An Autonomic Ontology-Based Multiagent System for Intrusion Detection in Computing Environments

Ryan Ribeiro de Azevedo,
Eric Rommel Galvão Dantas, Fred Freitas
Center of Informatics - Federal University of Pernambuco,
Semantic Web and Ontology Research and Development Group - SWORD

Cleyton Rodrigues, Marcelo J. Siqueira C. de Almeida, Wendell Campos Veras and Rubean Santos
Center of Informatics - Federal University of Pernambuco
Semantic Web and Ontology Research and Development Group - SWORD

Abstract

Computational security has become a worry among corporations. Indeed, the use of specific tools, procedures and policies that cover requirements to keep computation and IT infrastructure protected of malicious agents has been proved necessary. In order to address the problem of protecting computational resources, in this work we propose an autonomic model called AutoCore, constituted by a Multi-Agent system, an Intelligent Interface (CoreEditor) and a formal ontology (CoreSec). This proposal has been implemented, tested and validated in real scenarios to assist the safety activities and to minimize the administrator complexity.

1. Introduction

The various fields of human action, such as military, business or academics, need transparent means to plan and manage problems and information related to computational security, specifically the information security. The problems regarding information security have become a strategic issue for business. IBM (International Business Machines) presented a manifesto [1] alerting the industry of Information Technology (IT) to a new crisis. According to it, the computational systems have evolved considerably since that they had appeared, at the same time they had become more complex to be managed. As some evidences, it highlights: the security of critical mission systems, the increasing need for interconnectivity, the integration of heterogeneous technologies and finally, the efficient allocation of computational resources [2]. Still according to this manifesto, this complexity is incompatible with the vast majority of IT professionals, which generates very high costs, jeopardizing the quality of the services offered [3]. Therefore, nowadays security has evolved to a crucial and, in consequence, a complex task of design and plan, requiring the adoption of means for handling, processing and interpreting the information.

On the other hand, the set of technologies that make the Semantic Web, along with systems based on intelligent agents and autonomic computing, allows computational systems to handle, interpret and process this information through ontologies, which have been applied in various contexts that include sharing, organization and use of semantic information. Ontologies have contributed to facilitate the communication and processing of semantic information, both between agent-based systems as among humans, thus, promoting interoperability among systems when representing data shared by several applications [4]. With a base of knowledge regarding issues related to security, the corporations will be able to develop and deploy more easily mechanisms for protection, correction and prevention in accordance with its security requirements. Such requirements are supported by Service Level Agreements (SLA) or by security policy required, so that their services are always available, fair and reliable.

In this context, a solution was proposed based on the human autonomic nervous system, known as autonomic computing [1]. The autonomic nervous system is responsible for, amongst other things, maintain the rhythm of the heartbeat, control the body temperature and blood pressure, independently, that is, without the human-being conscious intervention. Therefore, as the autonomic nervous system, an autonomic computing system must perform tasks related to administration, automatically and transparently, keeping its integrity, thus having little or no human intervention.

In this work we present the *AutoCore*, a multi-agent system based on Ontologies to support the activities undertaken by those responsible to manage computational security in corporative environments. Its purpose is to provide a formal model that reflects the security aspects for this type of environment, allowing a safe communication between the various elements of the autonomic system and enabling a process of inference concerning the different events that can put the security of systems at risk. Basically, AutoCore consists of:

- A domain-ontology with more generic and abstract concepts in the field of security, serving as the basis for construction of other specific security-domain-ontologies called *CoreSec*;
- A system called *CoreEditor* based on intelligent agents that take independent decisions, concerning optimization, configuration and repair of a resource, monitored by them.

The rest of this article is organized as follows: Section 2 presents some related works. Sections 3 and 4 present a theoretical referential for the general understanding of the theory concerning the proposed work: Autonomic Computing and Ontologies, respectively. Already in Section 5, we introduce the proposal, its architecture, components and functionalities. In Section 6, the results obtained in experiments carried through with AutoCore are presented and discussed. Finally, conclusions and future work are outlined in Section 7.

2. Related Work

In recent years, some efforts have been spent to obtain autonomic models that allow assisting the complexity to protect computational systems and IT security, providing, therefore, different levels of protection to assets of a corporation. Albeit with different focus and levels of detail, some works of literature deal with autonomous systems able to manage, evaluate and specify security, either of the information or not, in the most diverse environments. Amongst these works that had helped on the maturity and support in the development of the proposal here described, we highlight a few below.

