ROUTE DETECTING SYSTEM USING MULTI-AGENT FOR MOBILE AGENTS

Yuki KOTEGAWA

Graduate School of Information Science
and Electrical Engineering Kyushu Univ.
6-10-1 Hakozaki, Higashi-ku, Fukuoka
812-8581 Japan
kotegawa@itslab.csce.kyushu-u.ac.jp

Toshihiro TABATA, Kouichi SAKURAI

Faculty of Information Science
and Electrical Engineering Kyushu Univ.
6-10-1 Hakozaki, Higashi-ku, Fukuoka
812-8581 Japan
{tabata, sakurai}@csce.kyushu-u.ac.jp

ABSTRACT

The effects of mobile agent technologies are expected in mobile environment. However, the problem of security exists. Malicious host may tampers with a mobile agent. In this paper, we propose a system which records a migration route of a mobile agent in secure. Moreover our proposal corresponds, when the mobile agent is terminated unjustly.

1. INTRODUCTION

1.1. Background

Using mobile agent technologies, an Agent-Owner can leave a processing on a computer network to an agent. Moreover, communication between hosts can be localized to the communication between programs in a host, and it can also reduce the communication delay which is the bottleneck of processing using network communication. Especially, the effects of mobile agent technologies are expected in mobile environment because of low spec mobile PC and unstable network.

However, problems of security occur with mobility that is a feature of mobile agent technologies. Those problems can be divided into three problems. One is to protect hosts from agents. Another is to protect agents from agents. The other is to protect agents from hosts. Specialty, third problem is important since an agent is defenseless to a host. A malicious host is able to tamper with an agent or terminate it. That is, an agent might be tampered or terminated by a malicious host. Therefore, complete protection of an agent or detection of a tampering against an agent is required. Moreover, a secure recording of a migration route of an agent is also required. If a problem occurs to the agent, searching of the agent is required using the route record.

There are some ideas to protect agents completely such as "Tamper-proof devices" [1], "Obfuscation of programs" [2], and "Mobile Cryptography" [3]. However, there is a problem that the malicious hosts don't install "Tamper-proof devices. Moreover, about "Obfuscation", there is a problem that technique which evaluates the effect quantitatively has not been established. About "Mobile Cryptography", there is a problem that it can use only for limited functions. Thus, it can be said that it is very difficult to protect agents completely at present. Therefore, importance is attached to ideas which detect tampering such as "Execution Traces" [4].

On the other hand, there is an idea [5] using communications between two agents for recording of an agent's migration route. However, the idea isn't suitable on the case where an agent is terminated.

1.2. Our contribution

In this paper, we propose a system which records a migration route of a mobile agent in secure. Moreover our proposal corresponds, even if the mobile agent is terminated unjustly.

2. AGENT'S ROUTE RECORDING ALGORITHM

In this section, existing agent's route recording algorithm proposed by Roth.

2.1. Roth's Algorithm

Roth's proposal [5] is as follows.

Definition

$\boldsymbol{H_a}$	the host which is visited by agent a.
$\boldsymbol{h_i}(\in \boldsymbol{H_a})$	the ith host which is visited by an agent.
$\boldsymbol{id(h_i)}$	the identity of host h_i.
$\boldsymbol{prev_i}$	agent a's idea of the identity $id(h_{i-1})$ of its previous hop
$\boldsymbol{next_i}$	the identity of the next hop agent a wants to take while being on host h_i.
$\boldsymbol{h_n}$	the agent's routes end at their origin, $h_n = h_0$ for a route with n hops.

Initialization

Let h_0 be the origin of agents a and b. h_0 doesn't attack to

a and b.
Let $next_0$ be the first hop of their respective itineraries.

Step $i, i \in \{1, ..., n\}$
Agents a sends the $next_i$ and the $prev_i$ to agent b over the authenticated channel. Agent b learns $id(h_i)$ using the authentication. Agent b verifies that $id(h_i) = i-1 \bigwedge prev_i = id(h_{i-1})$ and appends $next_i$ to the stored route. Processing of the agent a and the agent b are shown in ListA and ListB.

ListA. Processing of the agent "a"

`A.1` arrives at h_i.
`A.2` sends messages "$prev_i$" and "$next_i$" to the agent b through h_i.
`A.3` performs tasks.
`A.4` migrates to $next_i$.

ListB. Processing of the agent b (Recording)

`B.1` receives messages "$prev_i$" and "$next_i$" over the authenticated channel
`B.2` if $(id(h_i) = next_{i-1} \bigwedge prev_i = id(h_{i-1}))$, appends $next_i$ to the stored route.

2.2. features

By using Roth's algorithm, under the conditions that H_a and H_b don't conspire, an agent's route is safely recordable.

