Protecting Mobile Agent against Malicious Hosts Using A Time-Limited Blackbox Approach

Xiaodong yang
Department of Computer Science
The university of Auckland
Xyan011@ec.auckland.ac.nz
Student ID: 2553127

Abstract

A mobile agent is autonomous software that can achieve a sequence of tasks automatically, such as a ticket booking system. The benefit of using it is to increase the network performance and utilization. As the Mobile Agent performs an important role in the e-commence world now days, the security problems rise rapidly: protecting mobile agent against malicious hosts; protecting mobile agent against others agent; protecting the hosts against malicious agent; protecting hosts against malicious agent; and protecting hosts against untrustworthy third party. This paper will focus on protecting mobile agent against malicious hosts, address the possible threats and describe the way to implement the solution. The approach to solve the problem is using a time- limited blackbox. A Time-limited blackbox is a new agent generated from the original agent. And it will exist in a range of time and does the same job as the original agent did. The idea is during the time the time-limited blackbox exist, it is nearly impossible for the attacker to analyses and modifies it, so the new agent is just like a blackbox for the outside world. To achieve blackbox approach, obfuscation technique is used. Obfuscation mess up the program, so that the program hard to comprehend.

1. Introduction

A Mobile agent is a program that can be distributed to hosts,and achieves some goals. It has become a favored technique in the business world, because it can increase network performance and utilization; finish tasks independently; and satisfy the customer's requirement. As a result of this usage, security problems are an important issue. The most difficult problem is how to protect the mobile agent against the malicious host. It's Generally believed that it is impossible for protecting the mobile agent against the malicious host. The reason for this is that the mobile agent will run in the remote host, uses the resources of the remote host. Therefore all its actions are under control by the remote host. This paper will exam one of the approaches that trys to stop the malicious host from attacking mobile agents. The approach uses a blackbox idea and adds a time-limited property.

The rest of the paper is constructed as following sections: section 2 will talk about the background of the mobile agent and mobile agent system, the structure of mobile agent, and how does it work? Also introduce some existing approaches. Section 3 will talk about malicious host, and describe an attack model of malicious host. Section 4 will illuminate all the possible attacks issued by malicious host to the mobile agent. Section 5 will represent the time-limited blackbox approach, and explain how the obfuscation algorithm works. Section 6 will have a look if the scheme satisfies the requirements of security and if the scheme can stop the possible attacks. Section 7 is some comment. Section 8 is conclusion.

2. Background

Since the first forms of distributed computing appeared in the late 1980's, there exist three major technologies: message passing system (MPS), remote procedure call (RPC) and distributed object system (DOS) [NE]. The mobile agent is a new technique, and it has three advantages over the previous three technologies: client customization, migration and autonomy. Under the approaches RPC and DOS, such as OMG CORBA, Microsoft DCOM, the program in the client side can access the remote functions or objects through interfaces. However all the functions and objects are pre-defined, so the client program only can do limited things with these interfaces. After the mobile agent was born, the situation was changed. Because of the three advantages, agents are more feasible than the three mentioned techniques. The agents can use the resources on the server, such as library code. Also the client can customize functions in the agents so that agents do the job that the customer wants them to do.

A Mobile agent system provides an environment for running mobile agents. The mobile agents only can travel the host running mobile agent system. First a mobile agent lives in home host, and then the mobile agent migrates from one host to another host to achieve a sequence of tasks. After the mobile agent finished its tasks, it will back to home host. E.g. There is a mobile agent in the home host, it need to go through each New Zealand banks' server and find out which bank has a best

NZD-USD exchanged rate, then it has to choose the bank with the best exchanged rate, and does the currency exchange if the exchanged rate are acceptable for customer, finally it brings the currency home. All the jobs have done by agent own without interact with home host. (See Fig 1.)

