An Agent System Protection Mechanism for Secure Action of Mobile Agent in Open Network Systems

Chang-Ryul Jung* · Hong-Sang Yoon** · Jin-Gwang Koh***

Open Network 시스템에서 이동에이전트의 안전한 역할 수행을 위한 에이전트 보호메커니즘

정창렬* · 윤홍상** · 고진광***

ABSTRACT

In the resent years, the term mobile agent is probably one of the most overused words in many appliable areas of distributed open systems as electronic commerce and electronic data interchange, and it has very different meanings in the area of artificial intelligence, network management, or distributed systems. However, the use of mobile agent adds significant problems, primarily in the area of EC/EDI. Therefore it is very important to control the roaming agents to keep one's privacy or property in distributed open networks. The surge in secure intranets for commercial applications provides a robust, secure environment to which trading partners can increasingly entrust their interactions to some mobile agents. In this paper, we introduce a mechanism to protect mobile agent itself from the malicious server he is visiting and also we introduce a mechanism to protect vital resources of the open systems as internet

요 약

최근 몇 년간 이동에이전트는 분산시스템의 전자상거래와 전자데이터의 교환과 인공지능, 네트워관리와 같은 여러 응용분야에서 많이 이용되고 있는 분야중 하나이다. 특히 EC/EDI의 영역에서 본래의 이동에이전트 서명 문제를 추가하여 사용되어야 한다. 그러므로 분산네트웍에서 이동하는 에이전트의 프라이버시와 속성을 보호하는 것은 매우 중요하다. 상거래에 응용되기 위해서는 각각의 이동에이전트들 상호간에 거래의 신뢰성을 향상할 수 있는 안전한 인트라넷 환경이 필요하다.

본 연구에서는 이동에이전트를 악의적인 서버로부터 이동에이전트를 자신을 보호하고 인터넷과 같은 오픈 시스템의 자원들을 보호하는 안전한 이동에이전트 역할을 수행하도록 하는 에이전트 보호 메커니즘을 설계하여 신뢰하는 실행환경을 제안한다.

키워드

Mobile Agent, EC/EDI, Protection Mechanism, Agent Protection

* 순천대학교 컴퓨터과학과 박사과정
** 광주대학교 컴퓨터정보통신공학부 교수
*** 순천대학교 정보통신공학부 정교수
접수일자 : 2002. 4. 15

Ⅰ. Introduction

As the use of internet has grown rapidly, the advances of the computer networking are extending the vision of on-line control beyond the walls of the single plant or even the individual firm. Electronic Data Interchange(EDI) is making many firms more comfortable with on-line interactions that cross company boundaries. The use of web browsers as a Common Gateway Interface(CGI) for a wide range of applications, and the ability of Java to support common programs across multiple distributed platforms, provide a common computational platform that can span a wide geographic range[1]. Adding the mobile agents into the EC/EDI makes innovative approach to structure distributed applications[2]. Also, the use of mobile agent requires that each cooperating host in the distributed system should provide a facility for executing them.

In open distributed systems, such as the internet, it is unwise to assume that all agents are benign, and thus a certain amount of effort must be spent to ensure that sensitive resources are protected from unauthorized access. This can be accomplished by using a system of capabilities and by predicating resource access on possession of the appropriate capability. It is unreasonable, however, to expect that every use of every resource in a network be thus verified dynamically; such a requirement surely would degrade performance unacceptably. Thus it is attractive to develop some mechanisms that guarantee controlled access to host resources.

For introducing a mobile agent security mechanism, we first study about the structure of mobile agent and security considerations of the mobile agents recently developed and released. After surveying the security abilities of mobile agents, the appropriate security mechanism is introduced for giving the incoming agents a trusted execution environment of the system resource access. It requires some environmental facilities to the incoming agents. Then we explain the mechanisms to protect mobile agents from malicious agent server, and protect agent servers from malicious mobile agents. Finally, the contributions of our work and further works are mentioned in the last section of this paper.

