A Solution CBR Agent-Based to Classify SOAP Message within SOA Environments

Cristian Pinzón, Belén Pérez, Angélica González,
Ana de Luís y, and J.A. Román

University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
{cristian_ivanp,lancho,angelica,adeluis,zjarg}@usal.es

Abstract. This paper presents the core component of a solution based on agent technology specifically adapted for the classification of SOA messages. These messages can carry out attacks that target the applications providing Web Services. An advanced mechanism of classification designed in two phases incorporates a CBR-Agent type for classifying the incoming SOAP messages as legal or malicious. Its main feature involves the use of decision trees, fuzzy logic rules and neural networks for filtering attacks.

Keywords: SOAP message, XML security, multi-agent systems, case-based reasoning.

1 Introduction

The communication among services based on Service Oriented Architecture Web Services (SOA) is carried out by XML-based messages, called SOAP messages. This message exchange process is one of the key elements required in SOA environments for system integration [1]. The SOAP message payload often consists of sensitive information, which is sent through insecure channels such as HTTP connections. If a malicious user playing the role of a middleman intercepts a message between sender and recipient, it can result in a series of malicious tasks carried out over the captured message. A number of technologies and solutions have been proposed for addressing the secure exchange of SOAP message. Some WS standards such as WS-Security [2], WS-Policy [3], among others, continually strive to provide real security. Within academia some solutions in the research & development phase focusing on web service security in greater detail are [1], [4], [5]. However, both the WS-Security Standards and the given solutions still do not provide full security, leaving gaps that can be exploited by any malicious user.

This paper presents the core component of a strong solution based on a multi-agent architecture for tackling the security issue of the Web Service. This core is embedded in a CBR-BDI [6] deliberative agent based on the BDI (Belief, Desire, Intention) [7] model specifically adapted for preventing many attacks over web services. Our study applies a solution in two phases that include novel case-based reasoning (CBR) [8] classification mechanisms. The first phase incorporates decision tree and fuzzy logic rules [9] while the second phase incorporates neural networks capable of making short term predictions [10]. The idea of a CBR mechanism is to exploit the experience

E. Corchado et al. (Eds.): HAIS 2009, LNAI 5572, pp. 492–499, 2009.

gained from similar problems in the past and to adapt a successful solution to the current problem. The CBR-BDI agent explained in this work uses the CBR concept to gain autonomy and improve its problem-solving capabilities. The approach presented in this paper is entirely new and offers a different way to confront the security problem in SOA environments.

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 structure of the classifier agent which facilitates classification of SOAP message, and section 4 provides a detailed explanation of the classification model integrated within the classifier agent. Finally, section 5 presents the conclusions obtained by the research.

2 Web Service Security Problem Description

A web service is a software module designed to support interaction between heterogeneous groups within a network. 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 standard XML (eXtensible Markup Language) and are primarily exchanged using HTTP (Hyper Text Transfer Protocol) [11].

Security is one of the greatest concerns within web service implementations. 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 different types of attacks, compiled from those noted in [4], [5], [12].

- Oversize Payload: When it is executed, it reduces or eliminates the availability of a web service while the CPU, memory or bandwidth are being tied up by a massive message dispatch with a large payload.
- Coercive Parsing: Just like a message written with XML, an XML parser can analyze a complex format and lead to a denial of service attack because the memory and processing resources 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 or the message is captured and modified during its transmission.
- Parameter Tampering: A malicious user employs web service entries to manually or automatically (dictionaries attack) execute different types of tests and produce an unexpected response from the server.
- SOAP header attack: Some SOAP message headers are overwritten while they are passing through different nodes before arriving at their destination. It is possible to modify certain fields with malicious code.
- Replay Attack: Sent messages are completely valid, but they are sent en masse over a small time frame in order to overload the web service.

Standards such as WS-Security [2] and WS-Policy [3], among others, have set the standard for solutions to security breaches. One solution proposed by [1] takes information from the actual message structure and adds a new header named *SOAP Account* that contains information on the message structure. One solution based on the

XML firewall [4] was proposed to protect web services in more detail. 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 [5]. Based on a hybrid focus that combines agents, data mining and fuzzy 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 [13] as a highly flexible security tool. The focus incorporates 3 components: data extraction based on honeyd, 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.

The following sections detail the internal model of the CBR-BDI agent, as well as the classification process for SOAP message for identifying malicious messages.

3 Classifier Agent Internal Structure

Agents are characterized by their autonomy; which gives them the ability to work independently and in real-time environments [14]. Because of this and their other capacities, agents are being integrated into security approaches such as IDS [15]. However, the use of agents in these systems focuses on the retrieval of information in distributed environments, which only takes advantage of their mobility capacity.

The classification agent presented in this study interacts with other agents within the architecture. These agents carry out tasks related to capturing messages, syntactic analysis, administration, and user interaction. As opposed to the tasks for these agents, the classification agent executes a classification of SOAP messages in two phases that we will subsequently define in greater detail.

In our research, the agents are based on a BDI model in which beliefs are used as cognitive aptitudes, desires as motivational aptitudes, and intentions as deliberative aptitudes in the agents [7]. However, in order to focus on the problem of the SOAP message attack, it was necessary to provide the agents with a greater capacity for learning and adaption, as well as a greater level of autonomy than a pure BDI model currently possesses. This is possible by providing the classifier agents with a CBR mechanism [8], which allows them to "reason" on their own and adapt to changes in the patterns of attacks. When working with this type of system, the key concept is that of "case". A case is defined as a previous experience and is composed of three elements: a description of the problem that depicts the initial problem; a solution that describes the sequence of actions performed in order to solve the problem; and the final state, which describes the state that has been achieved once the solution is applied. To introduce a CBR engine into a BDI agent, we represent CBR system cases using BDI and implement a CBR cycle which consists of four steps: retrieve, reuse, revise and retain [16].

