Threats and security safeguards in a Multi–agent System Medical Applications

D. Cabanillas, S. Willmott and U. Cortés

Software Department, Technical University of Catalonia,
Jordi Girona 1 and 3,
08034 Barcelona, Spain
{dconrado , steve, ia}@lsi.upc.es

October 17, 2002

Abstract

This paper describes threats and security safeguards over agent applications. Areas where the security has a considerable importance and cover a wide range of applications. In our case we are focused in eHealth which is an area where terms such as privacy and trust have a great importance. We hope however that our own conclusions are applicable in the rest of the areas where security is demanded.

Key words: Security, Multi Agent Systems, FIPA, Jade.

1 Introduction

This paper is preliminary report to obtain credits research inside Ph. D. program in Artificial Intelligence , it will point out the aspects related with the security in medical scenarios with agents. Medical applications demand security guarantees due to the importance of the information, critical service and dangerous outcomes managed in this area [3].
Concretely in this paper we explain how to improve the security Multi–Agent Systems (MAS) in medical application with special emphasis on *Carrel* [2] system. We have focused our efforts on threats' classification and proposed safeguard cases based on the JADE multi–agent platform in a real scenario.

Our work is framed in Agent Platforms (APs) used in open environments as Agentcities[1]. Agentcities is a worldwide initiative to help to realize a worldwide, open network of platforms hosting diverse agent based services.

[1] http://www.agentcities.org/

Concretely we are working with *Carrel*, which is an Agent Mediated Institution for the distribution of Human Tissues.

The main objectives of *Carrel* to:

- Decrease the tissue seek time:
  - Increasing transplantation success
  - Avoiding tissue loss in Tissue Banks (TB) (due to aging for example)
- Maximize tissue compatibility
- Ensure fair distribution of tissues
- Ensure compliance with the norms and legislation regulating transplants

In *Carrel*, we have a second e–institution that represents the infrastructure and staff of a real Transplant Coordination Unit (UCTx [8]). UCTx permits the successful conclusion of an organ or tissue procurement and extraction process for transplantation. In addition, it deals with the management of the requests for pieces made by the surgeons in order to transplant them into recipients. Also we are working under a multi–agent environment is possible to include tasks such as: the coordination with the TBs about tissues reception, coordination of the surgical teams, information requests about past transplantations and so on.

Carrel and UCTx components work together to made the assignment of the tissues such as model based on a sealed envelope auction among autonomous agents with conflicting interests. The pieces are offered by TBs whereas are requested by the surgeons.

The main result obtained with this paper is described in the next list:

- Publication: Scientific document like research report. Concretely the number assigned at this document is LSI—02–76—R[2].
- Presentation: Inside of Agentcities iD3[3], the third Information Day organised by the EU IST funded Agentcities.NET project presentation sessions event in which the document has been accepted. Specifically we will show this work, in Track C and Session: Security Session with the name "Threats and security safeguards in a Multi–agent System Medical Applications".

In Section 2 we present the background of the application in order to identify its components in Section 3 we describe with more detail its elements and in Section 4 we describe the behaviour, which serves to show us the behaviour waited. In Section 5 and 6 we describe threats and security safeguards respectively. Next, Section 7 we make a list of proposals safeguards. Finally, Section 8 contains some conclusions.

[2] http://www.lsi.upc.es/dept/techreps/techreps.html
[3] http://www.agentcities.org/EUNET/ID3/

2 Background

We have a short description about the environment of our application. This section describes the *Carrel* application and highlights the areas of the system that might be attacked.

With occasion of explaining the problems in *Carrel* application we first describe its architecture to locate the different elements related in *Carrel*. Figure 1 shows the locations where the security needs to be reviewed.