Ⅱ. Mobile agent execution

The basic task of any mobile agent system is the actual execution of mobile agents. This means that the code of a mobile agent has to be executed in the context of the data and facilities available at an agent server, which were the original reason for the migration of the mobile agent to this agent server. Figure 1 shows a mobile agent system in general. The execution of an agent's code is accomplished in an execution environment that is provided from the agent server. The execution environment is an integral part of the agent server and is also responsible for the provision of a standardized runtime library with a well defined API(Application Programmers Interface)[5]. This runtime library implements the interface between the mobile agents and the other services offered by the agent server. It also allows to reduce the size of mobile agents since they do not need to carry the code which they can be confident to find on the agent server.

The primary identifying characteristic of mobile agents is their ability to autonomously migrate from host to host. An agent can request its host server to transport it to some remote destination. The agent server must then deactivate the agent, capture its state, and transmit it to the server at the remote host. The destination server must restore the agent state and reactivate it at the remote host, thus completing the migration.

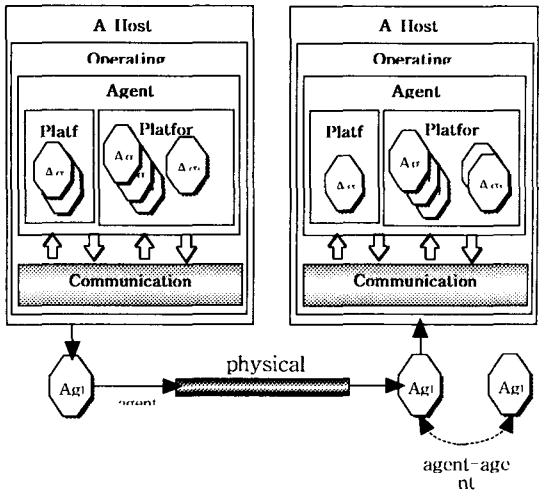

Fig. 1 A mobile agent system

Table 1. Agent mobility support

System	Naming	Migration
Agent Tcl	Location dependent based on DNS	Absolute Command : agent-jump
Aglets	Location dependent URLs based on DNS names	Absolute Command : dispatch
Concordia	Location dependent based on DNS	Absolute Based on the Agent's Itinerary
Tacoma	Location dependent based on DNS	Absolute and Relative Command : meet
Voyager	Location independent based on global ID	Absolute and relative Command : moveTo

The agent code can then direct the control flow appropriately when the state is restored at the destination. However, this only captures execution state at a function-level, in contrast to the machine instruction-level state provided by the thread context.

Another issue in implementing agent mobility is the transfer of agent code. One possibility is for the agent to carry all its code as it migrates. This allows the agent to run on any server which can execute the code. Some systems do not transfer any code at all, and require that the agent's code be pre-installed on the destination server. This is advantageous from a security perspective, since no foreign code is allowed to execute. However, it suffers from poor flexibility and limits its use to closed, local networks. During the agent's execution, if it needs to use some code that is not already installed on its current server, the server can contact the code base and download the required code. This is sometimes referred to as code-on-demand. The table 1 shows Agent mobility supported by some mobile agent systems.

Ⅲ. Design of Protection mechanism for mobile agent system

The security problems in mobile agent systems are due to the fact that the various principals in a mobile agent system who are the operators or owners of the different components and who lend their authority to these components, can not be assumed to trust each other[5]. Security in a mobile agent system is concerned with the protection of the two primary components that we have identified: the mobile agent and the agent server. The mobile agent has to be protected from tampering and from disclosure while it is in transit, but it should also be protected when it is executing on an agent server. The agent server in turn must be protected from malicious or simply buggy mobile agents[8].

3.1 Protection of the mobile agent

Protecting the mobile agent while traveling over an untrusted network, when it is marshaled into its transport format, amounts to the well known problems of confidentiality and integrity protection of messages[6]. Due to the special structure of mobile agents, the differentiation between the changeable and the unchangeable parts of an agent is exist, which can be independently protected. Provided that the mechanisms used for integrity protection support origin authentication, for instance

by using appropriate public-key signatures, the unchangeable part can be signed by the agent owner to assert that it has not been tampered with while in transit. Also, the unchangeable parts of a publicly available agent may not have to be confidentiality protected, unless the user wants to hide the fact that he is using this mobile agent. The changeable part of a mobile agent can be integrity and confidentiality protected with the help of well-known cryptographic mechanisms. If additional accountability is required, the intermediate states of the mobile agent can be signed by the agent executors and gathered in the mobile agent. The two parts should be tied together with appropriate cryptographic mechanisms to prevent an attacker from arbitrarily composing these parts[7].

