available at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/cose
Computers & Security

Security for a Multi-Agent System based on JADE

X. Vila*, A. Schuster, A. Riera

Department of Electronics and Computer Science, University of Santiago de Compostela, Campus Universitario s/n 15782 Santiago de Compostela, Spain

ARTICLE INFO

Article history:
Received 6 September 2006
Accepted 11 December 2006

Keywords:
Multi-agent
JADE
JADE-S
Learning management
Authentication
Authorization
Encryption
Signature
Availability

ABSTRACT

The present paper explores the challenges, issues and solutions to satisfy the security requirements of a Multi-Agent System (MAS) based on the JADE framework. By means of a prototype system used for Learning Management, an adequate security concept for MAS in general is presented. Hereby several security features are considered, ranging, among others, from the authentication of users over encryption of the exchanged data up to the authorization of the access to services designated only to a determined group of users.

1. Introduction

The agent concept originated in the area of Artificial Intelligence. Nowadays, an agent can be considered software components that are capable of acting with a certain degree of autonomy, reactivity and pro-activeness in order to accomplish tasks on behalf of its user (Wooldridge, 2002).

Multi-Agent Systems (MAS) arise when several agents are grouped together in a single system. While the single agents still aim at their own local tasks, a global object normally is followed by the MAS through the communication between the system agents. Thus a Multi-Agent System can manifest self-organization and complex behaviours even when the individual strategies and capabilities of all their agents are simple.

For the development of such distributed Multi-Agent Systems, the JADE Framework (Java Agent Development Framework) offers a Java middleware based on a peer-to-peer architecture (JADE) with the overall aim to provide a runtime support for agents. To guarantee interoperability between agents, JADE is compliant with the FIPA specifications (FIPA). The Foundation for Intelligent Physical Agents (FIPA) is an international non-profit association of companies and organizations sharing the goal and the effort to produce standard specifications for agent technology.

Although the agent paradigm appears promising, it is important to dedicate part of the research to the security issues that need to be addressed before the resulting system can be presented as a viable solution in real scenarios. Otherwise, data exchanged between agents could be spied, modified

* *Corresponding author*. Tel.: +34 981 563100x13565; fax: +34 981 528012.
E-mail addresses: vila@dec.usc.es (X. Vila), alx.s@web.de (A. Schuster), eladolfo@usc.es (A. Riera).
0167-4048/$ – see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2006.12.003

or denied. With an adequate security concept, however, it can be guaranteed that the Multi-Agent System acts in an expected way in all scenarios.

Considering an Intelligent Learning Management System, called EUME, as an example, this paper explores the challenges of security in an MAS and then presents a security concept handling these issues.

Section 2 constitutes an overview of this exemplary Multi-Agent System (EUME). In Section 3 the JADE-S add-on will be presented to familiarize the reader with the possibilities offered by JADE to secure the MAS. Setting up on this information, Section 4 provides a study of security requirements and presents a concept to solve them.

Finally, a description of the planned future work is provided based on the results presented in this paper.

2. The EUME system

This paper presents a security concept by means of an exemplary Multi-Agent System. The system, called EUME (Spanish acronym for Multimedia Ubiquitous Environment for Education), was implemented at the University of Santiago de Compostela in order to improve the efficiency of the classic learning methodologies (Vila et al., 2004; EUME).

Currently installed and tested in a classroom, EUME is equipped with a principal server where its services are executed; a central database, where all information is saved; a classroom server, situated in and managing the resources of the classroom (projectors, cameras, applications, electronic blackboard, etc.); and finally PDAs used as Graphical User Interfaces for the users.

In the resulting EUME platform the agents are grouped according to the function of their tasks in three well differentiated layers:

- *Client layer*: agents that provide the Graphical User Interfaces (GUIs) that are necessary for the users to interact with the system.
- *Service layer*: contains agents that provide high level services for clients by managing the resources.
- *Resource layer*: contains agents that manage directly the resources.

The distributed nature of the system is reached by locating each layer on a different network host, thus providing its tasks as a loosely and transparently coupled platform managed by the JADE framework. The service layer is entirely installed on the centralized principal server and is incorporated by two Agent Containers and their agents (one container executing the teacher and one the student services). The resource layer on the other hand is distributed among the classroom servers each managing the resources of its classroom and being incorporated by an Agent Container and its agents, respectively. The central database is also located on an own network host and managed by another Agent Container and its agent. The client layer is likewise distributed over the users' PDAs each executing an own Agent Container and its agents as shown in Fig. 1.

3. JADE-S and IMTPoverSSL

JADE-S (Version 2) (JADE, 2005) is a plug-in for the JADE framework providing some security features to the platform. The current version 2 of the security plug-in replaces completely the anterior version 1. Extending the Java security model based on a customizable "sandbox" (Sun Developer Network) it uses the following security features to comply the needs for security of a Multi-Agent System:

- User authentication (authentication);
- Authorization of the actions performed by agents (authorization);