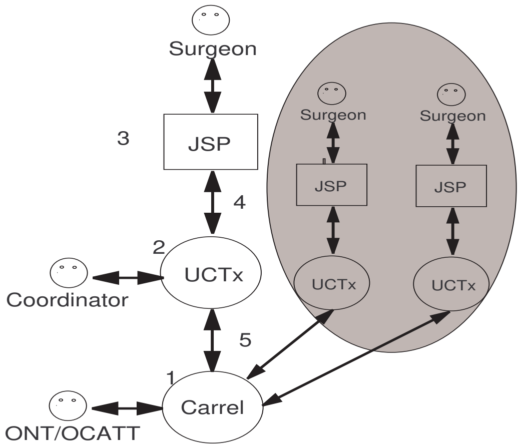

Figure 1: Important areas for security assessment

1. Application *Carrel*
2. Application UCTx
3. JSP (interface between user–UCTx)
4. Communication JSP–UCTx
5. Communication UCTx–*Carrel*

Of the 5 points, only 1, 2 and 5 will be addressed in this paper. This is because areas 3 and 4 are not related directly to agents and these can be solved with standard techniques.

For developing the MASs we have been using Jade 2.1 [5] and JSPs[4] are over Apache Tomcat 4.1.12[5]. In development phase we used one machine for MASs and another one for JSPs (to be able to work in parallel). Finally, in delivery

[4] http://java.sun.com/products/jsp/
[5] http://jakarta.apache.org/tomcat/

phase, we have JSPs and UCTx in different hospitals and *Carrel* in the Organització CATalana de Trasplantaments[6] (OCATT), the interaction among these e–institutions is realized over Internet. In summary, in this moment we have a prototype version, which will be the base for the implanted application.

3 Implementation

Now, we have a short description and notion about the architecture of *Carrel* (figure 1), we could explain the different aspects related with the points 1, 2 and 5 in the diagram.

Points 1 and 2 are very similar, for instance, technologies used in the two points are equivalent. In both of them we have worked with Jade[7] and an Oracle database[8]. The more important differences are in the functionalities and tasks of both.

In UCTx (point 2) we are talking about the "front–end" local application. The different users of the system (surgeons and coordinator) enter the system access in validated once with a password based login.

UCTx acts as the interface between the user and the agent platform, the two more important tasks are:

1. Local actions such planning the equipment with the appropriate surgeon or to consult and manage the information associated with the hospital.
2. Consultations and petitions of pieces over *Carrel*. We understand a piece such whatever sort of element that a TBs is able to keep such as organs, tissues and bones.

In *Carrel* (point 1) we are talking about the "back–end" local application. *Carrel* is designed to offer services management pieces at Hospitals (UCTx) that are members and TBs. The principal task is to assign the best piece to the recipient that shows the best match with the available pieces.

Architectures are depicted in figure 2 UCTx and in figure 3 *Carrel*. At these we should add platform–agents.

The communication between point 1 and point 2, point 5 in figure 1 is by means of public networks (hence cannot assume 100 % communication) using FIPA[9] transport mechanisms. We have different UCTx (one for each hospital) that communicate with an unique *Carrel* (in the future probably we will have a different *Carrel*, one for each administration / country). This is an important

[6]Catalan Transplant Organization
[7]http://sharon.cselt.it/projects/jade/
[8]http://www.oracle.com
[9]http://www.fipa.org

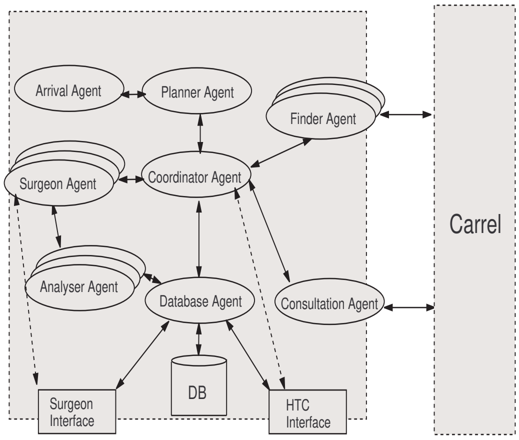

Figure 2: Architecture UCTx

risk because the information could be intercepted (and therefore read by unauthorized people or manipulated). For example, a message from a hospital may contain private information about a patient and pieces (illness, diagnosis and so on). We must assure the confidentiality of the information between these two points. Also this is a possible attack point for manipulating messages changing them and thereby impeding the correct operation of the application.

