Multi-Agents based Architecture for IS Security Incident Reaction

Gateau, Benjamin; Feltus, Christophe; Khadraoui, Djamel; de Remont, Benoît

Published in:
Proceedings of RIVF'08 : IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies, Ho Chi Minh, Vietnam.

Publication date:
2008

Document Version
Early version, also known as pre-print

Multi-Agents based Architecture for IS Security Incident Reaction

Benjamin Gâteau, Djamel Khadraoui, Christophe Feltus and Benoît de Rémont
Centre for IT Innovation
Public Research Centre Henri Tudor
29, Avenue John F. Kennedy, L-1855 Luxemburg
{benjamin.gateau}@tudor.lu

***Abstract*— The main focus of this paper is to provide a global architectural solution built on the requirements for a reaction after alert detection mechanisms in the frame of Information Systems Security and more particularly applied to telecom infrastructures security. These infrastructures are distributed in nature, therefore the targeted architecture is developed in a distributed perspective and is composed of three basic layers: low level, intermediate level and high level. The low level is dedicated to be the interface between the main architecture and the targeted infrastructure. The intermediate level is responsible of correlating the alerts coming from different domains of the infrastructure and to deploy smartly the reaction actions. This intermediate level is elaborated using multi-agents system that provide the advantages of autonomous and interaction facilities. The high level permits to have a supervision view of the whole infrastructure, and to manage business policy definition. The proposed approach has been successfully experimented for data access control mechanism.**

Keywords- Security Policy, Multi-agents systems, Architecture, Distributed networks

I. INTRODUCTION

Today telecommunication and information systems are more widely spread and mainly heterogeneous. This basically involves more complexity through their opening and their interconnection. Consequently, this has a dramatic drawback regarding threats that could occur on such networks via dangerous attacks. This continuously growing amount of carry out malicious acts encompasses new and always more sophisticated attacks techniques, which are actually exposing operators as well as the end user.

State of the art in terms of security reaction is limited to products that detect attacks and correlate them with a vulnerability database but none of these products are built to ensure a proper reaction to attacks in order to avoid their propagation and/or to help an administrator deploy the appropriate reactions [1]. In the same way, [3] says that at the individual host-level, intrusion response often includes security policy reconfiguration to reduce the risk of further penetrations but doesn't propose another solution in term of automatic response and reaction. It is the case of CISCO based IDS material providing mechanisms to select and implement reaction decision.

The realm of security management of information and communication systems is actually facing many challenges [5] due to the fact that it is very often difficult to:

- Establish central or local permanent decision capabilities;
- Have the necessary level of information;
- Quickly collect the information, which is critical in case of an attack on a critical system node;
- Launch automated counter measures to quickly block a detected attack;

Based on that statements, it appears crucial to elaborate a strategy of reaction after detection against these attacks

Our previous work around that topic has provided first issues regarding that finding and has been somewhat presented in [5]. This paper has proposed an architecture to highlight the concepts aiming at fulfilling the mission of optimizing security and protection of communication and information systems which purpose was to .achieve the following:

- Reacting quickly and efficiently to any simple attack but also to any complex and distributed ones.
- Ensuring homogeneous and smart communication system configuration, that are commonly considered and the main sources of vulnerabilities.

One of the main aspects in the reaction strategy consists of automating and adapting policies when an attack occurs. It exists in the scientific literature a large number of policy's definitions and conceptual model. For the purpose of that paper, we prefer the one provided by Damianou et al. in [14] that is "*Policies are rules that govern the behavior of a system"* (actors and sub components). The foreseen policy adaptation is considered as a regulation process. The main steps of the policy regulation are described in Figure 1, which shows the process that takes the business rules as input, and maps them into technical policies. These technical policies are deployed and instantiated on the infrastructure in order to have a new state of temporary network security stability adapted to the ongoing attack. This policy regulation is thereafter achieved in modifying/adding new policy rules to reach a new standing (at least up to the next network disruption) policy based on the observation of the current situation of the system. It must be

specified that this regulation process rely also on policies adaptation to a specific context. Those contexts and the modeling of concepts of org, role, activity, view are explained in [10]. Efficiently react against an attack, especially if this needs a change on an equipment configuration, often necessitates many checks that have to be performed in order to avoid bad side effects (conflict creation, services stability, etc.)

