ISBN 978-83-60810-27-9
ISSN 1896-7094

Selected Security Aspects of Agent-based Computing

Piotr Szpryngier, Mariusz Matuszek
Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology
Email:{piotrs,mrm}@eti.pg.gda.pl

***Abstract*—The paper presents selected security aspects related to confidentiality, privacy, trust and authenticity issues of a distributed, agent-based computing. Particular attention has been paid to authenticity of and trust in migrating mobile agents executables, agent's trust in runtime environment, inter agent communication and security of agent's payload. Selected attack vectors against agent-based computing were described and risk mitigation methods and strategies were proposed and discussed based on presented cryptography measures. In summary expected effectiveness of proposed countermeasures was described.**

I. Introduction

AGENT-BASED computing based on conceptual software entities named *agents* (often *intelligent agents*) plays an important role in integration of various distributed systems and services. Agent paradigm describes agents as closed entities, existing within agent runtime environments (e.g. [1]), capable of percepting and effecting their environment and aware of their *mission*. Agents typically have a large dose of autonomy in pursuing their goals.

Systems incorporating agent paradigm rarely utilise single agents. More often whole *societies* of agents are employed, interacting socially to reach common goals. Typical examples of such interactions are negotiations, contracts, cooperation and coordination of common tasks. All such interactions are based on communication, which in most cases follows established patterns and norms [2]. Runtime environments for agents also try to follow a set of established rules and guidelines [3].

In heterogeneous, open agent environments, where both agents and agent environments originate from and are hosted by different organisations special attention must be paid to security issues—both agent's security and payload (data) security.

Risks may originate from social attacks (e.g. attempts at fraud, cheating, deception) and technical attacks (e.g. agent environment modifying agent's code or data; agents attacking their environment and creating a denial of service by overloading of selected resources or crashing the container etc.). Many authors addressed selected aspects of agent-based computing in recent years. Extensive research of threats and countermeasures was presented by NIST [4]. In a followup to this paper, Jensen [5] described a taxonomy of protection techniques for agents and agent platforms. The taxonomy was divided into two main groups:

- detective measures—methods to detect modifications of agent's code, agent's payload and agent environment;
- preventive measures—obfuscating of code, data, agent's route etc.).

A major role in those techniques is played by cryptography—digital signatures, encrypted code, one time passwords, session keys. A review of agent protection measures reflecting the social element of agent architectures (agent's context, autonomy, communication with other agents, mobility) was presented by Borselius in [6]. Moreover, author proposes the following ways of addressing threats coming from malicious servers:

- contractual agreements—where server platform operators enter into contract with customers, declaring safety of the environment and administrative practices crafted to avoid violation of customers' privacy or integrity of their data, agent code and computations;
- trusted hardware;
- trusted network nodes;
- fault tolerant computing techniques—i.e. cooperation of many agents, partitioning and duplication of data and redundancy in agents functionality;
- execution tracking—gathering and offering logs of agents activities;
- data encryption (however it should be noted that, once decrypted, data will be easily available to interested parties and also that communication is required to obtain decryption key);
- hiding of code and data in blackboxes [7];
- undetachable signatures carrying encoded constraints on permitted agents' code use—if those constraints are not met, messages sent by agents will not be signed by a host;
- sandboxes and code signing when agents' code meets requirements of security policy requirements (proof carrying code).

Computations carried out by use of encrypted functions were deemed prospect less.

Another attempt at protection methods taxonomy was made by Leszczyna [8]. In his paper he proposed to classify protective measures into following subgroups:

- runtime data protection,
- ensuring of mobile code immutability and security,
- event logging.

In addition, he proposed an extension of JADE agent platform protecting agents against tracking. His proposed solution [8] assumes use of trusted parties, encrypting agents route and possibly other data it carries en route between hosts, thus making it impossible for an attacker to determine where a given agent is coming from and its route, which in turn makes it difficult to deduct on agents' goals and findings. A drawback of this approach is a necessary requirement for agents to always return to a recent forwarding trusted party. In addition, nodes forming trusted parties must securely store encryption keys and correctly match them to returning agents.

Yet another JADE extension was proposed in [9], where Zwierko and Kotulski proposed an original method of ensuring mobile agent's integrity, combining protection of code, data and computational state. Their method was based on secret sharing scheme and zero-knowledge protocol.

A secure agent platform, based on Public Key Infrastructure (PKI), was described by Ismail [10]. Author points at the importance of protection of objects containing data from other agents or server data, because such objects may be referenced by a mobile agent. Java environment (sandboxes, memory protection) provides only a partial protection of data against destruction by a malicious mobile agent, because it lacks mechanisms which would allow definition of various access rights based on authentication or permit runtime evolution and delegation of such access rights during execution and communication as well as carrying such rights during agents migration. A proposed solution [10] is to introduce a mechanism of dynamic permissions exchange between cooperating agents. When migrating between hosts an agent needs to carry its permissions and export them to other agents already working on target host. To allow it agents must have means to mutually authenticate themselves. It was proposed that such mechanisms are to be provided by hosts using an external, trusted PKI. A trusted CA generates a key pair for each host (platform) and signs the public keys. Initially agents provide only minimal rights to other cooperating agents. As cooperation progresses those rights may be gradually extended. A *right* in this context means a particular method able to act on a particular object. In her deliberations author did not address agents' mobile nature, methods of rights protection, ease of implementation or expected gains. In her paper she defined requirements and analysed various authentication scenarios:

- of a client agent created by a local server;
- of a foreign (visiting) agent—a problem in this case is the necessary communication between a local and a foreign agent because of its impact on processing performance.

