ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

A Survey on Protection Techniques of Mobile Agents from Malicious Hosts

Somnath Dey[1], Prof. Dr. Devadatta Sinha[2]

Assistant Professor, Department of Computer Science &Engineering, Sabita Devi Education Trust – Brainware Group of Institutions, Kolkata,India[1]

Professor, Department of Computer Science &Engineering, University of Calcutta, Kolkata, India[2]

ABSTRACT: Mobile agent technology has become a new paradigm of distributed computing that can replace the conventional client-server model. Despite its many practical benefits, it results in significant new security threats from malicious agents and from malicious hosts. Many techniques for the first have been developed. The second problem seems to be much harder. We will focus on the second one. We found that there is not a single, comprehensive solution that provides complete protection of agents against malicious hosts. Existing solutions either only detect or to some extent prevent attacks on agents. This paper gives a comparative survey on of the main security issues related to the mobile agent paradigm. These issues include security threats, security requirements and techniques for keeping the mobile agent secure against malicious hosts.

KEYWORDS: Security; Mobile Agent; MaliciousHost Problem; Masquerading;Denial of Service; Eavesdropping; Alteration

I. INTRODUCTION

Duringthe last few years we have seen several fundamental changes in distributed and client-server computing environment. This is due to the appearance of mobile agents. Mobile agent technology is getting popular as means for an efficient way to access to remote resources on computer networks. Mobile agents are processes that migrate from one node to other in the network autonomously to achieve user’s requests. 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 leaving the previous host [1].

However, one of the main technical obstacles to a wider acceptance of the mobile agent is security. Sander and Tschudin [2] present two types of security problems that need to be solved. The first is host protection against malicious agents. The second is agent protection against malicious hosts. Many techniques have been developed for the first kind of problem, such as access control, password protections, sand boxes etc. But the second problem appears to be difficult to solve. The fact is that computers have complete control over all the executing programs. As a consequence, it becomes very difficult to protect mobile agents from malicious hosts.

The rest of the paper is organized as follows: Section 2 deals with malicious host problem, Section 3 gives an overview of the main solutions for keeping a mobile agent secure against malicious hosts, Section 4 gives the comparative analysis of different solutions and finally section 5 gives some concluding remarks.

II. MALICIOUS HOST PROBLEMS

In a mobile agent system a malicious host is defined as a host that executes an agent and tries to attack the agent in some way. When an agent is executed on a host it must use the resources available on that host. The host can monitor an agent’s memory usage and each instruction given by the agent to the host. A malicious host may then attempt to attack an agent in a number of ways.

The four main forms of attacks by malicious hosts on mobile agents are:

- A host masquerading as another host.
- Denial of service by the host to the agent.
- Eavesdropping on an agent's activity.
- Alteration of the agent by the host.

A. *Masquerading:*

A masquerading host may try to trick the agent into believing it is another host and cause the agent to give the host sensitive information. Once the masquerading host is able to gain the trust of the agent, it may then be able to read or modify any of agent's code, data and state if mechanisms are not put in place to protect this type of attack. The main solution to prevent this type of attack is to use a strong authentication protocol to authenticate a host to an agent.

B. *Denial of Service:*

A host may deny an agent a specific service provided by the host. It is possible for a host to both intentionally and unintentionally deny an agent a service. A host may deny an agent service so that the agent is not able to complete its task. Another possible attack is the host could terminate the agent altogether. Furthermore, a host may deny a request from an agent on a time-sensitive task so that the agent is unable to complete its task in its allotted time.

C. *Eavesdropping:*

The next attack that can be performed by the host on an agent is eavesdropping. In a client/server environment, the typical eavesdropping attack comes from the monitoring of a communication channel. In the case of a mobile agent system, malicious hosts may try to determine the code, data or flow control held by the agent. This form of attack is difficult to prevent and detect.

Even when all of the information is hidden from the host, the host may still be able to infer some information from the agent. The main problem is that the agent must execute on the host so the host is able to record each instruction given to it by the agent.

D. *Alteration:*

The final form of attack by a host on an agent is the alteration of the agent. The host can alter an agent by changing the data, code and control flow. A malicious host may try to change the code of an agent so that the agent performs other tasks than were intended by its creator. A host may also try to change the data contained in the agent.

III. **TECHNIQUES FOR MOBILE AGENT PROTECTION**

The approaches to protect a mobile agent can be broadly classified into two:
(i) Detection mechanisms to detect unauthorized modification of code, state etc.
(ii) Prevention mechanisms for making it impossible to access or modify code, state or data.

