

A Distributed and Compromise-tolerant Mobile Agent Protection Scheme

Oguz Kaan Onbilger, Richard Newman, Randy Chow
Dept. of Computer and Information Science and Engineering, University of Florida,
Gainesville, FL 32611, USA
E-mail: {onbilger, nemo, chow}@cise.ufl.edu

Abstract

In this extended abstract paper we address the problem of protecting mobile agents from possibly malicious hosts they could visit in an open e-commerce environment, the Internet. The novel approach presented is to split up a task that otherwise could be employed by a single agent into a group of agents, which communicate and cooperate by executing on different hosts. The approach has its roots in what is known as Information Dispersal for security to provide fault- and compromise-tolerance. An open distributed support environment to realize the approach is also briefly presented.

1 Introduction

Mobile agents (MA) are an approach to distributed computing employing the mobile code concept. A mobile agent is an autonomous entity, which is composed of code, data and state information. They visit hosts (e.g., servers) possibly using an itinerary, do some execution on those hosts using their codes and migrate by carrying their state information from host to host. As in the case with stationary agents, they act on behalf of their owners (e.g., senders). They are autonomous in the sense that, they have all the knowledge to perform the assigned task on behalf of their owners. There are several application areas of MA systems (e.g., e-commerce, network management, etc.).

However, there are two important limitations of MA paradigm: security and interoperability among different MA systems. There are two types of security threats introduced by mobile code systems. One comes from potential malicious hosts and the other from malicious agents. The malicious agents problem is rather old and many protection mechanisms have already been proposed and implemented. On the other hand, the malicious hosts problem is more difficult and considered unsolvable without dedicated hardware. The problem is relatively new since the computation has never been defined to take place in a remote environment that could possibly be malicious. We are concerned with the latter problem: protecting the MAs from malicious hosts.

In this paper, we present a fault- and compromise-tolerant distributed MA security scheme for an open e-commerce environment, the Internet. An open support environment to realize the approach is also briefly described.

2 Related Work

There are several approaches proposed in the literature. Although we do not give a precise taxonomy, a rough classification and brief description is presented below.

Most of the MA systems ignore security by the assumption of organizational or social trust. For example, an individual user may have trust to a reputable well-known company. Therefore he/she may not expect any hostile behavior from the company that operate the MA system, against his/her agent. But this cannot be applied to a virtually unknown company,

and such an e-commerce MA system may not be fair to businesses that have not yet established a public trust. On the other hand, social trust does not apply to a business-to-business e-commerce system based on MAs. For example two competing companies could not assume the same trust to one another as in the individual user case, even if these were reputable companies. This potential aspect of lack of trust could severely limit the use of MAs in an open e-commerce environment.

Hardware support is recognized as the only tractable solution known since no single software solution proposed so far addresses every possible attack. The idea is to store and execute agents in cleartext form only in isolated and protected computers. (See for example, Tamper-proof Environment [1]).

One class of approaches is based on obscurity. They are specifically based on cryptography or obfuscation. A cryptographic approach is called Mobile Cryptography, which relies on cipherprogram concept [4]. The idea is to encrypt agents as a whole and apply computing with encrypted functions and data. Although it is limited to certain functions (e.g., polynomials), this approach is a good example for a software-based solution based on cryptography. Another approach called cryptographic traces [3] aims at detecting any kind of tampering with a MA. It relies on code execution verification using traces based on cryptography. Therefore only detection of tampering is addressed. Blackbox security [2] is an obfuscation scheme, which does not rely on cryptography. The idea is to mess up code and data of an agent so that they do not reveal the function of the code. Code obfuscation is a well-known method and there are many examples especially for Java. However, there are also tools to defeat known obfuscation methods and this is an arms race as indicated in [10].

Another approach protects the computation of agents by relying on trusted third parties [6]: some sites that offer a trusted environment for mobile agents to perform secure operations. Another example [7] assumes a “neutral trusted” host for e-commerce applications which also employs multiple agents similar to our approach presented in this paper. But it is not clear, why and how untrusted hosts guarantee to send the agents to so-called trusted servers to compute with secrets.

The last category can be called fault-tolerant schemes. One case-specific example [5] employs clones of an agent for a single task and at the end of the mission results of clones are compared. In the one malicious server case it is possible to find out if any malicious action against the agents result took place. Another example [9] relies on replication (of every server) and voting schemes.

