World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:11, 2008

Dynamic Adaptability using Reflexivity for Mobile Agent Protection

Salima Hacini, Haoua Cheribi, and Zizette Boufaïda

***Abstract*—**The paradigm of mobile agent provides a promising technology for the development of distributed and open applications. However, one of the main obstacles to widespread adoption of the mobile agent paradigm seems to be security. This paper treats the security of the mobile agent against malicious host attacks. It describes generic mobile agent protection architecture. The proposed approach is based on the dynamic adaptability and adopts the reflexivity as a model of conception and implantation. In order to protect it against behaviour analysis attempts, the suggested approach supplies the mobile agent with a flexibility faculty allowing it to present an unexpected behaviour. Furthermore, some classical protective mechanisms are used to reinforce the level of security.

***Keywords*—**Dynamic adaptability, malicious host, mobile agent security, reflexivity.

I. Introduction

THE mobile agent is generally defined as an autonomous software entity, having an own activity, acting by delegation for the account of a person or an organization, can migrate from a host to another over a network and can communicate with other agents [14]-[11]. The mobile agent paradigm offers some very interesting perspectives for a lot of applications, like electronic business, Web information research and data bases manipulations. This paradigm provides several advantages to design and control distributed applications (e.g., autonomy, dynamic adaptation, fault-tolerance, heterogeneous computing, better use of the network resources and reduction of communication with respect to latency and connection time). However, this technology creates serious theoretical and practical problems like the problem of heterogeneity, the preservation of communications, the shared resource management and particularly the security problem that represents a crucial point for the use of mobile agent applications.

The security of a mobile agent system covers four different aspects [2]: the security of the agent migration, the protection of the platform against the malicious agents, the agent's protection against others malicious agents and the agent's protection against a malicious platform. The problem of platform's protection has received a considerable attention.

Whereas, the protection of mobile agent against malevolent hosts remains an open problem because of its complexity, since the host has a full control on the mobile agent execution.

We will be interested in protection of a mobile agent against malicious hosts. The suggested protection approach is based on the dynamic adaptability. The latter is used in order to offer the mobile agent the property of flexibility allowing the modification of its behaviour and thus complicating its analysis. The co-existence of several approaches for the realization of the dynamic adaptability shows that there is no universal solution which reconciles the efficiency and the easiness. However, the reflexivity seems to be a promising method for the implantation and the realization of the dynamic adaptability. It constitutes a support of the application development and provides mechanisms enabling to express treatments in extremely generic terms. The intrinsic properties of the reflexivity (introspection and intercession) give the mobile agent the possibility to reason and to act on itself

The paper is organized as follows: Section II exposes some related works from which we inspired our idea. The section III explains in a concise and precise manner the notion of adaptability and the reflexivity. At the level of the section IV, the proposed approach is described. Finally, Section V concludes this article.

II. Related work

A. Related Works on Mobile Agent's Protection

In this subsection, we summarize only the techniques that have been proposed in the literature to protect agents against attacks perpetrated (by malicious hosts) and that will be combined to enhance our mobile agent security.

Several attempts approach these security threats in a total or partial way. They principally aim at making the attacks useless or detectable. Among the existing approaches, we find:

The approach of Riordan and Schneier [13] consists of using data found on the environment where the agent executes to construct decryption keys. When the proper environment information is located, the key is generated, the cipher-text is decrypted, and the resulting plain-text it acted upon. Without the environmentally supplied input, the agent cannot decrypt its own message and can be made cryptographically resistant to analysis aimed at determining the agent's function. Until the data has been collected, neither the agent nor the host is able to execute or understand the agent's mission or strategy.

Manuscript received September 30, 2006

Salima Hacini is with Lire Laboratory, Computer Science Department, Mentouri University of Constantine, Algeria; (phone: 00 213 31 81 88 17; fax: 00 213 31 81 88 17; e-mail: salimahacini@gmail.com).

Haoua Cheribi is with Lire Laboratory, Computer Science Department, Mentouri University of Constantine; Algeria; (e-mail: haoua_eva@yahoo.fr).

