RISK ANALYSIS AND DEPLOYMENT SECURITY ISSUES IN A MULTI-AGENT SYSTEM

Ambra Molesini, Marco Prandini, Elena Nardini and Enrico Denti
ALMA MATER STUDIORUM—*Università di Bologna, Italy*

Keywords: RBAC, MAS, Deployment, Risk analysis, Security engineering.

Abstract: Multi-agent systems (MASs) are a powerful paradigm enabling effective software engineering techniques: yet, it easily lets the designer be oblivious of the emergent security problems. This can be a critical issue, especially when MASs are exploited as an infrastructure to provide secure services. This paper performs a security analysis of such a scenario, identifying threats and assessing risks that could interfere with the achievement of the application goal – e.g. access control – as a consequence of its MAS-based implementation.

1 INTRODUCTION

Access control is a key topic in the security area. Building an access control system (Samarati and Capitani de Vimercati, 2001) means deploying a theoretically sound model on a structurally secure platform: the former cannot be effective without the latter.

In this context, Multi-Agent Systems (MASs) are finding interesting applications as providers of security functionalities. The flexibility of the agent paradigm proves very valuable in modelling the different aspects of security schemes, accurately capturing the concepts needed for achieving a robust design at the most appropriate abstraction levels. On the other hand, a MAS needs a complex underlying infrastructure, whose intrinsic security is fundamental for the correct behaviour of agents, and for the correct implementation of the policy to be enforced.

Various solutions exist (Yamazaki et al., 2004; Bordini et al., 2006; JADE, 2005) for the design of MAS-supporting platforms and for exploiting a MAS as a security provider, but the field of their security assessment is largely unexplored. We acknowledge that a layered approach is key to the proper engineering of complex systems, yet we claim that the existing literature fails to capture the peculiar security needs of MASs when exploited as security providers.

In this work, we analyse a previously proposed access control system (Section 2) to assess whether its peculiar MAS-based implementation affects the ability of the system to reliably pursue its ultimate goal. The attack sources fall into three broad layers: the *users*; the *system*, implementing the access control policies; and the *infrastructure*, supporting the MAS.

After shortly reporting on related works (Subsection 2.1), this paper first carries out a systematic risk analysis of the first two layers (Section 3), then analyses the advantages and drawbacks of different placement strategies (Section 4), emphasising the impact of deployment choices on the opportunity for attacks and their effects. Conclusions follow in Section 5.

2 BACKGROUND

In order to highlight the key aspects concerning the deployment issues in a secure MAS, we take the access control to a building as our reference. This choice is effective both because the access control problem has been widely studied in the literature (Samarati and Capitani de Vimercati, 2001), and because the access to some kind of building is at the same time simple enough to make most aspects clear, and not-so-trivial enough to expose most of the key aspects we mean to address. In particular, (Molesini et al., 2009) considers the case of a university building: roles, situations and policies are therefore tailored to the specific university environment, but this does not limit its generality, since a clear separation was adopted between the *access policy*, and the hardware & software *mechanisms* exploited to enforce such rules. Accordingly, since our interest concerns the deployment issues, we shortly report only the outcome of the mechanism design.

The global mechanism was conceived as com-

Molesini A., Prandini M., Nardini E. and Denti E. (2010).
RISK ANALYSIS AND DEPLOYMENT SECURITY ISSUES IN A MULTI-AGENT SYSTEM.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 143-148
DOI: 10.5220/0002704201430148

posed of two complementary sub-mechanisms, one controlling the access to the whole building (Figure 1, top) and another for the single room/office department—just "room" in the following (Figure 1, bottom). Both mechanisms were designed exploiting *Agents* and *Artifacts* (A&A) (Omicini et al., 2006) building blocks: agents were used for the system's active part, and artifacts for the system resources.

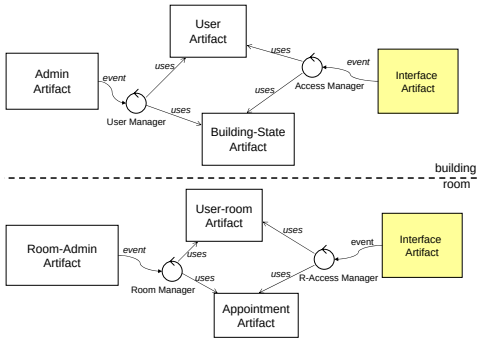

Figure 1: Access control mechanisms for the whole building (top) and the room (bottom) (Molesini et al., 2009).