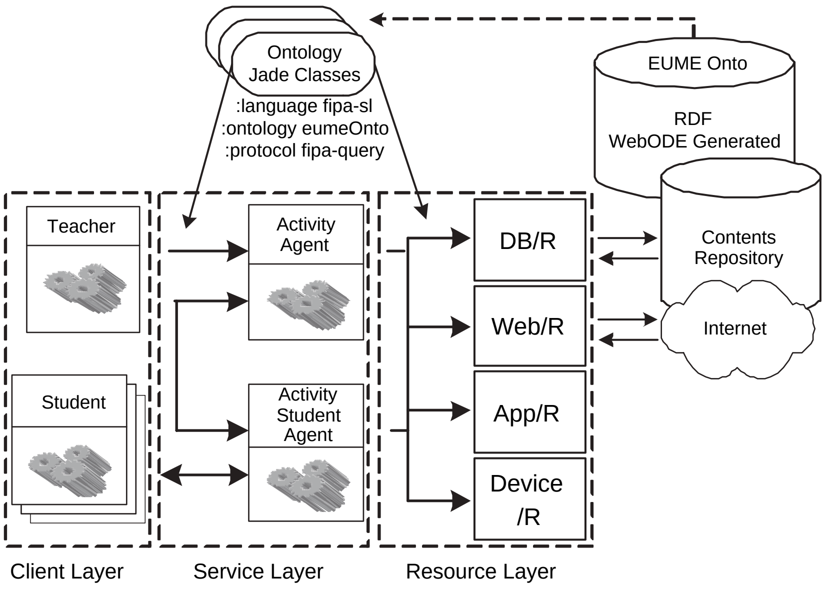

Fig. 1 – EUME architecture.

- Message signature (data integrity, non-repudiation);
- Encryption (confidentiality).

The JADE-S plug-in does this by providing four JADE kernel services (Care, 2005), one for each point:

- Security Service (authentication);
- Permission Service (authorization);
- Signature Service (message signature); and
- Encryption Service (message encryption).

Beside the obligatory Security Service as a base service to run JADE-S, the other ones can be executed independently of each other.

The Agent Containers in a JADE platform normally are distributed over different network hosts. In order to dominate security in such an open and distributed environment JADE-S introduces the concept of a *multi-user system* where all components (agents, containers) in a platform are owned by authenticated users, who in turn are authorized by the platform administrator to perform only certain privileged actions. Moreover, each agent owns a public and private key pair by means of which it can sign and encrypt messages.

A basic scheme of secure agent platform using JADE-S can be seen in Fig. 2.

The Main Container has access to a password store, like a local password file or a remote LDAP server, and is in possession of a global platform wide permission file. Whereas, a normal Agent Container only has his local permission file. Each agent holds an asymmetric key pair (public and private key) and a certificate to attest his identity. This identity certificate, containing among others the agents principal name and owner, is digitally signed by the AMS agent (Agent Management System) which acts like a Certification Authority to guarantee the certificate's authenticity. Containers and Agents are owned by a user.

Below the single services that are offered by JADE-S are explained more in detail beginning with authentication and authorization/permission up to signature and encryption.

3.1. *Authentication with JADE-S*

In JADE-S authentication is carried out by the Security Service and is the base of the plug-in and all its other services. As explained above each component of the platform belongs to an authenticated user. Based on the JAAS API (Sun Developer Network, JAAS) of Java the authentication system is composed of two elements: a *CallbackHandler* (like a dialog box) that allows the user to provide his username and password and a *LoginModule* that checks against the password store if the username and password are valid.

A set of standard LoginModules is supported by JADE-S: the *Unix* and *WindowsNT* (both checking against the current user of the corresponding OS session), the *Kerberos* (checking against a Kerberos server) and *Simple* (checking against a local plaintext password file) login modules.

Each user starting a container and its agents has to authenticate itself thus owning this container and its agents. This implicates that the user who boots the platform, by starting the Main Container, also owns this one, the AMS and the DF (Directory Facilitator). Over the CallbackHandler he provides his username and password locally to his container which sends this information to the Main Container which on its part checks them with a LoginModule against the corresponding password store. The authenticated username from now on is used as owner name of the user's components (agents, container). If the authentication is successful the Agent Container and his agents may join the platform. If any problem occurs during authentication, or the user fails to be correctly authenticated, then the local JADE system will exit and generate an appropriate error message.

3.2. *Authorization with JADE-S*

Thanks to the authentication mechanism, which assigns agents to a user, the Permission Service of JADE-S is able to provide authorization for the multi-user JADE system.

All actions that authenticated users or their principals can perform in the platform can be permitted or denied according to a set of rules (Access Control List). This Access Control List is saved in a policy file, usually named *policy.txt*, that follows the default Java/JAAS syntax to define permissions that configure the java security's sandbox accordingly. Every action that is not explicitly allowed in this policy file will be denied to the agents.

Beside the permissions provided by Java for the access to local resources (JVM, network, file system, etc.) an extended policy model was implemented in JADE-S thus providing

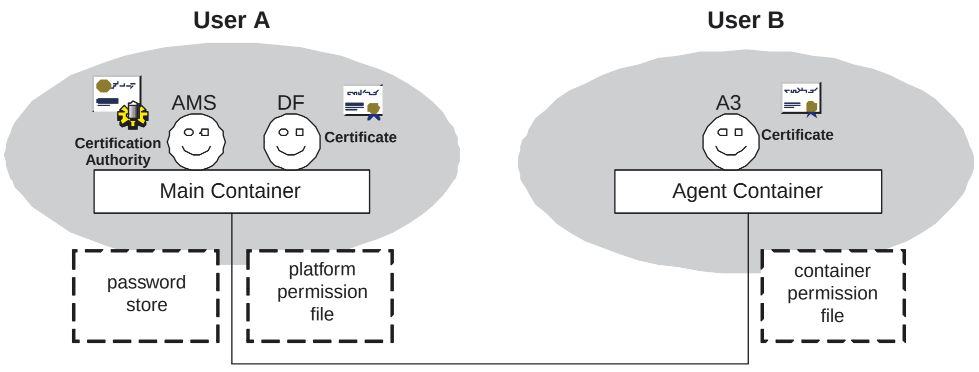

Fig. 2 – JADE-S architecture.

a greater flexibility for a distributed Multi-Agent System. In this extended JADE-S model authenticated users or their principals can be granted the following rights in the policy file:

- *Platform Permission*: right to create/kill a Main Container;
- *Container Permission*: right to create/kill an Agent Container;
- *Agent Permission*: right to create/kill an agent in a container;
- *AMS Permission*: right to register/deregister agents in the AMS;
- *Message Permission*: right to send messages to other agents.