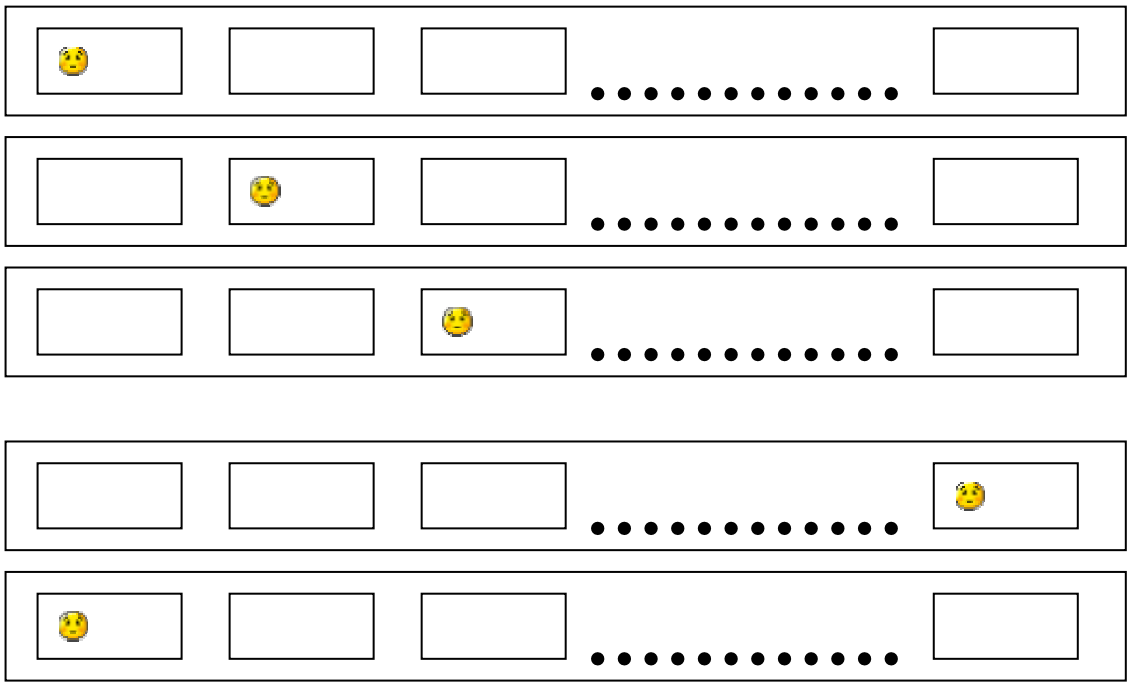

Fig. 1

Mobile agents consist of three sections: code, data and state [HO98]. There are three forms for mobile agents: source code, binary code, and intermediate code [NG00]. Each of the forms has its own characteristic. The mobile agents distributed in source code are easy understood, relative small size and compatible; and the mobile agents distributed in binary code or intermediate code is faster for execution.

As the mobile agents execute on the remote host, there are few aspects of security problems involved. This paper will focus on one of the aspects: protecting mobile agent against malicious host. Currently there exist lots of approaches trying to solve this problem: 1) using hardware tamperproofing; 2) all mobile agents only can run in the trust remote host domain; 3) use encrypted mobile agent [SA98]; 4) use time-limited blackbox. For the first approach, we need the remote host to do extra work base on hardware token, this introduces delay problem. However, if agents run in the remote host, they do need a software tamperproofing technique rather than a hardware tamper resistance to protect them. For the second one, it is not a good approach for the open web; the areas are limited for executing agents. Also there is

not clear for what the “trust” means is and what happens if the “trust” host has done something silly on the agents. For the third one, somehow, it is a kind of blackbox, it trys to achieve running encrypted program. Nevertheless it can only apply for the rational function and polynomial function. If the cryptography approach could apply for every function, it would be a good solution. Unfortunately the mobile agents are executable programs. it’s not easy to run the encrypted mobile agents.
When mobile agents travel the network, the security is a pain. To improve security of mobile agents, first things to do are to identify the requirements for the security. In [NG00], there are five goals to be achieved: confidentiality, integrity, accountability, availability, and anonymity. In the late section of this paper, the time-limited blackbox scheme will be exam with these five requirements.

3. Malicious host

The definition of the malicious host in [HO98] is **“**a party that is able execute an agent that belongs to another party and that tries to attack that agent in some way.**”** As the mobile agents are executed in the host, the host allocates all the resources, so the host can launch lots of different kinds of actions to attack the agents. How does the malicious host attack the agent? In [HO98a], Hohl introduced a model of attacks of malicious host. The model use abstract machine RASPS (Random Access Stored Program plus Stack). A RASPS contains four components, a set of memory; a set of stack; a program counter; and a stack pointer. The mobile agent program is stored in the memory of RASPS; values are preserved in the stack of RASPS. Program counter points to the memory of RASPS , and stack pointer points to stack of RASPS. (see Fig.2)

