UACEE International Journal of Advances in Computer Networks and its Security – IJCNS
Volume 3 : Issue 2 [ISSN 2250 – 3757]
Publication Date : 05 June 2013

A Novel Approach for Introducing Advanced Security in Mobile Agents

Rajan Sahota[1], Ankur Chauhan[2], Priya Suneja[3]

Abstract -Mobile agents can travel autonomously through a computer network in order to perform some computation or gather information on behalf of a human user or an application. With the concept of mobile agent, the execution process will go to the place where the data are available, data will not send to the place of execution process. However, it has not become popular due to some problems such as security, fault tolerance etc. The fact that computers have complete control over all the programs makes it very hard to protect mobile agents from untrusted hosts. So, the issue of protecting a mobile agent from a malicious host is a more difficult problem than protect a host from a malicious agent. This paper proposes advanced security model for the mobile agent security against malicious hosts by combining few techniques so that it can provide a better solution.

Keywords*:* security, mobile agents, mobile code, malicious host.

I. INTRODUCTION

Today so many computer networks are connected to each other and spreading all over the world, and we can use various distributed computer resources through the computer networks. However, when a user tries to use these resources, he has to understand the location of distributed resources, predict their current status, and select some suitable resources. Mobile agent technologies are getting popular as means for an efficient way to access to remote resources on computer networks. Mobile agents, in these technologies, are processes that migrate from a server to server in the network autonomously to achieve user's requests. The user using mobile agents can get result of request without any knowledge about the network environment. Usage of mobile agents also brings in achievement of load balancing in whole the network by agent migration [2].

Mobile agents are composed of code, data, and state. Agents migrate from one host to another taking the code, data and state with them. The state information allows the agent to continue execution from the point where it was before it left in the previous host [3].

However, one of the main technical obstacles to a wider acceptance of the mobile agent paradigm is security. Achieving security is fundamental for the successful deployment of mobile agent systems, especially in the electronic commerce area [1]. Sander and Tschudin present two types of security problems that must be solved [4]. The first is host protection against hostile agents. The second is agent protection against hostile hosts. Many techniques have been developed for the first kind of problem, such as access control, password protections, and sand boxes, but the second problem appears to be difficult to solve. Yee proposed an approach that uses a secure coprocessor that executes critical computations and stores critical information in secure registers [5].

The rest of the paper is organized as follows. Section 2 deals with various security issues in mobile agent paradigm, Section 3 deals with the malicious host problem which can be caused by spying the code, data or state of the mobile agent by malicious hosts, Section 4 gives an overview of the main solutions for keeping a mobile agent secure against malicious hosts such as code obfuscation, partial result encapsulation etc. Section 5 gives architecture of novel approach; Section 6 gives experimentation and results. Finally Section 7 gives conclusions and future work.

II. SECURITY ISSUES IN MOBILE AGENT PARADIGM

Different security requirements that the mobile agent paradigm needs to satisfy [22]:

a) *Confidentiality*

It is important to ensure that the information carried by a mobile agent or stored on a platform is accessible only to authorized parties. This is also the case for the communication among mobile agent paradigm components.

b) *Integrity*

It is essential to protect the mobile agent's code, state, and data from being modified by unauthorized parties. This can be achieved either by preventing or by detecting unauthorized modifications.

c) *Availability*

Platforms typically face a huge demand for services and data. In the case that a platform cannot meet mobile agents' demands, it should notify them in advance.

d) *Accountability*

Platforms need to establish audit logs to keep track of all visiting mobile agents' actions in order to keep them accountable for their actions. Audit logs are also necessary when the platform needs to recuperate from a security penetration or a system failure.

e) *Anonymity*

As mentioned above, platforms need to keep track of mobile agents' actions for accountability purposes. However, platforms also have to balance between their needs for audit logs and mobile agents' needs to keep their actions private.

III. THE MALICIOUS HOSTS PROBLEM

Malicious host's problem is a commonly agreed security issue in the area of agent security. In the mobile agent paradigm, the hosts have full control over the mobile agents running in them, which no longer works for them like that in the traditional computer system. Some of the attacks that could be performed by malicious hosts to the mobile agents, which are totally controlled by them [6]:

a) Spying

Spying focuses on understanding the code, data and network communication of the mobile agent. It is called spying attack fast-spying if the environment has no knowledge of whether the agent has been spied. Otherwise, it is called tardy-spying.

b) Thieving And Pirating

Based on successful spying, the host could either steal data (thieving) or pirate code (pirating) from the agent.

c) Manipulation

Based on successful fast-spying, the host could modify the code, data, and network communication of a mobile agent or return wrong system call result without being known by the agent's environment.