Zizette Boufaïda is with Lire Laboratory, Computer Science Department, Mentouri University of Constantine; Algeria; (e-mail: boufriche@hotmail.com).

The technique allows the agent owner to specify some constraints on the environment where the agent will execute.

The approach proposed by Wang and Guan [16] preserves the integrity of the mobile agent by the gradual construction of the agent's code in which new modules can be added and those redundant can be entrenched at the runtime. This approach increases the confidentiality of the code, reduces the cost of transport and facilitates the agent's recuperation after the malicious attacks.

The approach proposed by Grimley and Monroe [8] uses the factor of the time to identify a malevolent host. If the amount of time needed to execute a mobile agent on a host is limited, then the chance that it would tamper with is minimized. Once the maximum amount of time needed by a mobile agent to execute safely on an untrusted host is elapsed, the agent must shut down or move to the next host specified on its itinerary.

B. Related Works on Mobile Agent Dynamic Adaptability

This subsection presents some dynamic adaptability related works: they use

Ledoux and Bouraqadi-Saâdani [10] have proposed an approach permitting the adaptation during execution of the mobile agent in order to guarantee a better QoS (Quality of Service). They have used the reflection to support this adaptation. In order to increase the performances of the system, they have proposed the introspection of the environment characteristics to choose dynamically the best execution policy of the resource reallocation and the relationships configuration.

Amara-Hachmi and El Fallah [1] have proposed a model of architecture based components for a mobile agent to increase the modularity, the extensibility, and self-adaptability. The context of the mobile agent changes when it moves from a host to another. The result of the dynamic adaptation is the selection of the adequate components for the new context and their relationships.

Eyal de Lara *et al* [7] have elaborated Puppeteer: a component-based adaptation system for extending component based applications to support adaptation in mobile environments. It could support adaptation without modifying the applications. Puppeteer performs data and control adaptations by repeatedly using the policies of subsetting and versioning. A subsetting policy renders parts of the original document like text, or the first-slide of a document. A versioning policy allows the choices among multiple instantiations of a component, such as instances of an image with different resolution.

In order to protect the mobile agent against the host that receives and executes it, our approach exploits opportunities offered by the dynamic adaptability mechanism. The latter is used to offer the mobile agent the possibility to modify its behaviour. This ability makes it unpredictable and complicates its analysis. The idea is that the mobile agent must verify the customer trustworthiness and present to him, according to the trust he inspires, an appropriate behaviour.

III. Used Concepts

A. Adaptability

The adaptation designates the action to react facing variations of environment constraints. The Adaptability refers to the capacity or the degree of adaptation [12].

The adaptation can be static or dynamic: The static adaptability is done before the execution according to environment knowledge detained. At the level of the dynamic adaptability, two cases can be presented:

- Adaptation specified statically and done dynamically: This solution consists of estimating, during the construction of the application, the different variations of the environment and defining actions of adaptation. Consequently, it defines adaptability rules.
- Adaptation specified and done dynamically: This approach enables the adaptation during the execution as well as the definition and the dynamic setting up of the used strategy. Currently, we are not informed of a serious implementation of this idea, it is only considered.

We essentially distinguish two strategies for the realization of the adaptation [12]-[5]:

- Implicit Strategy: It concerns the application that uses a protocol which encapsulates a solution to the adaptation for a given problem, without having the control right or replacement. (e.g., facilities proposed by CORBA, JAVA - RMI…). The adaptability proposed in these cases has a limited domain of validity.
- Explicit Strategy: It concerns the applications that can control the definition and/or the choice of activities. This strategy can be provided by the application, either by procedural, declarative or a reflexive manner. This intermediate solution has benefit of a large popularity. The visibility that it offers can be placed between the transparency of the declarative approach and the total visibility of the procedural approach.

B. Reflexivity

The reflexivity is a manner of internal organization of a system to facilitate its development, its adaptation and its reuse, by offering a conceptual setting allowing program functionalities separation. In fact, the reflexivity is the capacity for a program to manipulate as data something representing the state of this program during its execution. It is the possibility to reason and to act on oneself [15]-[6].

