IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

A Novel Security Approach in Mobile Agent Using Encryption

Nidhi Gupta[1], Dr. Anurag Dixit[2]

[1]M.Tech Scholar1, M.D.U Rohtak, BRCM CET Bahal
nidhi_gupta_11@yahoo.co.in

[2]Professor and Head (CSE), M.D.U Rohtak, BRCM CET Bahal
anuragdixit@gmail.com

Abstract

The appearance of software agents has given rise too much discussion of what such an agent is and how it differs from programs in general. An agent is anything that can be viewed as perceiving its environment through sensors & acting upon that environment through actuators. The existing systems can be classified in the context of single-agent systems and multi-agent systems. Mobile agents can transport themselves from one host to another. Mobile agents have been developed as an extension to and replacement of the client-server model. The proposed system is Mobile Agent System. It reduces network load and latency in which there is usually no transmission of intermediate result. This conserves the network bandwidth .Since the agents are autonomous; the mobile device that dispatches the agent need not be connected all the time.

Keywords: *Mobile Agent, Encryption, Aglet.*

1. Introduction

An agent is a computational entity which acts on behalf of other entities in an autonomous fashion, performs its activities with some level of pro-activity and/or reactivates exhibits some degree of the key attributes of learning, co-operation and mobility.

The existing systems can be classified in the context of single-agent systems and multi-agent systems. In single-agent systems, an agent performs a task on behalf of a user or some process. While performing its task, the agent may communicate with the user as well as with local or remote resources, but it will never communicate with other agents. In contrast, the agents in a multi-agent system may extensively cooperate with each other to achieve their individual goals. Of course, in those systems, agents may also interact with users and system resources.

There is a significant difference between mobile agents and simple "traditional" mobile Code. This difference can be described by two kinds of mobility:

a) Remote Execution (which means that a program is sent to a remote location before its activation and remains at this location during its entire life time) and

b) Migration (which means that a program/mobile agent is able to change its location during its execution.

Mobile agent is a program that can migrate from machine to machine in a heterogeneous network. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport itself to another machine and resume execution. It contains:

•Code - the program that defines the agent's behavior.

•State - the agent's internal variables etc., which enable it to resume its activities after moving to another host.

•Attributes - information describing the agent, its origin and owner, its movement history, resource requirements, authentication keys, etc. for use by the infrastructure. Part of this may be accessible to the agent itself, but the agent must not be able to modify the attributes.

Mobile agents can be regarded as an alternative of the traditional client-server paradigm. While the client-server paradigm relies on remote procedure calls across a network, mobile agents can migrate to the

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

desired communication peer and take advantage of local interactions.

2. Mobile Agents and Mobile Agent Environment

A mobile agent environment is a software system, which is distributed over a network of heterogeneous computers.

A mobile agent is a software entity, which exists in a software environment known as mobile agent (MA) environment as shown in Fig. 1.

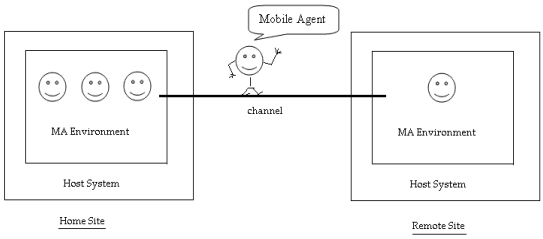


Fig : Mobile Agents and Mobile Agent Environment

The mobile agent environment is built on top of a host system and its primary task is to provide an environment in which mobile agents can execute. Mobile agents can travel between mobile agent environments. They can communicate with each other either locally or remotely.

2.1 Mobile agent based approach

A mobile agent is created and is sent to the remote web – server for information retrieval and all the communication and processing is done by the agent locally and eliminates lots of communication overheads that occur in a normal client - server based approach.