Access control lists (ACLs) were proposed for rights management and secure naming services and servers were proposed to allow communication where symbolic names are used as addresses. Each server uses a key-store mechanism to store its private key. Every agent has three relations (sender, owner, writer) with other entities (servers, users): Each of them signs agents' code, data and rights.

A protection against hostile hosts hosting agent platforms was proposed by Hohl [7]. Author stresses that protection of mobile agents against malicious servers and protection of agent environments against malicious agents are of equal importance. He describes attacks which can be carried by servers against agents' data, code and execution. Two main attack classes are distinguished:

- passive attacks—where agents' data, code and communication are spied upon / intercepted;
- active attacks—where agents' data, code, execution patterns and communications are subject to modifications, alterations and impairment.

A main concept addressed in [7] is a blackbox mechanism—blocking access to code and data. Author states, that it is not possible to construct an agent with a generic blackbox, but only with a time limited one. The stated reason is threat of dictionary attacks. He proposes to construct such a time limited blackbox with the use of transformations using random parameters (so called mess-up algorithms). However, this method ca not be applied to every kind of code and data. In addition, blackboxes may be vulnerable to various attacks, including sabotage, testing of limitations (e.g. finding agents' upper limit on price it is willing to pay) which lets an attacker uncover partially agents' data. Another problem is finding the right time limit of a blackbox. Too short limit may impede achievement of agent's goals and reveal its data and mission before it is accomplished. Setting the limit too long increases the chance of a successful attack against the blackbox.

II. Security issues in typical agent environments

Each type of computing has security considerations. Agent based computing is no different in this respect. Typical problems include: *authenticity* (who is the sender/author of a message), *confidentiality* (of sensitive/valuable data) and *privacy* (when personal data is carried/processed). Sometimes non-repudiation is also important. Cryptographic protocols are used to address these issues, mostly digital signatures and data encryption. Agents may sign messages to try to ensure its authenticity. As discussed later in this paper, this approach may be insufficient. All the mentioned security aspects of computations influence users trust in results—also known as results authenticity. Agent-based computing security mainly consists of:

- ensuring agent authenticity (is it really the agent sending data and not a malicious phantom (fake));
- trusted computing environment (including trusted agent platform — i.e. not introducing changes to agent's code or data; not interfering with agent actions unless it is necessary to protect itself against a malicious agent);
- authenticity of results—obtained results can be trusted to be original, not falsified;
- maintaining confidentiality of results during their transfer to recipient.

Typical agent environments [11][12][13] assume full trust in hosts and provide only a limited set of security tools — user authentication and cryptographic data tunnels (TLS/SSL). Moreover agent containers may be authenticated using digital signatures. Typically combinations of DSA/SHA1 and RSA/MD5 are used. MD5 algorithm is considered weak [14] as collision possibilities were detected. To start data signing it is necessary to provide a password deciphering a private key. It implies, that during agent activity either the password or the deciphered private key is accessible and visible, otherwise user engagement would be necessary on each signing attempt. Public key certificates are self-signed by corresponding private keys and must be distributed in advance to agent containers with the assumption, that each agent container uses identical combination of hash and signing functions. It implies full control over agent and hosting environment by users deploying agent-based applications. In addition, this environment should be isolated from threats coming from the Internet to ensure trust in results.

III. Attacks on agent environment and security assessment

Typical attacks on agent environments and agent-based computing lead to falsifications of computing results or disruption of agent activities. They include:

- undetected injection of hostile agent(s) posing as legitimate ones and deforming the results of agent interactions;
- snooping or substitution of messages;
- posing as a legitimate agent to damage or destroy data stored on machines hosting agent environment.

Unfortunately, neither JADE [11][12] nor other environments provide security to agents or authenticity of results because of several reasons: agents acting within a remote agent container are in an alien environment, thus are susceptible to remote interference (eavesdropping, intrusion, decompiling, interception of results, etc.). Agents in this situation are unable to securely store any sensitive data (e.g. signing keys, passwords, etc.) because at any time such data may be read by a host. Even SSL session keys are not protected because of these issues.

Current solutions in agent-based computing do not address any of the security postulates [10][4][5][8], i.e. neither really confidential communication is provided nor it is authentic. Proposed remedies are selective in their nature (i.e. nothing is gained by SSL encryption when session keys are readily available). Any solution in this area requires cooperation between agent and a trusted host to be effective [8].