The work [5] proposes the use of an ontology for information security management in autonomic computing environments called OSAKA. Despite the merits, OSAKA is a premature work. The ontology was not implemented using Semantic Web languages such as Resource Description Framework (RDF) [6] or Web Ontology Language (OWL) [7]. Further, it has no restrictions or axioms in DL (Description Logic) that are *de facto* the standards established. The work also does not present results, not ensuring its practical usefulness.

The work [8] has developed an ontology named *OntoSec*, with a standardized way of representing information about security incidents, facilitating sharing and reuse of this information, management and generation of knowledge of incidents and a tool to aid those responsible for security, the CSO, Chief Security Officers, where queries are held to obtain answers to possible incidents. The main difference between the OntoSec and AutoCore is that the former is only an ontology of application and, therefore, not related to the domain of security as the *CoreSec* of AutoCore, and the system developed to assist the CSO is not based on agents, not having the autonomy neither to perform preventive corrections nor to analyze attacks in real time.

It is worth to reference also the *eAutomation* [9], which was developed focusing a system on the correlation of the same name, but it does not address specifically the issue of information security, beyond describing a reduced set of classes in the proposed ontology.

The approach presented in this article shows significant improvements with respect the related work. In the next sections, the theories, namely autonomic computing and ontologies, are introduced.

3. Autonomic Computing

In 2001, IBM [1] presented to market a solution to the problem of excessive complexity of corporative software, known as autonomic computing. The idea follows a common approach, like the various branches of engineering, which is to seek inspiration in the human autonomic nervous system, which is able to maintain vital functions in order to understand and solve many different types of problems without any or little initiative, participation or direct human intervention.

An autonomic computing system can be defined as being able to manage itself in accordance with the high-level goals set by the administrator [2]. One of the main purposes of this computation is the ability of self-management, leaving the system administrator free, with respect the concerning about the operational details of the system and, further, allowing the use of the machines with better performance along all the time.

Since the currently systems are too dynamic, with constant changes, it is essential that they can adapt to changes arising from new demands, business politics and security restrictions, as far as they are occurring. Therefore, a system can be classified as an autonomic computing if it is able to fulfill four basic services in accordance with [10]:

- *Self-Configuring*: ability of the system to configure itself at run time, with little or no human intervention. Thus, the IT environment can be dynamically tailored to the new settings, by the inclusion, update or removal of components;
- *Self-Optimizing*: ability of the system in optimizing the allocation of resources, so that users' needs are met without compromising the other features. The idea is that, for example, issues related to performance and quality of service can be handled in accordance with high-level policy in a transparent way;
- *Self-Healing:* ability of the system to identify failed situations and implement corrective actions, without stopping the execution of services. Thus, the system will be able, after the occurrence of a failure, to return to a stable state without the administrator intervention;
- *Self-Protecting:* ability of the system to detect malicious or hostile behaviors, taking efficient and effective decisions transparently, about which is the most appropriate action to prevent or minimize an attack.

It is necessary that these four requirements work in synergy to reach the self-management. For example, during the automatic insertion of a new element in the environment, the responsible part for self-configuring must configure it and notify the responsible for self-protection on this event so that the necessary measures to protect the security are taken.

The autonomic system architecture is based on the theory of control, together with tools for planning. It comprises two main parts [11]:

- A functional unit that performs a typical IT environment operations, such as processors, databases managers systems and servers;
- A management unit, which monitors constantly the operation of the functional unit. This unit must ensure that the functional one is always safe, performing, amongst other things, the analysis of the flow of input and output requests in order to detect malicious behavior, and, thereby, preventing the attacks.

4. Ontologies

Since century XVII, the term ontology has been used to name the general discipline of metaphysics, in the traditional *first philosophy* of Aristotle, as the science of *being qua being*. It is, many times, faced as a complement to the idea of epistemology (science of knowledge) [12].

Various definitions have appeared to describe what an ontology in the computer field is. The best known is "a formal and explicit specification of a shared conceptualization" [13], where:

- Formal refers in being declaratively defined, therefore, understandable for agents and systems;
- Explicit means that the elements and their constraints are clearly defined;
- Conceptualization deals with an abstract model either of an area of knowledge or a limited universe of speech;
- Shared means a consensual knowledge, i.e., a common terminology of the modeled area, or agreed amongst the developers of the communicative agents.

Thus, ontologies, specified in a higher level of abstraction, provide a common and non-ambiguous terminology for the domain under discussion.