The problem of protecting a mobile agent when it is executing on an agent server consists of protecting it from other agents on the same agent server and from the agent server itself, which may be operated by a malicious agent executor. The former problem is a well-known problem from operating systems that has to be addressed in the resource management. The agent execution service can rely on memory protection and access control mechanisms to prevent local agents or other entities from accessing or tampering with the code or state of a mobile agent[3]. The latter problem, which is concerned with preventing undesired access to or manipulation of the mobile agent's state or code by the agent server, can be solved by assigning all the statements of an agent in logically separated registry as base and bounds registers in the memory. All of the base and bounds are assigned in a different access entries as "R: read only" and "X: execute only". All of the access request from the agent server or even the other agent must be followed by the restrictions of the access entry. If any violations occurred during the agent execution, it will be reported to the host and stop execution. Then the agent will migrate to the next itineraries.

Fig. 2 Logical assignment of a mobile agent

Logical separation	Contents of an agent	
Base1(R)	Agent Code	Agent statement
:		
Bound1(R)		
Base2(R)	Agent data	
:		
Bound2(R)		
Base3(X)	Agent state	
:		
Bound3(X)		

This mechanism is designed to protect mobile agent's data from destroying, manipulating, or disclosing it. A simple diagram of base and bounds registry protection is shown in the fig 2.

When an agent executes on a host's agent server, it is in effect completely exposed to that host. The act of migrating to a server implies a certain level of trust in that server and its host. If the server happens to be malicious, it can affect the agent in many different ways: ① It can simply destroy the agent and thus impede the functioning of its parent application. ② It can steal useful information stored in the agent, such as intermediate results gathered by the agent during its travels. ③ It can modify the data carried by the agent, for example changing the price quoted by a competitor in a shopping mall, to fool the parent application into favouring the malicious server. ④ It can attempt to change the agent's code and have it perform malicious actions when it returns to its home site. This is especially dangerous, since the home site could believe its own agents as trusted entities, and possibly allow them to bypass access controls to its own resources.

An agent server must of necessity have access to the agent's code and state in order to execute it. Parts of state in fact must usually change, in order to store the results of computations or queries. Thus it is not possible to provide a general guarantee that the agent will not be maliciously modified. However, the parent application must have some mechanism for detecting such modifications. If it determines

that the agent has been "attacked", it can take appropriate measures, such as executing it in a restricted environment or even discarding it altogether. When an agent is dispatched, it has an initial itinerary of hosts to visit. Different parts of the agent may be intended for different hosts, and some parts may need to be kept secret until the agent arrives at the intended host[12]. This allows the application to ensure that the agent followed the intended itinerary.

One of the toughest challenges in our agent security mechanism is the protection of agent code and state from malicious servers they happen to execute on. While no general- purpose solutions seem feasible at this stage, we designed and implemented a mechanism for protecting parts of the agent state. Some modules in the state can be made read-only; others can be inserted into unmodifiable containers. Parts of the state can be targeted towards specific agent servers, such that other servers cannot read or tamper with them.

3.2 Protection of the agent server

The remainder of this section is concerned with the problem of protecting the agent server from malicious mobile agents that try to circumvent the protection mechanisms established by the agent server. Another security mechanism is to protect the facilities such as excessive and uncontrolled use of cpu, memory, or communication bandwidth of an agent server, which are the primary assets of the agent executor[9]. In addition, they are also responsible to protect other mobile agents executing on the same agent server from malicious agents. This problem is very similar to the problem of protecting a host from downloadable mobile code.