4 Behaviours

This section show the system behaviour which allows to depicted representative aspects about the system.

One brief description about an usage behaviour of *Carrel* is:

- Start the application *Carrel* and one or more UCTx at more hospitals
- In first place the surgeon fills (name and password) in the form connecting with UCTx
- This data is validated by UCTx
- Then the surgeon can makes a petition
- The UCTx associated to the surgeon sends the petition to *Carrel*
- *Carrel* confirms the authentication and starts the request
- Finally *Carrel* sends the result of the petition to the UCTx and this responds to the surgeon

In this short list we can see different aspects related with the security such as:

1. The agent interaction sequence:

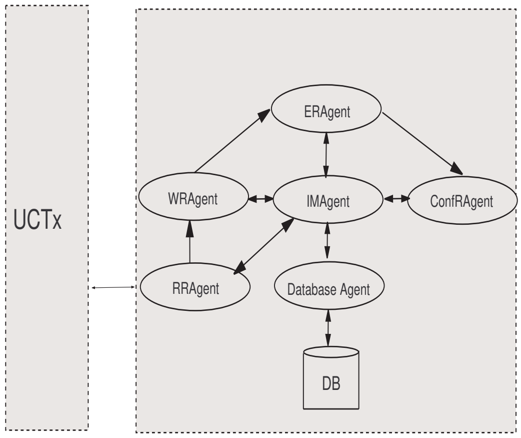

Figure 3: Architecture *Carrel*

- Inside whatever platform: Normal conditions that take place when we start an agent and it registers his services[10]. In figure 4 we describe the sequence and the agents related in this case.

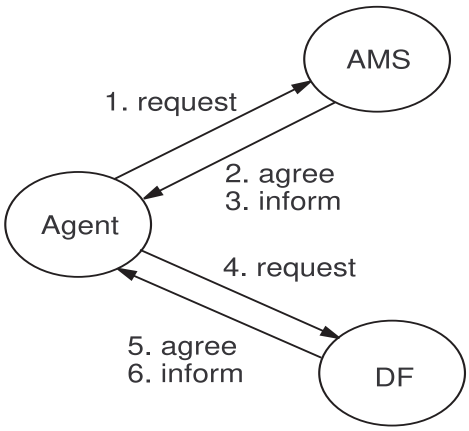

Figure 4: Start an agent

- Between platforms: This sequence (unlike with the other sequence) is specific of our application. In this case the communication is between two different platforms UCTx–*Carrel*; this is shown in figure 5.

[10]Services are the actions that our agents offers to others agents.

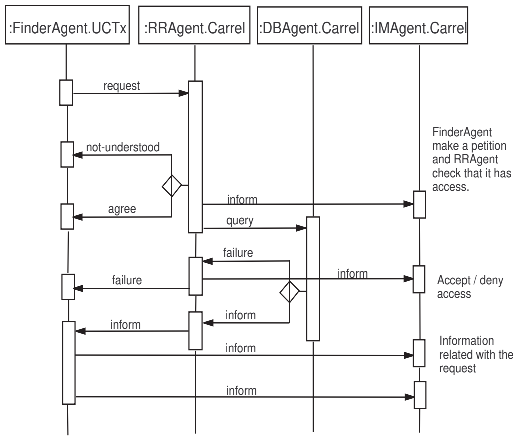

Figure 5: Communication between UCTx and *Carrel*

2. The hierarchy of users is (roles of the different users):

   In UCTx:

   - Coordinator: One unique coordinator for each UCTx. He creates the surgeon–users. This assignment is decided by hospital's members. She / He is assumed trusted by the UCTx owners.
   - Surgeon: They work with UCTx and they could make petitions or/and consultations at the system *Carrel*.

   In *Carrel*:

   - Super–user: She/He manages the *Carrel* application and database.
   - Membership: The hospitals that have access to *Carrel*.

3. Distributed applications and distributed information: On one hand, hospitals have a Jade–platform to manage the information associated (patients, users and so on). On the other hand, *Carrel* should guarantees a fair and equitable distribution of pieces available in TBs. The institution should warranty that they provide enough security about his systems information.