For Guarino [14], ontologies are a computational artifact composed of a vocabulary of concepts, their definitions and possible properties, a graphical model showing all possible relationships amongst the concepts and a set of formal axioms that constrain the interpretation of concepts and relations, representing in a clear and unambiguous way the knowledge of the domain. Several advantages have been presented in literature for the adoption of ontologies. Amongst them, the following are highlighted [15]:

- It encourages the developers to reuse knowledge, even with adaptations and extensions. This is explained by the fact that the construction of bases of knowledge is one of the most expensive, complex and slow task of a specialist and/or agent system. Therefore, reusing ontologies promotes a significant gain in terms of effort and investment;
- The wide availability of "ontologies-off-the-shelf", ready for use, reuse and communication between agents, which may be further extended and complemented with concepts of specific domains;
- Online access to servers of ontologies, capable of storing thousands of classes and instances, giving support to companies or research groups, once they work as tools to keep the integrity of shared knowledge between them, ensuring thus, a uniform vocabulary.

Recently, the use of ontologies has been spread through several other sub-areas of Computer Science, such as: Software Engineering, Database and Information Systems, motivated by the Semantic Web (which is indeed a direct consequence of the use of ontologies [16]).

5. Proposal

In this work we present the AutoCore, an autonomic ontology-based multi-agent system, which provides a flexible and adaptive system to the environments in which it is part of to support real time computational security.

The computational security of a corporation will be more efficient if it is based on autonomic models, such as AutoCore. Therefore, it aims to be the basis for possible security solutions as well ass to assist those responsible for the Management of Information Security in its current activities.

AutoCore was implemented using the JADE[1] - Java Agent Development Framework. The agents are organized in containers: a set of agents modeled in accordance with the basic autonomic system services (Self-Optimizing, Self-Healing, Self-Protecting and Self-Configuring) as explained in section 3. The AutoCore makes use of CoreSec as an ontology knowledge base, which is a domain-ontology with high-level concepts for information security, giving aid the management of IT. CoreSec is itself a common base of information, which improves the processing and use of this knowledge. Furthermore, it reveals a shared and explicit knowledge between those involved with the activities of computational security. A snapshot of a part of CoreSec is depicted out in Figure 1.

As a domain-ontology, the CoreSec enables the creation of domain-specific sub-ontologies such as ontologies of attacks, vulnerabilities and incidents of computational security.

Figure 2 shows the AutoCore architecture and components. The system consists of a Manager Agent, responsible for defining the objectives and tasks to be performed by each type of agent: Checkers, Appraisers, Actuators and Viewers.

