An Adaptive Multi-agent Solution to Detect DoS Attack in SOAP Messages

Cristian I. Pinzón, Juan F. De Paz, Javier Bajo, and Juan M. Corchado

Departamento Informática y Automática, Universidad de Salamanca,
Plaza de la Merced s/n 37008, Salamanca, Spain
{cristian_ivanp,fcofds,jbajope,corchado}@usal.es

Abstract. A SOAP message can be affected by a DoS attack if the incoming message has been either created or modified maliciously. The specifications of existing security standards do not focus on this type of attack. This article presents a novel distributed and adaptive approach for dealing with DoS attacks in Web Service environments, which represents an alternative to the existing centralized solutions. The solution proposes a distributed hierarchical multi-agent architecture that implements a classification mechanism in two phases. The main benefits of the approach are the distributed capabilities of the multi-agent systems and the self-adaption ability to the changes that occur in the patterns of attack. A prototype of the architecture was developed and the results obtained are presented in this study.

Keywords: Multi-agent System, CBR, Web Service, SOAP Message, DoS attacks.

1 Introduction

The Web services processing model requires the ability to secure SOAP messages and XML documents as they are forwarded along potentially long and complex chains of consumer, provider, and intermediary services. However all standards that have been proposed to date, such as WS-Security [1], WS-Policy [2], WS-Trust [3], WS-SecureConversation [4], etc. focus on the aspects of message integrity and confidentiality and user authentication and authorization [5].

Until now, denial-of-service (DoS) attacks have not been dealt with in Web Services environments. A DoS attack on Web Services takes advantage of the time involved in processing XML formatted SOAP messages. The DoS attack is successfully carried out when it manages to severely compromise legitimate user access to services and resources. XML messages must be parsed in the server, which opens the possibility of an attack if the messages themselves are not well structured or if they include some type of malicious code. Resources available in the server (memory and CPU cycles) can be drastically reduced or exhausted while a malicious SOAP message is being parsed.

This article presents a novel distributed multi-agent architecture for dealing with DoS attacks in Web Services environments. Additionally, the architecture has a four-tiered hierarchical design that is better capable of task distribution and error recovery. The most important characteristic of the proposed solution is the two-phased mechanism that was designed to classify SOAP messages. 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. Each of the phases incorporates a CBR-BDI [6] agent with reasoning, learning and adaptation capabilities. The CBR engine initiates what is known as the CBR cycle, which is comprised of 4 phases. The first agent uses a decision tree and the second a neural network, each of which is incorporated into the respective reuse 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.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research. Section 3 focuses on the design of the proposed architecture. Finally, section 4 presents the results and conclusions obtained by the research.

2 DoS Attacks Description

One of the most frequent techniques of a DoS attack is to exhaust available resources (memory, CPU cycles, and bandwidth) on the host server. The probability of a DoS attack increases with applications providing Web Services because of their intrinsic use of the XML standard. In order to obtain interoperability between platforms, communication between web servers is carried out via an exchange of messages. These messages, referred to as SOAP messages, are based on XML standard and are primarily exchanged using HTTP (Hyper Text Transfer Protocol) [7]. The server uses a parser, such as DOM, Xerces, etc. to syntactically analyze all incoming XML formatted SOAP messages. When the server draws too much of its available resources to parse SOAP messages that are either poorly written or include a malicious code, it risks becoming completely blocked.

Attacks usually occur when the SOAP message either comes from a malicious user or is intercepted during its transmission by a malicious node that introduces different kinds of attacks.

The following list contains descriptions of some known types of attacks that can result in a DoS attack, as noted in [8, 9, 10].

- **Oversize Payload:** It reduces or eliminates the availability of a Web Service when the CPU, memory or bandwidth are being tied up by the processing of messages with an enormous payload.
- **Coercive Parsing:** Just like a message written with XML, an XML parser can analyze a complex format and lead to an attack when the memory and processing resources of the server are being used up.
- **Injection XML:** This is based on the ability to modify the structure of an XML document when an unfiltered user entry goes directly to the XML stream. The message is captured and modified during its transmission.

- **SOAP header attack:** Some SOAP message headers are overwritten while they are passing through different nodes before arriving at their destination.
- **Replay Attack:** Sent messages are completely valid, but they are sent en masse over short periods of time in order to overload the Web Service server.