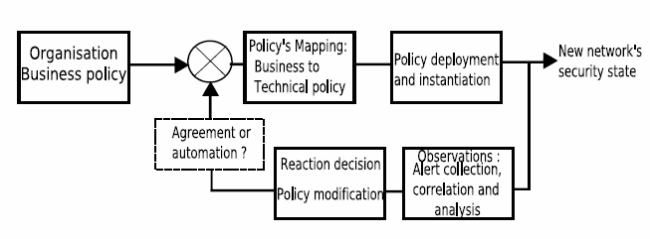

Figure 1. Policy regulation

Consequently, policy regulation's automation needs in one hand the existence of a hierarchy between the rules in case of multiple choices due to multiple attacks, and in second hand an automatic method to validate the policy's modifications. At the business level, the targeted foreseen solution will be able to improve the resilience to attacks of core IP networks and, by extension to large information systems, which form critical infrastructures for communication and services today.

The second section of this paper introduced requirement that has to be taken into account for the definition the presented architecture, section III introduce the architecture and section IV illustrate it through a use case in telecommunication networks.

II. REQUIREMENTS ANALYSIS AND DESIGN

The architecture of such a reaction system must respect some classes of requirements that has been synthesized in the following: (TABLE I.)

TABLE I. REQUIREMENT ANALYSIS

Requirement list	Description
Business needs	Laws and regulations dedicated to private sector exist and are continuously improving requirements that enforce the top management to be responsible regarding the needs for information security (SOX, Basel 2, ISO27000). Corporate policy and security policies are tools under the cover of the business that face IS security issues. In that sense, security requirements are dictated by the business and IT staff implements them. Accordingly, a business requirement is: when an attack occurs, the technical IT committee adapts the basic policy to solve the problem. This emergency modification of the consign policy needs to be validated or improved by the policy business owner before being introduced in production.
Scalability	The system should be able to manage and ensure security of several sub-systems (e.g. LAN and subs-LAN) called "managed systems".
Availability	There's always in IT systems a single element, component, system, device, or person that is crucial for the mission and of course the security; these item are called "single points of failure" and the management system should avoid them.
Confidence	Current usage of automatic reaction technologies is narrowed by end-user confidence into the system. As a result, operators often deactivate automatic features of the system. Strong confidence can be established by design, ensuring that reaction don't contravene known business policies. Besides, a confidence measure must be provided for each non-trivial process, where low confidence involves human support (Agreement, Manual investigation, Reaction selection). Thus the system should provide granularity in the automatic process.
Autonomy	However, certain autonomy should be provided to the managed systems, to avoid paralyzing situation in case of loss of connection with the global system. This autonomy could enable the system on highly scalable network (as P2P or Ad-Hoc networks) and specifically when a peer could be a managed system or part of it.
Survivability and robustness	The management system should implements means for being able to continue to function during and after a damage or loss due to intentional malicious threats (i.e. survivability), and unintentional hardware failures, human errors, etc. (e.g. robustness).
Reaction applicability	A reaction should be applicable to several managed systems or to targeted objects. The reaction applicability should be specified and adaptable considering the reaction. Furthermore, a time defining the validity of the reaction should be specified (temporary reactions for a certain time, or permanent).
Alert management correlation	Relatively to the alerts management, a global correlation between the alerts coming from different managed systems should be realized. The existing intrusion detection tools generate alerts and the system just collect and process them, as observation input. The alert should be used immediately by the local level, for an rapid reaction but also in a second time for a more adapted reaction (if needed).
Global supervision	Furthermore, a global supervision (common to all the managed systems) must available in order to manage detection and reaction (based on policy) on widely spread systems. Indeed, alerts from all the managed systems should be correlated together at the higher level of hierarchy. This supervision should be useful to check if the business policies are respected at both management levels.

III. AGENT BASED POLICY MANAGEMENT ARCHITECTURE

A. *Overview and definitions*

A Multi-Agent System (MAS) is a system composed of several agents, capable of mutual interaction. The interaction can be in the form of message passing or producing changes in their common environment.

Agents are pro-active, reactive and social autonomous entities able to exhibit organized activity, in order to meet their design objectives, by eventually interacting with users. Agent is collaborative by being able to commit itself to the society or/and another agent.

An agent, like an object, encapsulates a state and a behavior and provide moreover a number of facilities that are ::

- An agent has control on its behavior
- An agent decides in which state it is, even if external event may influence this decision..
- An agent exerts this control in various manners (reactive, directed by goals, social)
- MAS have several control flows while a system with objects has a priori only one control flow.