2.3. problems

When an agent is terminated, this algorithm cannot judge whether it is responsible for which of h_i and h_{i+1}.

3. OUR PROPOSAL

In this section, we propose a system that can detect an agent's migration route even if the agent was terminated by a malicious host. We realize our proposal using three kinds of agents. The agents are classified into the followings.

1. **Task-Agent**
   An agent who processes tasks instead of an Agent-Owner. The Task-Agent is performed at arbitrary hosts.
2. **Route-Agent**
   An agent who records a Task-Agent's migration route. The Route-Agent is performed at a trusted host.
3. **Search-Agent**
   An agent who is generated and used by a Route-Agent in order to search whether a Task-Agent was terminated. The Search-Agent is performed at arbitrary hosts.

The outline figure of the proposal system is shown in Fig. 1.

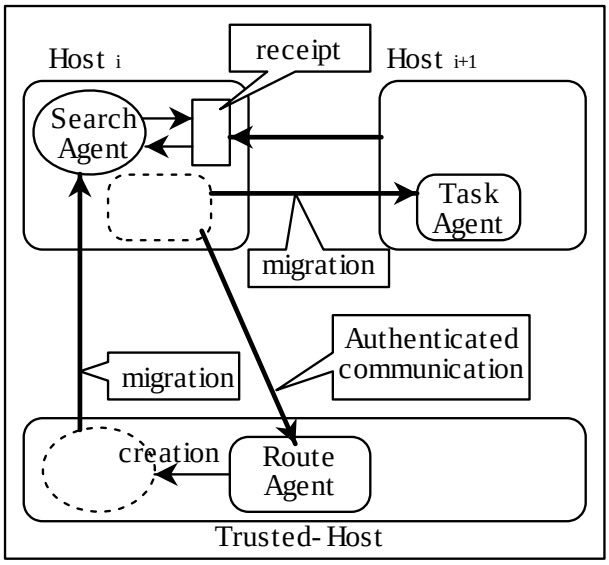

Fig. 1. proposed system

3.1. Premises

Our proposal is premised on the following conditions.

1. Each host can sign data in which oneself has responsibility using a public key.
2. A Task-Agent, a Route-Agent and a Search-Agent are given the authentication channel by hosts.
3. Each Host other than Trusted-Host may terminate agents.
4. Communication channels between hosts are secure. Therefore, an agent is never tampered by a third person during migration.

3.2. Definition

$\boldsymbol{Host_i}$	the host a Task-Agent visits to the ith. This host mignt terminate a Task-Agent.
$\boldsymbol{id(X)}$	the identity of a entity "X".
$\boldsymbol{prev_i}$	an agent a's idea of the identity $id(Host_{i-1})$ of the agent's previous hop.
$\boldsymbol{next_i}$	the identity of the next hop agent a wants to take while being on host h_i.
$\boldsymbol{Sign_x(M)}$	a signature to a message M by $Host_i$.

3.3. Processing of our system

We explain Processing of our proposed system and discribe security of the proposal.

3.3.1. Task-Agent

First, After a Task-Agent arrives at $Host_i$, the Task-Agent sends some data to a Route-Agent executed at a Trusted-Host. Some data are "$prev_i$" and "$next_i$". "$prev_i$" is the identity of the host that the Task-Agent believes that the

Task-Agent visits before $Host_i$. "$next_i$" is the identity of destination that the Task-Agent wants to migrate to. When these data are sent, these are signed by $Host_i$. Then, after the Task-Agent performs their tasks, the Task-Agent migrates to next host $Host_{i+1}$. Although The migration of the Task-Agent is performed by $Host_i$, if the migration to $Host_{i+1}$ is failure, the Task-Agent changes next host $Host_{i+1}$ to $Host'_{i+1}$. This process is continued until movement is successful. The flow of processing of a Task-Agent is shown in List 1.

3.3.2. $Host_i$

When $Host_i$ receives a Task-Agent and a signature "$Sign_{i-1}(id(Host_{i-1}), id(Host_i))$" from $Host_{i-1}$, $Host_i$ verifies the signature. If verification is ture, $Host_i$ returns a signature "$Sign_i(id(Task\text{-}Agent))$" as the receipt to $Host_{i-1}$.

Then the Task-Agent is executed at $Host_i$. If the Task-Agent wants to send messages "$prev_i$" and "$next'_i$" to a Route-Agent, $Host_i$ creates signatures $Sign_i(prev_i, next_i)$ and $Sign_{i-1}(id(Host_{i-1}), id(Host_i))$. And $Host_i$ sends these signatures to the Route-Agent.