[1] *http://jade.tilab.com/.*

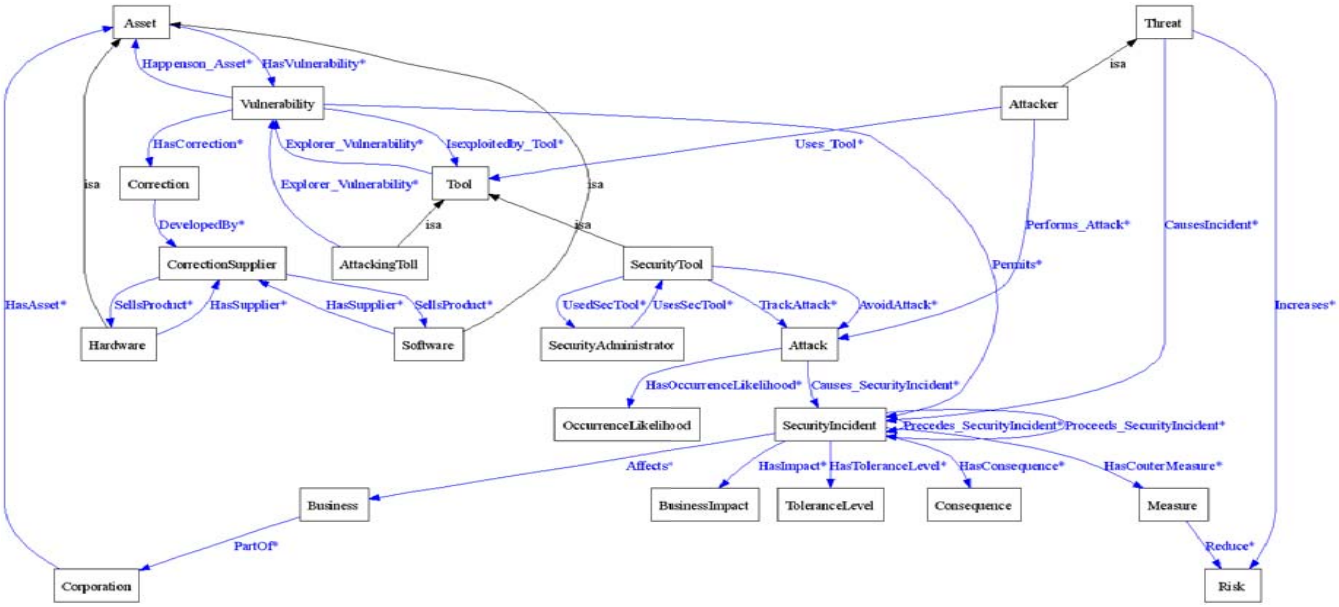

Figure 1.Some Classes and Relationships of CoreSec Ontology

The Manager agent enables self-configuration, since they provide for the colonies, at runtime, the rules to be followed. The Checkers are in constant analysis of the related resources, and they are also responsible for retrieving and analysis of network traffic, besides detecting and classifying hostile behavior. For each category of attack, there is a set of specialized agents, enabling the detection of possible malicious activities. At the moment they are instantiated, these agents consult a sub ontology, created from the CoreSec, and keep in an internal memory all current settings for the detection of abnormalities associated with resources. In case that any non-expected change is detected, through the CoreEditor, the ontology is updated with the new incoming information. The Checkers communicate with the Appraiser agent, sending the status of the system. Together, these agents set the self-protection, since while the former detects a bad functioning, the second takes efficient decisions to solve the problem.

Once the status of the environment was received, the appraisers consult the CoreSec to find the best action to be taken, and then, they send the new information to Actuators agents. In fact, Self-Optimization is possible due the queries carried out by the appraisers, since they allow the optimization of resources without compromising the others. The appraisers build solutions based on metrics for the resolution of some vulnerabilities detected. Following, they send to Actuators agents some indicators such as: Impact on Business, Time Resolution, Effectiveness, Efficiency, Quality and Resolution cost.

The Actuators, in turn, receive the assessments and execute the healing action. Any action performed, even those not running automatically as a measure of prevention, can be viewed by the CSO using the CoreEditor, through the viewers

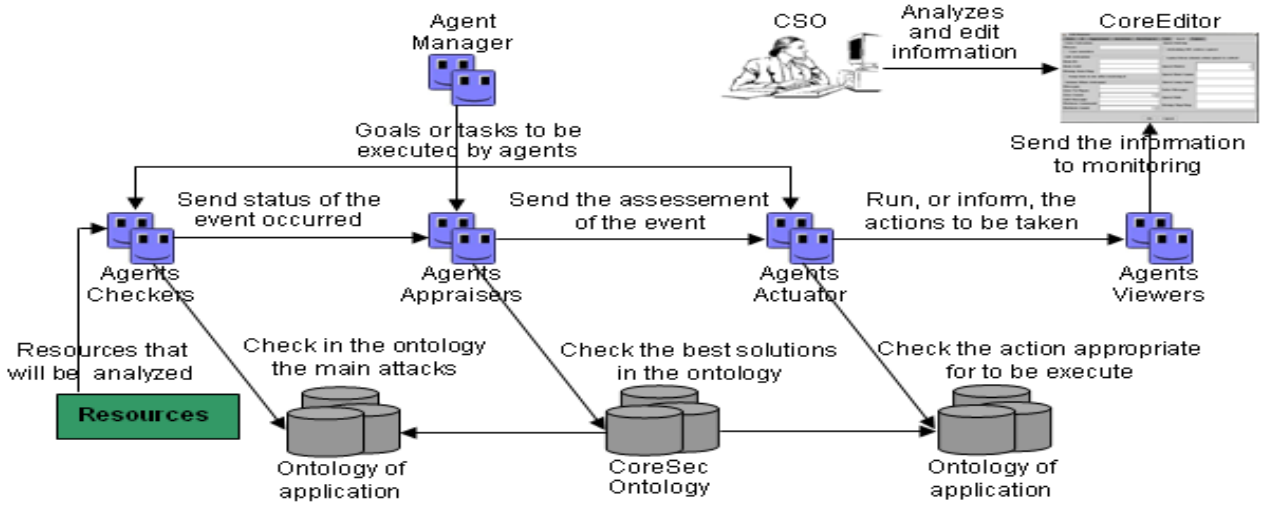

Figure 2. AutoCore Architecture

agents. It is through this CoreEditor that the CSO team may intervene in any action requested by actuator agents. These agents also carry out self-healing, since after identifying the failures, they perform preventive and corrective actions without paralyze the functioning of services, allowing return to a stable state without human intervention. These agents also consult a sub-ontology, created from the CoreSec, always before executing an action.

Viewers are the agents responsible for displaying graphs and information reports through the CoreEditor. Thus, those responsible for information security will be always aware regarding the current status of the system, e.g., possible interventions at critical moments to the systems are idealized from the view obtained.