This method considers two abstractions levels

- The base level: designates the standard application and manipulates domain entities. It describes the application functionality.
- The meta-level: is the level that reasons and acts on the base level. It manipulates abstractions of its entities. It describes the non functional aspect of the application.

Two types of reflexivity can be distinguished: the structural reflexivity and the behavioural reflexivity. The first one concerns the most static aspects (like graphs of inheritance, of composition or types of data). Whereas the second one, concerns the more dynamic aspects, related to the state and the execution strategy. A system of this type can propose two alternatives implementations of a same module [6].

IV. The Proposed Approach

A. Problem Position

Security is a very large and delicate problem and ensuring it at 100% is an impossible task. However, it is always possible to ensure an acceptable level of security by applying a well studied protection strategy. The aim of this paper is the protection of the mobile agent against the malicious hosts' attacks. These attacks are numerous and of various types. They include the passive attacks as well as the active attacks which modify the data, the code, the state or messages emitted by the mobile agent [3]. Furthermore, security requests are multiple. They relate to the Authentication, the Confidentiality, the Integrity and the Availability. Moreover, security measures are of various levels. They consist of structural measures, prevention measures, palliative measures, protection measures, detection measures and recovery measures. The proposed mobile agent protection approach uses prevention measures supported by dynamic adaptation technique. It mainly protects the mobile agent against *behaviour analysis*. In order to get a satisfying level of security, classical security mechanisms, like symmetric and asymmetric cryptography and digital signature, are used.

The eavesdropping attack involves the interception of confidential communications. Its threat is exacerbated in mobile agent systems because the host has a full control on the execution of the mobile agent. If the malicious host cannot directly reach the mobile agent confidential data or code, it will try to analyze the agent behaviour to be able to deduce its strategy [4]. Consequently, the mobile agent security implies protecting its information and code against their unauthorized disclosure.

B. The Adopted Strategy

Our approach consists of the protection of the mobile agent via a dynamic adaptation policy. During its life cycle, the mobile agent presents different and unexpected behaviours. In order to allow a modification of its behaviour, an aptitude of flexibility is affected to the mobile agent. This ability prevents the visited host to deduce the followed strategy. Thus, any behaviour analysis attempt becomes difficult and inefficient.

In fact, the change of the behaviour must be carried out by taking account of the environment variations represented by the collected data. These latter could be related to the host security detected level, the requested service, the service requirements or circumstances. Their analysis generates the selection of a suitable beforehand defined behaviour. This adaptation is statically conceived and dynamically performed. An *interface* allows communication with the host and permits data acquisition. These data are used for the mobile agent treatment selection. Collected data are stored in a *memory* and the modules of the various selected treatments are in *a library*.

C. Mobile Agent Structure

The considered mobile agent reflexive structure should support the previously described strategy and offer a better efficiency of the adaptive treatment. So, a behavioural reflexivity is suggested. It highlights two conceptual levels: (i) *a basic level* which contains the functional code and formulates the application logical implementation as well as its semantics. It comprises three components: the *Interface*, the *Memory* and the *Library* (ii) *a meta-level* which contains the non-functional code presents a variety of services that belong to the application and describes the adaptation process (see Fig. 1). The latter is ensured by an *adaptor* component.

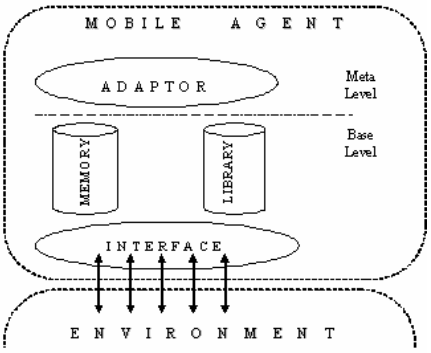

Fig. 1 Adaptive mobile agent reflexive structure

The proposed mobile agent reflexive architecture is presented by Fig. 1. The environment corresponds to the visited host. The refinement of this architecture is illustrated by Fig. 2. The mobile agent protective structure contains:

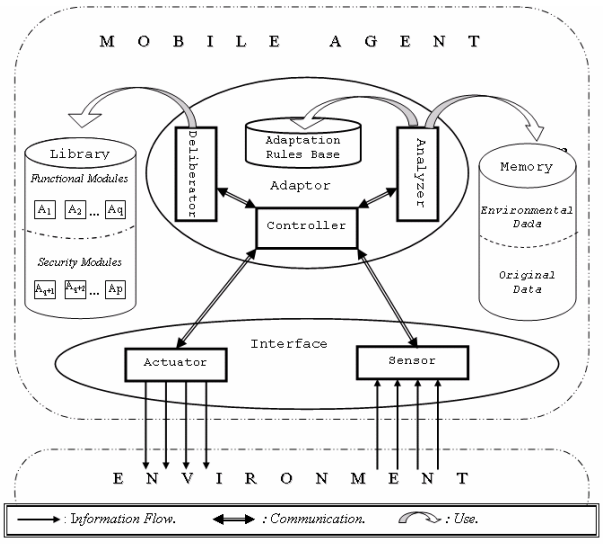

Fig. 2 Mobile agent protective structure

- Interface: this component allows the mobile agent-environment communication. It comprises two sub-components; a Sensor and an Actuator:
  - The Sensor: it is responsible for the environment perception and the data acquisition. Three types of acquisition can be considered: the observation, the inspection and the interaction. The type of acquisition is related to required data: identity, certificate, private data (e.g., password, reference contract…), infrastructure…
  - The actuator: it allows the execution of the sequence of selected actions. These actions, can be a simple alarm, a treatment modification, a stop of the execution or/and a migration.
- The memory: it contains the data collected by the sensor and that represent *environment data* as well as the data provided by the mobile agent owner named *origin data*. They will be used to check the validity of collected data.
- The library: it contains the modules that compose the mobile agent code (micro-components) and from which various compositions generate the different behaviours. The purpose is to offer more confidentiality and more flexibility to the treatment. In order to vary the mobile agent behaviour, alternatives for some modules are proposed. Furthermore, some fictive modules are used to complicate the analysis task and to increase the security level. Various combinations of the micro-components are supplied by a set of abstract expressions used to implement the mobile agent behaviours.

  Let E= $\{E_1, E_2...E_n\}$ be a set of abstract expressions and let A= $\{A_1, A_2...A_p\}$ be a set of adaptive micro-components. Each expression Ei (i<=n) is a sequence of calls of subset of adaptive micro-components Aj and can be viewed as a sequence of bits (each bit indicates a specific micro-component) [9]. The set A can be subdivided on two subsets:
  Let F= $\{A_1, A_2…A_q\}$ be a set of functional modules that corresponds to the application and let S= $\{A_{q+1},…,Ap\}$ be a set of security modules (e.g., cryptography, hashing, digital signature,….).
- The adaptor: it determines the adaptive treatment. It comprises a *controller*, an *analyzer*, a *deliberator* and a *base of adaptation rules*.
  - The analyzer: it analyzes the data stored in the memory component and uses the base of adaptation rules to determine the appropriate behaviour.
  - The deliberator: it determines the actions to be performed by selecting the micro-components specified by the selected abstract expression and which are stored into the library.
  - The controller: it begins the work and ensures the coordination and the synchronization of the mobile agent components.
  - The adaptation rules base: it comprises the rules of the type: *<If Condition Then Action>*. It constitutes the core of our architecture. Rules are generic and independent from any application (see Table I).