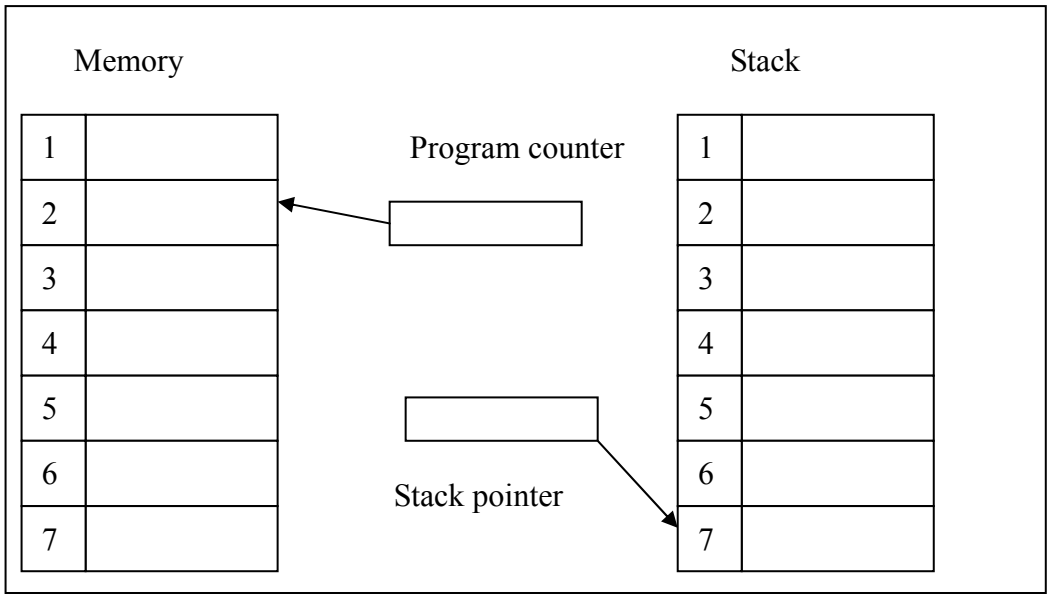

Fig.2 RASPS: Random Access Stored program plus Stack

When a new agent arrives at the host, it will be loaded into RASPS. The agent use

system call to communicate with the runtime environment or agent partner. Before an agent is loaded into agent RASPS, the attack program wrote by the attacker has already loaded into attack RASPS. Therefore, there are two RASPS machines in the host, one is agent RASPS, the other is attack RASPS. And the agent cannot directly communicate with environment and agent partner anymore. It has to communicate with them through the attack RASPS in this attack model. All the system calls and statements that the agent launched can be eavesdropped by the attack program. The attack program can modify the statement in the agent RASPS memory and the value in the agent RASPS stack; can store that agent program's statements in its own RASPS stack; can store that agent program's values in its own RASPS stack; can access the agent RASPS memory and stack. In this attack model, the attack program is so powerful, so that each statement executed by agent is under control by the attack program.

In [HO98a], there are seven steps that the attacker RASPS will do before a new statement of the agent is executed:

1. Fetch and decode the next statement of the agent RASPS
2. Store the next statement and parameters of next statement on the stack of the attacker RASPS
3. Compute the future program counter of the agent RASPS and store this value on the stack of the attacker RASPS
4. Analyses the statement of the agent program and execute attack program in attacker RASPS memory
5. Execute the statement with stored parameters
6. Preserve future program counter stored on the stack of the attacker RASPS into the program counter of the agent RASPS

Understanding this attack model is important and foundational, it will show the concept that the malicious host attack mobile agent, and help to know how the malicious host attack the mobile agent. Comprehending the concept and theory of the attack model is the start point of developing the protection scheme for protecting mobile agents.

4. Attacks

The attacker RASPS can generalize variant of attacks. Follow the concept of security breaches that Pfleeger given, the possible attacks show below:

- Interruption
  - Incorrect execution of code

    Without modifying the code, the host applies another way to execute the code.
  - Denial of execution

    The malicious host gives an unacceptable delay for executing the program of the agent.