The Main Container is in possession of a platform policy file specifying platform-wide permissions while each other container is in possession of a local policy file specifying corresponding local container. Rights given to a user are transmitted automatically to all his owned agents.

When coming to the point to enforce the access rights during the platform execution the signed identity certificates held by each agent are used to identify the agents name and owners and to apply the rules to them.

3.3. *Signature and Encryption with JADE-S*

When sending an ACL messages, both to an agent running on the same or a foreign platform, signature guarantees the data integrity and non-repudiation whereas encryption guarantees confidentiality.

In JADE-S signature and encryption are enabled by the fact that each agent holds an asymmetric private and public key pair. The Signature Service, applying its signing/verifying operation, and the Encryption Service, applying its encrypt/decrypt operation, always operate on the whole payload in order to protect all the sensible data of the ACL message. The security-related information for the Signature Service, as the public key, signing algorithm and message's signature, thereafter, is placed into the envelope.

Agents only have to request the signature or encryption of a message, by calling the corresponding methods of their *SecurityHelper* class (i.e. setUseSignature(msg)), before sending it and do not need to deal with decryption and verification of a signature. This is done transparently in the corresponding service when proceeding a received message in the incoming filter chain. If some error occur, decrypting or verifying a signature, the message is discarded automatically and a FAILURE ACL message is returned. This implicates that each time an agent receives a signed and/or previously encrypted message the signature and/or encryption was already checked and/or decrypted at a lower level and is valid.

3.4. *IMTPoverSSL*

Apart from JADE-S, another security concept for JADE is presented in the paper. This concept provides confidentiality, data integrity and mutually authenticated connections amongst JADE Containers by running its Internal Message Transport Protocol (IMTP) over TLS/SSL (Vitaglione, 2004), i.e. RMI over SSL. This container-to-container based structure is different from the agent-to-agent based one of the JADE-S plug-in. Each container is endowed with a certificate, holding among others his public key. Additionally, all certificates of all other containers of the platform are held in the trust store of each container.

The mutual authentication, encryption and signature then is performed by the TLS/SSL protocol (Dierks and Allen, 1999): each container presents his own certificate to the communication partner and looks if the one presented to it is saved in its trust store. Successfully passed that phase the TLS/SSL protocol continues with encryption and signature of all information exchanged between the containers.

This mechanism can be used independently of the JADE-S plug-in.

4. A security concept for MAS

In spite of its functionality of providing authentication, authorization, message integrity, non-repudiation and confidentiality, it has to be admitted that JADE-S is still in an early phase of development. That is why it suffers from security shortcomings or does not cover all necessary features to secure an MAS entirely as necessary. By means of these shortcomings and the additional necessary features now a complete security concept for a Multi-Agent System will be introduced in this section.

In order not to reinvent the wheel, however, it was decided to use one of the two presented concepts in Section 3 as a base for security of the system. In the end the selection fell in favour of JADE-S and not of the IMTP-SSL concept for the following reasons. In the first place, IMTP-SSL does not provide authorization, which may result very important in some systems. Secondly, it does not provide an agent aware security by applying its security features at a container level thus authenticating, encrypting and signing all information exchanged between containers. This fact, however, does not enable the agents to determine autonomously the grade of security they desire, based on the message destination for example. Finally, IMTP-SSL is a very rigid concept that assumes that all containers have to hold the certificates of all others with which they want to communicate, thereby restricting the extensibility of the MAS platform at a high level.

So JADE-S will be used and applied to provide the basic security features for the EUME system. Further additional security features, to compensate the uncovered fields and shortcomings of JADE-S, will be analyzed, singled out, implemented and installed on top of its fundamental traits. By means of our study of JADE and JADE-S these lacks could be identified and the resulting security concept will be explained and improved step by step in the next subsections.

4.1. *The authentication concept*

Using JADE-S, the Security Service as a base service always has to be started thus providing yet authentication for the system. The varieties of the possible password stores for authentication, used from the system to check the username and password, however, are kind of limited. LoginModules to check the password against the one of the user extracted from the current Operating System session of Unix or Windows on the Main Container are provided. This, however, implicates that all users need an account on that machine. Inherently an unusual behaviour for a great system beyond that the location independency is not guaranteed. Great problems occur when trying to migrate the system for example.

This is only one of the disadvantage that the system independent Kerberos LoginModule corrects by providing a secure and widely accepted infrastructure that allows centralized authentication and authorization for a heterogeneous server environment. JADE hereby could use this infrastructure to uncouple the password store from the Main Container and provide a secure authentication. Generally a good method if such an infrastructure is already existent, it is quite an overhead, however, to install and use a Kerberos server only for a single JADE based system.

Last but not least a simple plain text password file saved on the Main Container could be used but does not provide the desired security and it is advised by JADE-S to use it only for testing purposes.