If the Task-Agent wants to migrate to next host $Host_{i+1}$, $Host_i$ creates a clone of the Task-Agent for backup. Then $Host_i$ sends a signature $Sign_i(id(Host_i), id(Host_{i+1}))$ and one of the Task-Agents to $Host_{i+1}$. After $Host_{i+1}$ receives the Task-Agent, $Host_{i+1}$ returns a signature "$Sign_{i+1}(id(Task\text{-}Agent))$" as the receipt to $Host_i$. If $Host_i$ receives the receipt, the migration is succusessful and then the remain of the Task-Agents is deleted.

But if $Host_{i+1}$ has malice, $Host_{i+1}$ will not return the receipt. In this cace, $Host_i$ judges that migration is failure. if the migration is failure, $Host_i$ repeates the migration process similarly against another host. The flow of processing of $Host_i$ is shown in List 2.

3.3.3. Route-Agent & Search-Agent

A Route-Agent records the route of a Task-Agent. The Route-Agent receives signatures "$Sign_i(prev_i, next_i)$" and "$Sign_{i-1}(id(Host_{i-1}), id(Host_i))$" from a Task-Agent thr ough $Host_i$. Then the signatures verified. If the verification is ture, the Route-Agent compares $id(Host_i)$ with $next_{i-1}$ then compares $prev_i$ with $id(Host_{i-1})$. If the result of the comparisons is true, the Route-Agent appends $id(Host_i)$ to the stored route.

After recording of $Host_i$ is finished, the Route-Agent measures communication delay "D". Then let NEXTTIME be func(D)+TaskTime. func(D) is a suitable function which takes D to an argument. And TaskTime is a prediction processing time of a Task-Agent in one host. If the Route-Agent doesn't receive next signatures from the Task-Agent through $next_i$ within NEXTTIME, the Route-Agent creates a Search-Agent. Then the Search-Agent migrates to $Host_i$ and gets a signature $Sign_{i+1}(id(Task\text{-}Agent))$. If the Route-Agent receives the signature from the Search-Agent, $Host_{i+1}$ is the malicious host. If not, $Host_i$ is the malicious host.
The flows of processing of a Route-Agent and a Search-Agent are shown in List 3 and List 4.

List1. Processing of a Task-Agent

1.1 arrives at $Host_i$.
1.2 sends two messages "$prev_i$" and "$next_i$"
to a Route-Agent through $Host_i$.
(At this time, $Host_i$ signs the two messages.)
1.3 performs tasks.
1.4 migrates to $Host_{i+1}$.
1.5 if can't migrate to $Host_{i+1}$,
while (migration=failure){
1.5.1 changes $Host_{i+1}$ to $Host'_{i+1}$
1.5.2 sends two messages "$prev_i$" and "$next'_i$" to the Route-Agent through $Host_i$.
(At this time, $Host_i$ signs the two messages.)
1.5.3 migrates to $Host'_{i+1}$.
}

List2. Processing of $\mathbf{Host_i}$

2.1 receives a Task-Agent and
$Sign_{i-1}(id(Host_{i-1}), id(Host_i))$ from $Host_{i-1}$,
2.2 if the signature is true
sends the message "$Sign_i(id(Task\text{-}Agent))$"
to $Host_{i-1}$.
2.3 executes the Task-Agent.
2.4 if the Task-Agent sends two messages "$prev_i$"
and "$next'_i$" to the Route-Agent through $Host_i$,
sends $Sign_i(prev_i, next_i)$
and $Sign_{i-1}(id(Host_{i-1}), id(Host_i))$.
2.5 if the Task-Agent migrates to $Host_{i+1}$,
$Host_i$ creates a clone of the Task-Agent.
$Host_i$ sends $Sign_i(id(Host_i), id(Host_{i+1}))$ and
one of two Task-Agents to $Host_{i+1}$.
2.6 if receives "$Sign_{i+1}(id(Task\text{-}Agent))$",
deletes the remain of the Task-Agents.
else sets "migration=failure".
2.7 if a Search-Agent needs received signature,
passes "$Sign_{i+1}(id(Task\text{-}Agent))$"
to the Search-Agent.

List3. Processing of a Route-Agent

3.1 receives signatures "$Sign_i(prev_i, next_i)$"
and "$Sign_{i-1}(id(Host_{i-1}), id(Host_i))$"
from a Task-Agent through $Host_i$.
3.2 verifies the signetures.
3.3 if the verification is falth,

processing are stoped.

3.4 if $(id(Host_i) = next_{i-1} \bigwedge prev_i = id(Host_{i-1}))$, appends $id(Host_i)$ to the stored route.

3.5 measures communication delay "D" with $Host_i$.

3.6 sets "$NEXTTIME = func(D) + TaskTime$"

3.7 if doesn't receive next signatures within NEXTTIME from $next_i$, creates a Serch-Agent.

3.8 received $Sign_{i+1}(id(Task\text{-}Agent))$ from the Serch-Agent.

3.9 if ($Sign_{i+1}(id(Task\text{-}Agent))$=null), the malicious host is $Host_i$.