5 Threats

In this section, we aim is to show the different threats that might exist to the correct running of our application.

We have divided them in two groups: possible threats where we make a theoretical study and verified threats, where we demonstrate how to realize the possible threats.

5.1 Possible threats

Each of the next subsections describe possible threats, following with the idea displayed in ([6], [7]).

5.1.1 Attacks on AP

Probably these sort of threats are the most dangerous because an attack on the platform would cause the fall of all SMAs in the same one.

1. Actions: To execute managed actions on agents in AP.

Actions	
#	Description
1	Create / kill η agents. By default agents creation and destruction inside a platform is not controlled. For example, the AMS can send a kill–agent action to DF.
2	Register/ Deregister η services in DF. Anybody be able to register and deregister services. For example, that one agent registers services that in fact it does not implement.

2. Communication: To send messages that could provoke errors.

Communication	
#	Description
1	Flooding: Adding / creating useless data entries agents (to overload the system). For example, making the number of petitions increases enormously and finally crashing the application.
2	Outgrow the allowed size: The platforms have one important part to communicate with others agents and this has a buffer to save the messages receives. If the size of a message is greater than the capacity of the buffer this can be broken. For example, sending huge large content in a message with the idea to crash the buffer of the receiver.

5.1.2 Eavesdropping: Unauthorised Disclosure of Information

Eavesdropping	
#	Description
1	Disclosure: To obtain a data that is not directed towards the same one . For example, an entity eavesdrops on the interaction between agents and extracts information on the goals, plans, capabilities etc. of others agents.

5.1.3 Data resend: Unauthorised Modification of Information

Data resend	
#	Description
1	Copy and Replay: To obtain a message that not this directed towards the same one and to retransmit it. For example, a message from an agent is copied without permission and retransmitted.
2	Alteration: To modify the content of a message. For example, modifies the content of messages.
3	Spoofing and Masquerading: To take the identity from another agent, doing to believe to the rest of agents who we are the one who really we are not. For example, an agent registers as a DF and therefore receives information from other registering agents.

5.1.4 Identity theft

The meaning of term reputation is defined as "the common or general estimate of a person with respect to character or other qualities". In most of cases taking like pattern its previous actions. The risk in this case is the identity theft.

Identity theft	
#	Description
1	Identity theft: If we have an identification system and different roles for different identifications, we have confidence in this agents. A possible attack is to identity theft. For example, in a system that we have a role of super–user this is a treasure for malevolent users to obtain this identification.

5.1.5 Application level threats

We understand application level such as threats and problems apply to application layer directly i.e. that are application specific. These problems are often related with rules, norms and policies.

We realize that this type of threat are outside the subjects which they are wanted to touch in this paper, other papers ([4], [1]) explain with more detail these aspects.

5.2 Verified threats

In the next list, we describe from the possible threats we verified as they are possible. All verifications are realized over Jade.

5.2.1 Attacks on AP

1. Actions: The environment to make these different examples is shown in figure 6.

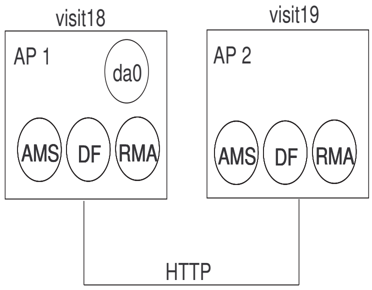

Figure 6: Environment in actions

Example 1: In this example, the agent da0 sends a messages to kill RMA in visit19. In the moment that agent da0 sends this sort of message the receptor dies. In this case as the target of the message is RMA the interface of it disappear. A similar answer we will receive if you try to send this message at the rest of the agents in your platform or in other platform. This means that we can arbitrarily kill any agent (AMS, RMA, DF or other agent) always that it had a standard behaviour when the kill–agent action is received.