IV. TECHNIQUES FOR MOBILE AGENT PROTECTION

For wide scale application, the approaches to protect an agent can be broadly classified into two main mechanisms [7]:

- Detection mechanism attempt to detect unauthorized modification of code, state or execution of mobile agent.
- Prevention mechanisms try to make it impossible to access or modify code, state or data in a manner that is meaningful to the perpetrator.

a) Code Obfuscation

Obfuscation is a technique in which the mobile code producer enforces the security policy by applying a behavior-preserving transformation to the code before it sends it to run on different platforms that are trusted to various degrees [8,23]. Obfuscation aims to protect the code from being analyzed and understood by the host. Consequently, the host should not be able to modify the mobile code's behavior or expose sensitive information that is hidden inside the code such as a secret key, credit card number, or bidding limits [8].

Typically, the transformation procedure that is used to generate the obfuscated code aims to make the obfuscated code very hard to understand or analyze by malicious parties. There are different useful obfuscating transformations [17,20,21,24]. Data Obfuscation concentrates on obfuscating the data and data structures in the code without modifying the code itself.

Hohl [18] suggested using the Obfuscation technique to obtain a time limited black box agent that can be executed safely on a malicious platform for a certain period of time but not forever. D'Anna et al [8] pointed out that Obfuscation could delay, but not prevent the attacks on agent via reverse engineering. They also argue that an attacker with enough computational resources, such as enough time, can always de-obfuscate the code. Barak et al [19] studied the theoretical limits of Obfuscation techniques and showed that in general achieving completely secure.

The main advantages of this technique includes flexibility and low cost. This technique has number of drawbacks, in this every transformation introduce extra cost in memory and computation time necessary to execute the obfuscate program.

b) Partial Result Encapsulation

Partial Result Encapsulation (PRE) is a detection technique that aims to discover any possible security breaches on an agent during its execution at different platforms. PRE is used to encapsulate the results of agent execution at each visited platform in its travel path. The encapsulated information is later used to verify that the agent was not attacked by a malicious platform. The verification process can be done when the agent returns to its home platform or at certain intermediate points in its itinerary.

To ensure the confidentiality of its results, the agent encrypts the results by using the public key of its originator to produce small pieces of cipher text that are decrypted later at the agent's home platform using the corresponding private key. This is one scenario of PRE where the agent itself does the encapsulation process. The agent uses a special implementation of encryption called "Sliding Encryption" that was suggested by Young and Yung [9]. Sliding Encryption encrypts small amounts of data within a larger block and thus obtains small pieces of cipher text. Sliding Encryption is particularly suitable for certain application where storage space is valuable such as smartcards [10].

Yee [15] suggested "Partial Result Authentication Code" (PRAC), where again the agent does the encapsulation of the results. However, the agent's originator also takes part in this scenario by providing the agent with a list of secret keys before launching it. For each visited platform in an agent's itinerary, there is an associated secret key. When an agent finishes an execution at a certain platform in its itinerary, it summarizes the results of its execution in a message for the home platform, which could be sent either immediately or later. It is important to note that the agent erases the used secret key of the current visited platform before its migration to the next platform. Destroying the secret key ensures the "forward integrity" of the encapsulation results. Forward integrity [15] guarantees that no platform to be visited in the future is able to modify any results from the previously visited platforms, as there is no secret key to compute the PRAC for these results.

Karjoth et al [16] proposed a "strong forward integrity", which, in addition to forward integrity, also requires that the visited platform cannot later modify its own results. Karjoth et al's approach depends on the visited platform doing the encapsulation process instead of the agent doing it. The visited platform encrypts the agent's results by using the originator's public key to ensure the confidentiality of the results [16].

The PRAC technique has a number of limitations. The most serious occurs when a malicious platform retains copies of the original keys or key generating functions of an agent. If the agent revisits the platform or visits another platform conspiring with it, a previous partial result entry or series of entries could be modified without the possibility of detection.

c) Execution Tracing

Execution tracing [11] is a technique for detecting unauthorized modifications of an agent through the faithful recording of the agent's behavior during its execution on each agent platform. The technique requires each platform involved to create and retain a non repudiatable log or trace of the operations performed by the agent while resident there, and to submit a cryptographic hash of the trace upon conclusion as a trace summary or fingerprint. A trace is composed of a sequence of statement identifiers and platform signature information. The signature of the platform is needed only for those instructions that depend on interactions with the computational environment maintained by the platform. For instructions that rely only on the values of internal variables, a signature is not required and, therefore, is omitted.