List4. Processing of a Search-Agent

4.1 gets $id(Host_i)$ and $next_i$ from the Route-Agent who created the Search-Agent.

4.2 migrates to $Host_i$.

4.3 gets $Sign_{i+1}(id(Task\text{-}Agent))$ from $Host_i$.

4.4 sends $Sign_{i+1}(id(Task\text{-}Agent))$ to $Trusted\text{-}Host$ where the Route-Agent performed.

4. DISCUSSION

In this section, we discuss the security against attacks from malicious hosts.

4.1. The case where a sender host has malice

Although a malicious $Host_i$ doesn't send a Task-Agent to $Host_{i+1}$, it is assumed that $Host_i$ claims that he sends the Task-Agent to $Host_{i+1}$. In this case, A Route-Agent can't receive messages $Sign_i(prev_i, next_i)$ and $Sign_{i-1}(id(Host_{i-1}), id(Host_i))$ from the Task-Agent through $Host_{i+1}$. Thus the Route-Agent creates a Search-Agent and sends the Search-Agent to $Host_i$. Although the Search-Agent is going to get a signature "$Sign_{i+1}(id(Task\text{-}Agent))$", $Host_i$ doesn't have the signature. More over $Host_i$ can't also create the signature. As a result, it becomes clear that $Host_i$ did injustice.

4.2. The case where a receiver host has malice

It is assumed that a malicious $Host_{i+1}$ incriminate a honest $Host_i$. $Host_{i+1}$ might claim to have received Task-Agent from $Host'_i$ In this case, $Host_{i+1}$ must send "$Sign'_i(id(Host'_i), id(Host_{i+1}))$". But $Host_{i+1}$ doesn't have the private key of $Host'_i$. Thus $Host_{i+1}$ can't create the signature. Therefore $Host_{i+1}$ can't incriminate a honest $Host_i$.

On the other hand, $Host_{i+1}$ may terminate the Task-Agent. First, it is assumed that the Task-Agent is terminated without $Host_{i+1}$ sends a notice of reception "$Sign_{i+1}(received)$" to $Host_i$. However, $Host_i$ stores the clone of the Task-Agent. Thus the Task-Agent can migrate to another host from $Host_i$ again. Next, it is assumed that the Task-Agent is terminated after $Host_{i+1}$ sends a notice of reception "$Sign_{i+1}(id(Task\text{-}Agent))$" to $Host_i$. In this case, since A Route-Agent can't receive messages $Sign_i(prev_i, next_i)$ and $Sign_{i-1}(id(Host_{i-1}), id(Host_i))$ from the Task-Agent through $Host_{i+1}$, the Route-Agent creates a Search-Agent and sends the Search-Agent to $Host_i$. Then, since the Search-Agent can get a reception signature "$Sign_{i+1}(id(Task\text{-}Agent))$", it becomes clear that $Host_{i+1}$ has malice.

5. CONCLUSIONS

In our proposal, a Task-Agent performs only tasks given by an Agent-Owner. And a Route-Agent and a Search-Agent are performed independently of the Task-Agent. By using three kinds of agents, recording of a Task-Agent's route and execution of the Task-Agent's task are processed independently. Thus, the recording of the Task-Agent's route can be performed without affecting the execution of the Task-Agent. Moreover, only when a problem arises in a Task-Agent, a Search-Agent is created. Thus, when satisfactory, the increase in useless network load can be prevented as possible.

The following model is assumed when using our proposed system. 1) An Agent-Owner uses mobile PC and wireless LAN. 2) Hosts that perform an agent are connected in static network.

For example, first, an Agent-Owner transmits a Task-Agent, a Route-Agent and a Search-Agent to a trusted host through wireless LAN. Then, the trusted host transmits the Task-Agent to the first host. Moreover, the Route-Agent and the Search-Agent are performed by the trusted host. Then those agents supervise a migration route of the Task-Agent instead of the Agent-Owner. Thus, once the Agent-Owner transmits agents, it is not necessary to continue connecting to the network until he receives an agent's result.

It is necessary to perform quantitative evaluation of the execution efficiency by implementation of our proposed system etc. as future work.

6. REFERENCES

[1] *Cryptographically Protected Objects*, EcolePolytechnique Fèdèrale de Lausanne, Switzerland, 1997.

[2] *Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts*, Springer LNCS 1419, pp. 92-113, 1998.

[3] *Protecting Mobile Agents Against Malicious Hosts*, Springer LNCS 1419, pp 44-60, 1998.

[4] *Cryptographic Traces for Mobile Agents*, Springer LNCS 1419, pp 137-158, 1998.

[5] *Secure Recording of Itineraries through Co-operating Agents*, 4th Workshop on Mobile Object Systems, 1998.