Information Dispersal is a technique used for fault-tolerance and intrusion-tolerance of information. The study in this area consists of three phases. In the first phase, Shamir [11] showed how to construct robust key management schemes for cryptographic systems, and Rabin [12] similarly showed how to disperse a file into pieces and to use a subset of pieces to combine them later. The second phase includes the techniques Fragmentation-Redundancy-Scattering (FRS) and Fragmented Data Processing (FDP) (See [13], [14] and further references). The main goal of the FRS is to employ several hosts to provide fault-tolerance of the systems and intrusion-tolerance against deliberate faults. FDP is designed to employ parallel processing techniques to be able to process fragmented and scattered pieces of data.

We consider our work as part of the third phase in this field. There are however important differences between the third phase and the former ones. FRS and FDP utilize a distributed static infrastructure to provide tolerance which otherwise centralized. In our case the target environment is already distributed and extremely dynamic. The assumption with those techniques that there are fixed available hosts does not hold for the MA problem. FRS merely relies on a spatial technique (replication) but we need to consider both spatial and temporal solutions for efficiency. Moreover, a trusted execution environment assumption as part of the system does not hold for MAs. Also, in addition to Shamir and Rabin's work we also need mechanisms not only to distribute the secrets but also to be able to compute with them again in a distributed fashion. These differences inevitably add new challenges to the known problems and solutions.

3 The Approach

This section consists of two parts; first one presents the MA security scheme, the second part introduces the infrastructure to realize the proposed solution.

3.1 Compromise-Tolerant Autonomous Group of Agents

There are three crucial requirements to the solution for the malicious hosts problem in an open e-commerce environment. First the solution proposed must be fault- and compromise-tolerant. There is one very easy-to-devise attack against MAs: immediate denial-of-service. A host simply may not run an agent at its own discretion. This is equivalent to the malfunction of the host while the agent resides in it. There is no known solution proposed so far against this kind of attack. Second, our target environment is an open distributed environment. In such an environment, a Network Trusted Computing Base (NTCB) which is supposed to be a "secure host" is a vague definition and is not acceptable. Such a centralized base is a single point of failure as well as a target of attacks. Even if we assume the existence of such host(s), the location of the host(s) could still be a problem. Moreover, such trust relationship hurts the MA autonomy. Third requirement is related to the reverse problem of MA security. A solution to protect agents must not jeopardize the protection of hosts. So, a cryptographic solution as in [4] or self-modifying code as proposed in [2] may not be acceptable. These requirements are actually what make the problem extremely hard.

We consider our work as part of the third phase in this field. There are however important differences between the third phase and the former ones. These differences inevitably add new challenges to the known problems and solutions.

The problem of protecting MAs against malicious hosts comes from the fact that they are, by definition, autonomous. Since this aspect of MAs is the most important among others and actually it is what make them special, it does not make sense to get rid of it for the sake of making them secure. But if we define "autonomy" not for a single MA but a set of MAs communicating and cooperating that rely on nothing but each other to perform a single task to achieve a goal, we will be able to reach autonomous secure MA groups.

The goal is to make MAs "meaningless" when they are treated as a single entity as much as possible since there is a trade-off between openness and security. This makes them resistant against malicious behavior of the hosts where they execute. A given task is split up into two or more MAs. These MAs are located in different hosts (see 3.2 for the infrastructure to realize

this) and migrate when necessary. They exchange information and partial results of execution at certain synchronization points. So, the approach presented here is based on information (data and code) dispersal supported by known cryptographic techniques.

Data Confidentiality is the easiest-to-achieve goal in our approach especially if the data needs to be kept confidential from certain hosts that possibly could gain advantage from knowing it. It is possible to place sensitive information (i.e. credit card) in an agent encrypted with a nonce and place the nonce in a cooperating agent. Other more complex means are also possible.

The detection objects idea has been proposed by Meadows [8] to detect possible modifications to MA data. We extend the detection objects idea and introduce detection code to achieve Correct Code Execution and Data Integrity. Predetermined but random code fragments are injected into the original code of the agent (possibly by also introducing dummy data). These code fragments can be produced by a tool and the modified program can be guaranteed to give the exact same result as the original when executed. At certain synchronization points, agents exchange the results of the modified program together with the results of the original program. Since the dummy code results have to be fixed, another agent can easily check them against the precalculated values. Together with realizing time limitations and dense injection (i.e. one line of dummy code for every original line) it can be guaranteed that the original code was executed correctly.