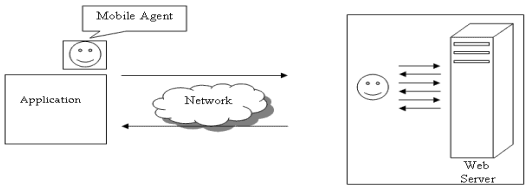


Fig : Mobile agent based approach

3. Aglet 2.0.2

Mobile agents are the basis of the emerging technology which makes it easier to design, implement and maintain distributed systems. Mobile agents have the unique ability to transport themselves from one system in a network to another. As mobile agents reduce network traffic, overcome network latency and most importantly their ability to operate asynchronously and autonomously. It helps to construct more robust and fault tolerant systems. Aglets are Java objects that can move from one host on the internet to another. That is, an aglet can execute on one host, can stop its execution, dispatches itself to other host and resume its execution on new remote host. On moving from one system to another aglet carries its code and data with it. The word aglet means “lightweight agent” in much the same way that applet means lightweight application. The term aglet is a combined work of agent and applet Aglet is developed by research team at the IBM Tokyo Research Laboratory in Japan in early 1995 and is now open source. Aglets are hosted by an aglet server in a similar way in which an applet is hosted by a web browser. The Aglet server provides an environment where agents can execute and Java language and Aglet security manager make the agents transfer safe. The Aglets Software Development Kit (ASDK) is an implementation of an Aglet API. The ASDK includes Aglet API packages, documentation, sample aglets, and the Tahiti Server.

3.1 Basic Elements

The aglet object model explains some abstraction and the behavior which is used to take full advantage of this agent technology. The abstractions which are used are:
1. Aglet: an aglet is a java object which moves in a network and gets executes on host which are aglet enabled. It is autonomous and run in its own thread.
2. Proxy: a proxy is a representative of an aglet. It also protects the aglet from direct access to its public methods. The proxy also provides the location transparency for the aglet.
3. Context: a context is where an aglet executes. It is a stationary object that provides a means for maintaining and managing aglets.
4. Identifier: an identifier is bound to an aglet. This identifier is globally unique an immutable throughout the lifetime of the aglet.
The following list has summarized the fundamental operation of an aglet

• Creation: the creation of aglets takes place in a

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

Context. The new aglet is assigned an identifier, inserted into the context, and initialized.

• Cloning: the cloning produces exact copy of the original aglet in the same context.

• Dispatching: dispatching an aglet from one context to another will remove it from its current context and insert into the destination context, where it will restart execution. This process is termed as dispatching.

• Retraction: the retraction will pull aglet from its current context and insert it into the context from which the retraction was requested.

• Activation and Deactivation: the deactivation of an aglet will halt its execution for the mentioned amount of time and store its state in secondary storage. Activation will again restore it in the same context.

• Disposal: the disposal of an aglet will halt its current execution and remove it from its current context.

3.1.1 Security Model

This section describes the security model that provides an overall framework for aglet security as is shown by the Luca Ferrari in its aglets manual.

3.1.2 Principals

Principals in agent system are authenticated identities that are used to enforce the policies that are defined by authorities and to authenticate the developer of the program or the host it is communicating with.

• Aglet: as they are autonomous in nature, it is reasonable to assume that they can define their own security policies. There are three roles for aglet principles:

1. Aglet: an aglet object is the thread responsible for executing the aglet.

2. Aglet manufacturers: the aglet manufacturer represents the person or organization that implemented the aglet program. The behavior with proper permissions is also set by these manufacturers.

3. Aglet owners: the aglet owner represents the person or organization that launched the aglet. Because the owner is responsible for its aglet, this principal is used for authorization of the aglet

• Context and Server: contexts and servers are responsible for keeping the underlying operating system safe by protecting it from malicious aglets. A server defines a minimal security policy to protect local resources. On the other hand, each context is responsible for hosting visiting aglets, and it enables the access to various local resources. There are three roles for context and server principals:

Context: a context is a place that hosts aglets.

Context manufacturer: this is the manufacturer of a context server. As with aglets, it is in a manufacturer's concern that no one be able to claim damage caused by a malfunctioning context and server.

Context owner: the context represents the context owner. This Principal is used for authenticating hosts in the role of sender and receiver of aglets.

Network Domain: the domain can be represented as the group of servers. The Principal of the domain authority is used to authenticate whether a server is member of its domain or not. A network domain is responsible for keeping its network secure so that all incoming aglets can complete their tasks safely.

3.1.3 Permissions

Permissions define the capabilities of executing aglets by implementing the access restrictions and limits on resource usage. Permission is a resource, such as a local file, together with appropriate actions such as reading or writing a file, listening to a network port, or creating a desktop window. Several permissions are available for aglets:

File permission: Access to the local file system is also subject to control. The aglet can be granted access to a specific file or an entire directory File Permission "/tmp/*" "read Network permissions: the aglet can be granted access to a specific host or to listen on a specific port.