Fortunately JAAS, that is utilized in JADE-S to provide authentication, offers a pluggable architecture and an API with which new LoginModules can be written and used to authenticate against every kind of password store. After a study of the available and appropriate systems that can provide authentication and how to include them into JADE-S the decision fell upon a LDAP server (OpenLDAP Foundation, 2005) as a password store. LDAP normally provides a sophisticated directory service. The crucial reasons for its election are:

- LDAP nowadays has the state of a standard mechanism for authentication with defined interfaces (accessible from java).
- It provides a high level of security by supporting different secure authentication mechanisms.
- It can act as an independent single sign on for many or only one server.
- Beside authentication it can be used in any MAS for authorization purposes or simply as a directory service storing user information.

The LDAP server will be installed on an own network host and prepared correctly for EUME. Connecting to it over the network, the Main Container is able to check the password, introduced by the user during authentication, against the password field of the corresponding user entry in the LDAP server. Therefore, an adequate LoginModule have to be implemented and added to JADE-S. Our *LDAPLoginModule* has to implement the Java interface *javax.security.auth.spi.LoginModule*. By doing this it complies the norms and can be plugged in every application using JAAS.

In order to access the LDAP server from this module, we can use an unified interface provided by Java for multiple naming and directory services, called *JNDI* (Java Naming and Directory Interface) (Sun Developer Network, JNDI). Using methods of this standard we can access the LDAP server, authenticate with an authentication mechanism provided by the server and check if the user password is correct. To guarantee security we chose *DIGEST MD5* (Leach and Newman, 2000) as a secure authentication mechanism and encrypt the connection between Main Container and LDAP server by using TLS.

4.2. *The authorization concept and an architecture*

Having the authentication running, the JADE-S Permission Service can be used to provide a basic low level authorization for agent, container and platform creation as well as communication between agents. Furthermore, it is necessary, like in the case of authentication, to introduce additional features that warrant an all-embracing security.

The examination of JADE and JADE-S brought these basic shortcomings in the field of authorization to the surface.

The Message Permissions are quite generic and limited by only providing the possibility to determine the permission to send and receive messages to/from owners or agents. Finer grained access to specific functionality provided by a special agent cannot be defined, however. Either it is permitted to send a message to an agent and thus having the possibility to access (in an autonomous way) the whole agent's functionality or it is not permitted to send to it at all.

As an example the interaction with the *DF* could be mentioned. If it is permitted to an owner/agent to send to the DF then automatically all of its provided actions regarding the yellow page service are accessible: register or deregister a service description of a service provided by an agent, modify a registered service description or search for agents that meet a given constraint (i.e. that offer a certain service).

JADE-S provides also user/agent based permissions but it is not possible to determine own user specific permissions to regulate the access to services provided by agents.

In the case of a Learning Management System for example, it is important to distinguish between services provided for teacher agents that have to be accessible only from teachers and services provided for students that should be accessible only by students (but also could be by teachers for example). To clarify this in the context of EUME Fig. 1 can be consulted.

Due to these insufficiencies, an architectural change of our MAS was conceived, tested in a prototype and implemented finally in the system. On its base stands the division of agents of the platform in two categories: *system agents* and *client agents*.

The *system agents* consist of all agents of the resource layer (i.e. projector agent or database agent), service layer (i.e. test service agent or presentation service agent) and Mediator layer (explained afterwards). These agents were implemented by the EUME programmers, tested in a trusted environment and their source code is physically unaccessible and unchangeable by others. In this way they do exactly what we want them to do if they are not influenced maliciously by non-system agents (i.e. killed or modified by others) during runtime. The system agents will not have any constraint on the actions they perform regarding agent creation as well as the two Main Container service agents, thus receiving the full access to the AMS (deregister, register, modify, search, send-to) and the DF (deregister, register, modify, search, send-to).

On the other hand, the *client agents* consist of all agents of the client layer. These are agents that consume or manage the learning services provided by the MAS. In the actual development state the client agents either are executed by teachers or students. Even if Graphical User Interfaces are provided by EUME for the clients in order to interact with the system, we do not have any control over these agents. The source code could be changed or newly programmed to perform malicious and dangerous actions (i.e. deregister or kill other agents). That is why it was decided to deny them the access to the DF at all and only allow them to register, deregister and create agents in their own container running on their PDA.

By reason of this categorization of agents it is possible to overcome the lack of authorization left by JADE-S. This will be done with a change of the system's architecture. Therefore, an own layer will be introduced between the client layer (''bad agents'') and the service layer (''good agents''): a Mediator that acts as security check. The resulting security architecture for EUME is shown in Fig. 3.

Client agents are negated to communicate directly with the DF at all and in case of the AMS only are allowed to act for their owned agents running on their container (deregister, register, etc.).

Forced in the same way, with Messaging Permissions of the JADE-S Permission Service, the client agents only are allowed to send messages to the Mediator agent(s). To have their needs processed a client now requests a service from the Mediator instead of directly to the service layer agents. The Mediator then performs a high level permission check of the sender's access rights and, thereafter, forwards the messages to the destination if the access is granted or drops the message and returns a FAILURE message to the sender otherwise. For performance reasons several Mediator agents can be launched to distribute the work among them.

In this way, the following two level authorization is implemented in the system:

- *low level authorization*: provided by JADE-S; offering basic permissions like agent, container and platform creation; enabling the new layered security architecture with convenient message permissions thus enforcing the communication through the Mediator.
- *high level authorization*: provided by the Mediator; offering solution for partial access to services provided by agents (i.e. DF) as well as the high level permission check to access group services (i.e. for teacher).

