International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August 2013
ISSN 2229-5518

Critical Misbehaviour Prevention for a Secure Mobile Agent System

Jasiya Jabbar, Jyothi B, Mala J B

[1,2]M.Tech (Software Engineering), TKM INSTITUTE OF TECHNOLOGY, KOLLAM

[3]ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE, TKM INSTITUTE OF TECHNOLOGY, KOLLAM

Abstract— A software agent is an intelligent program that acts as a user's personal assistant. Software agents endowed with the property of mobility are called mobile agents. Mobile agents perform a user's task by migrating and executing on several hosts connected to the network. The main drawback of mobile agents is the security risk involved in using mobile agents. Security risks in a mobile computing environment are two-fold. Firstly a malicious mobile agent can damage a host. On the other hand a malicious host can tamper with the functioning of the mobile agent. Any behaviour exhibited by a machine that is not taken into account at the time of programming, regardless of its favourable or unfavourable outcome, is considered to be misbehaviour. The surveys conducted on mobile agent security gives idea about different techniques that can be used for ensuring security of mobile agents. In this section we uses digital signature and RSA encryption algorithm for providing mobile agent security. In this work, we use JADE for the development of mobile agents. JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in Java language. JADE is free software and is distributed by Telecom Italia. In this work, JADE is enhanced to confirm major security goals-confidentiality, integrity and availability.

Index Terms— Software agent, Mobile agent, Misbehaviour, Digital signature, Encryption, JADE, Security

1 Introduction

Software agents are probably the fastest growing area of Information Technology (IT). They are being used, and touted, for applications as diverse as personalised information management, electronic commerce, interface design, computer games, and management of complex commercial and industrial processes. For the past couple of years, agent technology has been a hot topic, and most likely, this is mainly due to the popularity of the Java programming language, which represents an ideal language for implementing software agents as it is the "*Write Once Run Anywhere*" language. This is an important feature for software agents as it allows them to run on all platforms of the Internet.

An agent can be defined as an entity that acts on behalf of others in an autonomous fashion, performs its actions in some level of proactivity and reactivity, exhibits some levels of the key attributes of learning, co-operation, and mobility. Software agents can be classified as *static agents* and *mobile agents*. Static agents achieve their goal by executing on a single machine. On the other hand, mobile agents migrate from one computer to another in the network and execute on several machines. Mobility increases the functionality of the mobile agent and allows the mobile agent to perform tasks beyond the scope of static agents.

MAS can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver (Durfee and Lesser 1989).

2 Related Works

Mobility makes them a powerful tool for implementing distributed applications in a computer network. MAS can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver. Any behaviour exhibited by a machine that is not taken into account at the time of programming, regardless of its favourable or unfavourable outcome, is considered to be a misbehaviour [1]. Misbehaviours among the agents in a network might be intentional or unintentional, they might cause a system-wide failure or they might improve the performance or even enable us to achieve an objective.

Eventhough an agent misbehaving in a network does not necessarily result in catastrophe, one cannot discard the importance of the detection of the situations where an agent is misbehaving, that is, not following the original programming and the rules that were defined for it.

For detecting misbehaving agents in a network, observer-based approach is used. In model-based FDI, every node in the network has a model of the entire system. In observer-based FDI technique [1,2], each node in network monitors its neighbouring nodes. It calculates a residual value based on the real and expected outputs for each of its neighbours, if this value exceeds a threshold then the agent is said to misbehave. Every node in network is informed about the presence of misbehaving agents.

To accomplish some of the tasks in networked systems it is required that one agent or a subset of them act as leaders of the group. Since success in accomplishing tasks depends on choosing a leader, one may ask how this leader is being selected in multi-agent systems. With Distributed leader selection algorithm [1,3] one and only one leader is selected. In this, every agent in the network bid for leadership. If more than one agent bid at same time then they wait for a random time and bid again. This continues until a single agent is elected as a leader. If the leader is detected as misbehaving or if leader fails leader selection procedure is repeated.