An application called CoreEditor for exclusive use of the CSO was developed in order to assist the monitoring and feeding of the knowledge base (CoreSec) used by AutoCore. CoreEditor gives support to transmission, generation and distribution of knowledge, handling, evaluation and use of CoreSec. It is worth to note that the CoreEditor is not only a tool for manipulation of CoreSec, but also, a framework for creation and edition of ontologies. It was developed using the Jena framework, produced by Brian McBride of Hewlett-Packard for the creation and manipulation of RDF graphs [17]. The CoreEditor is an exclusive application for manipulation of the knowledge base when a human intervention is necessary.

The key features and functionalities of CoreEditor are: the language OWL to create ontologies (concepts, properties and individuals) and the Pellet inference engine for generation of hypotheses from the information in the knowledge base; the manipulation/generation of Java code for applications of the developed ontology; the generation of the *OntoDoc*, similar to Java javadoc as well as UML class diagrams of the developed ontology and a query interface using the language SPARQL[2].

The application has the following main modules:

- *Inference Module:* It allows to infer new information concerning the CoreSec on OWL and RDF by the mean of Pellet inference engine;
- *Query Module:* The module in which the queries are carried through using the CoreSec as knowledge base, by users of the application. The queries are performed through the SPARQL language and they are transparent to the end user. This module interacts with the inference one.
- *Manipulation Module:* It is the responsible for creating, editing and deletion of classes and subclasses, properties (Data Type and Objects), instances, types of relationships (Transitive, Symmetric, Functional and Inversely Functional), creation of new ontologies, RDF, OWL and UML code generation, besides Java code on the ontology developed.
- *Viewer Module:* It allows the visualization of graphs with instances, inheritances and other types of relationships, configured according to the needs of end users through the integration with the graphics generator: *Graphviz*[3];
- *Ontology Engineering Module:* It allows the creation of ontologies with support the *Methontology* methodology [18]. Furthermore, the ontologies are created and documented automatically. This module also has applications for reengineering, since one can use predefined developed ontologies.

The experiments and results using AutoCore are presented in the following section.

6. Experiments and Results

The experiments were performed for Denial of Service (DoS) attacks SYN Flooding, which are characterized for the sending of multiple requests to open connection to a single host on a single port, in a short period of time. DoS attack goal is to let the network unavailable [19]. This type of threat is easily identified, however, very difficult to prevent.

We have developed a simulator for this type of attack, in which we can set up: the amount of requests to the server, the attacked port and the attacked IP. Figure 3 depicts the simulator running.

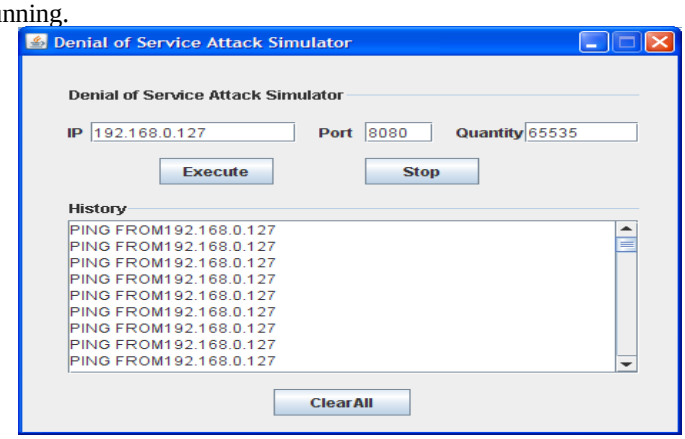

Figure 3: DoS attack Simulator

IP and *Port* fields are filled with the IP address and the target port of the DoS attack, respectively. The amount of requests is informed in the *Quantity* field. When you click in the *Execute* button, the simulator starts the DoS attack in accordance with the data informed. The *Stop* button ends the requests before reaching the set amount. The *History* field presents a description of requests made and the *ClearAll* button, in turn, excludes this history information. At the moment the AutoCore is initialized the agents are created as shown in Figure 4. For the sake of clarity, we have used the tool to monitor the process of packet communication (*sniffer*) of JADE.