Context Permission: an aglet can be given permission to use services provided by the context. This may include access to methods for creating, cloning, dispatching, retracting, deactivating and activating aglets.

Aglet Permission: the methods provided by individual aglets also need some control. An aglet can be allowed to invoke methods in another aglet

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

owned by a principal given by a name.

3.1.4 Protections

Although an aglet may be granted access to a resource or other aglets, it may also want to protect itself from access by other entities. For example, it is reasonable for user to request that an aglet should be disposed by you only, whereas other methods may be publicly accessible.

3.1.5 Policy and Authority

A policy authority is the person or organization for resources consumed by other entities. In this security model there are three authorities:

- Aglet owner: as the aglets are autonomous in behavior, it reasonable to assume that they can define their own security policy. The main objective of the aglet owner is to protect the aglet from attacks. The aglet owner defines the security policies for aglet. When an aglet migrates to remote host or a context, it requests the host to implement the security policy.
- Context owner: a context authority keeps the server and its system safe from malicious agents. It defines the action that an aglet can take in a particular context.
- Network domain owner: network domain is responsible for keeping its network of system secure so that aglets can securely execute and finish their task.

3.2 Agent Transfer Protocol

ATP is a simple application-level protocol designed to transmit an agent in an agent system-independent manner. An ATP request consists of a request line, header fields, and content. The request line specifies the method of the request, while the header fields contain the parameters of the request. ATP defines the following four standard request methods:

Dispatch

The dispatch method requests a destination agent system to reconstruct an agent from the content of a request and to start executing the agent. If the request is successful, the sender must terminate the agent and release any resources consumed by it.

Retract

The retract method requests a destination agent system to send a specified agent back to the sender. The receiver is responsible for reconstructing and resuming the agent. If the agent is successfully transferred, the receiver must terminate the agent and release any resources consumed by it.

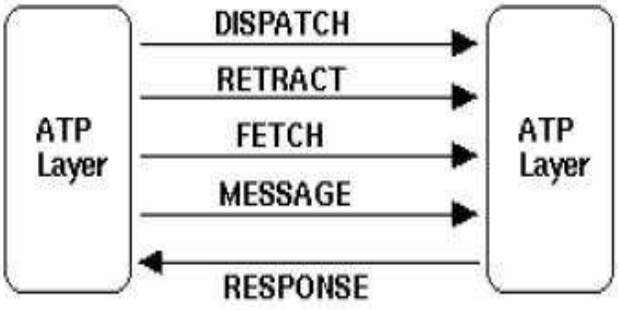
Fetch

The fetch method is similar to the GET methodin HTTP; it requests a receiver to retrieve and send any identified information (normally class files).

Message

The message method is used to pass a message to an agent identified by a agent-id and to return a reply value in the response. Although the protocol adopts a request/reply form, it does not lay down any rules for a scheme of communication between agents.

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com



4. Java Aglet Characteristics

Java is excellent for designing and programming mobile agents. Java has many built in features that qualify it to be the first choice in agent programming. Because of the byte-code concept the Java compiler offers, java applications are able to be executed anywhere, on any computer of the network, no matter of the underlying Operating System or Hardware. Java does not support dangerous pointer operations that are able to overwrite memory parts and corrupt data in such ways. The "Sandbox" principle the Java Runtime creates makes it extremely safe to execute code received from the network. In short, the most positive aspects of using Java are:

Platform independence

Secure execution

Further aspects are: dynamic class loading

multithread programming object serialization

reflection There are several drawbacks to the Java Virtual Machine Concept, these are:

1. Inadequate Support for Resource Control (may result into Denial Service Attacks)
2. No Object Ownership of References
3. No Support for Preservation and Resumption of the Execution State (only internal agent attributes and some external events may be used to reconstruct the execution state of a Mobile Agent)

4.1 Agent Execution Environment

Aglets Tahiti server is used as the agent execution environment which provides support for agent creation, arrival, dispatch and agent management. Tahiti is an application program that runs as an aglet server and provides a User Interface (Figure) that can be used to view and activate events on an aglet. It also enables the user to set access and network privileges for the aglet server.