There, "Interface Artifacts" represent the wrappers to the hardware resources which acquire the user credentials: there is one such artifact for each hardware device that monitors the building physical access points, plus one further Interface Artifact for each room. Interface Artifacts generate an event whenever a user enters the place they control: such events are then handled by "Access Manager" agents (for the whole building) and "Room-Access Manager" agents (for each room). Such agents exploit further artifacts – the "User Artifact" and the "User-room Artifact" – to check if the user can access the building/room: if so, the state of the "Building-State Artifact", which traces the people inside the building, is updated accordingly. Moreover, to grant room access also to occasional users who have an appointment with staff people, the "Appointment Artifact" is introduced to store the list of the appointments for a given room.

User Artifacts and User-room Artifacts store all the roles permanently qualified to access the corresponding place along with their access privileges, thus providing the knowledge base needed to decide whether an access is to be granted or not.

Users authorised to enter the building are managed by the "User Manager" agent, while users authorised to enter a room are managed by the "Room Manager" agent: both perceive the events generated by the "Admin Artifact" or by the "Room-Admin Artifact", respectively, which represent the interface between the human administrator and the mechanism itself.

2.1 Related Work

Some recent research work is reported in the literature about software engineering techniques for capturing security issues in the software development process. In particular, (Lodderstedt et al., 2002) introduces a methodology based on *SecureUML* which supports the specification of access control policies. Although this work makes an important integration effort, it is mainly focused on the process design stage: other aspects which still affect the software development process from the early stages are not considered— nor is there any reference about the system deployment and how the chosen deployment could impact on the reliability of the resulting system.

In the MAS field, integrating security in the development process is largely an unexplored area: one of the main contribution is represented by (Mouratidis and Giorgini, 2007), which proposes an extension of the *Tropos* methodology (Bresciani et al., 2004) to address security concerns, but risk analysis and system-deployment concerns are not explicitly considered.

In addition, in (Liu et al., 2002) authors present an approach for the analysis of security requirements for MAS development: yet, this work does not consider risk analysis explicitly.

3 RISK ANALYSIS

Risk analysis is a part of the more general process called *"Security risk assessment and management"* (Sommerville, 2007), where risk analysis should start from the identification of the system's *assets*—i.e., the system resources to be protected because of their *value*. So, first of all we have to determine the value of each asset inside the system, as well as its *exposure*, which represents the possible loss or harm that results from a successful attack (Sommerville, 2007).

Figure 2 reports the assets identified in our case study (presented in Section 2), along with their values and exposures. Assets are divided in three categories: artifacts (top), agents (middle), and what we call the "real world" (bottom), that is, the physical devices and the infrastructure where the software system will be deployed. Apparently, Figure 2 reports only the "material" assets, such as software and physical entities, while the "immaterial" assets – i.e., data such as the users' credentials and profiles – seem not to be taken into account. The reason is that immaterial data are considered indirectly: since the system is developed using the A&A metaphors, any external resource is wrapped by some suitable artifact—including the system's data.

Asset	Value	Exposure
Interface Artifact	high	medium
Admin Artifact	high	high
User Artifact	high	high
Building-State Artifact	low	low
Room-Admin Artifact	high	high
User-room Artifact	high	high
Appointment Artifact	medium	medium
User Manager	high	high
Access Manager	high	high
R-Access Manager	high	high
Room Manager	high	high
Physical Device	high	high
Infrastructure	high	high

Figure 2: System's assets.

Threat	Probability
Stealing admin credential	low
Stealing user credential	high
Personifying user	high
Social Engineering	high
Introducing malicious agent	medium - high
Disappearing agent	medium - high
Agent bugs	high
Modifying agent code	low - medium
Tampering artifact data	high - very high
Sniffing artifact data	high - very high
Artifact bugs	high
Replacing artifact	medium - high
Men in the middle	medium - high
Sniffing communication	medium - high
Damaging physical device	high

Figure 3: System's threats.

So, since in our case study the system's data are accessible only through artifacts, attackers have to force the artifacts in order to steal or modify such data. However, one must be aware that this is a critical aspect: in fact, artifacts were introduced in open and dynamic systems, where agents can join the MAS and discover the services provided by artifacts simply thanks to the artifact's "inspectability" property, which allows agents to inspect the state of an artifact and its content (Omicini et al., 2006). This is the reason for the *high* value and exposure levels assigned to some artifacts in Figure 2 – namely, to the Admin Artifact, the Room-Admin Artifact, the User Artifact, and the User-room Artifact – as they wrap the most sensitive application data.

