AIDeM: Agent-Based Intrusion Detection Mechanism

Cristian Pinzón, Martí Navarro, and Javier Bajo

Abstract. The availability of services can be comprimised if a service request sent to the web services server hides some form of attack within its contents. This article presents AIDeM (An Agent-Based Intrusion Detection Mechanism), an adaptive solution for dealing with DoS attacks in Web service environments. The solution proposes a two phased mechanism in which each phase incorporates a special type of CBR-BDI agent that functions as a classifier. In the first phase, a case-based reasoning (CBR) engine utilizes a Naïves Bayes strategy to carry out an initial filter, and in the second phase, a CBR engine incorporates a neural network to complete the classification mechanism. AIDeM has been applied within the FUSION@ architecture to improve its current security mechanism. A prototype of the architecture was developed and applied to a case study. The results obtained are presented in this study.

Keywords: Availability, Web Service Attack, Multi-agent, case-based reasoning.

1 Introduction

Security is one of the primary concerns in service oriented architectures (SOA) and Web services [1]. Some protective measures such as Web Service Security (WSS) [2], WS-Policy [3], WS-Trust [4], etc. focus on authorization and authentication aspects to ensure confidentiality and integrity. However, they do not contemplate security problems that put the availability of Web services at risk. The

Cristian Pinzón
Universidad Tecnológica de Panamá, Av. Manuel Espinosa Batista, Panama
e-mail: cristian.pinzon@utp.ac.pa

Martí Navarro
Departamento de Sistemas Informáticos y Computación, Universidad Polítecnica de Valencia, Camino de Veras s/n, 46022, Valencia, España
e-mail: mnavarro@dsic.upv.es

Javier Bajo
Escuela Universitaria de Informática, Universidad Pontificia de Salamanca, Compañia , 5 37002, Salamanca, Spain
e-mail: jbajope@upsa.es

Y. Demazeau et al. (Eds.): Trends in PAAMS, AISC 71, pp. 347–354.
springerlink.com

emergence of new threats that can interrupt the correct functioning of services is closely related to some of the components contained in this technology, such as the XML standard used to encode messages, and the hypertext transfer protocol (HTTP) used to the communication. Different types of threats, similar to denial of service (DOS) attacks, can incapacitate a web service and block access to authorized users by sending malicious requests to the web server.

This study presents AIDeM, an advanced detection method that can confront mechanisms or techniques that produce denial of service attacks within Web environments. AIDeM is intended to improve the initial security level within the FUSION@ architecture [5]. FUSION@ proposes a new and easier method to develop distributed intelligent ubiquitous systems, where applications and services can communicate in a distributed way with intelligent agents, even from mobile devices, regardless of time and location restrictions. FUSION@ did already include a security component within its structure consisting of an agent specialized. However, the security method employed by this agent is limited in scope making available services vulnerable to attack. AIDeM is based on a group of agents specially designed to work together intelligently and adaptively to solve the problem of the reliability of SOAP messages sent in service requests. The core of AIDeM is a classification mechanism that incorporates a two-phase strategy to classify SOAP messages. The first phase applies an initial filter for detecting simple attacks without requiring an excessive amount of resources. The second phase involves a more complex process that ends up using a significantly higher amount of resources. Each of the phases incorporates an intelligent agent that integrates a CBR engine with advanced classification capabilities. The idea of a CBR mechanism is to exploit the experience gained from similar problems in the past and then adapt successful solutions to the current problem [6]. The first agent uses a Naïves Bayes classifier and the second a neural network, each of which is incorporated into the respective re-use phase of the CBR cycle. As a result, the system can learn and adapt to the attacks and the changes in the techniques used in the attacks. Additionally, a strategy of a two phased classification mechanism is to use its resources (CPU, cycle, memory) and improve response time.

The rest of the paper is structured as follows: section 2 presents a general description of the FUSION@ architecture and the limitations of the current mechanism of security. Section 3 focuses on the details of the AIDeM architecture and the mechanism of classification. Finally, section 4 describes how the classifier agent has been tested inside a multi-agent system and presents the results obtained.

2 FUSION@ Architecture and Current Mechanism of Security

FUSION@ [5] combines a services-oriented approach with intelligent agents to obtain an innovative architecture that facilitates ubiquitous computation and communication, and high levels of human-system-environment interaction. It also provides an advanced flexibility and customization to easily add, modify or remove applications or services on demand, regardless of the programming language. FUSION@ framework defines four basic blocks: a) Applications represent

all the programs that can be used to exploit the system functionalities. They can be executed locally or remotely. b) Services represent the activities that the architecture offers. c) Agents Platform is the core of the architecture and integrates a set of agents, each one with special characteristics and behaviours. In FUSION@ services are managed and coordinated by deliberative BDI agents with distributed computation and coordination abilities. d) Communication Protocol allows applications and services to communicate directly with the agents platform. The protocol is completely open and independent of any programming language, facilitating ubiquitous communication capabilities. This protocol is based on SOAP specification [7] to capture all messages between the platform and the services and applications. Developers are free to use any programming language. The only requirement is that they must follow the communication protocol based on the transactions of XML (SOAP) messages. The communication among agents in the platform follows the FIPA Agent Communication Language (ACL) specification.