The agents also have global behavior into the MAS, such as:

- Cooperation: agents share the same goal
- Collaboration: agents share intermittently the same goal,
- Competition: incompatible goals between agents

An architecture description has been developed considering the requirements described in the previous section. To manage several different systems, due to their location, the focused business domain or organization type, a distributed system is appropriate. Furthermore, a distributed solution should be able to bring some autonomy to the managed systems; robustness, survivability and availability are also impacted.

The architecture will be composed of several components, called “nodes”, having different responsibilities. Theses nodes will be organized in two dimensions, as presented in Figure 2. .

The vertical dimension, structured in layers relatively to the managed network organization, allows adding abstraction in going upward. Indeed, the lowest layer will be close to the managed system and thus being the interface between the targeted network and the management system. The higher layer will expose a global view of the whole system and will be able to take some decisions based on a more complete knowledge of the system, business, and organization.

Intermediate levels (1 to n-1) will guarantee flexibility and scalability to the architecture in order to consider management constraints of the targeted infrastructure. Those middleware levels are optional but allow the system to be better adapted to the complexity of a given organization and the size of the information system.

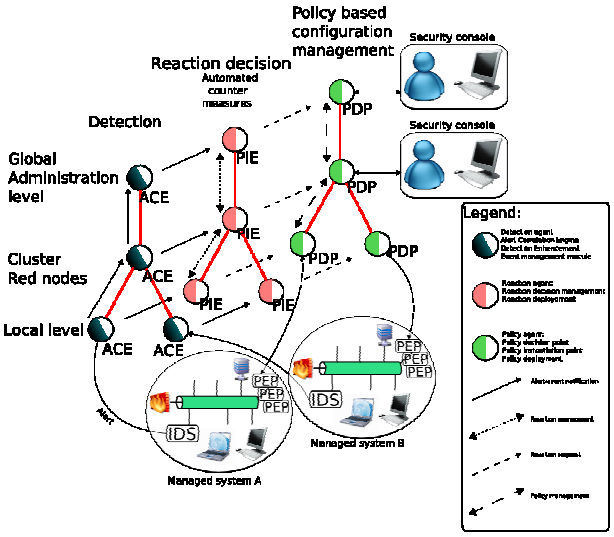

Figure 2. Architecture Overview

The horizontal dimension, containing three basic components, is presented in Figure 3. and its three main phases are described below:

1) Alert: Collect, normalize, correlate, analyze the alerts coming from the managed networks and representing an intrusion or an attack. If the alert is confirmed and coherent, it is forwarded to the reaction decision component. (Alert Correlation Engine-ACE).

2) Reaction Decision: Receive confirmed alerts for which a reaction is expected. Considering the knowledge of: the policy, the systems organization and the specified behaviour, this component decide if a reaction is needed or not and define the reaction, if any. The reaction will be modification(s), addition(s) or removal(s) of current policy rules. (Police Instantiation Engine-PIE).

3) Reaction: Instantiation and deployment of the new policies, on the targeted networks. The deployment (Policy Deployment Point – PDP) and enforcement (Policy Enforcement Point – PEP) of these new policies, lead to a new security state of the network. The terminology in italic used in this section 4 is extracted from both: XACML [9] and OrBAC Model [11].

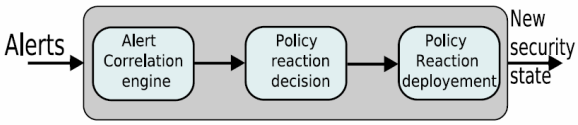

Figure 3. The three basic components

An issue is raised considering which layer will be allowed to take a decision reaction: only one layer, two, several or all? If more than one layer can trigger a reaction on the same object(s), there will be a conflict issue. Thus, the system should be able to provide mechanisms to solve conflicts between several selected reactions. Another issue concerns the agreement: at which level should it be asked? : A solution could be to ask it at the same level (or at an upper one) that the reaction decision is made, this should be specified by the user. A possible solution is a distributed, vertically layered and hierarchical architecture. The layer's number could be adapted according to the organization of the managed systems. In our case, three layers are sufficient (local, intermediate and global). The reaction system is composed of three main parts: the alert management part, the reaction part and the police definition-deployment part. Three trees (alert, reaction and policy) could be placed side by side, as presented in Figure 2. These trees are the same but their nodes have different functions. The alert tree collects the alerts with the local nodes and correlate them in several steps, one step by layer. A certain response time is used by the system from the intrusion detection to the reaction application. This time is increased if the reaction process is propagated to the upper layers, as presented in Figure 4. A global goal is of course to shorten it.

$$\text{Response time} = 2\times(\text{Propagation time between levels}) + (\text{Processing and deployment time})$$