This technique gives all information about path of code. It helps to analysis the performance of code in individual host. The approach has a number of drawbacks, the most obvious being the size and number of logs to be retained, and the fact that the detection process is triggered occasionally, based on suspicious results or other factors.

d) Environmental Key Generation

Environmental Key Generation [12] describes a scheme for allowing an agent to take predefined action when some environmental condition is true. The approach centers on constructing agents in such a way that upon encountering an environmental condition (e.g., string match in search), a key is generated, which is used to unlock some executable code cryptographically. The environmental condition is hidden through either a one-way hash or public key encryption of the environmental trigger.

The technique ensures that a platform or an observer of the agent cannot uncover the triggering message or response action by directly reading the agent's code.

e) Computing With Encrypted Functions

The goal of Computing with Encrypting Functions [13] is to determine a method whereby mobile code can safely compute cryptographic primitives, such as a digital signature, even though the code is executed in untrusted computing environments and operates autonomously without interactions with the home platform. The approach is to have the agent platform execute a program embodying an enciphered function without being able to discern the original function; the approach requires differentiation between a function and a program that implements the function. Essentially, the problem the author would like to solve is the following: agent's program computes some function f, and the host is willing to compute f (x) for the agent, but the agent wants the host to learn nothing substantive about f . The protocol presented works in

the following way, where E is some encryption function:

- The owner of the agent encrypts f.
- The owner creates a program P(E(f)) which implements E(f) and puts it in the agent.
- The agent goes to the remote host, where it computes P(E(f)) (x), and returns home.
- The owner decrypts P(E(f))(x) and obtains f(x).

Strength of security is directly proportional to strength of encryption function. It is best suitable technique for application which requires high security. However this approach has a serious drawback: no information about the encrypted computation must leak to the host and only originator may receive any output.

V. NOVEL APPROACH FOR SECURITY

We are proposing a security model for making our agent more secure as it is using both the techniques of cryptography and obfuscation for its protection. The working of this model is shown in Fig.1 as flow of agent from host to remote server and vice-versa. The following are some useful points which we get from proposed model:

a) If the attacker is able to get the code of the agent, he will look for the private data which is been encrypted. So this data is protected as far as he compromised the secret keys.

b) On obfuscating the whole agent code, it will make it more difficult for the attacker to understand the code also obfuscation makes private data look more ordinary. So, it will take attacker much more time to crack the agent and its private information.

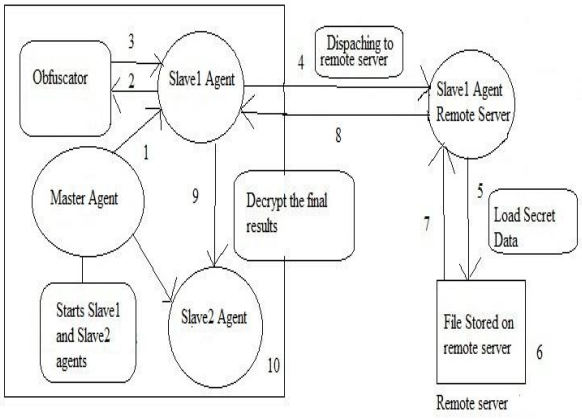

Fig.1: Architecture of proposed security model

The various steps of proposed model are given below:

1. The Master agent instantiates the Slave1 and Slave2 agents.
2. The Slave1 agent's code is obfuscated using obfuscator and then dispatched to remote server for retrieving the secret data from the file stored on remote server.
3. The Slave1 agent encrypts the secret data using the encryption algorithm used.
4. Then Slave1 agent returns back to the home with decrypted data.
5. At home it passes the encrypted information to the agent Slave2.
6. After getting the message from Slave1 agent, Slave2 agent decrypts the results back to original form and starts processing.

VI. EXPERIMENTATION AND RESULTS

The mobile agent system used in this paper is aglet software development kit (ASDK) 2.0.2. Aglets software development kit was originally developed at IBM Tokyo Research Laboratory. When the installation process is done, we can run the Aglets server called Tahiti which prompts for login name and password which we can use default values given in manual.

Some experimental results are shown below:

a) The execution time on protected agents is 40% higher than the execution of unprotected agents on average.

Number of Host	2	4	8
Encrypted Agent	17500	32400	46200
Unprotected Agent	11050	22500	33800

Table 1: Execution time for number of hosts

b) Rate of successful Iterations increases nearly 50% with the help of proposed security model.