To realize this high level authorization the Mediator provides two different behaviours up to now: a *regulated access to the DF* and a *high level permission check*. As client agents may need to access the DF, to register a new service description of a service they provide or search for a service they want to use, it is necessary to make these features available to them. As the direct communication with the DF is not allowed any more for the clients, the Mediator agent, as a system agent, has to provide a regulated access to the DF. When a client for example wants to know which agents provide a specific service, it sends a search request with the corresponding service description to the Mediator. This one on his part checks if the client has access to this DF functionality. If so it starts an adequate request to the DF and then returns to the client the agent identifiers (AID) of the agents that provide the desired service.

The second point, regarding authorization implemented by the Mediator, is the *high level permission check*. Hereby, it is considered to obtain an access control to teacher and student services. In the actual development state the client agents either can be grouped into teachers, authorized to access teacher services, or students, authorized to access student services. The teachers and students have to authenticate themselves as many different users (i.e. myTeacher) thus owning their agents (i.e. UIpresentationControlAgent has the owner myTeacher). In order to distinguish the requesting agents according to their

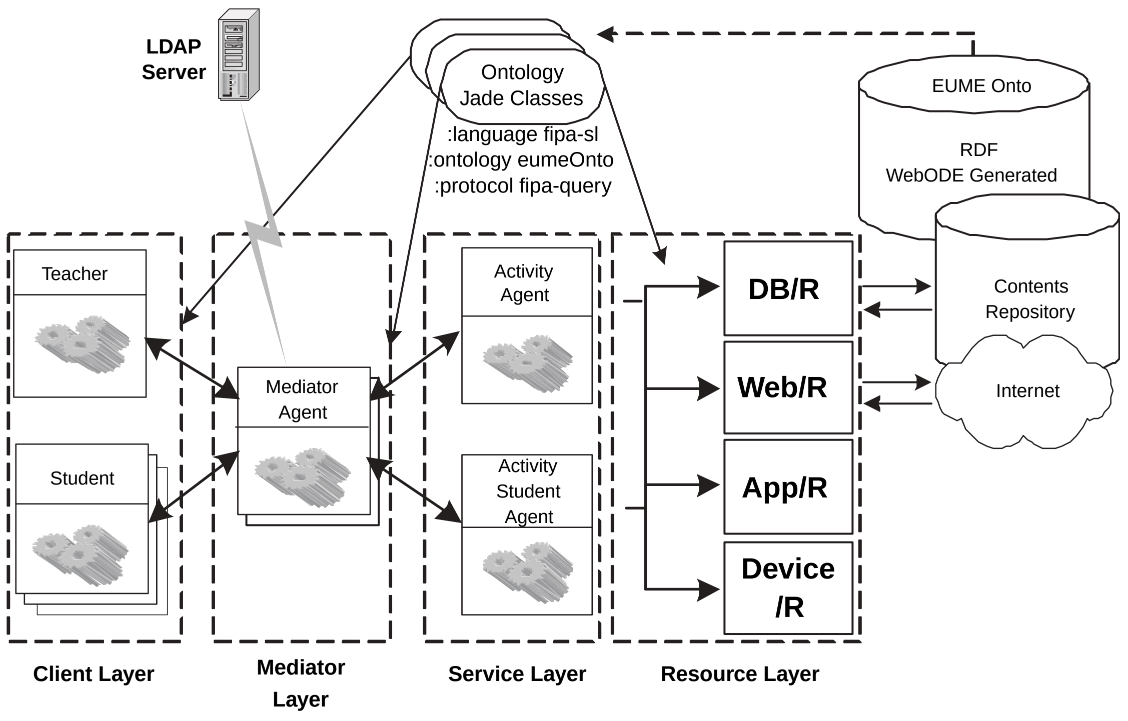

Fig. 3 – Security architecture of EUME with a Mediator.

group it is necessary for the Mediator to relate their owner to one of the groups (i.e. myBadestStudent appertain to the group of students). Therefore, the Mediator consults the LDAP server, also used for authentication, over a secure channel. In this directory the group, to which a user belongs, is saved in the corresponding user entry. Based on predefined authorization rules and equipped with the group information, the Mediator can control the access to the service layer.

It has to be append that *high level authorization exceptions* can be defined with which a professor can enable a specific student to access temporally some well specified teacher services (i.e. myTeacher grants to myBestStudent the rights to access the presentation service and lets the student explain some of the teacher's PowerPoint slides).

With a *MessageTemplate* it is possible for the Mediator to differentiate the aim of the received ACL message thus able to distinguish if the agent needs a regulated DF request or wants to access some services on the service layer.

Thanks to the extensibility of the concept it will be possible in the future to amply the Mediator's functionality and add further behaviours that provide partial access to services of other agents or adapt it to further services prepared for other groups.

With the JADE-S Permission Service it is also possible to assure the desired layer based architecture of EUME: Clients communicate, over the Mediator, with their services on the service layer. The service agents on their part access the resource agents to accomplish their tasks. With the Message Permissions this architecture now can be enforced entirely by controlling that agents only can send messages to other agents that are located on a neighbouring layer.

4.3. *Encryption and signature*

Up to this point data integrity, non-repudiation and confidentiality are not considered. The first two features can be guaranteed with a signature of a message digest of the exchanged messages whereat the last feature can be assured with message encryption. JADE-S, therefore, provides the Encryption and the Signature Service. Both of these kernel services utilize the asymmetric key pair that JADE-S creates for each agent.