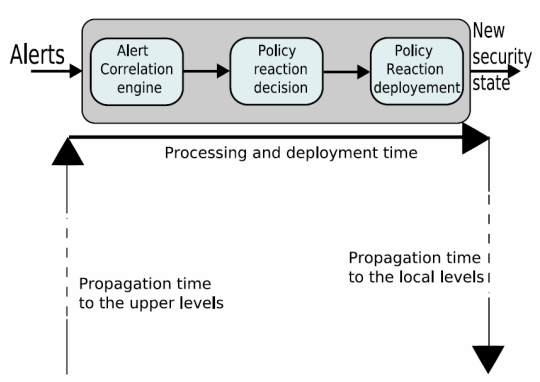

Figure 4. Response time.

The next step of our research development is firstly the definition of a reaction engine that encompass both architecture components defined in that paper and communication engine between these components. This engine will be based on a message format and on a message exchange protocol based on standards such as [12]. Secondly, real cases must be studied in order to experiment with the architecture and its associated protocol.

The message format will be defined in XML format and will be structured around a number of attributes that will specify the message source, the message destination and the message type (alert, reaction, policy request, policy modification, policy modification validation, decision and synchronization). The protocol will define the exchange format and the workflow of messages between the architecture components. It will encompass a set a rules governing the syntax, semantics, and synchronization of communication. In the section relating to technical requirements, we have seen that nodes structure must be flexible in order to be able to reorganize itself if a node fails or disappears. Each node must also be autonomous in order to permit reorganization. Given these requirements, we think that the use of Multi-Agents Systems is a solution to provide autonomy, flexibility and decision mechanisms to each node by representing them by agents.

As studied in the state of the art presented in [4], a set of agents could be managed and controlled through an organization. An organization is a set of agents playing roles, gathered in a normative structure and expecting to achieve some global and local objectives. Several models like the roles model, the tasks model, the interaction model, the norms models etc specify an organization.

In our context we need an interaction definition in order to specify communication protocols between agents representing nodes. We also need roles in order to specify what agent will have to communicate or act in order to detect intrusions and then react. Based on this needs, the use of an electronic institution based on agents is one of the possibilities that we will investigate.

The main goal of the reaction policy enforcement engine is to apply policies in terms of specific concrete rules on “technical” devices (firewall, fileserver, and other systems named PEP). For that, we need means to make PIE, PDP and PEP interacting and collaborating. As we will see in the following section, the multi-agents systems concept already defines architectures and models for autonomous agents’ organization and interaction. Existing platform like JADE (Java Agent DEvelopment Framework) [18][19] implements agents’ concepts as well as their ability to communicate by exchanging messages and could simplify the reaction components integration. This is the solution, which is detailed hereafter. Foundation for Intelligent Physical Agents (FIPA) [16], promotes the success of emerging agent-based applications, services and equipment. Making available in a timely manner, internationally agreed specifications that maximize interoperability across agent-based applications, services and equipment pursues this goal. This is realized through the open international collaboration of member organizations, which are companies and universities active in the agent field. FIPA's specifications are publicly available. They are not a technology for a specific application, but generic technologies for different application areas, and not just independent technologies but a set of basic technologies that can be integrated by developers to make complex systems with a high degree of interoperability.

The multi-agent framework that will be used her is JADE. We base ourselves on a survey made in [15] to argue that this agent platform responds to the expectations in terms of agents' functionalities, security, safe communication between agents, performance and standardization.

The following sections present the specification of the policy enforcement engine deployment based on agents. After motivating this solution, we introduce agents and multi-agents

theory and we detail the Policy Enforcement Point, Policy Decision Point and the communications between them.