Figure 4.1 Tahiti: The Aglet Viewer

To run Tahiti you simply have to open a command window, e.g. UNIX shell or DOS window, and by executing a script

- **Agletsd**

  which launches Tahiti on the host machine.

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

Once launched the server runs on a port on the computer, the default being port 4434 i.e. the daemon which the aglets runs on will listen for incoming aglets on port 4434. The launching script has a number of switches that the user can enable to modify the setting up of the server. For example if user wanted to launch on a specific port on user's host machine user would use the -port switch e.g. agletsd -port 8000 which would launch the server on port 8000 of the host machine The first time Tahiti is run a registration panel is launched in which the user enters information with which he/she uses to identify themselves to the agent system. This information (which is simply the user's name, organization, and e-mail address) is also used to tag aglets that are created by this server with their owner's identity. This way if an aglet of user's is running on a remote server when it executes a line of code which causes the aglet to throw an exception that stops the aglet returning to user's system then the user of this remote server now has access to contact information with which they can notify user of the status of aglet .

Once this registration has been successfully completed the primary Tahiti window is launched from which the user can now perform various functions to monitor and control an aglet's lifecycle with a simple and intuitive GUI. This GUI has a series of clickable buttons which in turn launch windows that enable the user to perform the major events that affect an aglet's lifecycle.

- **Aglet**: This menu allows the user to select actions to modify an aglet's lifecycle and duplicates the Create, Dialog, AgletInfo and dispose clickable buttons that are present on the server panel as well as methods to kill an aglet (i.e. dispose of it, overriding the ondisposal method) and to shutdown the server.
- **Mobilty:** This menu allows the user to send requests to an aglet to manuplate its mobility and as well as duplicating the Pispatch,Retract clickable buttons it also has option to Deativate and Activate an aglet.
- **View**: In this menu the user can open window panels which detail memory usage and also a log of the aglet's behaviour (actions) on the server. It also contains two options which are yet to be implemented: age, which presumably for getting the time that the aglet has been running in the server, and the Java Console which would be for the new Java Visual compiler
- **Options**: This menu panel allows the user to modify the setting options for Tahiti, each of which are detailed below
- **General Preferences**: This panel allows the user to modify the font of the text in the GUI, specify an aglet that is launched when the server starts up, clear the Class cache and to modify their User information which is initially inputted in the registration panel.
- **Network Preferences**: Allows the user to specify a HTTP proxy through their aglets can be launched if their network uses a firewall.
- **Security Preferences**: This panel is used by the user to specify the security privileges for trusted and untrusted aglets on the File System, Network Access,

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

Properties and others.

Related Work

John K. Ousterhout presents Safe Code Interpretation. In safe Code Interpretation Agent systems are often developed using an interpreted script or programming language. The main motivation for doing this is to support agent platforms on heterogeneous computer systems. The idea behind Safe Code Interpretation is that commands considered harmful can be either made safe for or denied to an agent.

John K. Ousterhout, Jacob Y. Levy, and Brent B. Welchbased suggested Safe Tcl System which was used in the early development of the Agent Tcl system. Safe Tcl employs a padded cell concept, whereby a second "safe" interpreter pre-screens any harmful commands from being executed by the main Tcl interpreter. The term padded cell refers to this isolation and access control technique, which provides the foundation for implementing the reference monitor concept.

Günter Karjoth, Danny B. Lange, and Mitsuru Oshima presented Signed Code technique. It is a fundamental technique for protecting an agent system or other objects with a digital signature. A digital signature serves as a means of confirming the authenticity of an object, its origin, and its integrity. The author of the agent either an individual or organization, may use a digital signature to indicate who produced the code, but not to guarantee that the agent performs without fault or error.

William Farmer, Joshua Guttman, and Vipin Swarup State Appraisal suggested State Appraisal technique. The goal of State Appraisal is to ensure that an agent has not been somehow subverted due to alterations of its state information. It is not clear how well the theory will hold up in practice, since the state space for an agent could be quite large, appraisal functions for obvious attacks may be easily formulated.

G. Necula and P. Lee presented Proof Carrying Code technique. Proof Carrying Code is a prevention technique, while code signing is an authenticity and identification technique used to detect, but not prevent the execution of unsafe code. They include a standard formalism for establishing security policy, automated assistance for the generation of proofs. This technique is tied to the hardware and operating environment of the code consumer, which may limit its applicability.