The next step in the risk analysis process is threat identification (Sommerville, 2007), Threats can be divided into two different classes: *(i)* fortuitous events, such as earthquakes, flooding, storms, etc., and *(ii)* deliberate attacks, such as sniffing, spoofing, etc. For the sake of simplicity, we assume here that all the fortuitous events can be viewed as different examples of physical devices damaging.

Now the probability that a threat actually materialised as an attack has to be estimated. Figure 3 shows threats and probabilities for our case study, divided in three categories based on the threatened application layer: threats to users (top), threats to software entities (middle), and threats to physical devices.

Perhaps surprisingly, one might note that we did not report the threats to the infrastructure level: this was done intentionally, just because infrastructural security issues are so wide to require a specific investigation, which is outside the scope of this paper.

The probabilities of the threats belonging to the first and third categories are expressed as single values (Figure 3), as they are independent from the specific system deployment that will be chosen at the end of the system design phase. Instead, the threats of the second category are expressed as a range of probability values, as they strictly depend on the specific system deployment. In fact, the *locus* where an artifact or an agent will be allocated affects both the protection of the communications and the mechanisms that are required for the artifact's / agent's protection. The only exceptions in this category are the "Agent bugs" and "Artifact bugs", which are not tied to to the deployment of the software entities.

Two threats in Figure 3, "Tampering artifact data" and "Sniffing artifact data", assume very high values of probability: this is because they threaten the most valuable system assets (see Figure 2)—i.e., the artifacts. In addition, the probabilities of such two threats are also conditioned by the probabilities of other threats such as "Introducing malicious agent" or "Modify agent code", because of the risk of misusing the artifact's inspectability property. If, for instance, a malicious agent is introduced in the system, the inspectability of "User Artifact" or "Admin Artifact" could in principle allow such agent to steal the user credentials. In the same way, if the code of an agent has been changed, the agent could behave as a malicious agent and possibly steal or corrupt the system data. Of course, not all of the artifacts in the system are so critical: for instance, the "Building-State Artifact" does not contain any specially-critical data.

Finally, Figure 4 reports for each asset the set of threats that could be exploited by an attacker. This figure highlights the inter-dependencies between the artifact's threats and the "agent asset": for instance,

if the "Replacing artifact" threat becomes active, the new artifact introduced in the system could provide wrong information to agents, leading them to behave in an unpredictable way (see bullets in the whole line).

4 DEPLOYMENT ISSUES

Our aim in this section is to investigate the issues designers have to face when they have to determine the best deployment for a system. In order to discuss the different deployment strategies, we assume that infrastructures exhibit the same basic set of concepts:

Nodes — logical *loci* where agents and artifacts can be allocated.

Artifacts — passive components of the systems, that are constructed, shared, manipulated, and used by agents to support their activities; artifacts are classified as: *resource artifacts* – wrap external resources (see Section 3) –; *social artifacts* – mediate between two or more agents in a MAS – ; *individual artifacts*—mediate between an individual agent and the environment

Agents — pro-active components of the systems.

The different deployment choices stem from the simple consideration that increasing the complexity of a component yields the opportunity for more sophisticated security measures, but also broadens the spectrum of possible attack paths and failure modes.

In the following, we present the general deployment issues tied respectively to the artifacts (Subsection 4.1) and the agents (Subsection 4.2) metaphors, then we discuss strengths and drawbacks of two different deployment strategies (Subsection 4.3).

4.1 Artifacts Security Deployment Issues

Risk analysis (Section 3) showed that artifacts are valuable assets, consequently, their deployment is critical from the security viewpoint.

In particular, resource artifacts abstract the functions and behaviours of devices. Where possible, the deployment of the artifact on the device it controls should have a *smart device* as a result. The smart device enjoys the useful properties of artifacts such as openness. On the flip side, openness can also mean more easy access for potential attackers, so the node where the smart device is allocated should be protected in order to prevent possible artifact tampering, replacement and sniffing. In addition, the designer should foresee mechanisms for protecting the physical devices, so that the "artifacts corruption" does not automatically damage the integrity and confidentiality of the controlled devices.

Social artifacts are the core of interactions: agents use them for communicating with each other, their deployment is critical and should take into account all the measures to ensure that they remain trusted.