The introduction of mobile code in a network raises several security issues. In a completely closed local area network, it is possible to trust all machines and the software installed on them[9]. Users may be willing to allow arbitrary agent programs to execute on their machines, and their agents to execute on arbitrary machines. However, in an open network such as the Internet, it is entirely possible that the agent and server belong to different administrative domains. In such cases, they will have much lower levels of mutual trust. Several types of security problems may arise:

① Servers are exposed to the risk of system penetration by malicious agents, which may leak sensitive information. ② Sensitive data contained within an agent dispatched by a user may be

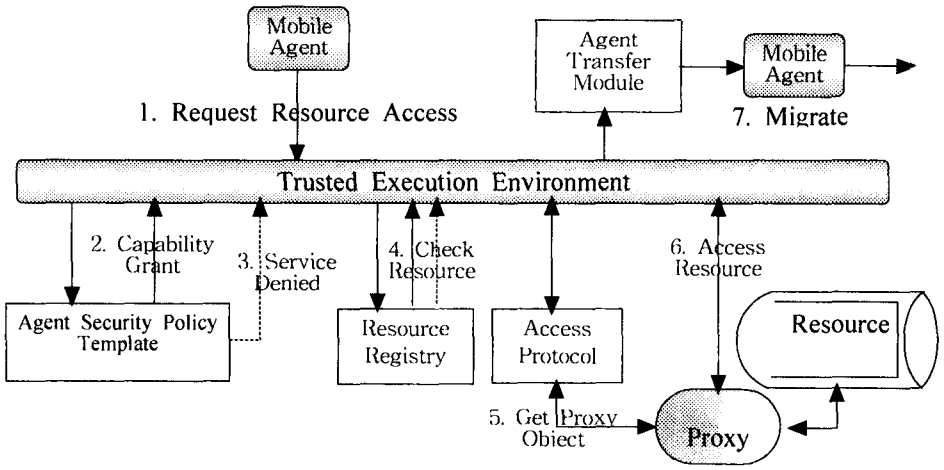

Fig. 3 Trusted Execution Environment

compromised, due to eavesdropping on insecure networks, or if the agent executes on a malicious server. ③ The agent's code, control flow and results could be altered by servers for malicious purposes. ④ Agents may mount "denial of service" attacks on servers, whereby they destroy server resources and prevent other agents from progressing.

Therefore, as we proposed in this paper, the mobile agent system must provide a strong security mechanisms for protecting either agent itself or host resources. These include privacy mechanisms to protect secret data and code, authentication mechanisms to establish the identities of communicating parties and authorization mechanisms to provide agents with controlled access to server resources. Building on the secured resource registry, a resource protection facility for servers was designed. Based on the concept of proxy interposition, this scheme allows servers to make resources available to agents securely, with dynamic, fine-grained control over their access rights[8]. Metering and accounting mechanisms can also be built in so as to allow servers to charge agents for the resources they use. The next section of this paper describes this resource access mechanism.

3.2.1 Build a trusted execution environment

The allocation of these resources must be guided by some policy of the agent server, which identifies what kind of and how many resources should be made available to a mobile agent.

Once certain resources have been allocated to an agent, confining the agent to these resources is a very important issue. Most of the mechanisms required for this purpose are well-known operating system mechanisms. The CPU can be protected from monopolization or excessive use through a clock interrupt, which limits the amount of time an agent can execute. The creation of new threads can be regulated via a limitation on memory usage and on the number of CPU cycles consumed by all the threads that belong to a certain agent. The access to memory can be protected with base and bounds registers or paging mechanisms[3]. This issue is very important in the context of security and allows us to protect the agent server from possibly malicious mobile agents and mobile agents from one another. We assume that any access to resources other than the CPU and memory will be mediated through functions in the runtime library offered by the TEE(Trusted Execution Environment) which includes the allocation of new memory. It is therefore the task of the execution environment to implement generic access control mechanisms for other resources and to enforce the access control policy defined for the agent server.

Whenever the agent calls a potentially dangerous operation, the Trusted Execution Environment acts as a monitor and screens the request based on its security policy. If it is allowed to proceed, it calls the trusted environment which provides the actual access to the resource. The resource provider, then retains complete control over the security policy, and can customize the policy for each resource type by using trusted execution environments as proxies. The proxy objects can be tailored to specific agents and dynamically generated as required(see figure 4). Once a safe proxy is made available to an agent, access control checks only require a minimal amount of computation since the security policy need not be consulted. Security policy checks are performed only once at the time of issuing a capability[1]. Resource access only involves checking the submitted capability for tampering. If the security policy changes, however, a previously issued capability may need to be cancelled.