The Viewer Agent processes the event to request the system boot. Then it sends the startup request to the Manager

[2] *http://www.w3.org/TR/rdf-sparql-query*

[3] *http://www.graphviz.org/*

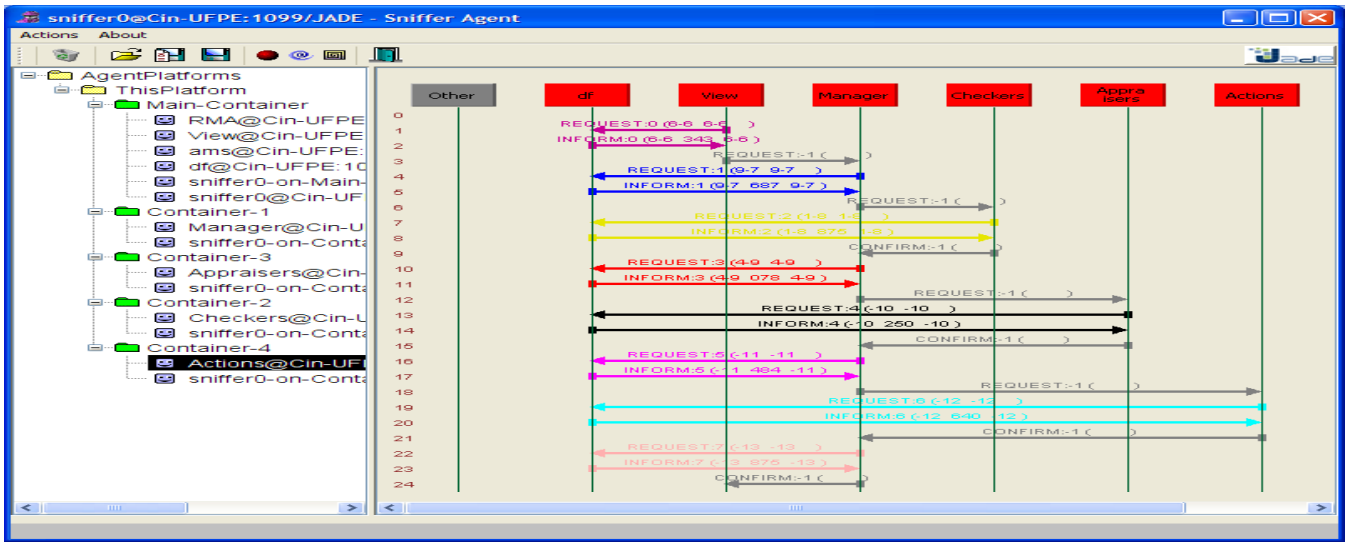

Figure 4: Startup and Creation of Agents

Agent and it, in turn, initializes the Checkers, Appraisers and Actuators. Before the connection with each agent, a query is made to the *Directory Facilitator* (DF), which is responsible for telling the agents identifier (ID), allowing therefore the communication and exchange of messages. Messages are encoded using the protocol of interaction *contract-net* [20], and use as vocabulary, the ontology CoreSec. After the confirmation of the initialization of each agent, the Manager confirms this information to Agent Viewer, and then the startup event is finished. Figure 5 shows the process of communication between AutoCore agents at the moment of the simulation of a DoS attack using the simulator.

Checkers Agents are responsible for checking the network traffic. At the moment they perceive a high rate of packets from the same host (taking into account the size and type of packets in a given period of time), they detect an anomaly, based on the rules in the knowledge base of an instance of CoreSec, loaded during startup. After consulting the DF to know for which agents must send these information, they send a request for the Appraisers agents with the (abnormal) network status.

The Appraisers, based on some descriptions informed by the Checkers, consult the CoreSec to evaluate the anomaly.

After consulting the DF, they communicate with the Actuators. As a possible solution can be found in the instance of CoreSec, they again consult it seeking specific information to resolve or, at least, to soften the anomaly. Firing the best action based on metrics received by the Agents Appraisers characterizes the Self-Healing. After AutoCore has detected a possible DoS attack, the following actions are taken by the agents: forward the packets to an invalid application, changing the flow of attack and releasing the application for use and inform the CSO on the occurrence of attack. Consequently, the agents reveal the source of forged packets, thereby assisting the creation of concise rules for the firewall configuration.