Generally, it is better that an agent is aware of security thus being capable to determine autonomously the grade of security it desires, based on the message destination for example. With the help of the Encryption and Signature Service, such an agent security awareness is given based on the fact that an agent has to enable the encryption and/or signature manually by itself for each message it sends.

Unfortunately, the studying of JADE-S and sniffing of the exchanged traffic between containers showed another shortcoming of JADE-S: the *Encryption Service* cannot guarantee a secure channel for authentication and platform managing commands.

With the network protocol analyzer Ethereal, we proved that passwords are sent over the network in plaintext to the Main Container, for authentication purposes, and cannot be encrypted by the Encryption Service. This results from the fact that the Encryption Service is restricted to encrypt only ACL messages but platform managing commands (i.e. join new container, create agent) are ignored. Having a password sent in plaintext over the network it is easy to spy and then incorporate the user's identity. This is a security lack that cannot be tolerated. Therefore, the Encryption Service will not be used and an alternative have to be found.

A complete new encryption service based on certificates could be mounted. In this case, however, emerges the same problem as when trying to adapt the existing JADE-S Encryption Service: it is necessary to penetrate in a closed framework and change lines of code to perhaps be able to encrypt also platform commands. This intrusion is not desired.

So our attention turned on the IMTPoverSSL concept, presented in Section 3.4, that runs the Internal Message Transport Protocol, RMI in the case of JADE and JICP in the case of LEAP (see Section 4.4), over a SSL secured connection to exchange information between containers. Thus the two implementations RMIoverSSL for JADE and JICPoverSSL for LEAP, provide confidentiality and data integrity for all exchanged information (even platform managing commands). The undesired characteristic of these implementations is the mutual container authentication, which takes place before the secure connection is established. The communicating containers authenticate each other through certificates. Therefore, all containers have to possess the certificates of the containers they want to communicate with. These certificates are saved, respectively, in a java trust store. Applying that to our Multi-Agent System this involves that each client (teacher or student) needs a certificate of its own. Furthermore, when being issued to a client, the certificate has to be imported in all trust stores of the client's communication partners: in the case of the current EUME architecture this would be the Mediator and Main Container.

Nevertheless, for our MAS we need to follow a protocol with which only the system's Mediator and Main Container authenticate themselves to the user clients. Using such a SSL based protocol, instead of one with mutual authentication, the advantage of authenticating the Main Container and Mediator remains, but on the other hand, the undesired client authentication may disappear. That is why we have decided to develop a new IMTPoverSSL module for EUME. The new module is based on the existing IMTPoverSSL implementation but, before the information is encrypted, it processes only a single-sided authentication instead of a mutual one. Such a decision involves that, in the EUME system, only the Mediator and the Main Container need certificates to be authenticated to the clients.

Regarding the JADE-S *Signature Service* it provides some additional security on top of the encryption presented above. Unlike the container-to-container based procedure of the IMTPoverSSL concept it provides an agent-to-agent signature that ensures that each message arriving at an agent really comes from the claimed sender that is specified in the message. Otherwise, agents could fake the sender's identity of a message without the possibility to detect it.

Resumed shortly the mechanism and their functionality introduced in this section are the following:

- *single-side authenticated SSL*: provides encryption and data integrity as well as the security to really communicate with the Main Container and Mediator.

- *signature*: provides data integrity and security that received messages really are sent by the sender.

4.4. *Availability*

When searching in *Wikipedia* (Wikimedia Foundation) for Availability in information systems the following statement can be found: "The first goal of modern information security has, in effect, become to ensure that systems are predictably dependable in the face of all sorts of malice, and particularly in the face of denial of service attacks." To provide such a timely dependable system we will take care of the availability in this section.

In an open system these dreaded DoS (denial of service) attacks partially can be prevented with a *firewall*. This one has the basic task of controlling traffic between different zones of trust, i.e. protect our local trusted network of the university from the untrusted Internet and let only pass traffic that is allowed by a predetermined security policy. The common modus operandi to control the network traffic is to close all communication protocols and ports between the two networks. Thereafter, only some specified protocols (i.e. TCP) and corresponding ports (i.e. 80 for http) are opened for information exchange. If the attacks are unsophisticated there might be a specific signature to the traffic. A careful examination of captured packets may reveal a trait on which the firewall rules can be based thus preventing damage. The installation and configuration of a firewall, however, go beyond the aim of this paper and a well configured and running firewall, protecting our system of attacks from insecure networks, will be presumed in the following.

With the distributed Multi-Agent architecture used in EUME, it is designed to allow the access to the system from an arbitrary network host, inside or outside the trusted network of the university. To maintain this functionality of outside reachability of an Multi-Agent System and enable students and professors to connect themselves to the platform for example from their home, the platform's hosts outside of the trusted network have to be able to traverse the firewall. This, however, is not possible when using JADE. By virtue of the utilized *Internal MessageTransport Protocol* Java RMI (Remote Method Invocation; Sun Developer Network, Java remote method inovation) the firewall cannot be crossed and thus all type of communication between these containers is denied.

A study of the RMI protocol and tests, to confirm the result, showed that it uses a dynamic port selection to transfer information from one Agent Container to another. This dynamic behaviour, however, does not conform to the few static ports that are opened in the firewall thus impeding RMI to traverse it.