Ⅳ. Conclusion

The specific research challenges related to

security that this paper addresses include the protection of host resources from malicious agents and the protection of agents from tampering by other agents or their hosts. Our contributions in addressing these challenges are detailed in the trusted execution environment that we designed and implemented. A mobile agent server must support the secure execution and migration of mobile agents. It should also provide a mechanism for agents to bind themselves to their host environment, and be able to access the resources available at the server. Access control and authentication are necessary so that the system can ensure that resources are not misused. Secure interfaces for application control over remote agents must also be provided. The trusted execution environment is designed to address these requirements. The trusted execution environment provide secure resource registry in which they isolate agents, preventing malicious agents from tampering with other agents. Resources are made available to specific secured domains in a controlled manner. All entities in the system are assigned global, location-independent names, and a name service is provided for translating entity names into their current locations. Agents are assigned credentials which identify them along with their ownership information.

Bibliography

[1] B.C. Neuman. proxy-based authorization and accounting for distributed systems. In Proceedings of the Thirteenth International Conference on Distributed Computing Systems, pages 283-291, May 1993.

[2] Colin G. Harrison, David M. Chess, and Aaron Kershen, baum. Mobile Agents: Are they a good idea? Technical report, IBM Research Division, T.J.Watson Research Center, March 1995. http://www.research.ibm.com/massdist/mobag.ps.

[3] Dag Johansen, Robbert van Renesse, and Fred B. Schneider. Operating System Support for Mobile Agents. In Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems, pages 42-45, May 1995.

[4] H. Van Dyke Parunak. Practical and Industrial Applications of Agent-Based Systems. http://www.iti.org/, Industrial Technology Institute, 1998.

[5] Hong-sang Yoon. Thesis: A Security Mech anism of Mobile Agent on the Distributed Communication Networks. Dec. 1999.

[6] Robert S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In Proceedings of the Fourth Annual Tcl/Tk Workshop(TCL '96), July 1996.

[7] Tomas Sander and Christian F. Tschudin. To wards Mobile Cryptography. Technical Report TR-97-049, International Computer Science Institute, Berkeley, California, November 1997.

[8] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for Mobile Agents: Issues and Requirements. In Proceedings of the 19th National Information Systems Security Conference, pages 591--597, October 1996.

[9] W. Stallings. Network and Internetwork Security Principles and Practice. Prentice Hall, Inc. Englewood Cliffs, New Jersey,

저 자 소 개

정창렬(Chang-Ryul Jung)
1995년 광주대학교 전자계산학과(학사)
1999년 순천대학교 교육대학원 컴퓨터교육과(석사)
2000년~현재 순천대학교 대학원 컴퓨터과학과 박사과정

※관심분야 : 정보보안, 전자상거래, 멀티미디어 보안, Image Processing

윤홍상(Hong-Sang Yoon)
1987년 Western Illinois Univ. 수학과(이학사)
1990년 WIU대학원 컴퓨터과학과(이학석사)
2000년 순천대학교 대학원 컴퓨터과학과(이학박사)
2000. 8~현재 광주대학교 컴퓨터정보통신공학부 전임강사
※관심분야 : 정보보안, 이동에이전트, 정보통신

고진광(Jin-Gwang Koh)
1982년 홍익대학교 컴퓨터공학과(이학사)
1984년 홍익대학교 대학원 컴퓨터공학과(이학석사)
1997년 홍익대학교 대학원 컴퓨터공학과(이학박사)
1984. 3~1988. 2 송원전문대학 전자계산과 전임강사
1988. 3~현재 순천대학교 공과대학 정보통신공학부 교수
1992. 3~1993. 2 홍익대학교 공과대학 컴퓨터공학과 국내교류교수
1997. 8~1998. 8 Oregon State Univ. 컴퓨터공학과 방문교수
2000. 12~2001. 2 일본 류큐대학 정보공학과 방문교수
2001. 3~2002. 현재 순천대학교 정보전산원장
※관심분야 : 데이터베이스, 전자상거래, 정보보안