An Overview of SECMAP Secure Mobile Agent Platform

Suat Ugurlu[1], Nadia Erdogan[1]

[1]Istanbul Technical University, Computer Engineering Department,
Ayazaga, 34390 Istanbul, Turkey
suat@suatugurlu.com, erdogan@cs.itu.edu.tr

Abstract. Mobile agent technology presents an attractive alternative to the client-server paradigm; however, the lack of a feasible agent security model seriously hinders the adoption of the agent paradigm. This paper describes a mobile agent platform, Secure Mobile Agent Platform (SECMAP) and its security infrastructure. SECMAP presents abstractions which ensure the protection of agents and system components through a shielded agent model. It provides secure agent communication and migration facilities, and maintains security policy information to examine agent actions and to prevent undesired/unauthorized activity.

1 Introduction

There exists a wide range of security issues in using mobile agents and, in spite of its several advantages; the lack of a feasible agent security model seriously hinders a wider adoption of mobile code based applications. When compared to traditional systems, mobile agents face several security risks. Both mobile agents during their life times and hosts executing mobile agents are under security threats [1], [2], [3].

A secure mobile agent system should not only support basic agent requirements such as facilities for agent communication and agent mobility, but also must provide a security model to protect agents and hosts without causing any overhead to the programmer.

This paper describes a new mobile agent platform, Secure Mobile Agent Platform (SECMAP) and its security infrastructure. Unlike other agent systems, SECMAP proposes a new agent model named as the *shielded agent* model for security purposes. A shielded agent is a highly encapsulated software component that ensures complete isolation against unauthorized access of any type. SECMAP provides secure agent communication and migration facilities as well, and maintains security policy information to examine agent actions and to prevent undesired/unauthorized activity. The system ensures protection of different agents and system components by enforcing security policies for various agent activities and continuously monitors and reports on the execution of an agent from its creation to its completion. SECMAP is written in Java and is therefore platform independent.

2 Security Model of SECMAP

In a mobile agent system, agents cannot be reliably associated with end users without taking certain precautions. The approach taken by SECMAP is to treat every agent as a distinct principal and to provide protection mechanisms that isolate agents. SECMAP differs from other mobile agents systems in the abstractions it provides to address issues of agent isolation.

SECMAP provides a light-weight implementation of agents; they are implemented as threads instead of processes. Each agent is an autonomous object with a unique identification and agents communicate via asynchronous message passing.

A Secure Mobile Agent Server (SMAS) resident on each node presents a secure execution environment on which new agents may be created or to which agents may be dispatched. A SMAS may operate in three modes according to the functionality it exhibits: standard mode(SM-SMAS), master browser mode (MB-SMAS), or security manager mode(SM-SMAS). SMAS working in standard mode provides basic agent services such as agent creation, activation, inactivation, destruction, communication, and migration. It also includes a policy engine that checks agent activity and resource utilization according to the rules that are present in a policy file, which has been received from a Security Manager. MB-SMAS an SM-SMAS are also both capable of supporting all functionalities of standard SMAS. However, they have additional responsibilities. MB-SMAS maintains the name-location directory of all currently active agents in the system. SM-SMAS, on the other hand, performs authentication of all SMAS engines and is in charge of the distribution of SMAS certificates.

SMAS provides functionalities that meet security requirements and allow the implementation of the *shielded agent* model. A shielded agent is a highly encapsulated software component that ensures complete isolation against unauthorized access of any type. On a request to create a new agent, SMAS instantiates a private object of its own, which is an instance of predefined object *AgentShield*, and uses it as a wrapper around the newly created agent by declaring the agent to be a private object of *AgentShield* object. This type of encapsulation ensures complete isolation, preventing other agents to access the agent state directly. An agent is only allowed to communicate with its environment over the SMAS engine through the methods defined in a predefined interface object, *AgentInterface*, which is made the private object of the agent during the creation process. The interface provides limited yet sufficient functions for the agent to communicate with SMAS. All variables of agents are declared to be private and they have corresponding accessor methods. Agents issue or receive method invocation requests through asynchronous messages over the secure communication facility of SMAS. Thus, a source that is qualified for a particular request, for example, that has received the rights to communicate with a target agent, is granted to pass its message.

SECMAP allows the concurrent execution of several agents on the same host and each agent runs as a separate thread in the same memory area of the host. In this mode of operation, the shielded agent model suffices to guarantee inter agent isolation and protection.

SECMAP employs cryptographic techniques to meet security constraints. Each SMAS owns a certificate which is used to identify its identity and to encrypt and decrypt data. A request from a SMAS is not processed before the validity of the SMAS identity is verified. A SECMAP agent's code and state information are kept encrypted during its life time using Data Encryption Standard (DES) algorithm. They are decrypted only when the agent is in running state on the host's memory. Thus, an agent is identified as a black box on a host, except while in memory. To protect agents during migration over the network, agent code and state data are encrypted as well while in transfer and can only be decrypted on the target host after retrieving the appropriate DES key from the security manager.

SECMAP monitors, time stamps and logs all agent activity in a file, in order to be later analyzed to determine the actions an agent has carried out on the host. In case an unexpected result is recognized, the route of the agent can be traced and how the agent has executed on each host can be detected. In addition, in case of a threat, SMAS has the privilege to end the execution of an agent.