Although a human can detect which portions of code are dummy by analyzing the code, it is an undecidable problem for machines especially if data flow analysis is prevented by mixing the data flow from the original code to the dummy code (e.g., $y=y*x$; $z=y$; $z=z/x$; $y=z$; y and z are dummy, x is not). Since the concern here is instant correct execution of code and time for that execution can be limited to a few seconds, an analysis by an human is prohibited. The approach presented is equivalent to randomized state information that could be maintained and exchanged among agents.

Code confidentiality is the most difficult of these problems. The difficulty of the problem comes from the fact that it is not possible to limit the time for analysis, therefore human analysis attacks are possible. Due to the difficulty of the problem, our approach is to restrict the confidentiality of code into certain decision functions (e.g., a shopping agent's purchase decision) and apply the code injection technique to those functions before splitting. In addition to arbitrary code generated by a tool randomly, other mechanisms can be applied. For example, code can be generated which consists of similar statements to the original code. Also, code libraries can be used to provide code that implements some relevant or irrelevant function to the original function.

The three goals above are the generalization of broad range of diverse security requirements of the MAs. There are certain situations where combinations of the three goals above overlap. One such requirement in a mobile e-commerce environment is computing with secrets in an untrusted environment. Any such secret (e.g., a private key) cannot be revealed to any third party since the information here is more sensitive. You may for example, reveal your credit card information to somebody to buy lunch but not your social security number. So, it is crucial to have the ability not only to keep certain data secret but also to compute with that data. Digital contract signing [4] is such an example and fortunately there are mechanisms to

do this remotely by a group of agents without divulging secrets or inventing new cryptographic algorithms.

3.2 Mobile Agent Collaboration and Execution Support (MACES) Environment

One drawback of our approach to MA protection is the communication overhead introduced by the necessary cooperation among the agent groups. This actually hurts one of the promising aspects of MAs, which is reducing the bandwidth requirements especially if large amounts of data need to be communicated. For this purpose MACES is proposed. This is a distributed system, which consists of hosts that run a MACES Entity (ME). Together with security, the most important aspect of MACES is to alleviate this problem by maintaining information about hosts connected to the network namely, location, availability, and network QoS measures (distance, bandwidth, etc.). So, MACES is responsible of finding the best combination of hosts for a mission that consists of a group of agents. The best combination of hosts is the one with most efficient network communication without any degradation in security. For instance, the hosts involved in a single mission should not belong to the same administrative domain. For this purpose MACES employs protocols similar to routing information protocols in computer networks together with multicasting. Another possibility is to query routers of the underlying network for this purpose and use the information together with the information maintained by MACES itself.

An ME is responsible for a network domain or a single host, maintains information about other MEs and MAs, and implements collaboration protocols. It is run on an ordinary computer, which runs an ordinary OS. This host can run any existing MA system or any combination of those that are available. It is important to note that there is no assumption on the trustworthiness of any ME or host on which it is run. Every such host (piece of the system) is considered untrusted as opposed to whole MACES.

The MACES is a self-healing system. With self-healing we mean that the system checks itself periodically and applies the appropriate countermeasures against possible intrusions into the system. For example, an ME can intentionally or unintentionally (a Byzantine failure) report that the best places to run a pair of agents are the same host. Thus, the target host could obtain all the secret information by analyzing the peer agents. To combat such an attack, MEs cooperate with the protocols mentioned and prevent such a placement and execution of agents.

It should also be noted that there is no trust relationship between agents and MEs. MACES tries to achieve its self-healing property while agents rely only on their peers. It can be said that an autonomous group of agents and MACES are a crosscheck mechanism.

It is also expected that MACES will help to tackle the lack of interoperability problem of MA systems and enable the Mobile Client/Server concept to be realized. There are many mobile agent systems both commercial and research purposes. The problem besides security for the mobile agent paradigm is that there is no standard followed by these systems. The obvious consequence is agents deployed from different vendors' systems cannot collaborate or even interact. A system as MACES can be capable of acting like a broker between agents and agent systems via a well-defined API. The Mobile Client/Server concept is aimed to combine MA paradigm with client/server computing. This hybrid approach will enjoy advantages of both. With an API designed for this purpose, it will be possible to standardize such functionality. One specific advantage is to isolate MAs to execute in a host that also runs an

ME and protect other servers against possible malicious behavior. However, this aspect of the system is outside the scope of this paper.