- Interception
  - Eavesdrop the code of mobile agent
    Attacker can read the code of mobile agent, e.g. statements.
  - Eavesdrop the data of mobile agent
    Attacker can read the data of mobile agent, such as variables.
  - Eavesdrop the control flow of mobile agent
    Attacker can read the control flow.
  - Eavesdrop the interaction with other agents
    Attacker can read the information transmitted between agents.
- Modification
  - Modify the code of mobile agent
    Attacker knows the location of the code; can modify the code of mobile agent.
  - Modify the data of mobile agent
    Attacker knows the location of the data; can modify the data of mobile agent.
  - Modify the control flow of mobile agent
    Attacker knows the location of the control flow; can modify the control flow of mobile agent
  - Modify the interaction with other agents
    Attacker can modify the information transmitted between agents
  - Sabotage
    Attacker can modify a single bit or few bits of the agent code, cause the damage.
- Fabricate
  - Masquerading of the host
    The malicious host pretends itself as another host
  - Returning wrong results of system calls issued by the agent
    The malicious host returns a wrong result of the system call to the mobile agent

- Others
  - Blackbox test
    The malicious host enters variant input to the agent, and gains the output data, so that it can analyses the data.

5. Time-limited blackbox protection scheme

Before introducing time-limited blackbox approach used for protecting mobile agent against malicious host, the early stage of this section will describe blackbox approach first.

Imaging there is a black iron box on the table, there is not chance for us to know what is inside the box, and there is also not chance for us to put something into the box. So

the object in the black iron box is safe. It is the idea that blackbox approach uses to protect mobile agent. To achieve the blackbox protection scheme, there are two goals: no one can read the code and data of the mobile agent; no one can modify the code and data of the mobile agent. Any approach can satisfy all these two requirements, it is a kind of blackbox protection scheme. The cryptography approach mentioned in section 2 can meet the requirements, so it is a kind of blackbox approach.

Also, Hohl in [HO98] represent an alternative approach, apply obfuscation algorithm and some parameters and convert the original agent to a new agent. Different new agents are created depend on the different parameters. Here convert means mess up the program. The original program will be messed up, the new agent with the mess-up program is distributed on the network. Well the new agent does the exactly the same things as the original agent does. After apply obfuscation algorithm on the original program, the new program is hard to understand and analyses for the attackers. If the attackers can never understand and analyses the code of mobile agent, the mobile agent is protected by a blackbox. Unfortunately, sooner or later, attackers will break the algorithm, and know the meaning of the code of the mobile agent. So this approach can only protect mobile agents during interval time. (See Fig. 3)

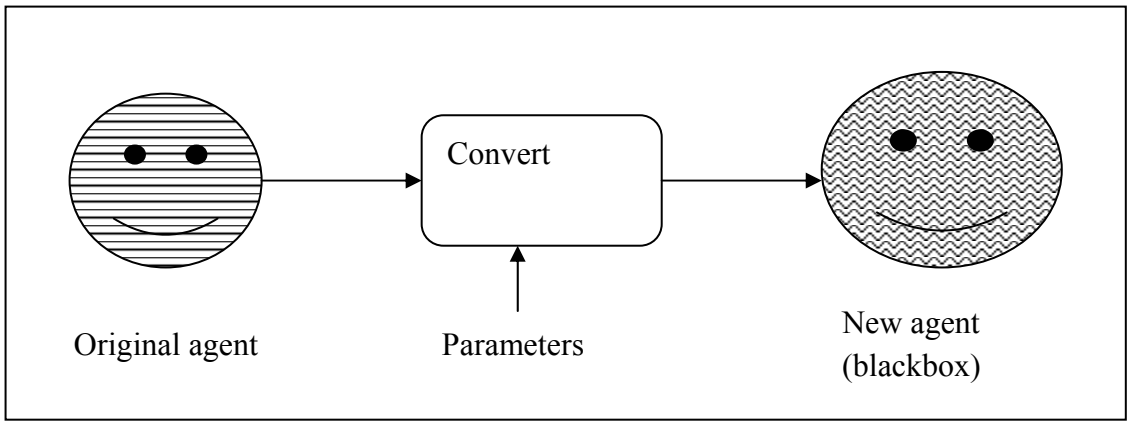

Fig.3

The approach represented before is an incomplete approach. So Hohl in [HO98] developed a new approach base on the approach described. The new agent created by the convert mechanism is added a time property, a new attribute, expired date, is added in the new agent. The idea is, before the expired date, it is not possible for the attacker to comprehend the code in the program. After this expired date, even the attackers can break the code, but the information that the attackers gained is not useful anymore. So the attack will not threat to the mobile agent. The requirements for such time-limited blackbox protection scheme as below:

- In a certain time interval
- No one can read the code and data of the mobile agent

- No one can modify the code and data of the mobile agent
- After expired date, the successful attacks will not have effects

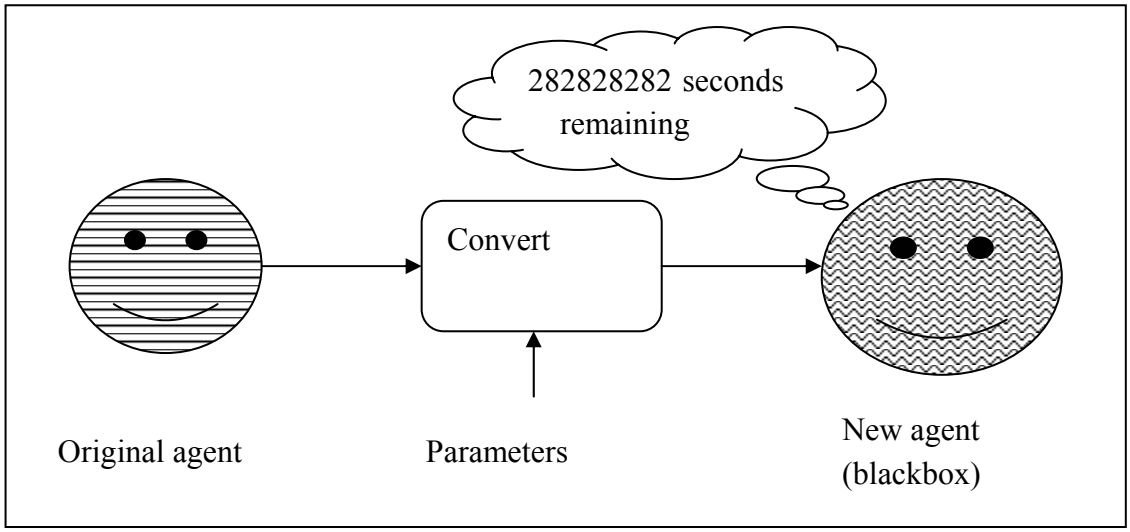

Fig.4

In [HO98], Hohl provides three approaches of obfuscation: (1) variable recomposition; (2) conversion of control flow elements into value-dependent jumps; and (3) deposited keys. In [TH00][TH98][TH97], the authors classify four targets of obfuscation: layout, control, data, and preventive. Each target has several operations, and each operation has several transformations.

Applying the taxonomy of obfuscating transformations in [TH97], the target of approach (1) is data, the operation is aggregation, and transformations involve both split array and merge arrays transformation. (See Fig. 5)

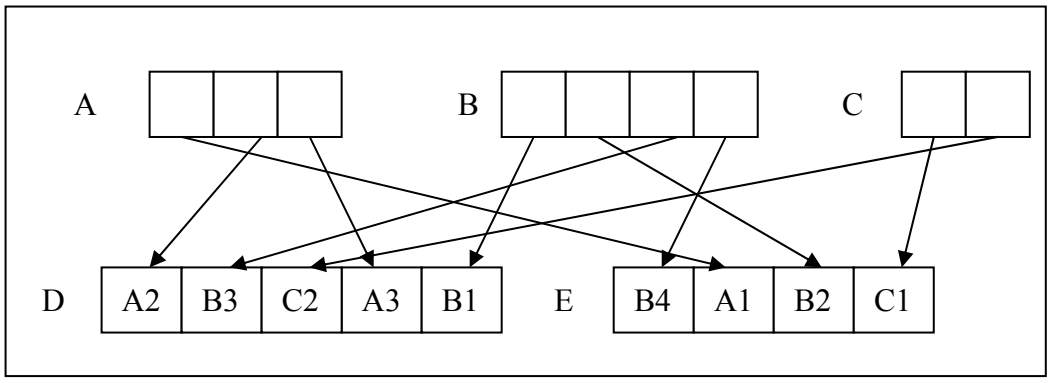

Fig. 5

In Fig.5, A, B, C, D and E are variables. Variable recomposition approach separates each variable into sections, respect to the split array transformation in [TH97]. And recombines them to different variables, respect to the merge arrays transformation in