All Web Services security standards focus on strategies independent from DoS attacks. Other measures have been proposed that do not focus directly on dealing with DoS attacks, but can deal with some of the inherent techniques. One solution based on the XML firewall was proposed to protect Web Services in more detail [8]. By applying a syntactic analysis, a validation mechanism, and filtering policies, it is possible to identify attacks in individual or group messages. An adaptive framework for the prevention and detection of intrusions was presented in [9]. Based on a hybrid focus that combines agents, data mining and diffused logic, it is supposed to filter attacks that are either already known or new. The solution as presented is an incipient idea still being developed and implemented. Finally, another solution proposed the use of Honeypots as a highly flexible security tool [11]. The focus incorporates 3 components: data extraction based on honeyd and tedpdum data analysis, and extraction from attack signatures. Its main inconvenience is that it depends too much on the ability of the head of security to define when a signature is or is not a type of attack. Even when the techniques mentioned claim to prevent attacks on Web Services, few provide statistics on the rates of detection, false positives, false negatives and any negative effects on application performance.

3 An Agent Based Architecture

Agents are characterized by their autonomy; which gives them the ability to work independently in real-time environments [12]. Furthermore, when they are integrated within a multi-agent system they can also offer collaborative and distributed assistance in carrying out tasks [13].

The main characteristic of the multi-agent architecture proposed in this paper is the incorporation of CBR-BDI [6] deliberative agents. The CBR-BDI classifier agents implemented here use an intrusion detection technique known as anomaly detection. In order to carry out this type of detection, it is necessary to extract information from the HTTP/TCP-IP protocol headers that are used for transporting messages, as well as from the structure and content of the SOAP message, the message processing tasks, and any activity among Web Services users.

Our proposal is a distributed, hierarchical multi-agent architecture integrated for 4 levels with distinct BDI agents. The hierarchical structure makes it possible to distribute tasks on the different levels of the architectures and to define specific responsibilities. Meanwhile, it is essential to maintain a continuous interaction and communication between agents, as they must be able to continue requesting services and deliver results. The architecture presented in figure 1 shows the four levels with BDI agents organized according to their roles.

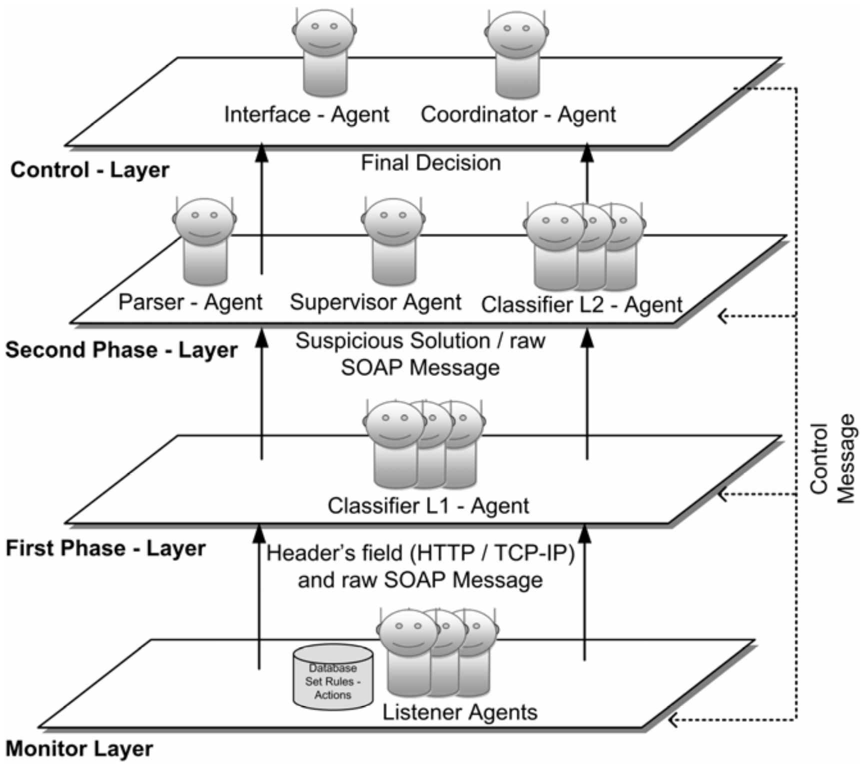

Fig. 1. Design of the multi-agent architecture proposed

The following section describes the functionality of the agents located in each layer of the proposed hierarchical structure.