TABLE I
EXAMPLES OF ADAPTATION RULES

RULE	CONDITION	ACTION
1	Failure of Key generation	Notify and Leave
2	Trust degree belongs to $[a_1,b_1]$	Assure a complete service
3	Trust degree belongs to $[a_2,b_2]$	Assure a degraded service
4	Trust degree belongs to $[a_3,b_3]$	Notify and Leave
5	Lack of facultative Resource	Assure a degraded service
6	Lack of critical Resource	Notify and Leave
7	Time of execution expired	Stop, Notify and Leave
8	Already visited Path	Choose another alternative
9	Blockage or Failure Execution	See exceptions
10	Failure Exception	Stop, Notify and Leave

D. Scenario of Execution

The mobile agent transports its itinerary from which it selects, in a random way, the next customer to visit. The customers' data are moved with the agent in an encrypted form. They are encrypted using the public key of the host of origin.

While arriving on the visited host, the agent must authenticate all data useful to the achievement of a requested service (e.g., contract reference, password…). Thus, the agent must obtain information from the host environment via the sensor. The analyzer crypts the collected data using the host of origin public key and compare them to the origin data in order to calculate the trust degree. The latter enables to specify the behaviour to be carried.

In order to reduce our mobile agent size, the origin data are removed and replaced by collected data. The new values will be used as a proof when the mobile agent returns to the host of origin.

The mobile agent behaviour depends on the release of rules belonging to the base of adaptation rules. The checked conditions are contained in the left parts of the rules, while the right parts correspond to the different behaviours expressed by the abstract expressions. The treatments vary from a service to another and are related to the degree of degradation of the service and to the provided alternatives.

TABLE II
EXAMPLE OF ESTIMATION INTERVALS WITH THEIR RELATED FEEDBACK

Interval of trust estimation	Feedback
0-20	Stopping service
21-60	Reducing service
61-100	Performing service

There are three trust estimation intervals. They respectively correspond to three ranges of trust value and generate different behaviours (see Table II). If the trust degree value belongs to a good interval (see rule n°2 in Table I), the agent starts the environmental key generation. The latter is used to decrypt the abstract expression related to the specified behaviour. When the agent cannot generate the appropriate environmental key because of missing or erroneous information, it stops its execution and leaves the host after having noted the cause of the failure [9]. In the case of the success of the generation of the environmental key, the analyzer decrypts the abstract expression and transmits it to the controller which sends it to the deliberator. The latter has to call the suitable modules which are in the library and provides the result to the controller which transmits them to the actuator with an execution order. The actuator executes deliberated actions and delivers a report of the execution to the controller so that it can pass at the following step.

With an aim of increasing the flexibility of the system, a set of exceptions is defined. If any problem occurs during one of the execution steps, the controller notes the non-progression for a determined duration. It then asks the analyzer to check the set of the exceptions for a possible re-establishment. If the whole of the exceptions does not correspond to the problem mentioned, it stops the execution immediately, notifies the problem and leaves the current host to migrate towards another host of the itinerary or to be allocated to the host of origin.

E. Trust Estimation and Environmental Key Generation

The trust degree estimation of the visited host is calculated using the collection of the values of certain parameters starting from the environment. The trust degree T is calculated according to importance I_J, weight W_J of the parameter J and factor S_J which is equal to 1 in the case of success (conformity of information), and equal to 0 in the case of a failure (non-conformity of information). The trust degree estimation is performed according to the following formula [9].

$$T = \sum_{j=1}^{k} w_j \; I_j \; s_j$$

Each parameter J has predefined values of its importance and its weight. These values are stored into the mobile agent memory.

For confidentiality reasons, the abstract expressions corresponding to the treatments to be carried out are encrypted, by the host of origin, using a symmetrical key. For the same reasons, this key should not be transferred through the network. It will be generated by the agent while arriving on the host of execution. The generation of this *environmental* key depends on a set of parameters values collected from the environment.

The generation of the environmental key follows the subsequent steps [9]:

1) Collect parameters values; let $\{d_1, d_2\ldots, d_K\}$ be the set of collected data.
2) Concatenate the collected data. Let $C = (d_1.d_2\ldots d_K)$ be the result of the concatenation.
3) Apply an SHS (Secure Hash Standard) one way function to C. Let be S = H(C)
4) Apply $S \oplus id$ (where id is a unique mobile agent identifier) to generate an environmental key Ks witch will be serve to decrypt the selected abstract expression.