[TH97]. The purpose of approach (1) is to hide the location of the variables. So that the attacker cannot find out the location of variables.(See Fig. 6)

Public Currency exchange(double A)	
Exchange(A);	Excange(E[1]+D[0]+D[3]);
Original code	Obfuscation code

Fig. 6

It's not easy to identify the second approach provided by Hohl. Respect to the taxonomy in [TH97], the target of approach (2) is control. The tough part is what kind of transformation it belongs to. The idea of this approach is to convert the format of some codes to a difficult understanding format. Example for approach (2) below:

If(a(b)<c){
b=s(d(e)+f);
}
	start=0;
	boolean abc=true;
	while(abc){
	switch(start)
	case0: t1=a(b); start=4; break;
	case1: t4=t3+f; start=3;break;
	case2:{ if (t2){ start=7}else {start=5;}break;}
	case3: b=t4; start=6; break;
	case4: t2=t1<c; start=2; break;
	case5: abc=false;break;
	case6: start=5; break;
	case7: t3=d(e); start=1; break;
	}

Fig. 7

Approach (3) is not belongs to the obfuscation scope. The mobile agent send a requirement to a trust host, ask for some information about which statement should execute next, and the trust host send the answers back to mobile agent. The ideal is similar as the one describe in [MA84].

It doesn't matter which kind of the obfuscation transformation used, all the

transformations defined in [TH97] can be used for achieving time-limited blackbox protection scheme. It is sure that applying more than one transformation are more secure than only applying one transformation.

6. Examine the time-limited blackbox approach

In this section, the time-limited blackbox approach will be examined whether this approach satisfies the security requirements, whether this approach can stop all the possible attacks issued by malicious host

.

✓ Examine the security requirements:

The five security requirements are: confidentiality, integrity, accountability, availability, and anonymity.

- **Confidentiality**:
  The private data in agent or host must keep secret. Only the authorized parties can access them. "The type of access is read-type access" [PL97]. The time-limited blackbox protection scheme doesn't allow malicious host has any chance to comprehend the code of mobile agent. However, the malicious host still can access the memory of agent RASPS, read the code of mobile agent. Imaging if someone stole a top secrete military file, the file is wrote by some cryptography, even the person can read the file, he cannot understand it, so the information in the file are safe. Hence, in some level, the time-limited blackbox protection scheme is satisfied confidentiality of security.
- **Integrity**:
  Prevent the unauthorized modification from modify the date, code, and state of mobile agent. Under the time-limited blackbox protection scheme protecting, attackers cannot understand the code of mobile agent, so they cannot modify the code for specific goals, for example, attackers cannot increase or decrease the foreign currency exchanged rate stored in the agent. But the attackers still can modify the code for just want to damage it, such as Sabotage attack mentioned in section 4. Hence, this protection scheme has low integrity.
- **Availability**:
  Unauthorized parties cannot access and use the data store in the mobile agent. Time-limited blackbox protection scheme has not problem to satisfy this requirement of security.
- **Accountability**:
  The actions of mobile agent and host are accountable. Time-limited blackbox protection scheme can satisfy this requirement of security.
- **Anonymity**:
  Mobile agents hide their owner's identifies for not exposing to public. Time-limited blackbox protection scheme can hide mobile agent owner's identifies.

✓ Examine the attacks:

As mentioned before, attackers can read code, control flow, but they would not understand them, so time-limited blackbox approach can stop attacks such as: read code, and read control flow. Also can stop modify code and control flow for specific purpose. After applying obfuscation transformations, the location of the data is hid, so the attacks such as read data, modify data, and incorrect execute the code are stopped.
However the scheme has nothing to do with the attacks such as modify the interaction information, read the interaction information, masquerade, denial of execution, returning wrong results, Sabotage, and analyses the I/O data.
After the third parties help to verify the result of the system calls, and the identity of the host, mobile agents can avoid the attacks like the malicious host gives wrong results of the system calls and masquerade attack. Mobile agent communicate with other agents or host using secure channel, this will stop the attacks such as read the interaction information and modify the interaction information.
All the information left after expired date of mobile agent are useless, this make the denial of execution attack to stack.
Sabotage is an attack that modifies one or more bit of the agent code for damaging the code. In [HO98], author solves such problem using CRC mechanism.
The method of stopping blackbox test attack is simply adding some dummy code, so that the result will confuse the analyzer.