No.of Iterations	2	4	6	8
Protected Agent	45	57	65	80
Unprotected Agent	20	30	35	38

Table 2: Successful Iterations in Protected and unprotected Agents Systems

c) The proposed model increases the size of program code because it uses both data cryptography

technique and code obfuscation but the failure rate is greatly reduced by this proposed model so, we can neglect that in case of complex applications.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents some of the main issues in the security of mobile agents against attack from malicious host. This paper presents the most important techniques for providing security in mobile agent systems. We concluded that none of the existing techniques provides an optimal solution for all scenarios. However, a combination of various techniques may yield powerful solutions. So, we proposed a hybrid security model that revolves around the security of agent's code, data and itinerary from malicious execution environment.

In future, a more advanced cryptographic technique can be applied, so that the mobile agent's data can be made more secure while migrating from one host to another. This solution addresses most of the problem but still it is very much dependent on the complexity of the algorithm used and the possibility that how soon the professional hacker can de-obfuscate the program. It still does not address the problem of denial of service.

REFERENCES

[1] A. Corradi, R. Montanari," Security Issues in mobile agents Technology", IEEE Internet Computing, Vol. 1, 1999.
[2] T. Taka Tadanori, M. Takashi Watanabe," A Model of mobile agents Services Enhanced for Resource Restrictions and Security", IEEE Internet Computing, 1995.
[3] H. Lee, "The Use of Encrypted Functions for mobile agent Security", Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.
[4] T. Sander, C. Tschudin, "Protecting mobile agents Against Malicious Hosts", In G. Vigna, editor, mobile agent Security, pages 44–60. Springer Verlag: Heidelberg, 1998.
[5] B. Yee, "Using Secure Coprocessors", PhD thesis, Carnegie Mellon University, 1994.
[6] X. D. Guan, Y. L. Yang, and J. Y. You, "POM - A Security Model against Malicious Hosts", DCTC Tech Report, IEEE Computer Society, Shanghai Jiaotong Univ. Dec. 2000.
[7] N. Karnik, "Security in mobile agent Systems" PhD Thesis, Department of Computer Science and Engineering, University of Minnesota, 1998.
[8] L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab, and P. LeBlanc, "Self- Protecting mobile agents Obfuscation Report", Network Associates Laboratories, June 2003.
[9] A. Young, M. Yung, "Encryption Tools for mobile agents: Sliding Encryption," In: E. BIHAM (ed), Fast Software Encryption, Springer-Verlag, Germany, 1997.
[10] G. Karjoth, J. Posegga, "Mobile agents and Telcos' Nightmares," Annales des Telecommunications VoL 55, No. 7/8, 29-41, 2000.
[11] G. Vigna, "Protecting mobile agents Through Tracing," Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, Jyvalskylä, Finland, June 1997.
[12] J. Riordan, B. Schneier, "Environmental Key Generation Towards Clueless Agents," G. Vinga (Ed.), Mobile agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
[13] Yan Li, Min Fu, Lina Yu, "E-Commerce Security Model Construction Based on Mobile Agent", IEEE International Conference on Networking and Digital Society, 2010.
[14] V. Roth, "Secure Recording of Itineraries Through Cooperating Agents," Proceedings of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, pp. 147-154, INRIA, France, 1998.
[15] B. Yee, "A Sanctuary for mobile agents," DARPA Workshop on Foundations for Secure Mobile Code, Feb. 1997.
[16] G. Karjoth, N. Asokan, and C. Glc, "Protecting the Computation Results of Free- Roaming Agents", Second International Workshop on mobile agents, Stuttgart, Germany, Sep. 1998.
[17] G. Wroblewski, "General Method of Program Code Obfuscation", PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002.
[18] F. Hohl, "Time Limited Blackbox Security: Protecting mobile agents from Malicious Hosts," To appear in mobile agents and Security Book edited by Giovanni Vigna, published by Springer Verlag 1998.
[19] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (Im)possibility of Obfuscating Programs," in Advances in Cryptology, Proceedings of Crypto'2001, Lecture Notes in Computer Science, Vol. 2139, pages 1-18.
[20] G. Hachez, "A Comparative Study of Software Protection Tools Suited for Ecommerce with Contributions to Software Watermarking and Smart Cards," Universite Catholique de Louvain, 2003.
[21] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Technical Report 148, Department of Computer Science, University of Auckland, July 1997.
[22] W. Jansen, T. Karygiannis, "Mobile agent Security," NIST Special Publication 800-19, National Institute of Standard and Technology, 2000.
[23] Wayne A. Jansen, "Countermeasures for Mobile Agent Security" March 01, 2010.
[24] S. Armoogum, A. Caully," Obfuscation Techniques for Mobile Agent code confidentiality", March 2010.
[25] S. Srivastava, G.C Nandi, " Detection of Mobile Agent's blocking in Secure Layered Architecture", IEEE International Conference on Communication Systems and Network Technologies, 2011.