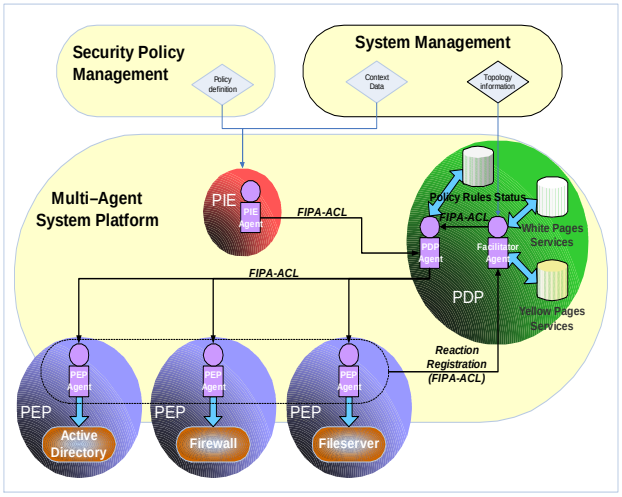

Figure 5. Multi-Agent System based enforcement process deployment

B. *Policy Enforcement Point*

We consider here the flow starting with a set of new policies to apply on physical PEP. We also consider that the main components of the "policy enforcement architecture" (PIE, PDP and PEP) are composed of an agent (or more) as depicted on Figure 5. The PIE decides to apply new policies. Its PIE Agent sends the policies to the PDP Agent, which decides which PEP is able to implement policies in terms of rules or script on devices (firewall, fileserver, etc,). Then, the PDP Agent sends to PEP Agent of which PEP are concerned by their corresponding policies. Finally, each PEP Agent knowing how transforming a policy into a rule or script understandable by the device interfaced implements the policy. Consequently, agents do not represent only PDP but each component of a node (in the enforcement loop at least). This solution provides a multi-agent framework making possible agents cooperating and communicating between them.

C. *Policy Decision Point*

Figure 5. represents the PDP architecture composed by several modules. For the multi-agent system point of view, the Component Configuration Mapper results from the interaction between the PDP Agent and the Facilitator Agent while the Policy Analysis module is realized by the PDP Agent. The Facilitator manages the network topology by retrieving PEP Agents according to their localization (devices registered with IP address or MAC address) or according to actions they could apply and their type (firewall, file server, etc.). For that the Facilitator uses white pages and yellow pages services. The JADE platform already provides implemented facilitator and searching services. Besides, the use of a multi-agent system as the framework provides flexibility, openness and heterogeneity. Actually, when we decide to add a new PEP, we just have to provide its PEP Agent able to concretely apply the policies that will register itself through the Facilitator that will update databases.

D. *Communication specifications using Jade*

JADE is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middleware, which is FIPA compliant. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. JADE ensures standard compliance through a comprehensive set of system services and agents in compliance with the FIPA specifications: naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used.

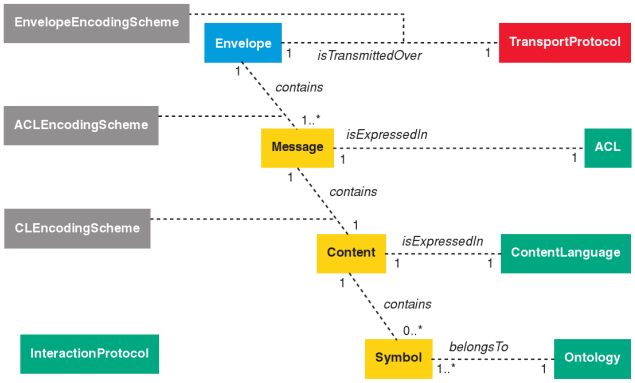

Figure 6. FIPA-ACL Overview

The AMS (Agent Management System) provides the naming service (i.e. ensures that each agent in the platform has a unique name) and represents the authority in the platform (for instance it is possible to create/kill agents on remote containers by requesting that to the AMS). The DF (Directory Facilitator) provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals. The ACC (Agent Communication Channel) is a high-level interface, through which messages are sent using a MTP (Message Transport Protocol). FIPA-ACL [17] is the standardization of ACLs developed by FIPA. ACLs (Agent Communication Languages) are high level languages based on speech acts (inform, request. cfp, agree, understood, ...) in order to establish collaboration, negotiation etc. A message written by using an ACL describes a desired state instead of procedure or method call. ACLs are based on low-level languages for messages transportation (SMTP, TCP/IP, IIOP, HTTP).

FIPA-ACL messages are structured among other things with performatives (type of communication acts), sender, receiver, content, a language in which the content is expressed and an ontology used to give sense to symbols used in the content expression. For instance, Agent A (the sender) can send a FIPA-ACL message to Agent B (the receiver) requesting (use of performative request) something (content of the message) in language SPL by respecting the protocol "policyApply". We choose SPL [7] to represent policies within the agent platform. Therefore, the content of the message will be a XML file defining the policy to apply. The full FIPA

communication model is implemented in JADE and its components have been clearly distinct and fully integrated: interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and, finally, transport protocols. The transport mechanism, in particular, is like a chameleon because it adapts to each situation, by transparently choosing the best available protocol. Java RMI, event-notification, HTTP, and IIOP are currently used, but more protocols can be easily added via the MTP and IMTP JADE interfaces. Inside this platform, a communication support is defined and agents communicate by exchanging messages structured in accordance with the FIPA-ACL formalism. As mentioned before, the full FIPA communication model is implemented in JADE. Being composed by agents, PIE, PDP and PEP are able to communicate by exchanging messages. As a consequence, using an agent platform as JADE is in concurrency with other PDP-PEP communications protocols and has the advantage to already been implemented. A multi-agent system is a solution to make the reaction components communicating and collaborating without defining specific communication techniques.