The different surroundings in which it has been proven, are the following ones:

(a) At home: Two computers connected by means of port USB. In both we are executing one platform JADE. From one of them, a kill-agent action is sent towards agent RMA of the other. The result, the disappearance of the interfaces of the platform that has received kill-agent.
(b) In Agentcities: In Agentcities[11] you can obtain the list of the platforms that are correctly registered with the agentcities.net site. When the platform is correctly working you can read up in status column and when this doesn't work the status column appears like down. We have tried to send kill–agent at three platforms that were working correctly and they pass to down .

```
( request
  :sender da0@visit18:1099/JADE
  :receiver ams@visit19:1099/JADE
  :content ( ( action ( agent-identifier
   :name ams@visit19:1099/JADE
   :addresses ( sequence http://visit18:7778/acc )
```

[11] http://www.agentcities.net/globalapd.jsp

```
   :resolvers (sequence))
    ( kill-agent :agent ( agent-identifier
    :name RMA@visit19:1099/JADE
    :addresses ( sequence )
    :resolvers ( sequence )
  ))))
  :communicative-act request
  :language: fipa-sl0
  :ontology: jade-agent-management
  :protocol: fipa-request
)
```

Example 2: In this example, the agent da0 tries to register da4 agent in visit19. The sequence of actions are: request, agree and inform, in this moment we have registered agent da4. If you repeat the register you obtain: request, agree and failure. The reason of this answer is because the agent da4 already had been registered in visit19.

```
( request
  :sender da0@visit18:1099/JADE
  :receiver ams@visit18:1099/JADE
  :content ( ( action ( agent-identifier :name ams@visit19:1099/JADE )
   ( register ( ams-agent-description
    :name ( agent-identifier
    :name da4@visit19:1099/JADE
    :addresses ( sequence http://visit19:7778/acc ))
    :state active))))
  :communicative-act request
  :language: fipa-sl0
  :ontology: fipa-agent-management
  :protocol: fipa-request
)
```

2. Communication: With the same environment that in the previous point. But in this case this has been proven only under Jade platform.

    (a) Flooding: The queue is an expandable data structure and can accommodate any number of messages, up to the available virtual memory limit. When the number of buffered messages exceeds a maximum queue size, older messages are discarded according to a FIFO replacement policy. In our case, we modify the queue size by means of the action setQueueSize at 1 in the receiver agent. At this moment, from other sender agents they sent messages in infinite loops to receiver agent.

(b) Outgrow the allowed size: In this case we have a message with a huge content, when this message is received this consume all platform memory. In this case, we have started the platforms with non–standard java options to manage heap size. In order to obtain it we have limited the heap size. The reason, is to be reproduced this type of error is necessary that the message created by the sender is greater than the maximum capacity of memory available of the receiver. In our case we have modified the PingAgent with the idea that instead of a ping it sends a message with an enormous content, in order to exceed the memory size of the receiver.

```
java -Xmx128M jade.Boot
s:SenderHugePing
-gui platform_that_send_the_message

java -Xmx20M jade.Boot
da0:jade.tools.DummyAgent.DummyAgent
-gui platform_that_receive_the_message
```

It results in that the receiver platform doesn't run out of memory and crashing.

5.2.2 EavesDropping: Unauthorised Disclosure of Information

Disclosure: Jade platforms (η UCTx in different hospitals and one *Carrel* in OCATT) are in separate machines and the communication among they could be disclosured with sniffer tools (figure 7), in our case we have proven concretely with Ethereal [12] . Ethereal is a free network protocol analyzer. The tool allows a user to examine data from a live network. You can interactively browse the capture data, viewing summary and detail information for each packet. In figure 8 we can disclose information related with Jade Platform (IOR, platform identifier and so on) and related to the application (agent identifier, message content, and so on). This information can be used by malevolent users.