Finally, individual artifacts both equip agents with all the protocols they can adopt for interacting in a particular society, and rule the agent behaviour. Even if individual artifacts do not play a crucial role in our specific case study, in general their deployment is particularly critical, since the corruption of this kind of artifact could allow a malicious agent to misbehave.

4.2 Agent Security Deployment Issues

In a system developed according to the A&A metamodel, only agents can take proactive security measures, for instance as a consequence of a suspicious activity against the artifacts or the controlled devices. A smart device then can be made even smarter by slightly modifying the proposed model so as to introduce device manager agents, that exploit "local intelligence" to detect and promptly face dangerous situations, both of malicious source (e.g. by intrusion detection and prevention functionalities) and of accidental nature (e.g. implementing fail-secure/ fail-safe policies). To some extent, the deployment of building access managers as closely as possible to the related interface artifacts can bring similar advantages, at least acting as a system-level barrier that helps blocking the spreading of problems out of the physical and logical context where they appeared.

However, the agents present several vulnerabilities and are subject to different threats (Figure 4). In particular, their autonomy, pro-activity and learning capabilities could also act as drawbacks from the security view point, since these properties restrict the designer's control on the agent execution flow. Other malicious agents and corrupted artifacts can induce data-driven agent misbehaviour.

4.3 Deployment Configurations

In this section our aim is to understand how issues presented in the previous Subsections can be resolved or reduced, and what is the role of the risk analysis during the deployment design.

For the sake of brevity, we restrict our deployment analysis to the sub-mechanism (Section 2) for the access to the whole building, since the architectures of the two sub-mechanisms are very similar.

First of all we need to analyse the "deployment requirements" coming from the physical world. For

Threat	Asset												
	Interface Artifact	Admin Artifact	User Artifact	Building-State Artifact	Room-Admin Artifact	User-room Art.	Appointment Artifact	User Manager	Access Manager	R-Access Manager	Room Manager	Physical Device	Infrastructure
Stealing admin credential	*	*	*										
Stealing user credential	*		*	*	*								
Personifying user	*		*	*	*	*	*						
Social Engineering	*	*	*	*	*	*	*						
Introducing malicious agent	*	*	*	*	*	*	*	*	*	*	*		
Disappearing agent								*	*	*	*		
Agent bugs								*	*	*	*		
Modifying agent code	*	*	*	*	*	*	*	*	*	*	*		
Tampering artifact data	*	*	*	*	*	*	*						
Sniffing artifact data	*	*	*	*	*	*	*						
Artifact bugs	*	*	*	*	*	*	*						
Replacing artifact	*	*	*	*	*	*	*	*	*	*	*		
Men in the middle	*	*	*	*	*	*	*	*	*	*	*		*
Sniffing communication	*	*	*	*	*	*	*	*	*	*	*		*
Damaging physical device	*	*										*	

Figure 4: Threats for each asset.

sake of simplicity we can consider four logical nodes – each of these nodes can have one ore more physical counterparts, we see below how to manage this situation – labelled *Node 1*, *Node 2*, etc. The physical resources, i.e., the device capturing the user credential, the administrator position, and the database are respectively allocated in *Node 2*, *Node 3* and *Node 4*. In addition, let us suppose that the protection of these devices is realised at the infrastructural level, since here we focalise only the MAS security deployment.

In Figure 5, we propose two different deployment configurations that represent the two extremes in the range of all possible configurations. In particular Figure 5 part a) presents a centralised MAS deployment since all the sub-mechanism software entities are allocated in the *Node 1*, while part b) presents a totally distributed MAS deployment.

The centralised deployment is apparently more convenient in terms of protection mechanisms because it is sufficient to build a "secure boundary" around *Node 1* to protect the communication channels among nodes in order to obtain a "secure" system. However, starting from the risk analysis results in Section 3 and analysing this configuration we can note that the MAS asset's exposures are all, equally increased, since the compromission of a single software entity means that the secure boundary of *Node 1* is broken. In addition, also the threat probabilities (Figure 3, middle) regarding the assets eventually increase, since an attacker will try to force *Node 1* for accessing the system, whereas he threat probabilities regarding the intra-MAS communications decrease, since these communications occur inside the secure boundary of *Node 1*. This configuration is only apparently simpler, security-wise, because the chosen protection mechanisms should be suitable for protecting the more valuable asset – such as the "User Artifact" – but the costly, effective countermeasures have to be sized to protect the whole *Node 1*, including less valu-

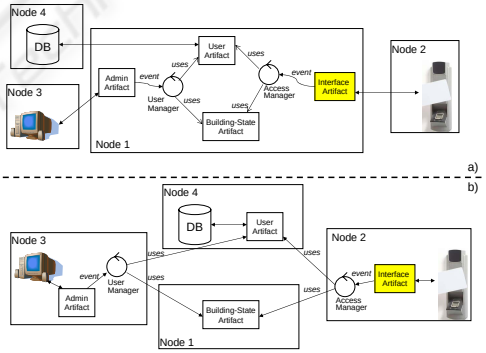

Figure 5: Centralised and distributed deployments.

able assets such as the "Building-State Artifact".