7. Comment

For the time-limited blackbox protection scheme, the major problem is how long a time interval of the mobile agent should have, if the time interval is too large, the attacker will get a chance to break the code. But if the time interval is too short, mobile agent might not finish the tasks that the home host given. To determine a perfect time interval is hard work.
Hohl in [HO98] use some techniques like CRC mechanism to stop attack such as sabotage. It is sure when the attacker cannot access to the agent code, only can communicate with mobile agent. On the other view, the attack program in RASPS can access the code in the mobile agent RASPS memory. What happen if the attack program directly access to the agent RASPS memory and modify the code inside it? At this situation, CRC can do nothing to it. Can mobile agent use some tamperproofing technology to protect agent code? The definition of tamperproofing in [TH00] is detecting the modification and fails the program when modification detected. In [TA01], author introduces a tamperproofing technique – dynamic self-checking techniques. Basely, it inserts some testers and correctors into the code, check if the code is modified. If it does, use correctors to correct the code. Apply self-checking techniques, mobile agent code can be protected from malicious host

directly access agent RASPS memory and modify the code in memory.
Mobile agent uses secure channel to communicate with other agents or hosts. To achieve this, mobile agent has to bring the private key to encrypt the messages or bring the symmetric key to encrypt the messages. This will raise the risk of disclosing the private key or symmetric key to malicious host. If the key disclose to the malicious host, then the malicious host can use the key communicate to other hosts or other agents and attack them.
Obfuscation is the approach that mix program so that the attacker can not understand the code in short range of time. Unfortunately obfuscation involves human to modify the code manually. Therefore the protected mobile agent cannot generate easily and efficiently.

8. Conclusion

This paper describes time-limited blackbox approach for protecting the mobile agent against malicious host. Also the paper examines the time-limited blackbox approach with security requirements and possible attacks. The approach can satisfy most security requirements and can stop most of attacks. However, the approach can not meet the integrity requirement very well. Also the major problem is how to determine how long should the mobile agent exist. Nevertheless, the time-limited blackbox approach still gives a way to protect mobile agent from attacking by the malicious host.

9. References:

[NE] Nelson Minar. Designing an ecology of distributed agents. http://www.media.mit.edu/ nelson/

[NG00] Ng, Sau-Koon: Protecting Mobile Agents against Malicious Hosts. Master Thesis. Division of Information Engineering, The Chinese University of Hong Kong, June 2000.

[HO98] F. Hohl, “Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts,” *Mobile Agents and Security*, pp. 92-113, vol. 1419, Lecture Notes in Computer Science, Springer Verlag, 1998.

[HO98a] Hohl, F. (1998a). *A model of attacks of malicious hosts against mobile agents*. In Proceedings of the ECOOP Workshop on Distributed Object Security and Jth Workshop on Mobile Object Systems: Secure Internet Mobile Computations, pages 105-120.

[SA98] T. Sander and C.F. Tschudin, “Protecting Mobile Agents against Malicious Hosts,” *Mobile Agents and Security*, 1998.

[TA01] B Horne, L Matheson, C Sheehan, and R Tarjan, “Dynamic Self-Checking Techniques for Improved Tamper Resistance”. In *Workshop on Security and Privacy in Digital Rights Management 2001*. Available: http://www.star-lab.com/sander/spdrm/papers.html, February 2002.

[TH00] C. Collberg and C. Thomborson. Watermarking, tamper-proofing, obfuscation – Tools for software protection. Technical Report 2000-03, University of Arizona, February 2000.

[MA84] Tim Maude and Derwent Maude, “Hardware protection against software piracy,” *Communications of the ACM*, 27(9):950-959, September 1984.

[TH98] C. Collberg, C. Thomborson, and D. Low, “Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs,” Proc. Symp. Principles of Programming Languages (POPL '98), Jan. 1998. http://www.cs.auckland.ac.nz/collberg/Research/Publications/CollbergThomborsonLow98a/.

[TH97] C. Collberg, C. Thomborson, and D. Low, “A Taxonomy of Obfuscating Transformations,” Technical Report 148, Dept. of Computer Science, Univ. of Auckland, July 1997, http://www.cs.auckland.ac.nz/~collberg/Research/Publications/CollbergThomborsonLow97a.

[PL97] C. Pfleeger, "Is there a security problem in computing?", Chapter 1 of Security in Computing, 2nd edition, Prentice Hall, 1997, pp. 1-19.