5.2.3 Data resend: Unauthorised Modification of Information

Copy and Replay: To show this problem, supposing that we have sniffer and catches packets information and we could send "bad–actions" such as: we try to resend the same petition to the receiver or to respond to the sender with a false answer. The behavior of the receiver / sender will depend on the type of application.
For example: In figure 8 agents in Host2 sends the next message:

```
( query-ref
  :sender da0@Host2:1099/JADE
  :receiver Ping@Host1:1099/JADE
```

[12] http://www.ethereal.com

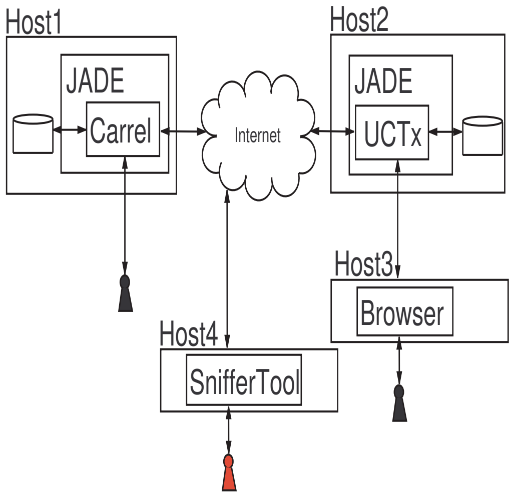

Figure 7: Application environment in eavesdropping

```
  :content "ping"
)
```

Agent in Host1 receives the message but Host4 too receives the message. And this could replay the message because he has a SnifferTool.

5.2.4 Application level threats

In our case, we describe application level threats related with UCTx. For example:

- What it happens if a surgeon makes a request for 300 eyes?
- What it happens if a surgeon always says that his requests are with urgency 0?
- What it happens if a surgeon requests a piece before he requires the piece?

One way of addressing this is by using a profile related with a user of the system. With this profile the system could obtain information about the user.

Summary threats that have been verified are shown in the next table:

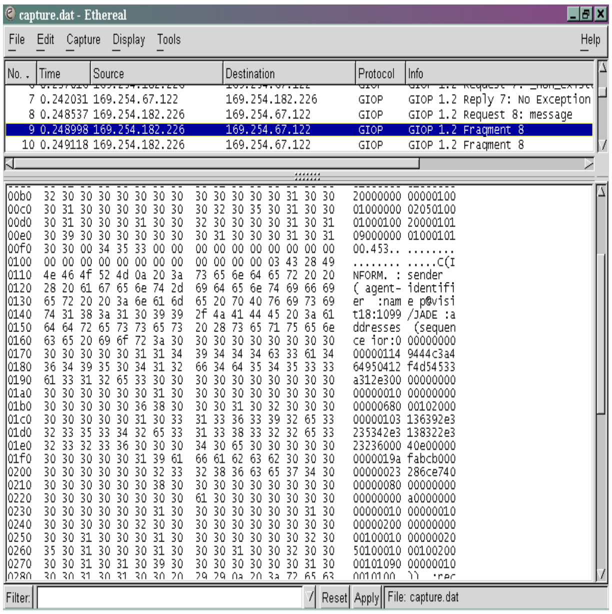

Figure 8: Snapshot of Ethereal showing an ACL INFORM message from p@visit18

Threats verified	
#	Description
1	Attacks on AP: Actions
2	Attacks on AP: Communications
3	Eavesdropping
4	Data resend: Copy and Replay

The threats alteration and spoofing are related with the agent identification. In Jade, the identification agent is associated to name @ host. We think that change the host name is a low level assumption. On the other hand, application level threats will be it specifies of each application.

6 Existing security safeguards

Next, we will describe security safeguards. We understand safeguards such as solutions or tools to defend from identified threats.

6.1 Jade–S

The first proposal is named Jade Secure AP (Jade–S [9]). It is an add-on for security and multi-user support. It includes features such as:

- User / agent authentication and authorization: To be able to own or perform actions on a component of the platform.
- Secure communication: To use Secure Socket Layer (SSL) protocol providing privacy and integrity for all intra–platform connections

In the next table, we summarize the different threats, solve and doesn't solve, by Jade–S:

Jade–S		
#	Solve	Does not solve
Agent Platform: Action		
1	$\surd$ It is not possible to send a kill–action if you don't have permissions to execute kill–action.	
2	$\surd$ If you like to register one agent but don't have permisions to do this you will obtain the next message: "Permissions to execute agent-receive-from on name–agent are not owned".	
Agent Platform: Communication		
1		$\surd$ If the number of messages is greater than the dimension of buffer these are lost.
2		$\surd$ If the dimension of message is greater than the space in the queue of messages the platform receiver is crashed.
Eavesdropping		
1	$\surd$ It uses Secure Socket Layer in order to realize a secure communication between containers but this doesnt work correctly between platforms .	
Data resend		
1		$\surd$ If two users have the same privileges over the actions to realize in one agent and one of them copies the message, this could resend the message.
Identity theft		
1	$\surd$ Each component in the platform belongs an authenticated user via user name and password. We should understand that authorizations and delegation are supported to allow or deny actions.	

6.2 X–Security

The second proposal is named, X–Security [13] is a project developed by Czech Technical University in Prague, Faculty of Electrical Engineering Department of Cybernetics, it proves agents identity and ensure secure communication. Security package offers these two basic functions:

- Security Certification Authority (SCA) - stand-alone agent, application independent, which allows certificate registration / deregistration and provides other agents in community with these certificates. It isn't a platform part (like AMS or DF).
- Security module - this module has to be included into an agent that wants to use communication security. Using this module we agents could use the methods that offer the security module such as (enCrypt, setSignature and so on).

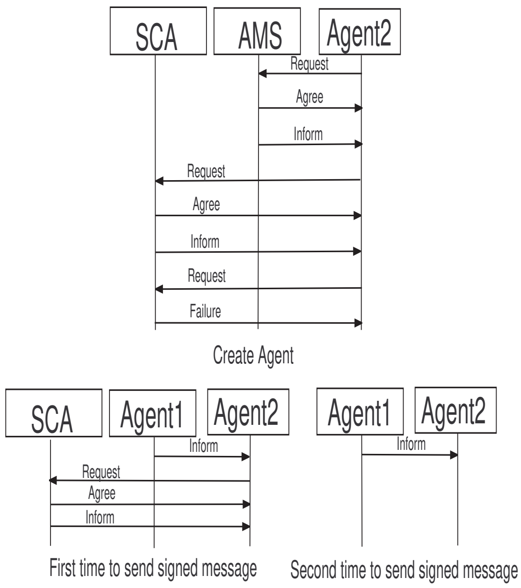

Figure 9: Communication in X–Security

In next table, we resume the different threats, solve and don't solve, by X–Security:

[13] http://agents.felk.cvut.cz/security/main/index.php

X–Security		
#	Solve	Does not solve
Agent Platform: Action		
1		$\surd$ The use of AMS is not limited, anyone could send a kill-agent.
2		Anyone could register in DF.$\surd$.
Agent Platform: Communication		
1 2		$\surd$ The channel of communication is protected but not the queue of messages.
Eavesdropping		
1	$\surd$The message between platforms are signed and encrypted.	
Data resend		
1	Before delivering the message X–Security detects correct certificate in queue of incoming ACL messages.	$\surd$ But, only the content is encrypted the content of the message and it is possible to catch other important information such as: sender, type of message and so on.
Identity theft		
1	$\surd$ The SCA generate a certificate for the agent. These data are saved in the private space of the agent.	

7 Safeguards Proposals

In this section we propose our own safeguards to protect of the threats described above. Future work is to implement and verify a number of these proposals. They also do not yet cover all the indentified threats.

7.1 Access control to key services

In the same way that when you start a platform, you could specifier the enable / disable mobility and clone support for the container, you should indicate if you like that other agents could register in your DF or that sort of actions (kill, register ...) could be able in certain agents. For example, you could wish that AMS don't make nothing if he receives one kill–agent.

7.2 Filter messages

To provide more intelligent queues of messages and only to accept message of one determined agent or agents. To solve threats related with the communication between agents. Work over queue of messages force to find a low–level solution, for example, to put the limits in the length of the messages.

7.3 Net Trust

Different agents can cooperate to accent security. Authentication can make use of agents persistence over time similar to the way people do, is possible to obtain the trust about one agent from the experience that has been had throughout the time. Agents can interact with agents to assess the behaviour of a particular agent to build up a distributed model trust.