Multi-agent system contains static and mobile agents. A mobile agent is a particular class of agent with the ability during execution to migrate from one host to another where it can resume its execution. Mobile agent technology, amongst other things, can help to reduce network traffic and to overcome network latencies. An agent's ability to move does however introduce significant security concerns. Security risks in a mobile computing environment are two-fold. Firstly a malicious mobile agent can damage a host. On the other hand a malicious host can tamper with the functioning of the mobile agent.

The major security goals of mobile computing include confidentiality, integrity and availability. The various properties of agents that produce security issues are mobility, autonomy, social ability, etc. Classification of Security threats in an Agent System

- Agents attacking Hosts [4] - Malicious agents can steal or modify the data on the host. Lack of sufficient authentication and access control mechanisms lead to these attacks. If resource constraints are not set, they can also commit Denial of Service (DoS) attacks by exhausting computational resources and denying platform services to other agents.
- Hosts attacking the Agents[4] – A malicious host can attack the agent, by stealing or modifying its data, corrupting or modifying its code or state, deny requested services, return false system call values, reinitialize the agent or even terminate it completely. It can also masquerade the agent by delaying the agent until the task is no more relevant. The Host may also analyze and reverse engineer the agent.
- Malicious Agent attacking another agent [4] – A malicious agent may invoke public methods of another agent to interfere with its work.
- Attack by other entities [4] – Some other entity in the network may manipulate or eavesdrop on agent communication.

Available technologies and research efforts addressing the security issues arising from the mobility property of mobile agents includes mechanisms addressing various security aspects of the mobile agent, and technologies protecting the executing hosts from agents.

2.1 Agent attacking the Host environment

Traditional methods such as authentication, access control, sand-boxing techniques, and cryptography can be used to secure the Host.

- ***Authentication and access control mechanisms*** [5] – This is the first line of defense against a malicious agent. If the Host can authenticate the agent and in turn the device that dispatched the agent, it can apply authorization and access control.
- ***Path Histories* [4, 5] -** An agent could reach the host by making a number of hops. During this transit a malicious host could have morphed the agent into a malicious agent. By storing the log of the travel of the agent, the current host can determine the route taken by the agent. . Each host platform to which the agent travels to appends a signed entry to the path. This entry indicates the hosts identity as well as the identity of the next host the agent intends to visit. The platform has to judge by looking at the log if the previous platforms can be trusted.
- ***State Appraisal* [4, 5]** – The author of the agent supplies a state appraisal function called maximum function. This function calculates, depending on the state of the agent, the maximum set of permissions to be granted to the agent. This function is packaged together with the agent. The user/owner of the agent also supplies another state appraisal function called the request function. This calculates the permissions the user wants the agent to have during execution. The host platform uses these state functions to verify the correct state of the agent and hence determines the privileges to give to the agent depending on its state. This ensures that the agent has not turned malicious due to alterations of its states.
- ***Proof carrying code* [4, 5]** – Proof carrying code requires the author of an agent to formally prove that the agent conforms to a certain security policy. The execution platform can then check the agent and the proof before executing the agent. The agent can then be run without any further restrictions. The major drawback of this approach is the difficulty in generating such formal proofs in an automated and efficient way.

2.2 Host platform attacking agents

Providing security against the attacks by the host is difficult due to the fact that the host needs to have the full knowledge of the code and the state in order to execute the agent. Traditional mechanisms are not sufficient to protect an agent from the attack of malicious hosts.