- **Listener agents:** Capture any traffic directed towards the server and process the HTTP and TCP/IP protocol headers that are used to transport SOAP messages. JPCAP is used to identify and capture any traffic that contains SOAP message packets. The information extracted from the HTTP/TCP-IP transport protocol message headers is sent to the next layer in order to carry out the classification process. In addition to these tasks, Listener agents use an IP register to monitor user activities. This type of monitoring makes it possible to identify suspicious activities similar to message replication attacks.
- **Classifier-L1 agents:** These CBR-BDI agents are located on layer 2 of the architecture and are in charge of executing the first phase of the classification process based on the data sent by the Listener agents. These agents initiate a classification by incorporating a CBR engine that in turn incorporates a decision tree strategy in the reuse phase. The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources. The case description for this CBR is shown in Table 1 and involves various fields extracted from the HTTP/TCP-IP transport protocol message headers.

Table 1. Case Description First Phase - CBR

Fields	Type
IDService	Int
Subnet mask	String
SizeMessage	Int
NTimeRouting	Int
LengthSOAPAction	Int
TFMessageSent	Int

Within the CBR cycle, specifically in the reuse phase, a particular classification strategy is used by applying a knowledge extraction method known as Classification and Regression Tree (CART), which creates 3 groups: 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.

- **Classifier-L2 agents:** These CBR-BDI agents carry out the second phase of classification from layer 3 of the architecture. In order to initiate this phase, it is necessary to have previously started a syntactic analysis on the SOAP message to extract the required data. This syntactic analysis is performed by the Parser Agent. 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. Table 2 presents the fields used in describing the case for the CBR in this layer.

Table 2. Case Description Second Phase - CBR

Fields	Type
SizeMessage	Int
NTimeRouting	Int
LengthSOAPAction	Int
MustUnderstandTrue	Boolean
NumberHeaderBlock	Int
NElementsBody	Int
NestingDepthElements	Int
NXMLTagRepeated	Int
NLeafNodesBody	Int
NAttributesDeclared	Int
CPUTimeParsing	Int
SizeKbMemoryParser	Int

The neural network is trained from the similar cases that were recovered in the retrieve phase. Once the neural network has been trained, the new case is presented to the network for classification as either legal or malicious. However, because there is a possibility of significant errors occurring in the training phase of the neural network, an expert will be in charge of the final review for classifying the message.

- **Supervisor Agent:** This agent supervises the Parser agent since there still exists the possibility of an attack during the syntactic processing of the SOAP message. This agent is located in layer 3 of the architecture.
- **Parser Agent:** This agent executes the syntactic analysis of the SOAP message. The analysis is performed using SAX as parser. Because SAX is an event driven API, it is most efficient primarily with regards to memory usage, and strong enough to deal with attack techniques. The data extracted from the syntactic analysis are sent to the Classifier L-2 agents. This agent is also located on layer 3 of the architecture.
- **Coordinator Agent:** This agent is in charge of supervising the correct overall functioning of the architecture. Additionally, it oversees the classification process in the first and second phase. Each time a classification is tagged as suspicious, the agent interacts with the Interface Agent to request an expert review. Finally, this agent controls the alert mechanism and coordinates the actions required for responding to this type of attack.
- **Interface Agent:** This agent was designed to function in different devices (PDA, Laptop, Workstation). It facilitates ubiquitous communication with the security personnel when an alert has been sent by the Coordinator Agent. Additionally, it facilitates the process of adjusting the global configuration of the architecture. This agent is also located in the highest layer of the architecture.

4 Results and Conclusions

This article has presented a distributed hierarchical multi-agent architecture for classifying SOAP messages. The architecture 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 decision tree. 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. This phase applies a CBR engine that incorporates a neural network. A prototype of our proposed solution was based on a classification mechanism and developed in order to evaluate its effectiveness. Figure 2 shows the results obtained for a set of SOAP messages.