A. *Trusted Hardware:*

One solution is to let a trusted third party supplied trusted hardware, in the form of tamper resistant devices, which are placed at the site of the host and can interact with the agent platform [3]. A tamper resistant device can be in the form of a smartcard. Such trusted hardware can then either protect the complete execution environment of the agent or perform certain security tasks.

The major drawback of trusted hardware is the cost of such a solution.

B. *Mutual Itinerary Recording:*

Mutual Itinerary Recording is a general scheme for allowing an agent's itinerary to be recorded and tracked by another cooperating agent and vice versa [4] in a mutually supportive arrangement. When moving between agent platforms, an agent conveys the last platform, current platform, and next platform information to the cooperating peer through an authenticated channel. The peer maintains a record of the itinerary and takes appropriate action when inconsistencies are detected. Attention is paid so that an agent avoids platforms already visited by its peer. The

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

rationale behind this scheme is founded on the assumption that only a few agent platforms are malicious, and even if an agent encounters one, the platform is not likely to collaborate with another malicious platform being visited by the peer. Therefore, by dividing up the operations of the application between two agents, certain malicious behavior of an agent platform can be detected. The scheme can be generalized to more than two cooperating agents. For some applications it is also possible for one of the agents to remain static at the home platform. Since the path records are maintained at the agent level, this technique can be incorporated to any appropriate application.

Drawbacks of this technique include the cost of setting up the authenticated channel and the inability of the peer to determine which of the two platforms is responsible if the agent is killed.

C. *Itinerary Recording with Replication and Voting:*

A faulty agent platform can behave similar to a malicious one. Therefore, applying fault tolerant capabilities to this environment should help countering the effects of malicious platforms. One such technique for ensuring that a mobile agent arrives safely at its destination is through the use of replication and voting [5]. The idea is that, rather than a single copy of an agent performing a computation, multiple copies of the agent are used. Although a malicious platform may corrupt a few copies of the agent, enough replicates avoid the encounter to successfully complete the computation.

This approach is suitable where agents can be duplicated without problems and survivability is the major concern. The drawbacks are the additional resources consumed by replicated agents and message complexity increased.

D. *Computing with Encrypted Functions:*

The goal of Computing with Encrypted Functions [2] 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. Theapproach is to have the agent platform to 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 that 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 in this technique is directly proportional to the strength of encryption function. It is best suitable technique for application which requires high security.

The drawback is that this technique does not prevent denial of service, replay, experimental extraction and other forms of attack against the agent.

E. *Partial Result Encapsulation:*

Partial Result Encapsulation (PRE) is a detection technique that aims to discover any possible security breach 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.

The PRE technique has different implementations. In certain scenarios, the agent itself performs the encapsulation, while in others the platform does it. To meet essentially security requirements such as integrity, accountability and privacy of the agent, PRE makes use of different cryptographic primitives such as encryption, digital signatures, authentication codes and hash functions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

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 [6]. Sliding Encryption encrypts small amounts of data within a larger block and thus obtains small pieces of cipher text. It is particularly suitable for certain application where storage space is valuable such as smartcards [7].

Yee [8] 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. In order to produce the PRAC, the agent uses the associated secret key for the current platform to compute a Message Authentication Code (MAC), which is encapsulated together with the message to produce PRAC. 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 [8] 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. Only the agent's originator has a copy of all used secret keys and thus can verify the encapsulated results. The result verification enables the originator to detect any modification (tampering) of the agent's results. Yee [8] suggested that the results could also be encrypted using the originator's public key, in order to guarantee both privacy and integrity.

Karjoth et al [9] proposed a "strong forward integrity" which 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. Then the visited platform uses its private key to digitally sign the encrypted results together with a hash chain. The hash chain links the results from the previous platform with the identity of the next platform to be visited. This prevents the platform from changing its results later and thus ensures strong forward integrity [9].

The PRAC technique has a number of limitations. The most serious of them 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. Since the PRAC is oriented toward integrity and not confidentiality, the accumulated set of partial results can also be viewed by any platform visited, although this is easily resolved by applying sliding key or other forms of encryption.

F. *Environmental Key Generation:*

Environmental Key Generation [10] 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 (for example 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. The procedure is somewhat akin to the way in which passwords are maintained in modern operating systems and used to determine whether login attempts are valid.