FUSION@ is a modular multi-agent architecture, where services and applications are managed and controlled by deliberative BDI (Belief, Desire, Intention) agents [8] [9]. There are different kinds of agents in the architecture, each one with specific roles, capabilities and characteristics:

- CommApp Agent. Responsible for all communication between applications and the platform.
- CommServ Agent. Responsible for all communications between services and the platform. This agent also periodically checks the status of all services to know if they are idle, busy, or crashed.
- Directory Agent. Manages the list of services that can be used by the system.
- Supervisor Agent. Supervises the correct functioning of the other agents in the system.
- Security Agent. Analyzes the structure and syntax of all incoming and outgoing XML messages.
- Admin Agent. Decides which agent must be called by taking the QoS and user preferences into account. Admin Agent has a routing list to manage messages from all applications and services. This agent also checks if services are working properly to ensure that QoS is always current.
- Interface Agent. This particular agent was designed to be embedded in user applications.

Developers can add new agent types or extend the existing ones to conform to their projects needs. However, most of the agents' functionalities should be modelled as services, releasing them from tasks that could be performed by services. More specific details of the architecture can be found in [7].

Security is considered an important element within the FUSION@ architecture. As a result FUSION@ incorporates a security mechanism that validates all incoming messages dealing with service requests. The strategy is centered on the role of the Security agent and its attempt to protect services that are facing potential attacks hidden within service requests (embedded within a SOAP message). The security mechanism contained within the Security agent is simple and efficient for known attacks, although it presents a series of limitations when it comes to

protecting the architecture and services during a more complex attack. The majority of the attacks made against service based environments use complex techniques that are difficult to detect with a simple XML code review found in the SOAP message. One example of a complex attack directed at service based environments is the denial of service attack, which can implement many mechanisms of attacks within service based environments [1]. A DoS attack causes the resources available in the server of the provider (memory and CPU cycles) to be drastically reduced or exhausted while a malicious SOAP message is being parsed.

In summary, the initial security mechanism incorporated within FUSION@ presents the following limitations: a) It can cause a bottleneck during an instance of high service requests and negatively affect the architecture's performance. b) Its strategy can only detect and block a limited number of known attacks, and cannot handle attacks that are more complicated in nature. c) The security mechanism is incapable of adapting to new attack patterns. This limitation prevents the security mechanism from confronting new attacks or fast-paced changes in known attack patterns.

3 AIDeM: Agent-Based Intrusion Detection Mechanism

AIDeM is based on the incorporation of a new security block composed of a set of agents with special capabilities. The new proposed mechanism is based on our previous research in SQL injection attacks [10] [11] which developed a multi-agent SQLMAS architecture. In this way, some resources are reused and the knowledge acquired from previous work is adapted in order to provide an evolution of the mechanism proposed.

Figure 1 presents the AIDeM architecture. As shown in Figure 1, AIDeM is comprised of the Security and Admin agent, both of which were already included

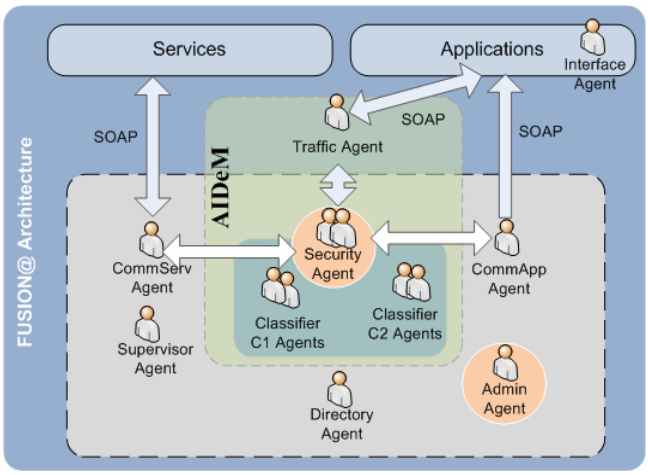

Fig. 1 Representation of AIDeM with the set special agents

in the FUSION@ architecture, as well as the Traffic agent and two others that will function as classifiers: C1 agent and C2 agent. AIDeM was designed as a two phased classification mechanism for classifying SOAP messages, as explained by [10] and [11]. The first phase applies the initial filter for detecting simple attacks without requiring an excessive amount of resources. The second phase involves a more complex process which ends up using a significantly higher amount of resources. This two-phased strategy improves the overall response time of the classification mechanism, facilitating a quick classification of any incoming SOAP messages that were thought to contain significant features during the first phase. The second phase is executed only for those SOAP messages with complex characteristics identified as suspicious during the first phase and requiring a more detailed evaluation. Each of the phases incorporates a CBR-BDI [12] agent with reasoning, learning and adaptation capabilities.