IV. Improving security of agent-based computing

To solve the problem of authentic and confidential delivery of agent-based computing results to the originating party several assumptions should be made:

- hosts taking part in agent-based computing should be methodically designed, tested and reviewed, i.e. they fulfil the requirements of ISO Evaluation Assurance Level 4 [15]. Such requirements are in use when designing digital signature environments that are considered valid by authorities (government, legal system). Similarly, system projects should follow generally recognised software engineering safe practices, i.e. automatic architecture management and testing to ensure system resistance to most potential attacks.
- Public Key Infrastructure with PK certificates issued by recognised Certification Authorities (CA) should be designed and implemented.
- Agent environments should be extended to provide communication interfaces to host based services like signing, time stamping, session key generation, etc.

The solution proposed below is similar to the one described in [10]. However, it is based on the assumption of trusted hosts availability. We postulate, that without the support of trusted hosts the security of agent-based computing can not be attained. With these limitations in mind we envision a secure agent-based computing in a following way:

1) User (computation originating party) signs the agent's code and data, assuring the receiving host of agent's credibility.
2) Agent executes inside a monitored container, trusted not to modify agent's code or results.
3) For communication with other agents or originating host agent calls local trusted host provided services like: digital signing or time stamping. After obtaining either a time stamp certificate or a signature agent combines it with a message and sends it to a recipient. To provide in-transit data security either SSL tunnels may be utilised or, in their absence, messages can be encrypted with a session key enciphered by a recipient's public key. Encrypted session key is attached to a cipher text.
4) A receiving party, using trusted PK certificates of computers involved in computations, verifies the signature of a message, therefore gaining trust in its contents and its origin.

V. Summary

We analyse the security of a proposed approach:

1) A signed agent should convince each hosting computer about agent's code authenticity. Public key certificate of the originating party should be commonly available to allow any party to validate the signature and therefore identify agent's origin.
2) Data and results gathered or produced by an agent will be trustworthy if the environment in which they were obtained is sufficiently secure. Access to a public key certificate issued by a commonly recognised CA attests to the hosts credibility. The procedures of obtaining such a PK certificate ensure, that the host and the organisation running it is sufficiently trustworthy.
3) Similar arguments can be used to indicate, that trust can be extended to a hosting computer and its signature. Host security requirements create a similar level of confidence in its private (signing) key. Agents can not store any signing keys—even in an encrypted form — because at signing it would have to be uncovered and in a such case accessible to all observers.

Implementation of our proposals should significantly improve trustworthiness of agent based computing. They are not perfect, and additionally require a very high level of security for all computers and a PK infrastructure. In our approach it is also mandatory to develop new mechanisms of cooperation between an agent and a hosting computer, because agents can not hide any secret within themselves and such secrets are necessary to use cryptographic protocols. Currently we are working on implementation of our proposals as extensions to Jade environment. We expect to have some results sometime around half of next year and after verification of their efficacy and efficiency we plan to publish our findings.

References

[1] "JADE (Java Agent DEvelopemnt Framework) Online Documentation," Telecom Italia Lab. [Online]. Available: http://jade.tilab.com /doc/index.html

[2] "FIPA — IEEE foundation for intelligent physical agents." [Online]. Available: http://www.fipa.org

[3] "MASIF." [Online]. Available: http://www.omg.org/cgi-bin/doc?orbos /97-10-05

[4] W. Jansen and T. Karygiannis, *Mobile Agent Security*. Gaithersburg, MD 20899: NIST, 1999, vol. NIST Special Publication 800-19.

[5] W. Jansen, *Countermeasures for mobile agent security*. Elsevier, 2000.

[6] N. Borselius, "Mobile agent security," *Electronics & Communication Engineering*, vol. 14, no. 5, pp. 211–218, 2002.

[7] F. Hohl, *Time limited blackbox security: Protecting mobile agents from malicious hosts*, ser. LNCS. Berlin: Springer-Verlag, 2006, vol. 1419, pp. 92–113.

[8] R. Leszczyna, "Architecture supporting security of agent systems," Ph.D. dissertation, Gdańsk University of Technology, Faculty of Electronics Telecommunications and Informatics, 2006.

[9] A. Zwierko and Z. Kotulski, "Integrity of mobile agents: A new approach," *Intl. Journal of Network Security*, vol. 4, no. 2, pp. 201–211, 2007.

[10] L. Ismail, "A secure mobile agents platform," *Journal of Communications*, vol. 3, no. 2, 2008.

[11] "Jade Board: Jade Security Guide," Telecom Italia Lab S.p.A., 2004. [Online]. Available: http://jade.tilab.com/doc/index.html

[12] G. Vitaglione, *Mutual-authenticated SSL IMTP connections*. Telecom Italia Lab, 2008.

[13] V. Gunupudi and S. Tate, "SAgent: a security framework for JADE," in *Proceedings of the 5th intl. joint conference on Autonomous agents and multiagent systems*. AAMAS, 2006, pp. 1116–1118.

[14] B. Schneier, *Applied Cryptography*, 2nd ed. New York: John Wiley & Sons, 1996.

[15] "ISO15408 Common Criteria." [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/