IV. RESUTS

A. *Use case*

Our use case focuses on accessing files through telecommunication networks where we have to apply access or restriction, in writing, reading and executing rights on a file. To achieve that, the mechanism is based on the real application of the access rights permission on a file server for a specific user, under Windows and Linux environment.

Jade environments are called containers. Typically, in a multi-agent application, there will be several containers (with agents) running on different machines. The first container started must be the main container that maintains a central registry of all the others in order to permit that agents discover and interact with each other.

After the platform start, the following operations are executed:

1) PIE registers with its IP address to the main container.
2) PDP registers with its IP address to a container of the main container.
3) PEP also registers with its IP address to another container of the main container.
4) The policy is send to the PIE.
5) PDP receives the message (policy) from the PIE.
6) The PDP knows all PEP services, and according to this, it identifies and sends the policy to the right PEP.
7) The PEP parses the policy in order to remove necessary element for the mapping.
8) The PEP then mapped the policy to a specific execution command (setfacl).
9) The policy is finally applied.

The PEP will afterward send node after node to the console, through the communication channel the state of the policy application and whether or not it is successfully applied. The dash lines with OK annotation illustrate it on Figure 7.

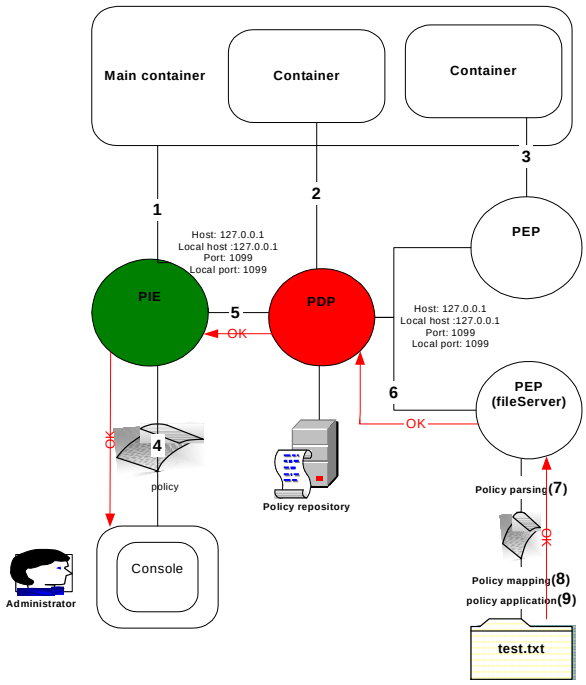

Figure 7. Use case mechanism

Practically, concerning our prototype, for a sample of the DTD (Document Type Definition), we have:

```
<!DOCTYPE POLICY[

<!--Object-->

<!ELEMENT object (target,path)>
<!ELEMENT target href #PCDATA #REQUIRED>
<!ELEMENT path  #PCDATA>

<!--Subjects-->

<!ELEMENT subject(uid?,role*,group*)>
<!ELEMENT uid  (#PCDATA)>
<!ELEMENT role (#PCDATA)>
<!ELEMENT group  (#PCDATA)>

<!--Actions-->

<!ELEMENT action (parameter*, provision*)>
<!ATTLIST action name (read¦write¦create¦delete) #REQUIRED
                                        permission(grant¦deny)
#REQUIRED>
<!ELEMENT provisional_action(parameter*)>
<!ATTLIST provisional_action name #CDATA #REQUIRED
timing(before¦after) "after">

<!--Condition-->
```

```
<!ELEMENT condition  ANY>

<!--Policy-->
<!ELEMENT policy   (property?,xacl*,policyType+)>
<!ELEMENT xacl   (object+,rule+)>
<!ELEMENT rule   (acl)+>
<!ELEMENT acl   (subject*,privilege+,condition?)>
<!ELEMENT policyType (#PCDATA)
<!ELEMENT property  (propagation?,conflict-resolution?,default?)>
<!ELEMENT propagation EMPTY>
<!ATTLIST propagation read(no¦up¦down) "down"
                      write(no¦up¦down) "down"
                      create(no¦up¦down) "down"
                      delete(no¦up¦down) "up">
<!ELEMENT conflict-resolution EMPTY>
<!ATTLIST conflict-resolution read(dtp¦ptp¦ntp) "dtp"
                              write(dtp¦ptp¦ntp) "dtp"
                              create(dtp¦ptp¦ntp) "dtp"
                              delete(dtp¦ptp¦ntp) "dtp">
<!ELEMENT default EMPTY>
<!ATTLIST default read (grant¦denial) "denial"
                  write (grant¦denial) "denial"
                  create (grant¦denial) "denial"
                  delete(grant¦denial) "denial">  ]>
```