The following section provides a detailed description of the characteristics and tasks related to each of the agents that constitute the AIDeM platform.

- Traffic Agent: This agent has a type of sensor feature that allows it to identify and capture SOAP messages that have been sent from external applications and that request a particular type of service. The agent captures the messages and redirects them to the Security agent for evaluation.
- Security Agent: This agent carries out tasks similar to those assigned with the original FUSION@ security mechanism. The agent is in charge of receiving SOAP message that contain service requests. It performs a quick analysis of the message, and the data obtained are sent to the agent at the first phase of the classification mechanism. With cases that are considered suspicious, the Security agent submits the XML message to a more comprehensive syntactic analysis in order to obtain the necessary data for carrying out the second phase of the classification mechanism. There can be more than one Security agent, depending on the amount of workload.
- Classifier C1 Agent: This is one of the key CBR-BDI agents in the classification process. These agents initiate a classification by incorporating a CBR engine that in turn incorporates a Näives Bayes strategy in the re-use phase. The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources. The fields of the case are obtained from the headers of the packages of the HTTP/TCP-IP transport protocol. Table 1 shows the fields taken into consideration to describe the problem.

Table 1 Problem Description First Phase – Classifier C1 Agent

Fields	**Type**	
IDService	Int	*i*
Subnet mask	String	*m*
SizeMessage	Int	*s*
NTimeRouting	Int	*n*
LengthSOAPAction	Int	*l*
TFMessageSent	Int	*w*

Within the CBR cycle, specifically in the re-use phase, a particular classification strategy is used by applying a Näives Bayes strategy, which gives 3 possible results: legal, malicious and suspicious. Messages that are classified as legal are sent to the corresponding web service for processing. Malicious messages are immediately rejected, while suspicious messages continue through to the classification process executing the second phase of the classification mechanism. There can be more than one Classifier C1 agent depending on the amount of workload.

- Classifier C2 Agent: This CBR-BDI agent completes the classification mechanism. In order to initiate this phase, it is necessary to have previously initiated a syntactic analysis on the SOAP message to extract the required data. Table 2 presents the fields used in describing the problem for the CBR in this layer. Once the data have been extracted from the message, a CBR mechanism is initiated by using a Multilayer Perceptron (MLP) neural network in the reuse phase. There can be more than one Classifier C2 agent, depending on the amount of workload.

Table 2 Problem Description Second Phase – Classifier C2 Agent

Fields	Type	variable
IDService	Int	*i*
MaskSubnet	String	*m*
SizeMessage	Int	*s*
NTimeRouting	Int	*n*
LengthSOAPAction	Int	*l*
MustUnderstandTrue	Boolean	*u*
NumberHeaderBlock	Int	*h*
NElementsBody	Int	*b*
NestingDepthElements	Int	*d*
NXMLTagRepeated	Int	*t*
NLeafNodesBody	Int	*f*
NAttributesDeclared	Int	*a*
CPUTimeParsing	Int	*c*
SizeKbMemoryParser	Int	*k*

- Admin Agent: In addition to the functions already mentioned in the FUSION@ architecture, this agent is responsible for overseeing the correct functioning of the classification process and for coordinating the distribution of tasks.

4 Results and Conclusions

This article has presented AIDeM, a mechanism for classifying incoming requests based on SOAP messages. The mechanism was designed to exploit the distributed capacity of the agents. Additionally, an advanced classification mechanism was designed to filter incoming SOAP messages. The classification mechanism was structured in two phases, each of which includes a special CBR-BDI agent that functions as a classifier. The first phase filters simple attacks without exhausting an

excessive amount of resources by applying a CBR engine that incorporates a Naïves Bayes strategy. The second phase, a bit more complex and costly, is in charge of classifying the SOAP messages that were not classified in the first phase.

A small case study was used to evaluate the efficacy of the integration of AIDeM within the FUSION@ architecture. The ALZ-MAS 2.0 multi-agent system [5] was used to carry out the test. It was implemented through FUSION@ and used to construct a tool for dependent environments. In order to evaluate AIDeM, two specific services available in ALZ-MAS 2.0 were selected for external users: *RequestScheduleDoctor()* which is used to consult the agenda of a doctor via Internet and *RequestAppointment()* which is used to request an appointment with a doctor via Internet.