The distributed deployment configuration seems indeed more expensive from the protection mechanisms viewpoint, since all the system entities and the communication channels need to be protected.

However, this configuration makes it possible to decouple the exposures level of assets, choosing the most suitable protection mechanism for each: this is why the exposure levels for this configuration are the same as in Figure 2. In addition, this deployment could lead to reduce the inter-dependency between threat probabilities: for instance, if the artifact controls the identity of the requesting agent and adopts secure channels and cryptographic algorithms, the threats to agents are decoupled from the threats to artifact and vice versa. Of course, this configuration presents higher probability values associated with intra-MAS communication than in the centralised scenario, since the communications between entities always occur between network nodes, exposing the vulnerabilities related to the interactions.

As a result, the distributed configuration seems more appealing than the centralised one, the key decision element being the former's resiliency: the compromission of one node does not automatically implies the compromission of the whole system.

Once a "deployment architecture" is chosen for the base scenario, it also influences the way risk analysis is updated when the scenario changes. So, the choice of a deployment configuration should be guided by the findings of an accurate risk analysis, but it also affects the way the risk analysis itself evolves with the system.

5 CONCLUSIONS

In this paper we explored the topic of security assessment in a MAS, taking a MAS-based access control system as our reference. We performed a detailed risk analysis then, we studied how the deployment choices can influence the opportunity for attacks and the effects of their success. From the viewpoint of the software development lifecycle, the deployment analysis we performed can be situated at the end of the design phase as the purpose of this study is precisely to identify the "most adequate" deployment strategy in terms of security assessment.

Of course, our work is just the starting point of the story. Much broader research is needed to devise a general model of the security requirements for MAS-based systems: in turn, this will open the way towards the integration of security aspects into a suitable agent-oriented design methodology. Further investigations are also required concerning the security issues at the infrastructural level, since the role of the MAS infrastructures is becoming more and more relevant in the whole MAS development process.

REFERENCES

Bordini, R., Braubach, L., et al. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44.

Bresciani, P., Perini, A., et al. (2004). Tropos: An agent-oriented software development methodology. *AAMAS Journal*, 8(3):203–236.

JADE (2005). Jade.tilab.com/doc/tutorials/JADE_Security.pdf.

Liu, L., Yu, E., and Mylopoulos, J. (2002). Analyzing security requirements as relationships among strategic actors. Raleigh, North Carolina. electronic note.

Lodderstedt, T., Basin, D. A., et al. (2002). SecureUML: A UML-based modeling language for model-driven security. In *Proc. 5th Int. Conf. on The Unified Modeling Language*, pages 426–441, London, UK. Springer.

Molesini, A., Denti, E., and Omicini, A. (2009). RBAC-MAS & SODA: Experimenting RBAC in AOSE. In *Engineering Societies in the Agents World IX*, volume 5485 of *LNCS*. Springer.

Mouratidis, H. and Giorgini, P. (2007). Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309.

Omicini, A., Ricci, A., and Viroli, M. (2006). *Agens Faber*: Toward a theory of artefacts for MAS. *ENTCSs*, 150(3):21–36.

RBAC (2004). http://csrc.nist.gov/rbac/.

Samarati, P. and Capitani de Vimercati, S. (2001). Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 137–196. Springer.

Sommerville, I. (2007). *Software Engineering 8th Edition*. Addison-Wesley.

Viroli, M., Omicini, A., and Ricci, A. (2007). Infrastructure for RBAC-MAS: An approach based on Agent Coordination Contexts. *Applied Artificial Intelligence*, 21(4–5):443–467.

Yamazaki, W., Hiraishi, H., and Mizoguchi, F. (2004). Designing an agent-based rbac system for dynamic security policy. In *Proc. 13th IEEE Int. Workshops on Enabling Technologies (WETICE'04)*, pages 199–204, Washington, DC, USA. IEEE CS.