If we work with one agency in a closed world we could protect but the most normal situation is that different agencies share services among different agencies and these could have different security level.

7.4 Guardian–Agent

Another idea is to include a central–agent that controls the communication between the different agents in the system, this agent could be called Institution Management (IM). This could be considered as "big–brother" in MAS *Carrel* and could start agents if they died or had problems.

We could see IM such a SnifferAgent. But SnifferAgent works at level of platform Jade and IM works at level of application. This idea is depicted in figure 10. But, this could be a double edged sword because of the important information managed in IM. It makes that we are obligated to safeguard the IM.

8 Conclusions

To sum up, in this paper we have showed the architecture of *Carrel* and UCTx and different threats and its safeguards. In this case we have talked about a medical application and this fact makes that we have paid special attention to every environment of our application.

We have made a deep study and proving in a real scenarios threats and safeguards. We have praised that solution is the use of different techniques to protect our application.

Finally, in the next list we describe future work:

- Implementation: To include some of safeguards proposed in this paper in Carrel system.
- Testing and assessment: To make the pertinent tests and verify that safeguards work correctly.

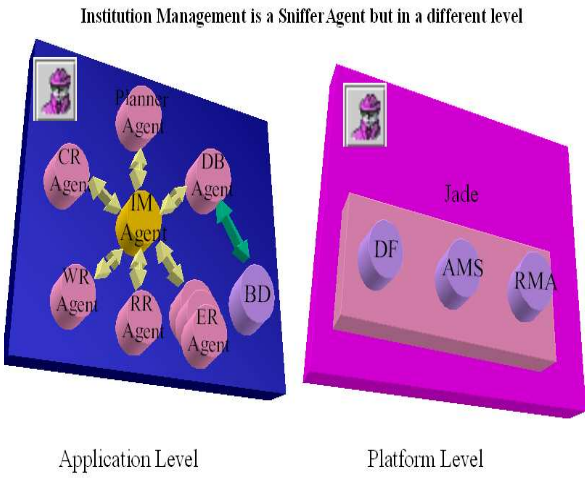

Figure 10: IM agent vs. Sniffer agent

9 Acknowledgements

David Cabanillas would like to express their thanks to Steven Willmot (UPC) because I have been able to make meetings periodically with him. And thanks also to Ulises Cortes (UPC) to give this opportunity to me. I could not have had better coordinators.

References

[1] C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberate normative agents: Principles and architecture, 1999.

[2] U. Cortés, A. López-Navidad, J. Vázquez-Salceda, A. Vázquez, D. Busquets, M. Nicolás, S. Lopes, F. Vázquez, and F. Caballero. Carrel: An agent mediated institution for the exchange of human tissues among hospitals for transplantation. In *3er Congrés Català d'Intel.ligencia Artificial*, pages 15–22. ACIA, 2000.

[3] U. Cortés, J. Vázquez-Salceda, and A. López-Navidad. Protección de datos médicos e internet. Technical report, Departament de Llenguatges i Sistemes d'Informació. Universitat Politécnica de Catalunya., April 2001.

[4] Marc Esteva, Julian Padget, and Carles Sierra. Formalizing a language for institutions and norms. In Milinde Tambe and Jean-Jules Meyer, editors, *Intelligent Agents VIII*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2001. to appear.

[5] Proceedings of the 4th International Conference, Exhibition on the Practical Application of Intelligent Agents, and Multi-Agents, editors. *JADE - A FIPA-Compliant Agent Framework.*, 1999.

[6] A. Poggi, G. Rimassa, and M. Tomaiuolo. Multi-user and security support for multi-agent systems, 2001.

[7] M. Calisti S. Poslad. Towards improved trust and security in fipa agent platforms. *AA2000*, June 2000.

[8] U. Cortés López-Navidad J. Vázquez-Salceda. Uctx: A multi-agent approach to model a transplant coordination unit. Technical report, Departament de Llenguatges i Sistemes d'Informació. Universitat Politécnica de Catalunya., April 2001.

[9] Giosué Vitaglione. Jade tutorial–security administrator guide.