That is why an alternative had to be found and after some research the solution to replace JADE with *JADE-LEAP* (Light Extensible Agent Platform) (Caire, 2005) was considered as the appropriate one. The LEAP add-on, when combined with JADE, replaces some parts of the JADE kernel forming a modified runtime environment that can be deployed on a wide range of devices varying from servers to Java enabled cell phones. Though different internally, LEAP still provides the same set of API like JADE. Furthermore, this extension of JADE implements an own Internal Message Transport Protocol, called JICP (JADE Inter Container Protocol), for information exchange between containers. In this way LEAP replaces RMI, as the platform's standard IMTP, with its own JICP protocol. Normally designed for mobile devices, that among others do not have access to RMI, LEAP allows to determine the used communication ports with the help of this proprietary protocol. Thus it is possible to enable the JADE containers to communicate over static ports which are adapted to the firewall configuration.

As the security concept for the EUME system, presented in this paper, partially relies on JADE-S, this plug-in now has to be run in combination with LEAP. Both JADE-S and LEAP in principal should work together, thanks to the service-based JADE kernel. However, as admitted by the JADE developers, code which uses the two components together has never been developed. Unfortunately, even if the contrary is stated in the JADE-S documentation, LEAP did not work with JADE-S. When the Security Service was activated, then remote containers could not join any more to the platform because of an error occurring in the framework. This, however, was not the only error that had to be eliminated by a lot of source code examination and tests until the two pieces worked properly together. In the end we accomplished it and thus availability can be guaranteed for the system. A very nice secondary effect is that JADE-S, apart from the use of LEAP for firewall traversal, now is also enabled for the actual use of LEAP, i.e. for mobiles phones.

Apart from the availability rendered possible by the firewall a *Main Container Replication* could be activated. This allows to start several replicated Main Containers for a single platform thus also providing a higher availability in case of failure, crash or network unavailability. JADE provides an own kernel service for this purpose, a good tutorial is available at JADE Board (2005) and, therefore, will not be highlighted any more in this paper.

5. The whole concept and a use case

Now, that the security concept has been developed step by step, it is time to give a short summary of all its defining features and their functionality:

- *user authentication*: provided by JADE-S + LoginModule + LDAP; authenticate the user; assign user as owner to all his components.
- *low level authorization*: provided by JADE-S; authorization of message exchange, components creation, AMS access.
- *high level authorization*: provided by Mediator; authorization of DF access and service access based on groups.
- *encryption*: provided by single-authenticated IMTPoverSSL; encryption of exchanged information; guarantees the authenticity of Main Container and Mediator.
- *signature*: provided by JADE-S; data-integrity; security that message really comes from sender.
- *availability*: provided by corrected LEAP; substitution of RMI, as standard IMTP, with JICP to cross firewalls; main

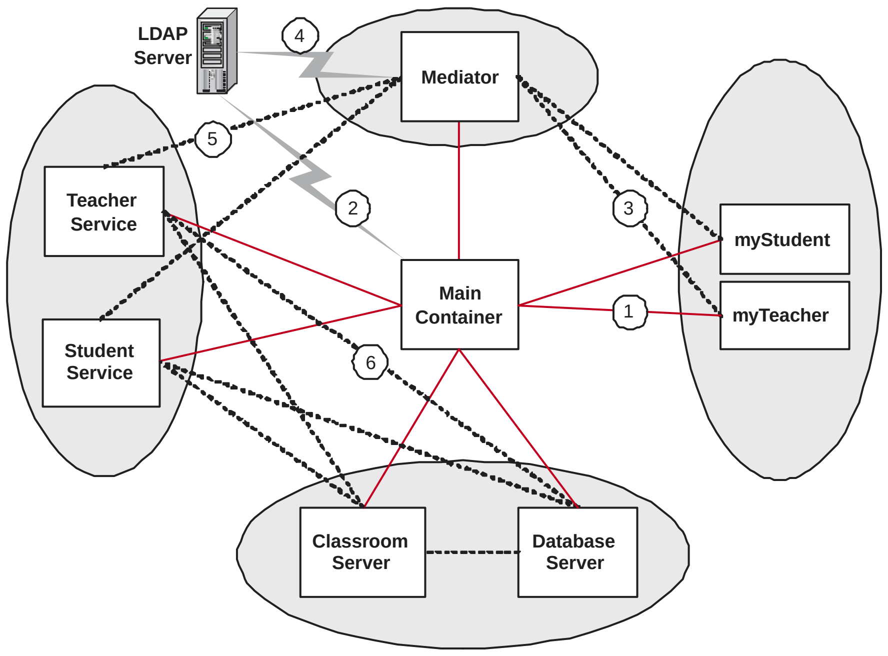

Fig. 4 – Use case.

container replication in case of failure, crash or network unavailability.

Fig. 4 shows an example that demonstrates the step by step appliance of the features of our security concept.

The structure results in a hybrid peer-to-peer architecture with the Main Container as a central node providing services for platform management. The EUME layers (resource, service, mediator, client) are marked with a grey oval. The containers are connected to the platform through the Main Container. The layered architecture can be recognized by the dotted lines that symbolize the message exchange between the Agent Containers and only exist between neighbouring layers. At the end, the LDAP server is accessed by the Main Container for authentication and by the Mediator Container for authorization purposes.