One weakness of this approach is that a platform that completely controls the agent could simply modify the agent to print out the executable code upon receipt of the trigger, instead of executing it. Another drawback is that an agent platform typically limits the capability of an agent to execute code created dynamically, since it is considered an unsafe operation. An author of an agent can apply the technique in conjunction with other protection mechanisms for specific applications on appropriate platforms.

G. *Execution Tracing:*

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

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 nonrepudiatable 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 is omitted. The technique also defines a secure protocol to convey agents and associated security related information among the various parties involved, which may include a trusted third party to retain the sequence of trace summaries for the agent's entire itinerary. If any suspicious results occur, the appropriate traces and trace summaries can be obtained and verified and a malicious host identified.

This technique 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. Other problems identified include the lack of accommodating multi-threaded agents and dynamic optimization techniques. While the goal of the technique is to protect an agent, the technique does afford some protection for the agent platform, providing that the platform can also obtain the relevant trace summaries and traces from the various parties involved.

H. *Obfuscated Code (Time Limited Black Box):*

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 [12][13]. 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, etc. [12]. 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 [14] - [17]. Layout obfuscation tries to remove or modify some information in the code, such as comments and debugging information, without affecting the executable part of the code. 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 [12] 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 obfuscation is impossible. In addition to protecting a mobile agent, obfuscation can also be used for other applications such as protecting digital watermarking, enforcement of software licensing and protecting protocols from spoofing [12][14]. As far as the performance is concerned, some obfuscation techniques reduce the size of the code and thus speed up its execution, while others achieve the opposite (control obfuscation) [15]. Obfuscation is considered resistant to impersonation and denial of service attacks [14].

This technique is flexible and inexpensive. Depending on the need for security an application can be obfuscated accordingly. In this technique there is a possibility to create different instances of one software application to battle global attacks. It has low maintenance cost due to automation of the transformation process and compatibility with systems. This technique is platform independent. This technique has number of drawbacks such as every transformation introduce extra cost in memory and computation time necessary to execute the obfuscate program. Obfuscation does not provide waterproof security. Most of these code transformations are not one way and it is hard to decide where to use which transformations.

IV. COMPARATIVE ANALYSIS

The existing approaches for protecting mobile agents from malicious hosts use different mechanisms either prevention or detection with different objectives. Here objectives are the network security requirements. All solutions

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

have a number of limitations as no solutions fulfil all the objectives. The following table summarizes the comparative analysis of approaches of securing mobile agents.

TABLE I: COMPARATIVE ANALYSIS OF SECURITY SOLUTIONS TO PROTECT MOBILE AGENTS

Proposed Mechanisms	**Parameters**						
	Prevent Masquerading	***Prevent Eavesdropping***	***Prevent Unauthorized Access of Information***	***Prevent Denial of Service***	***Prevent Copy & Replay***	***Detect Tampering***	***Implementation Available***
Trusted Hardware	Yes	No	Yes	Yes	Yes	Yes	Yes
Mutual Itinerary Recording	No	No	No	No	No	Yes	Yes
Itinerary Recording with Replication & Voting	No	No	No	No	No	Yes (fault tolerance)	Yes
Computing with Encrypted Functions	No	Yes	Partial	No	No	Partial	No
Partial Result Encapsulation	Yes (Digital Signatures)	Yes (Encryption)	Partial	Yes (Encryption)	Yes (Digital Signatures)	Partial	Yes
Partial Result Authentication Codes (PRAC)	No	No	Partial	No	No	Partial	Yes
Environmental Key Generation	Yes	Yes	Partial	No	No	Partial	No
Execution Tracing	Yes	No	No	Yes	Yes	Yes	No
Obfuscated Code (Time Limited Black Box)	No	No	Yes (Time Limited)	No	No	Yes (Time Limited)	Yes

V. CONCLUSION

Conventional network management approach is based on client server model, suffers from problems like network delays, lack of scalability, information bottleneck and excessive processing load at manager and heavy usage of network bandwidth. The mobile agent technology gives improvements in terms of network bandwidth utilization, significant reduction of network load etc.

This paper gives an overview about the security techniques of mobile agent against attack from malicious hosts. Some of those techniques are still at the theoretical level and are not yet widely used in practice. None of the existing techniques provides an optimal solution for all scenarios. However, a combination of various techniques may yield powerful solutions.

The protection of mobile code against a malicious host is still an open research topic. It can be observed that non-cryptographic techniques are generally not sufficient to protect a mobile code strongly. But on the other hand

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

cryptographic techniques are slow and computational expensive. So we need to select a balanced method for much secure and high speed ended application.