V. CONCLUSION

A malicious host can deduce mobile agent confidential data and understand its execution strategy by analyzing its behaviour.

To prevent this type of attacks, the proposed approach uses the dynamic adaptation techniques. These techniques allow the adaptation and the modification of mobile agent behaviour during its life cycle by taking account of the environment variations of the visited host. Thus, the mobile agent is characterized by an unexpected behaviour. This ability prevents the visited host to deduce the followed strategy. Thus, any behaviour analysis attempt becomes difficult and inefficient.

Furthermore, the use of the classical protection techniques, such as the cryptography and the hashing, conserves the confidentiality and the integrity of the mobile agent and increases the security level.

The suggested architecture emphasizes a generic strategy, independent of any low level implementation thus allowing its reuse and its extensibility. The reflexivity was adopted as a modelling support because it provides a good representation and an easy handling of oneself. This concept facilitates the adaptation implantation.

REFERENCES

[1] N. Amara-Hachmi1 and A. El Fallah-Seghrouchni, "Towards a generic architecture for self-adaptiv," Proceedings of 5th European Workshop on Adaptive Agents and MultiAgent Systems (AAMAS'05), Paris, 2005.
[2] P. Bellavista, A. Corradi, C. Frederici, R. Montanari and D. Tibaldi, "Security for mobile agents: issues and challenges, " in Invited Chapter in the Book Handbook of Mobile Computing, I. Mahgoub, M. Ilyas (eds.), CRC Press, Dec. 2004.
[3] E. Bierman and E. Cloete, "Classification of malicious host threats in mobile agent computing," in proceedings of SACICSIT2002, pp. 141-148.
[4] N. Borselius, "Mobile agent security," Electronics & Communication Engineering Journal, vol 14, No 5, IEEE, London, UK, pp. 211-218, October 2002.
[5] R. Brandt, H. Reiser, "Dynamic adaptation of mobile agents," in Heterogeneous Environments, " in Springer Lecture Notes in Computer Science 2240, December 2001.

[6] O. Charra, "Approche réflexive des liaisons entre objets répartis," DEA report, I.M.A.G., Ecole Doctorale Mathématique et Informatique, 2000.
[7] E. de Lara, Dan S. Wallach, and W. Zwaenepoel, "Puppeteer: Component based adaptation for mobile computing,," in Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, pp. 159–170, March 2001.
[8] Grimley, M.J. and Monroe, "Protecting the integrity of agents," in ACM Magazine, B.D, 1999.
[9] S. Hacini, Z. Guessoum, Z Boufaida, "Using a trust-based key to protect mobile agent code", will be published by CCIS 2006, Italy.
[10] T. Ledoux and Noury M.N.Bouraqadi-Saâdani, "Adaptability in mobile agent systems using reflection," in ECOOP 2000, Workshop on Reection and Metalevel Architectures, Cannes, France, 2000.
[11] .S. Leriche, J. Arcangeli., "Vers un modèle d'agent flexible," In : Journées Multi-Agent et Composant, JMAC'06, Nîmes, mars 2006.
[12] S. Leriche and J. Arcangeli, "Une architecture pour les agents mobiles adaptables", in Journées Composants JC'04, Lille, pp. 1-9, 2004.
[13] J. Riordan and B. Schneier, "Environment key generation towards clueless agents," in Lecture Notes in Computer Science 1419, pp. 15-24, 1998.
[14] K. Rothermel and M. Schwehm. "Mobile agents," Encyclopedia for Computer Science and Technology, Volume 40, Supplement 25, New York: M.Dekker, Inc., 1998.
[15] D. Spinellis, "Reflection as a mechanism for software integrity verification," in ACM Transactions on Information and System Security, 3(1), pp. 51–62, 2000.
[16] T. Wang, S. Guan, and T. Khoon Chan, "Integrity protection for code-on-demand mobile agents in e-commerce," in The Journal of Systems and Software 60, pp. 211-221, 2000.