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 and to evaluate the successfulness in repelling and detecting attacks. With the advantage of a distributed process for classification tasks, it would be possible to evaluate the effectiveness of the classification mechanism, and the response time.

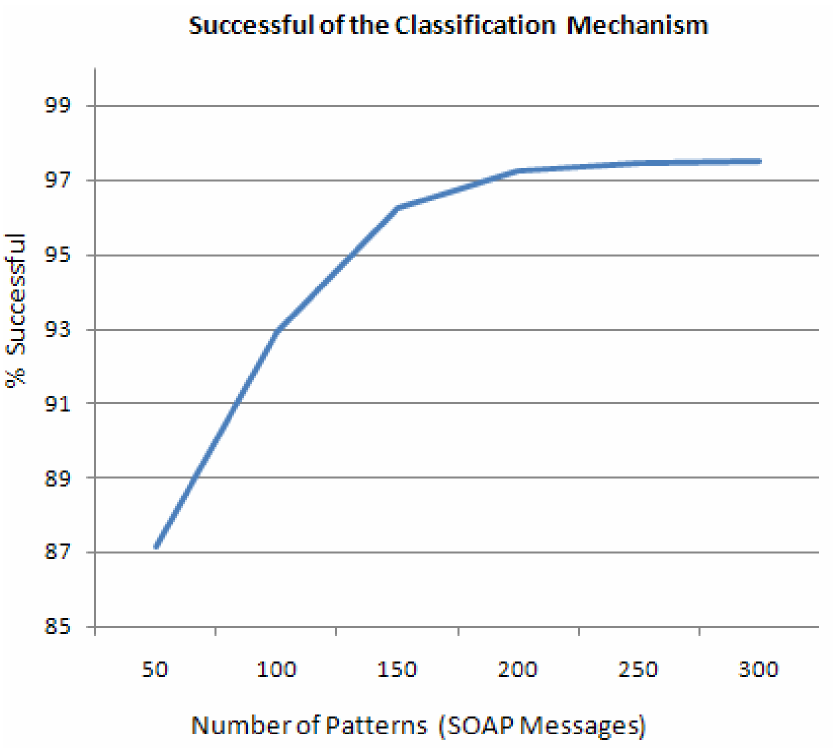

Fig. 2. Effectiveness of the classification mechanism integrated according to the number of patterns

Acknowledgments. 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. OASIS: Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), OASIS Standard 2004, `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf` (2006)
2. Bajaj, et al.: Web Services Policy Framework, WS-Policy (2004), `http://www.ibm.com/developerworks/library/specification/ws-polfram`
3. Web Services Trust Language (WS-Trust), `http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf`
4. Web Services Secure Conversation Language (WS-SecureConversation), `http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf`
5. Gruschka, N., Luttenberger, N.: Protecting Web Services from DoS Attacks by SOAP Message Validation. Security and Privacy in Dynamic Environments (201), 171–182 (2006)

6. Laza, R., Pavon, 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)
7. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture: SOAP. In: WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR, Englewood Cliffs (2005)
8. Loh, Y., Yau, W., Wong, C., Ho, W.: Design and Implementation of an XML Firewall. Computational Intelligence and Security 2, 1147–1150 (2006)
9. Yee, G., Shin, H., Rao, G.S.V.R.K.: An Adaptive Intrusion Detection and Prevention (ID/IP) Framework for Web Services. In: International Conference on Convergence Information Technology, pp. 528–534. IEEE Computer Society, Washington (2007)
10. Jensen, M., Gruschka, N., Herkenhoner, R., Luttenberger, N.: SOA and Web Services: New Technologies, New Standards - New Attacks. In: Fifth European Conference on Web Services-ECOWS 2007, pp. 35–44 (2007)
11. Dagdee, N., Thakar, U.: Intrusion Attack Pattern Analysis and Signature Extraction for Web Services Using Honeypots. In: First International Conference Emerging Trends in Engineering and Technology, pp. 1232–1237 (2008)
12. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multiagent architecture as a real-time problem-solving model. Expert Syst. Appl. 34, 2–17 (2008)
13. Corchado, J.M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences. IEEE Intelligent Systems 23, 19–25 (2008)