The experiments were carried out in two stages; the first stage was to obtain the test data used for training the classifiers, and the second stage was to evaluate the classification mechanism. In order to obtain the test data in the first stage, the Traffic agent was configured to capture the incoming SOAP messages without redirecting them to the services. In order to send the SOAP messages, 3 points (nodes) were established, from which various requests for selected services were executed. Each of these nodes belonged to a different network, i.e., each node connected to the internet using a different IP and subnetwork mask. In the first stage, each node was configured with 30 requests (SOAP messages) to be sent to each of the 2 selected services. Each node sent a total of 60 requests so that the total number of requests made by the 3 nodes to the 2 services was equal to 180 requests. The 30 requests sent by each node, included legal message and malicious message (incorrectly formed messages). For the second stage of testing, the number of nodes and services was the same as in the first stage, but the number of requests was configured at 15 requests per node. At this stage, once the requests were captured by the Traffic agent, they were sent to AIDeM to be evaluated and classified. A total of 90 requests (legal and malicious) were sent to AIDeM for evaluation. Figure 2 shows the results obtained for the set of SOAP messages evaluated.

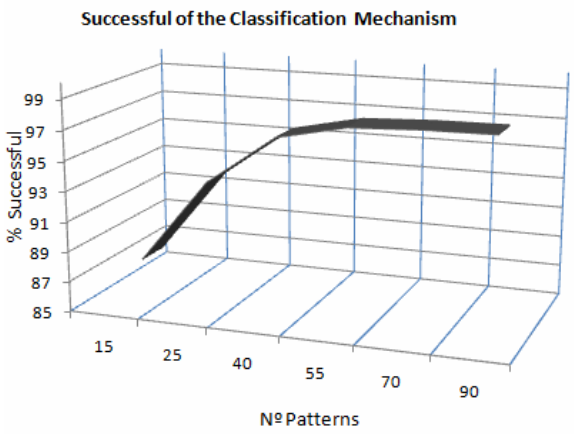

Fig. 2 Effectiveness of the classification mechanism prototype according to the number of patterns

Figure 2 shows the percentage of prediction with regards to the number of patterns (SOAP messages) for the classification mechanism. It is clear that as the number of patterns increases, the success rate of prediction also increases in terms of percentage. This is influenced by the fact that we are working with CBR systems, which depend on a larger amount of data stored in the memory of cases.

Future works are expected to develop the tools for obtaining a complete solution.

Acknowledgements. This development has been partially supported by the Spanish Ministry of Science project TIN2006-14630-C03-03 and The Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama.

References

1. Negm, W.: Anatomy of a Web Services Attack: A Guide to Threats and Preventative Countermeasures (2004)
2. Nadalin, A., Kaler, C., Monzillo, R., Hallam-Baker, P.: Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) (2006)
3. Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M.: Web Services Policy Framework (WS-Policy) version 1.2 (2006)
4. Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della, G., Dixon, B.: Web Services Trust Language (WS-Trust) (2004)
5. Corchado, J.M., Tapia, D., Bajo, J.: A Multi-Agent Architecture for Distributed Services and Applications. International Journal of Ambient Computing and Intelligence - IJACI, 15–26 (2009)
6. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7, 39–59 (1994)
7. Cerami, E.: Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, 1st edn. O'Reilly & Associates, Sebastopol (2002)
8. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355 (1988)
9. Jennings, N.R., Wooldridge, M.: Applying agent technology. Applied Artificial Intelligence 9, 357–369 (1995)
10. Pinzón, C., De Paz, Y., Bajo, J.: A Multiagent Based Strategy for Detecting Attacks in Databases in a Distributed Mode. In: Corchado, J.M., Rodríguez, S., Llinas, J., Molina, J.M. (eds.) International Symposium on Distributed Computing and Artificial Intelligence - DCAI 2008, vol. 50. Springer, Heidelberg (2008)
11. Bajo, J., Corchado, J.M., Pinzón, C., Paz, Y.D., Pérez-Lancho, B.: SCMAS: A Distributed Hierarchical Multi-Agent Architecture for Blocking Attacks to Databases. International Journal of Innovative Computing, Information and Control (2008)
12. Laza, R., Pavón, R., Corchado, J.M.: A Reasoning Model for CBR_BDI Agents Using an Adaptable Fuzzy Inference System. In: Conejo, R., Urretavizcaya, M., Pérez-de-la-Cruz, J.-L. (eds.) CAEPIA/TTIA 2003. LNCS (LNAI), vol. 3040, pp. 96–106. Springer, Heidelberg (2004)