2.1 Security Policies

SECMAP employs a policy based authorization mechanism to permit or restrict agents to carry out certain classes of actions. Agent communication, migration, disk I/O, access to system resources are some of the events that require enforcement of security policies. SECMAP allows for policies to be dynamically defined and be enforced by intercepting agent service requests. Two types of policies are defined: agent policies and host policies. Agent policies are specified by the agent owner when he is deploying the agent and are carried together with the agent while migrating over the network. An agent policy simply defines the rights an agent possesses, such as disk access rights or the right to create network connections. A user may modify agent policies after deployment. Host policies, on the other hand, determine restrictions on access to the host resources by the agents. Policies are kept as encrypted XML files by each SMAS.

2.2 Agent Communication

SECMAP agents communicate via messages. The platform supports asynchronous message exchange primitives through methods of *AgentInterface*. Agent communication is secured by transferring encrypted message content through SSL. Agents are provided with a flexible communication environment where they can question the results of a message send request, wait for a response for a specified period of time, and receive messages or replies when it is convenient for them.

2.3 Location Transparence

The system is managed with a decentralized control; several MB-SMAS and SM-SMAS may concurrently be active and they cooperate for a smooth execution. They share their data and communicate messages to keep them coherent. When initializing an S-SMAS on a node, the programmer specifies the addresses of the MB-SMAS and the SM-SMAS it should register itself to. Next, S-SMAS sends its agent list to MB-SMAS and, in return, receives the identities of all other agents active on the system. We call those S-SMAS that a MB-SMAS or a SM-SMAS cooperates with as its *partners*. When a MB-SMAS gets a request to return an agent identity, it cooperates with its partners to obtain the current agent identities. A similar mode of processing is true for SM-SMAS. If an SM-SMAS can not authenticate a request, it directs it to its partners for possible authentication. Additionally, when an S-SMAS communicates with its MB-SMAS and SM-SMAS, it obtains the addresses of their partners and saves them as well, in order to use as a contact address in case its communication to its MB-SMAS or SM-SMAS fails. This approach adds robustness against network or node failures.

2.4 Agent Migration

SECMAP supports weak migration of agents between remote hosts on a call to the *Move* method of *AgentInterface*. The agent that wants to migrate should specify the address of the remote host where it wants to be transferred. When agent transfer completes, its new location information is updated on MB-SMAS automatically so that any new message destined to this agent is redirected to the correct SMAS. An agent can not receive any messages while it is being transferred. Therefore, the agent programmer needs to question the result of the send operation and re-send the message if the operation has failed. All communication that is carried out between SMAS engines to complete the transfer of the agent is encrypted through SSL.

3 Related Work

Developers and researchers have taken a variety of approaches to provide security of mobile agent environments. Hohl [4] proposes what he refers to as Blackbox security to scramble an agent's code in such a way that no one is able to gain a complete understanding of its function. Proof carrying code [5] requires the author of an agent to formally prove that the agent conforms to a certain security policy. By digitally signing an agent, its authenticity, origin, and integrity can be verified by the recipient. The idea behind path histories [6] is to let a host know where a mobile agent has been executed previously. State appraisal [7] attempts to ensure that an agent's state has not been tampered with and that the agent will not carry out any illegal actions through a state appraisal function which becomes part of the agent code. There does not seem to be a single solution to the security problems introduced and most of the solutions are inadequate in protecting agent and host data, while others that provide adequate protection cause an unacceptable overhead to the programmer. No DOS protection is available in any system because of the difficulty of its detection and prevention. However, a system should at least have a monitoring and logging mechanism to analyze agent activities and use these data to later prevent DOS attacks. A dynamic policy based security management is also absent in most systems.

4 Conclusions and Future Work

This paper describes a mobile agent platform, SECMAP, and its security infrastructure. The system has been especially developed against security threats that both agents and hosts may be exposed to. Security features are inserted into the system core at design time. The system has an open and flexible architecture that can further be enhanced in the future to meet additional requirements.

SECMAP allows for completely isolated lightweight agents with flexible and efficient communication facilities. Sources of requests are authenticated before they are processed to verify that they really come from their stated sources. SECMAP introduces trusted nodes into the infrastructure; to which mobile agents can migrate when required, so that sensitive information can be prevented from being sent to untrusted hosts. This approach does not appear to be fully explored elsewhere. Currently, work is in progress on detection and resolution of policy conflicts and enforcement of security policies. Our future work also includes the addition of dynamic policy creation capability to the architecture with the help of log analysis.

References

1. Christian F. Tschudin "Mobile Agent Security" Department of Computer Systems, Uppsala University, Sweden
2. Karnik, N.M., Tripathi, A.R., 1998. Design issues in mobile-agent programming systems. IEEE Concurrency 6 (3), 52–61
3. V.Varadharan and D.Foster ,"A Security Architecture for Mobile Agent Based Applications" World Wide Web: Internet and Web Information System, 6,93-122,2003
4. F. Hohl, "Protecting mobile agents with black box security" Proc. 1997 Wksp. Mobile Agents and Security , Univ. of Maryland , Oct 1997
5. F. Sander, "On cryptographic protection of mobile agents" Proc. 1997 Wksp. Mobile Agents and Security , Oct 1997
6. Uwe G. Wilhelm and Sebastian Staaman "Protecting the itinerary of Mobile Agents" Laboratoire de Systemes d'Exploitation, Switzerland,1998
7. Vipin Swarup "Trust Appraisal and Secure Routing of Mobile Agents" The Mitre Corporation 1997