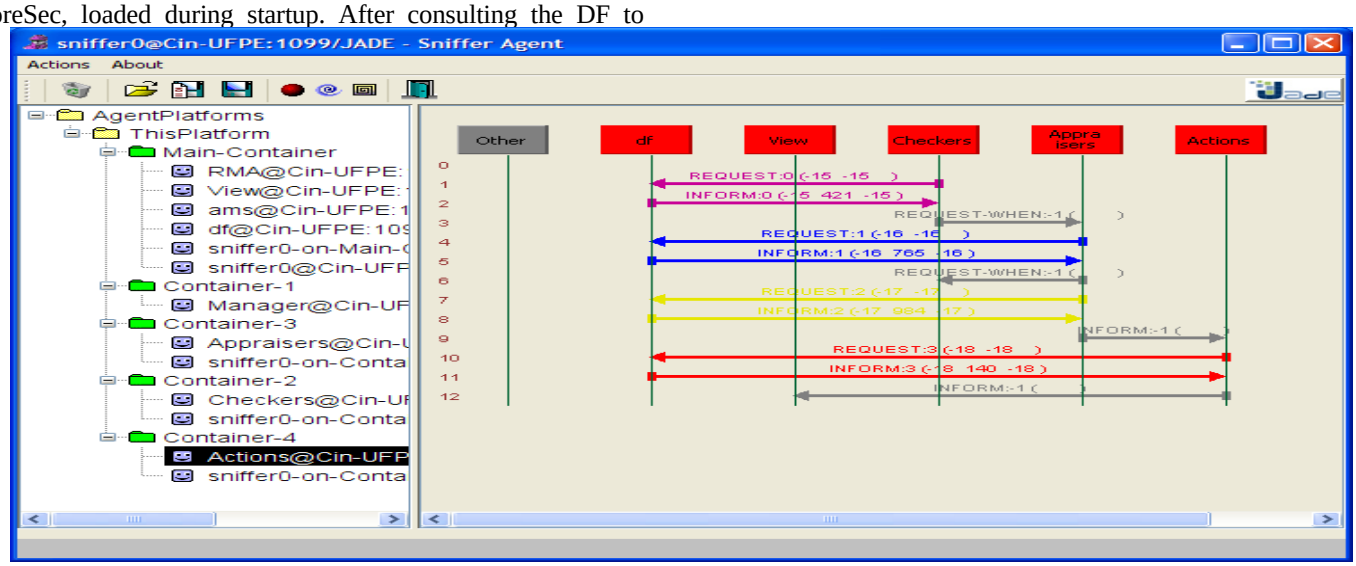

Figure 5. Communication between agents in the detection and correction of an attack

After consulting the DF, the Actuators send a message to the Viewers (remember that they interact with the CoreEditor) with the current situation, if a healing action has been done or not, and some other possible fixes to CSO intervene in the system, if necessary.

As the AutoCore has the CoreEditor in the visualization layer, it is feasible to use it to monitor and support the Risk and Security Management, thereby keeping the services offered by the available organizations without losing competitiveness in the market. Some queries were defined based on the following questions of ability that CoreSec should be able to answer: if a malicious agent explores any vulnerability and performs a DoS or a Distributed Denial of Service (DDoS) attack, for example, what is the consequence of these attacks for the organization? What is the probability of occurrence of this attack? What is the impact on business? What assets are affected? What level of tolerance is allowed? What controls to use to mitigate the exploitation of such vulnerability? What level of risk is acceptable?

The instances used to feed CoreSec were obtained from NIST (National Institute of Standardization and Technology) of the United States. This is an international database of vulnerabilities, a repository of free access, containing information about each vulnerability and how to fix them. By using the CoreEditor Query module, some queries were performed and the results are presented in Table 1. The queries that address the issues of abilities were defined according to what the experts should be able to answer, in order to mitigate real risks in their corporations. According to the results of the queries, those responsible for security should take strategic decisions to mitigate risks, besides maintaining the services offered by organizations available without losing competitiveness in the market.

Table 1. Results of queries using CoreEditor

Malicious Entity	Attack	Vulnerability	Resource	Occurrence Probability	Business Impact	Level Tolerance	Healing Action
Hacker	**UDP Packet Storm**	**Distributed Denial of Service**	**Servers**	**Hard**	**Hard**	**Not Acceptance**	**- Analyze Network Traffic (Packets TCP and UDP); - Use IDS.**
Hacker	**SYN Flooding**	**Denial of Service**	**Servers**	**Hard**	**Hard**	**Not Acceptance**	**- Analyze Network Traffic (Packets TCP and UDP); - Use IDS.**
Nature	**Fire**	**Lack of protection against fire**	**Servers Room**	**Low**	**Hard**	**Acceptance**	**- Fireproof coffer; - Duplication of servers in remote locations**

By observing Table 1, note that priority should be given to DDoS and DoS attacks, since they have high probability of occurrence, high impact on business and does not have a acceptable tolerance for the organization. For the attack called "Fire" would not be given any priority because the probability of occurrence is low and the risk in this case is acceptable. However, the impacts and values of security are high. Those responsible should consider levels of acceptance to take decisions to control risk. Some classes were not displayed in the results table for reasons of space. Without the use of AutoCore, decisions are taken manually, without the support of a knowledge base or intelligent computational tool, thus, jeopardizing the decision-making on security.

7. Conclusions and Future Work

AutoCore fulfills the gap for which it proposes, since it is a computational tool that can be used for the treatment and use of risks and security information, enabling those responsible for Risk Management and Information Security Management to take strategic decisions of aligning IT and security to business processes of organizations, providing a high level view for the involved ones.