- ***Environmental key generation***– Environmental key generation [4, 5] allows an agent to carry encrypted code or information. The encrypted data can be decrypted when some predefined environmental condition is true. Using this method an agent's private information can be encrypted and only revealed to the environment once the predefined condition is met. This requires that the agent has access to some predictable information source. Once the private information has been revealed, it would, of course, be available also to the executing host. However, if the condition is not met on a particular host, the private information is not revealed to the platform.
- ***Execution tracing*** – Execution tracing [4, 5] has been proposed for detecting unauthorised modifications of an agent through the faithful recording of the agent's execution on each agent platform. Each platform is required to create and retain a non-repudible log of the operations performed by the agent while executing on the platform. The major drawbacks of this approach are not only the size of the logs created, but also the necessary management of created logs.
- ***Trusted nodes*** – By introducing trusted nodes [4, 5] into the infrastructure to which mobile agents can migrate when required, sensitive information can be prevented from being sent to untrusted hosts and certain misbehaviours of malicious hosts can be traced. The owner's host, i.e. the platform from where the mobile agent first is launched, is usually assumed to be a trusted node. In addition to this, service providers can operate trusted nodes in the infrastructure.
- ***Co-operating agents*** – By using cooperating agents [5], a similar result to that of trusted nodes can be achieved. Information and functionality can be split between two or more agents in such a way that it is not enough to compromise only one (or even several) agents in order to compromise the task.

3 Proposed System

The surveys conducted on mobile agent security gives idea about different techniques that can be used for ensuring security of mobile agents. In this section we uses digital signature and RSA encryption algorithm for providing mobile agent security.

Digital signature is a method of verifying authenticity, and also ensures that no alterations are made to the data once the document has been digitally signed i.e., it confirms the authenticity of an object, its origin, and its integrity. Typically the code signer is either the creator of the agent, the user of the agent, or some entity that has reviewed the agent. For generating digital signature, the code signer will be given two keys: a public key and a private key. Here we sign the mobile agent code itself, it will be crunched using hash function to form message digest which will be encrypted using the private key to form the signature.

As we know mobile agents migrate from machine to machine within a network enviournment. It can suspend execution at any point in time, transport itself to a new machine and resume execution there. Here we consider a network of nodes; one among them is selected as a coordinator node most probably randomly. When a mobile agent is created, a checksum will be calculated and it will be stored in mobile agent registry in the coordinator. When the mobile agent migrates to a machine, before executing it, the checksum will be verified. The agent code will be encrypted using RSA algorithm and send to new machine. In the new machine, the mobile agent class will be extracted and the checksum will be calculated and send to coordinator for verification. After verification, only if the checksum are same, the agent can be executed in the new host.The working of the proposed system can be depicted as shown in fig 1 shown below.

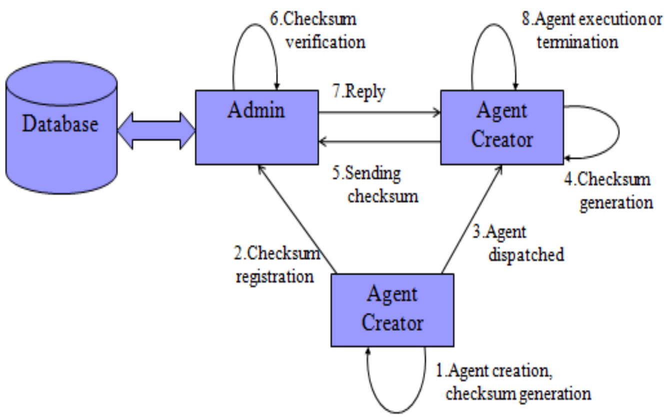

Fig 1. Working of Proposed Research Work

In this work, JADE framework is used for the development of mobile agents. In the current version of JADE framework, no security is provided and if the platform is started in all systems in the network, an illegal user can misuse it to access unauthorised data. As part of this work, JADE platform is enhanced by modifying the framework.In this "Agent" class of jade.core jar file is modified by modifying "aftermove ()" method in ts class.

3.1 General Concepts and Methodology

3.1.1 Co-ordinator node

Here we consider a network of nodes in which JADE platform is started. One of the systems is selected as a server or a co-ordinator. This is responsible for providing user authentication. It provides user with userid and password. When user logs in, it verifies it. It handles the checksum registry, which contains the checksum of agent classes registered by users. It contains the migration history.