This rule is written according to the DTD provided above and based on the access control language.

```
<policy>
 <policyType>FileServer</policyType>
 <xacl>
        <object>
        <object>
        <target>text.txt</target>
        <path>/tmp</path>
        </object>
        </object>
    <rule>
     <acl>
       <subject>
            <uid>bob</uid>
                <roles><role>Administrator</role></roles>
       </subject>
       <action name="read" permission="grant"/>
       <action name="write" permission="grant"/>
       <action name="execute" permission="deny"/>
      </acl>
  </rule>
</xacl>
</policy>
```

By applying the mentioned above policy, the administrator called Bob receives the permission to read and write on a given file ("TEST:TXT" in our case) located in a given directory ("TMP" in our case). However, the permission to execute the file is denied.

The approach presented here is based on a system that uses (1) multi agents' architecture, (2) the access rights permission on a file server, (3) the use case mechanism, (4) the policy's DTD, and (4) a policy example.

B. Implementation

The common package is composed of GenericAgent and MASPlatform classes. This package is necessary for all other packages. The application initializes service registration and deregistration into the directory facilitator. Then a PIE's agent registers into the main container and the other agents into the same or other containers. During the initialization and before starting the platform, it also checks if the configuration parameters are correct.

The following platform's configuration settings are verified:

- The name of the host where the main-container should listen
- The Port number where the main-container should listen for other containers.
- The platform-id.
- The Container local port number.
- The local port number.
- The local hosts IP address.

Figure 8. shows the two main classes used during the application implementation.

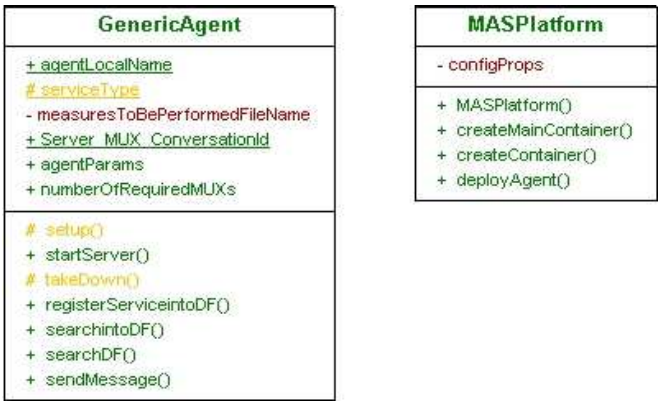

Figure 8. Common package class diagram

- **Parsing**

With the "JDOM" libraries, we implement a policy parsing class, which receives a XML policy file and extract the needed elements as for example the target, the action and the privileges concerning the policy to be applied

- **Mapping**

Policy rules are mapped to real commands. According to our use case, the prototype is focused on the Linux environment so as the kernel handles access right on files via ACL option.

- **Linux ACLs**

With this model, it is possible to give or restrict rights on a file. Basically, in the Linux environment, rights are given to the user, to the group to who belongs the file or to the other. But with the ACL's we can expand the right to a number of users and groups. Setfacl and getfacl are the basic ACL commands. Setfacl sets the rights by using the mounted ACL option of the kernel. Example:

setfacl –m u:Geronimo :rw,g :red :r-x,o:--- ./test.txt

This example enables read and write rights to the user Geronimo; it enables read and execute rights to the red group and finally no rights to the other. It is the setfacl command that we have implemented in our command mapping class. Getfacl shows the user, the group and the other files access right values. According to the version of the kernel used, we enabled the kernel to support Access Control List (ACL) by mounting the ACL option in the partition containing the files on which we want to extend rights. The policy java package creates and manages a vector with many policy rules. Each policy rule is a Hash Table and each Hash Table is characterized by its key and value. The Hash Table's keys are the XML tags and their values are the corresponding attributes. This package through its classes gets all the characteristics of the policy to be applied. The elements like the action, the policy rule and the policy type are extracted from the parsed policy file.

