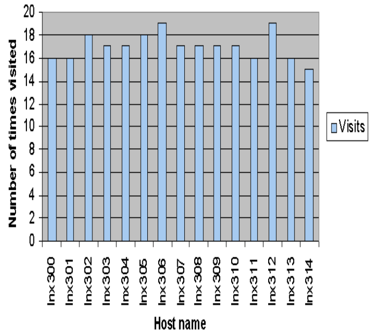

Figure 3: IDA Visit Pattern. The sentinel moves around nodes without affinity to any.

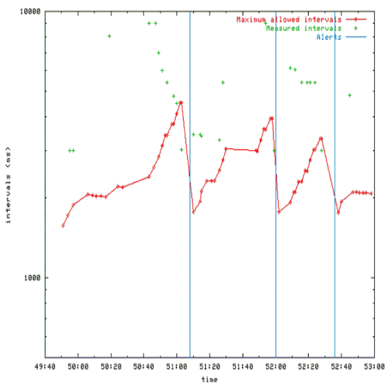

Figure 4: Threshold, suspicion relationship for a continuous long-term port scan. The Y-axis represents the average interval between packets as measured by the sentinel throughout the test execution. The alternate curve indicates the threshold value for this variable. X-axis represents the time.

- Exploration
  IRAs explore the deposited pheromone at each visited node thanks to the diffusion process. This exploration allows them to find new possibilities to reach the source of alerts. Table 1 illustrates this exploration effect. The higher the diffusion distance (the higher the exploration effect), the higher the percentage of success.

5 Conclusion

This paper describes a model for intrusion detection and response transposed from natural systems, its implementation in using mobile agents, and results showing the rapid detection and localisation of an attack in a port scan scenario. The IDRS offers decentralised control and emergent behaviour, obtained through self-organisation of these mobile agents using an electronic pheromone for adapting their behavior (travel direction, suspicion index, stress,...). The lack of efficient algorithms to detect and respond to attacks in near real-time is raised by the computer security community. Thus, this paper is placed from the point of view to bring new enriching paradigms based on natural life systems. These paradigms are specially adapted in the case of complex systems such as information networks.

Number of hops	Mean time to source [sec]	Success
7	14.60	100 %
3	17.20	90 %
2	8.14	70 %

Table 1: Response frequency according to the number of hops. The field success indicates the number of IRAs which answered in time.

6 Acknowledgements

Thanks to Salima Hassas for many helpful contributions and expert discussions dealing with self-organizing and complex systems.

References

[1] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 1997.

[2] N. Foukia, D. Billard, and P. J. Harms. Computer system immunity using mobile agents. HPOVUA Workshop, Berlin, June 2001.

[3] N. Foukia and S. Hassas. Towards self-organizing computer networks: A complex system perspective. in Proceedings of the First International Workshop on Engineering Self-Organising Applications (ESOA 2003), Melbourne,Australia., 14-15 July 2003.

[4] J-Seal2. http://www.coco.co.at/development/.