The results are encouraging and indicate success in the detection and correction of DoS attacks, besides aiding those responsible for the computational security of a corporation to take efficient decisions. It was possible to carry out the assessment and validation of AutoCore, resolving a real problem that affects many organizations. As expected, the proposed system behaved according to the characteristics of an autonomic system, thus improving the complexity of relative procedures of tools that implement similar mechanisms of security.

As future work, we are working on the detection of other types of attacks such as DDoS and variants of the DoS, SQL Injection, Buffer Overflow and others, extending the number of case studies carried out. Another task will be the junction of AutoCore with firewalls. Moreover, some new classes are being added to CoreSec as well as some features in CoreEditor, as the module of reports, giving further support to corporative management level.

8. References

[1] Horn, P. 2001 “Autonomic Computing: IBM´s Perspective on the State of Information Technology”; http://www1.ibm.com/industries/government/doc/contet/ resource/thought/278606109.html. Last access in 6th February 2009.

[2] Kephart, Jeffrey O. and Chess, David M. 2003. The Vision of Autonomic Computing. IEEE Computer Society, January, 41-50.

[3] IBM - International Bussiness Machine. 2006. An Architectural Blueprint for Autonomic Computing. Avaliable inhttp://www 03.ibm.com/autonomic/pdfs/ACBP2_2004 10-04.pdf. Last access in 15th March 2009.

[4] Uschold, M, Grüninger, M. 1996. *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review. Vol. 11, Nº 02. June.

[5] Almeida, M. J. S. C.; et al. 2007. An Ontology about Information Security Management for Autonomic Computing Environments. In: 2nd Latin American Autonomic Computing Symposium (LAACS 2007), 2007, Petrópolis - RJ. Proceedings of the 2nd Latin American Autonomic Computing Symposium (LAACS 2007), 2007.

[6] RDF. Web ontology language overview - w3c;. *[Online]. Available: http://www.w3.org/RDF/ [acessed Feb. 2009]*.

[7] OWL. Web ontology language overview - w3c;. *[Online]. Available: http://www.w3.org/TR/owl-features/ [acessed Feb. 2009]*.

[8] Martimiano, L. A. F. 2006. *Sobre a estruturação de informação de segurança computacional: o uso de ontologia.* 163 p. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC, Universidade de São Paulo - USP, São Carlos.

[9] Stojanovic, L. et al. 2004. The Role of Ontologies in Autonomic Computing Systems. IBM Systems Journal, Volume 43, Issue 3.

[10] Parashar, M. and Hariri S. 2007. Autonomic Computing: Concepts, Infrastructure, and Applications. CRC Press, USA.

[11] Chess, D.M., Palmer, C. C. e White, S.R. 2003. Security in an Autonomic Computing Environment. IBM Systems Journal, Volme 42, Issue 1, 2003.

[12] Freitas, F; Schulz, Stefan ; Moraes, Eduardo. 2009. Survey of Current Terminologies and Ontologies in Biology and Medicine. In revista eletrônica de comunicação, informação e inovação em saúde. Vol. 3, p. 1-13.

[13] Gruber, T. R. 1995. Toward Principles for the Design of Ontologies used for Knowledgement Sharing. In International Journal of Human-Computer Studies.Vol 43, Inssue 5-6: 907-928.

[14] Guarino, N. 1998. Formal Ontologies and Information Systems. In Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Itália. 6-8 june. P3-15. IOS Press.

[15] Freitas, F. 2003. Ontologias e a web semântica. In: Renata Vieira; Fernando Osório. (Org.). *Anais do XXIII Congresso da Sociedade Brasileira de Computação*. Campinas: SBC. Vol. 8, p. 1-5.

[16] T. Berners-Lee, O. Lassila, and J. Hendler. 2001. The semantic web. *Scientific American*, 5:34–43.

[17] Carroll, J, J. Carroll, et al. 2004. Jena: implementing the semantic web recommendations. In WWW (Alternate Track Papers & Posters), 2004.

[18] Fernández, M. A.; Gómez-Pérez, A.; Juristo, N. 1997. Methontology: From ontological art towards ontological engineering. *In Proceedings of the AAAI Spring Symposium Series*, p. 33-40.

[19] Khadraoui, D. e Herrmann, F. 2007. Advances in Enterprise Information Technology Security, Books24x7: IGI.

[20] FIPA (2002), FIPA Contract Net Interaction ProtocolSpecification,http://www.fipa.org/specs/fipa00029/SC0029H.pdf.