To facilitate its understanding let us assume that the professor "myTeacher" only wants to consult some information managed by one of his teacher services. Therefore, he starts EUME on his PDA with the activated Security Service. This requests a username and password of the professor and sends them over a secure channel provided by JICPoverSSL to the authenticated Main Container (Wooldridge, 2002). The Main Container checks this credentials against the LDAP password store over a secure channel provided by the LDAP LoginModule (JADE). If the authentication succeeds the teacher's container is added to the platform and he can send a service request to the authenticated Mediator. This also happens over the channel secured by JICPoverSSL (FIPA). The Mediator executes a high level permission check and controls if the teacher is allowed to access the type of service it requests. Therefore, it consults the LDAP server over a secure channel (Vila et al., 2004). In case of success the Mediator forwards the service request to the appropriate teacher service (EUME) which on his part for example may ask the database for some information (JADE, 2005) before returning the desired service to the teacher. It has to be appended that all exchanged messages are signed by JADE-S.

6. Conclusion

By means of the Intelligent Learning Management System called EUME, the paper discussed the issues required to add security to an Multi-Agent System based on JADE. First of all this paper identified a number of security requirements for such systems. Based on this account an innovative security concept was introduced to counteract the potential threads and guarantee the expected behaviour of the system on each scenario. A Java implementation of the described components was realized and integrated into the EUME system.

In order not to reinvent the wheel the JADE-S security plug-in for JADE was utilized to provide some basic security features. Using them as a base, we could design and implement a couple of additional, more complex and sophisticated security mechanisms on top. However, due to a multitude of analyzed security lacks and shortcomings of JADE-S, a great research effort had to be made to overcome its missing documentation and be able to fix the arising problems. Finally, by combining the newly designed security features with some fixed JADE-S elements, it was possible to satisfy the demands of the security concept.

Having provided these general solutions to guarantee a secure behaviour of an Intelligent Learning Management System in an open environment, we can concentrate on adding more specific features. Future work could face the problem of privacy and agent mobile security in agent-supported systems.

Privacy protection requires that each individual has the power to decide how his or her personal data is collected and used, how it is modified, and to what extend the data can be linked. Being an ample investigation area the exact requirements have to be analyzed in detail to be implemented correctly afterwards.

On the other hand, the security for mobile agent constitutes an interesting research area. As it seems impossible to find an all embracing mobile security concept without the use of trusted hardware, further research can be devoted to this part.

Acknowledgements

This work has been supported by Spanish Ministerio de Ciencia y Tecnología and Xunta de Galicia under projects TIC2003-09400-C04-03 and PGIDIT04SIN06003PR.

REFERENCES

Caire G. Light extensible agent platform user guide, Version 3.3, TILAB S.p.A, <http://jade.tilab.com/doc/LEAPUserGuide.pdf>; 2005.

Care G. The JADE services architecture. Utrecht: Jade Workshop. Available from: http://jade.tilab.com/papers/2005/JADEWorkshopAAMAS/Jade-the-services-architecture.pdf; 2005.

Dierks T, Allen C. The TLS protocol, Version 1.0, request for comments, vol. 2246; 1999. Available from: http://rfc.sunsite.dk/rfc/rfc2246.html; 1999.

Ethereal, Inc. Ethereal: a network protocol analyzer, <http://www.ethereal.com>.

EUME, Ubiquitous and Multimedia Enviroment for Education, <http://www-gsi.dec.usc.es/~eume>.

FIPA, Foundation for Intelligent Physical Agents, <http://www.fipa.org>.

JADE Board. JADE administrator's guide, Version 3.3, TILAB S.p.A, <http://jade.tilab.com/doc/administratorsguide.pdf>; 2005.

JADE Board. JADE Security Guide, Version 3.3, TILAB S.p.A, <http://jade.tilab.com/doc/SecurityAdminGuide.pdf>; 2005.

JADE, Java Agent DEvelopment Framework, <http://jade.tilab.com>.

Leach P, Newman C. Using digest authentication as a SASL mechanism, Version 2, request for comments, vol. 2831; 2000. Available from: http://rfc.sunsite.dk/rfc/rfc2831.html; 2000.

OpenLDAP Foundation. OpenLDAP Software 2.3 administrator's guide, <http://www.openldap.org/doc/admin23/>; 2005.

Sun Developer Network. Java authentication and authorization service (JAAS) reference guide, <http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html>.

Sun Developer Network. Java remote method invocation, <http://java.sun.com/products/jdk/rmi/index.jsp>.

Sun Developer Network. Java Naming and Directory Interface (JNDI), <http://java.sun.com/products/jndi/>.

Sun Developer Network. Security and the Java Platform, <http://java.sun.com/security/>.

Vila X, Riera A, Sánchez E, Lama M, Barro S. Beyond keyboard and mouse: a remote interface for a classroom management system, Conference on education multimedia, hypermedia & telecommunication (ED-MEDIA 2004). Switzerland: Lugano; 2004.

Vitaglione G. Java secure socket extension reference guide. Telecom Italia LAB. Available from: http://jade.tilab.com/doc/tutorials/SSL-IMTP/SSL-IMTP.doc; 2004.

Wikimedia Foundation. Wikipedia<http://www.wikipedia.com>.

Wooldridge M. An introduction to Multi Agent Systems. Chichester, England: John Wiley & Sons; 2002.

Dr. Xosé Vila is an Associate Professor of Computer Science. He is a member of the Department of Electronics and Computer Science in the University of Santiago de Compostela, Spain. His research interest include development of intelligent systems in education and biomedical signal processing.

Alexander Schuster is a graduate student in computer science at the Technical University of Munich. His research interests include security, Web Services and multiagent systems. He received his diploma in computer science from the Technical University of Munich.

Mr. Adolfo Riera is a consultant at GFI Informatique, an European IT service firm, and collaborator of the Department of Electronics and Computer Science in the University of Santiago de Compostela, Spain. His research interest is centered on intelligent systems in education.