3.1.2 Checksum generation

When a mobile agent is created, checksum is generated to corresponding agent class file and it will be registered to the co-ordinator. For checksum generation, MD5 algorithm is used. When a mobile agent is executed, it migrates to another system. When it reaches there, before execution the agent class will be extracted and checksum is calculated again. This is encrypted and sends to co-ordinator; there the verification is done for further execution. Only if both are same the agent will be executed in the new platform.

3.1.3 Checksum verification

Here we use JADE for the development of mobile agents. JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in Java language. It is used for the development of mobile agents. JADE is free software and is distributed by Telecom Italia. The current version of JADE doesn't provide security to the systems in the network. As part of this phase, to improve security checksum of mobile agent has to be registered in the co-ordinator. After migration the checksum will be again calculated and it will be verified. For doing this verification, JADE platform is modified. JADE contains a package named "core", it contains a class named "Agent". By extending the class "Agent" a mobile agent can be created. In this agent class "afterMove" method is modified to verify whether the checksum is same. Only if they are same the agent can execute in the new host.

3.1.4 Encryption

The encryption is another major mechanism in our design. The algorithm used is RSA. For registering checksum, the checksum will be first encrypted using the registering node's private key and thus forms the digital signature and is then encrypted using the public key of the receiver. This ensures integrity and confidentiality.

4 Performance Evalution

An agent's ability to move introduce significant security concerns. In this system, we focus on the security issues related to mobile agents. JADE framework is used for the implementation of mobile agents. In this work, JADE is enhanced to confirm major security goals-confidentiality, integrity and availability. To comply the needs for security we use the following security features

- User authentication (authentication)
- Digital signature (data integrity, non-repudiation)
- Encryption (confidentiality)
- Checksum verification(Framework modification)

Parameter	Enhanced JADE	JADE
A central server system	Present	Not present
Authentication	Yes	No
Agent execution	All agent programs	All agent programs
Execution after migration	Registered agent	All agents
Checksum verification	Done	No
Encryption	Done	No

Table 1 Performance comparison of new system

5 Conclusion

Security is an important issue for the widespread deployment of applications based on software agent technology. An agent's ability to move does however introduce significant security concerns. This article mainly focuses on the security issues of mobile agents and various countermeasures. Security risks in a mobile computing environment are two-fold. Firstly a malicious mobile agent can damage a host. On the other hand a malicious host can tamper with the functioning of the mobile agent.

For producing a secure mobile agent system digital signature, encryption and checksum verification is used in this work. This ensures authenticity and integrity. Thus, this system prevents misbehaviour in mobile agents rather than monitoring them. JADE framework is used for the implementation of mobile agents. In this work, JADE is modified to confirm the security goals. When a mobile agent reaches a host,there the verification is done and only if it is a registered agent it will be executed there, otherwise it will be killed.

References

[1] Iman Shames, André M. H.Teixeira,Henrik Sandberg, Karl H. Johansson, " Agents Misbehaving in a Network : A Vice or a Virtue?" IEEE Network, 2012, vol. 26, no.3, pp.35-40.

[2] I. Shames et al., "Distributed Fault Detection for Interconnected Second- Order Systems", Automatica, vol. 47, no. 12, 2011, pp. 2757–64.

[3] I. Shames et al., "Distributed Leader Selection Without Direct Inter-Agent Communication", 2nd IFAC Wksp. Estimation and Control of Networked Systems, Annecy, France, 2010, pp.221–26.

[4] W.A.Jansen, "Countermeasures for mobile agent security", Computer Communication, 2000.

[5] Niklas Borselius , "Mobile agent security",Electronics & Communication Engineering Journal, October 2002, Volume 14, no 5, IEE, London, UK, pp 211-218.

[6] Danny B. Lange and Mitsuru Oshima, "Seven Good Reasons for Mobile Agents", Communications of the ACM, March 1999/Vol. 42, No. 3.