Karjoth and his associates devised a platform oriented technique for encapsulating partial results, which reformulated and improved on the PRAC technique. The approach is to construct a chain of encapsulated results that binds each result entry to all previous entries and to the identity of the subsequent platform to be visited.

Vigna presented an approach that allows a mobile agent owner (under certain assumptions1) to detect any possible attempt to tamper with agent data, code, and execution flow. The proposed mechanism does not require dedicated tamperproof hardware or trust between parties, both advantageous when designing a generic solution for mitigating part of the malicious host platform problem.

Haiyan Che, Dali Li, Jigui Sun, and Haibo Yu advocated a novel understanding and definition of mobile agent: a data package describing the tasks user required and proposed the security architecture of TDBMA system. In the TDBMA system, the task-description-based mobile agents are used to behave on behalf of users. The Proxy/Manage Agency is responsible to create and dispatch agents to avoid any individuals sending agents.

Levent Ertaul, Jayalalitha Panda, discuss the implementation of two of the security approaches called Mixed Multiplicative Homomorphic Encryption scheme and Secure Dynamic Programming. These security approaches protect the mobile agents from malicious agent platforms. It also discusses our agent integrity checking mechanism that is implemented using SHA1 digest algorithm. These implementations are done in the IBM's JAVA Mobile agent system called Aglets and provide Confidentiality and Integrity services to the mobile agents.

Yang Kun, Guo Xin, Liu Dayou investigates the problems & approaches of Mobile agent system, which show that bi-directional & layered security model, may be a good idea to resolve the security problems in mobile agent systems. The main security problems faced by mobile agents, and propose bi-directional & layered security model to resolve them in horizontal direction and vertical direction respectively.

IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

Objective

•Propose to develop a Mobile Agent System which tends to provide security features that are specific to mobile agents but independent of any particular networked operating system.

•To perform Encryption algorithm that can be used to ensure that mobile agent and its data do not become compromised. To perform Decryption algorithm to acquire data in original perform.

•Comparative study of different algorithm such as MD5, SHA etc.

Conclusion

Mobile agents can be viewed as an alternative of the traditional client-server paradigm. While the client-server paradigm relies on remote procedure calls across a network, mobile agents can migrate to the desired communication peer and take advantage of local interactions. The mobile agent paradigm is often regarded as a replacement of the client-server paradigm but a mobile agent based system can be viewed as an extension of distributed client-server system. The most relevant design paradigms for current systems are Client-Server, Remote Evaluation, Code on Demand, and Mobile Agent. In the Mobile Agent paradigm the know-how and whole component are moved to the remote location and this transferred component executes this code. Mobile agents are autonomous software entities, which can migrate through a network of heterogeneous sites to perform tasks on behalf of their owners.

References

[1] Stamatis Karnouskos. “A Security Oriented Architectural Approach for Mobile Agent Systems”.
[2] R. Haghighat far, H. Yarahmadi. “A New Approach for Mobile Agent Security” World Academy of Science, Engineering and Technology 42 2008.
[3] Wayne Jansen, Tom Karygiannis. “NIST Special Publication 800-19 – Mobile Agent Security” National Institute of Standards and Technology Computer Security Division Gaithersburg, MD 20899.
[4] Levent Ertaul Jayalalitha Panda. “Mobile Agent Security” California State University, East Bay, Hayward, CA, USA.
[5] Marco Tranquillin, Carlo Ferrari, Michele Moro. “Using mobile agents for secure biometric Authentication “The University of Padova, Padova Ital.
[6] Harmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath . An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions.
[7] Nikola Mitrovi´c, Unai Arronategui Arribalzaga. “Mobile Agent security using Proxy agents and Trusted domains” University of Zaragoza, Maria de Luna 350018 Zaragoza, Spain.
[8] V. S. Shankar Sriram1, G. Sahoo2. A Mobile Agent Based Architecture for Securing WLANs.
[9] A Survey of Mobile Agent Systems by Syed Adnan, John Datuin, Pavana Yalamanchili.
[10] Yang Kun, Guo Xin, Liu Dayou. “Security in Mobile Agent System: Problems and Approaches”.
[11] Haiyan Che, Dali Li, Jigui Sun, and Haibo Yu. “A Novel Solution of Mobile Agent Security: Task-Description-Based Mobile Agent”.