REFERENCES

1. Lee, H.,Alves-Foss, J., and Harrison,S.,"The Use of Encrypted Functions for Mobile Agent Security", Proceedings of the 37th Hawaii International Conference on System Sciences, January 5-8, 2004.
2. Sander, T., and Tschudin, C.,"Protecting Mobile Agents against Malicious Hosts", in G. Vigna, editor, Mobile Agents and Security, LNCS, Springer-Verlag, Heidelberg, Germany, Vol.1419, pp.44-60,1998.
3. Wilhelm, U. G.,Staamann, S., and Buttyan L.,"Introducing Trusted Third Parties to the Mobile Agent Paradigm", in J. Vitek and C. Jensen, editors, Secure Internet Programming, LNCS, Springer-Verlag, New York, USA, Vol.1603, pp.471-491, 1999.
4. Roth, V.,"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, INRIA, France, pp.147-154,1998.
5. Schneider, F. B.,"Towards Fault-Tolerant and Secure Agentry", Proceedings of the 11th International Workshop on Distributed Algorithms, Saarbucken, Germany, September, 1997.
6. Young, A., and Yung, M.,"Sliding Encryption: A Cryptographic Tool for Mobile Agents", Proceedings of the 4th International Workshop on Fast Software Encryption, FSE'97, January, 1997.
7. Karjoth, G., and Posegga, J.,"Mobile Agents and Telcos' Nightmares", in Annales des Telecommunications, Vol.55, no.7/8, pp.29-41, 2000.
8. Yee, B.,"A Sanctuary for Mobile Agents", DARPA Workshop on Foundations for Secure Mobile Code, February, 1997.
9. Karjoth, G.,Asokan, N., and Glc, C.,"Protecting the Computation Results of Free-Roaming Agents", Second International Workshop on Mobile Agents, Stuttgart, Germany, September, 1998.
10. Riordan, J., and Schneier, B.,"Environmental Key Generation Towards Clueless Agents", in G. Vinga, editor, Mobile Agents and Security, LNCS, Springer-Verlag, Vol.1419, 1998.
11. Vigna, G.,"Protecting Mobile Agents through Tracing", Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, Jyvälskylä, Finland, June, 1997.
12. D'Anna, L., Matt, B., Reisse, A.,Vleck, T. V., Schwab, S., and LeBlanc, P.,"Self-Protecting Mobile Agents Obfuscation Report", Network Associates Laboratories, June, 2003.
13. Jansen,Wayne A.,"Countermeasures for Mobile Agent Security", Computer Communication, Special issue on Advances in Research and Application of Network Security, November, 2000.
14. Wroblewski, G.,"General Method of Program Code Obfuscation", PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002.
15. Hachez, G.,"A Comparative Study of Software Protection Tools Suited for Ecommerce with Contributions to Software Watermarking and Smart Cards", UniversiteCatholique de Louvain, 2003.
16. Collberg, C.,Thomborson, C.,and Low, D.,"A Taxonomy of Obfuscating Transformations", Technical Report 148, Department of Computer Science, University of Auckland, July, 1997.
17. Armoogum, S., and Caully, A.,"Obfuscation Techniques for Mobile Agent Code Confidentiality", March, 2010.
18. Hohl, F.,"Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts", G. Vigna, editor, Mobile Agents and Security,LNCS,Springer-Verlag,Vol.1419, pp.92-113, 1998.
19. Barak, B.,Goldreich, O.,Impagliazzo, R.,Rudich, S.,Sahai, A.,Vadhan, S., and Yang, K.,"On the (Im)possibility of Obfuscating Programs", in Advances in Cryptology, Proceedings of Crypto'2001,LNCS, Springer-Verlag,Vol.2139, pp.1-18,2001.

BIOGRAPHY

Somnath Dey is an Assistant Professor, Department of Computer Science & Engineering in Sabita Devi Education Trust – Brainware Group of Institutions. He received his M.Techin Computer Science &Engineering from University of Calcutta, Kolkata, India. His research interests are Cryptography, Distributed Systems, Mobile Computing, Mobile Agents etc.

Prof. (Dr.) Devadatta Sinha is a Professor, Department of Computer Science & Engineering, University of Calcutta, Kolkata, India. He received his PhD in the area of Computer Science from Jadavpur University. His research interests are Parallel Computing, Software Engineering, Distributed Systems etc.