4 A Brief Discussion of Attacks

The easiest attacks against MAs are denial-of-service and replay attacks. The main feature of the proposed solution is its resistance against these attacks by using autonomous groups of agents. It should be noted that cryptography does not help with such attacks. Other possible attacks, which are briefly presented by generalization in this paper has solutions by the help of cryptography, code injection (and state information) and other proposals cited in Section 2. One type of attack still open is returning wrong results to system functions. A partial solution, which is specific to application and system function under consideration could be invoking the function remotely by a cooperating agent that resides on another host and then verifying the result by comparing with the local function invocation.

Attacks specific to the approach presented in this paper are organizational attacks, which require cooperation of hosts, where cooperating agents execute. Similarly intrusions to multiple hosts might reveal information about agents that reside on these hosts. Since MACES recognizes domain boundaries, these types of attacks are highly unlikely. Another attack specifically against code confidentiality, which can be devised by a single host, is to match agents executed on this host based on the owner or manufacturer of the agents to identify the cooperating ones from different missions. The countermeasure could be to randomize the code injection to produce different agents at least to prohibit automated attacks.

5 Conclusion

In this paper we briefly presented a compromise-tolerant mobile agent security scheme. The basic idea is to employ an autonomous group of agents, which cooperate and communicate to protect themselves against possible intrusions, in an open e-commerce environment. This scheme is supported by an open, distributed system: Mobile Agent Collaboration and Execution Support environment. We also presented novel techniques to protect agents. It should be noted that other protection schemes like code obfuscation can still be applicable as additional security mechanisms. The approach presented in the paper provides different levels of security required by different MAs, application and users. By replication and further splitting agents into several MAs, desired higher levels of security can be achieved at the expense of efficiency. We are currently working on techniques to split up agents and their sensitive operations. MACES collaboration protocols are also under investigation.

References

- [1] U.G. Wilhelm, S. Staamann & L. Buttyan, On the problem of trust in Mobile Agent Systems, *Proceedings of NDSS'99*, March 1998.
- [2] F. Hohl, Time Limited Blackbox Security: protecting mobile agents from malicious hosts, *Mobile Agents and Security, LNCS 1419*, 92-113, 1998.
- [3] G. Vigna, Cryptographic Traces for mobile agents, *Mobile Agents and Security, LNCS 1419*, 137-153, 1998.

- [4] T. Sander & C.F. Tschudin, Protecting mobile agents against malicious hosts”, *Mobile Agents and Security, LNCS 1419*, 44-60, 1998.
- [5] B.S. Yee, A Sanctuary for Mobile Agents, *Proceedings of the DARPA workshop on foundations for secure mobile code*, 1997, Also available as <http://www-cse.ucsd.edu/users/bsy/pub/sanctuary.ps>.
- [6] A. Corradi, R. Montanari & C. Stefanelli, Security Issues in Mobile Agent Technology, *Future Trends of Distributed Computing Systems*, 7th IEEE workshop on FTDCS’99.
- [7] P.J. Marques, L.M. Silva & J.G. Silva, Security Mechanisms for Using Mobile Agents in Electronic Commerce, *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 1999.
- [8] C. Meadows, Detecting Attacks on Mobile Agents, *DARPA Workshop on Foundations for Secure Mobile Code*, March 1997 available via <http://www.cs.nps.navy.mil/research/languages/statements>.
- [9] Y. Minsky, R. Renesse, F.B. Schneider & S.D. Stoller, *Cryptographic Support for Fault-Tolerant Distributed Computing*, Technical Report TR96-1600, Department of Computer Science, Cornell University, July 1996.
- [10] Dave Dyer, Java Decompilers Compared, http://www.javaworld.com/javaworld/jw-07-1997/jw-07-decompilers_p.html, July 1997.
- [11] A. Shamir, How to Share a Secret, *Communications of the ACM*, Vol. 22, No.11, November 1979.
- [12] M.O. Rabin, Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance, *Journal of the ACM*, Vol.36, No.2, 335-348, April 1989.
- [13] J.M. Fray & J.C. Fabre, Fragmented Data Processing: An Approach to Secure and Reliable Processing in Distributed Computing Systems, *Dependable Computing for Critical Applications*, Springer-Verlag 1991.
- [14] J.C. Fabre, Y. Deswarte & B. Randell, Designing Secure and Reliable Applications using Fragmentation-Redundancy-Scattering: an Object-Oriented Approach, *LNCS 852, Proceedings of Dependable Computing EDCC-1*, October 1994.