V. CONCLUSIONS

In this paper we have presented an architecture developed for an incident reaction system based on policy. As explained in section 4, the main advantage of this architecture is its distributed structure. Moreover, the architecture covers the requirements needs described in section II. The future works of our achievements will be the specification of a protocol, specification of the messages and thus the reaction methodology. This protocol and methodology will be dedicated to the architecture presented in this paper and should reply to the issues raised in this paper.

We have tested our approach in many configurations. For example, we launched each node of the application on a different machine of the test bed network. And it works: by this we proved the distributed property of the application. Figure 30 represents the basic case where all agents are launched on a single machine. As it is shown on this figure, the host IP address and the local host IP address are the same (127.0.0.1)

For the testing phase we made independently, executable .jar files for each agent. They are composed as followed:

The PIE agent is composed of: The common (setting, starting the platform), the configfiles (platform's configuration files), the lib (libraries, jar files used by JADE), and the PIE packages.

The PDP agent is composed of: The common (setting, starting the platform), the configfiles (platform's configuration files), the lib (libraries, jar files used by JADE), the policyfiles (policy parsing class) and the PDP packages. The PEP agent is composed of: The common (setting, starting the platform), the configfiles (platform's configuration files), the lib(libraries, jar files used by JADE), the policy(policy mapping , policy execution command class) and the PEP packages. The following figure, represents the PIE, PDP, PEP agents running.

REFERENCES

[1] Leonard J. LaPadula. "State of the Art in Anomaly Detection and Reaction" Technical Report MP 99B0000020, Mitre, July 1999.

[2] G.L.F. Santos, Z. Abdelouahab, R.A. Dias, C.F.L. Lima, E. Nascimento (Brazil), E.M. Cochra. "An Automated Response Approach for Intrusion Detection Security Enhancement" in Proceedings of Software Engineering and Applications (SEA), 2003.

[3] M. Petkac and L. Badger" Security agility in response to intrusion detection" in 16th Annual Conference on Computer Security Applications (ACSAC '00), 2000.

[4] B. Gâteau. Modélisation et Supervision d'Institutions Multi-Agents. Ph.D. Thesis, Ecole Supérieure des Mines de Saint-Etienne, 2007.

[5] Christophe Feltus, Djamel Khadraoui, Benoît de Rémont and André Rifaut, Business Gouvernance based Policy regulation for Security Incident Response. Centre de Recherche Public Henri Tudor, Luxembourg. 2007. IEEE GIIS 2007 Global Infrastructure Symposium, 6 July 2007.

[6] A. Rifaut and C. Feltus, Improving Operational Risk Management Systems by Formalizing the Basel II Regulation with Goal Models and the ISO/IEC 15504 Approach, Proceeding, REMO2V'2006, International Workshop on Regulations Modelling and their Validation & Verification, to be held in conjunction with the 18th Conference on Advanced Information System Engineering (CAiSE'06), 6 June 2006, Luxembourg.

[7] Security Policy Language : http://www.positif.org/ispl.html

[8] Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance hierarchies in the Or-BAC Model and application in a network environment. In: Second Foundations of Computer Security Workshop (FCS'04), Turku, Finland (2004).

[9] http://xml.coverpages.org/draft-seitz-netconf-xacml-00.txt

[10] H. Debar, Y. Thomas, N. Boulahia-Cuppens, F. Cuppens; Using contextual security policies for threat response. Third GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA). Germany. Juillet 2006.

[11] F. Cuppens and A. Miège, Modelling contexts in the Or-BAC model, 19th Annual Computer Security Applications Conference, Las Vegas, December, 2003

[12] IDMEF/RFC4765, Network Working Group: Hervé Debar, France Telecom; D. Curry, Guardian; B. Feinstein, SecureWorks, Inc.; March 2007

[13] http://www.rfc-archive.org/getrfc.php?rfc=4765.

[14] N. Damianou, N. Dulay, E. Lupu, M. Sloman , The Ponder Policy Specification Language Workshop on Policies for Distributed Systems and Networks (Policy2001), HP Labs Bristol, 29-31. Springer-Verlag.

[15] E. Bulut, D. Khadraoui, and B. Marquet, Multi-Agent based Security Assurance Monitoring System for Telecommunication Infrastructures in Communication, Network, and Information Security conference (CNIS 2007), Berkeley, California, USA, september 2007.

[16] FIPA, http://www.fipa.org/

[17] FIPA, "Agent Communication Language", FIPA Specification, November 1997

[18] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa. JADE - A FIPA-compliant agent framework, CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, April 1999, pp.97-108

[19] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE - A White Paper. Sept. 2003