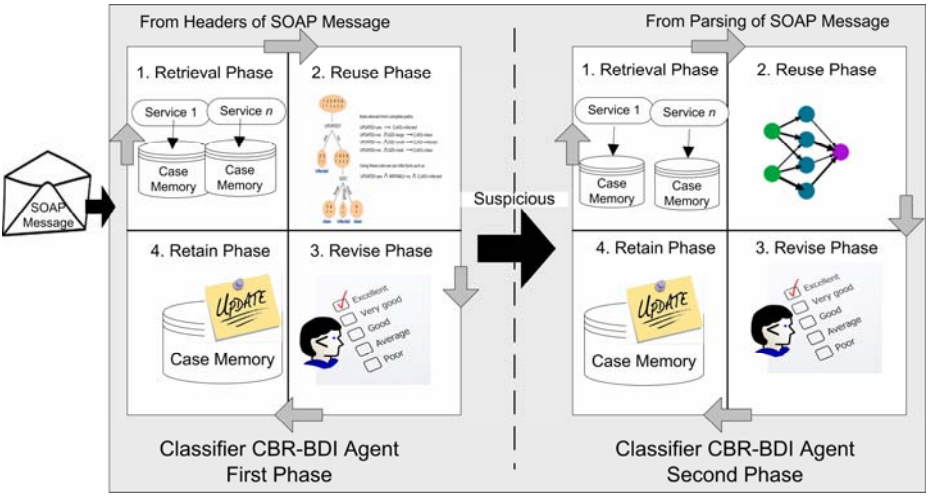

Fig. 1. Classifier CBR-BDI agents in each phase of the mechanism of classification

As previously mentioned, the classifier CBR-BDI agent is the core of the multi-agent architecture and is geared towards classifying SOAP messages for detecting attacks on web services. Figure 1 shows the classifier CBR-BDI agents in each phase of the mechanism of classification.

4 Mechanism for the Classification of SOAP Message Attack

The CBR-BDI classifier agent presented in section 3 incorporates a case-based reasoning mechanism that allows it to classify SOAP messages. The mechanism incorporated into the agent approaches the idea of classification from the perspective of anomaly-based detection. This mechanism requires the use of a database with which it can generate models such as the solution of a new problem based on past experience. In the specific case of SOAP messages, it manages a case memory for each service offered by the architecture, which permits it to handle each incoming message based on the particular characteristics of each web service. Each new SOAP message sent to the architecture is classified as a new case study object. The advantage that the CBR systems provide spans from automatic learning to the ability to adapt and approach new changes that appear in the patterns of attack.

Focusing on the problem that is of interest to us, we will represent a typical SOAP message which consists of a type of wrapping that contains an optional heading and a mandatory body of text with a useful message load, as depicted in figure 2.

Based on the structure of the SOAP messages and the transport protocol used, we can obtain a series of descriptive fields to consider: IPSource, SizeMessage, Time TravelMessage, NumberHopRouting, LengthSoapAction, NumberHeaderBlocks, NumberElementsBody, NestingDepthElements, NumberXMLTagRepeatedBody, NumberLeafNodesBody. Based on this information, we can present a two-part strategy for executing the classification process:

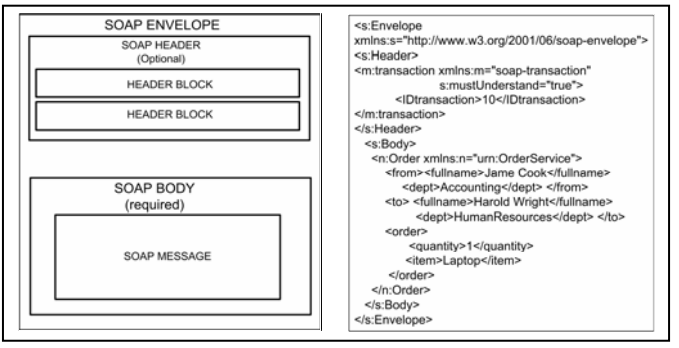

Fig. 2. SOAP Message Structure

The first phase executes a case-based reasoning mechanism that incorporates the Classification and Regression Tree (CART) knowledge extraction method to obtain fuzzy logic rules [9]. In order to execute this CBR mechanism, it is necessary to define the case as follows:

Table 1. Case Description – CBR (First Phase)

IP Source	Size Message	Time Travel Message	Number Hop Routing	Length SoapAction
String	Int	Int	Int	Int

In each phase of the CBR cycle, certain well-defined tasks are executed. Using CART, we can generate decision-making rules based on the IP Source field extracted from the transport protocol. In order to optimize the system, the reasoning cycle is only executed when it does not have the decision-making rules obtained from previous iterations for an IP from the same class and with similar bytes. In this case, it would only be necessary to recover the rules from the memory of rules. In the opposite case, the CBR cycle is executed in its entirety, as explained below:

- Recovery Phase: Cases will be filtered according to the original IP address so that the cases whose IP is from the same class and with a matching IP are selected. This way, we try to account for the network that the attacks are sent from.
- Reuse: Once the IP filtering process has taken place, we extract the fuzzy logic rules in order to classify the SOAP messages. Knowledge extraction is carried out by applying decision trees specifically with CART, which is then applied to the IPSource, SizeMessage, TimeTravelMessage, NumberHopRouting, LengthSoapAction variables. The final classification that is stored would include three groups: "Malicious", "Legal" and "Suspicious". For each of the decision tree rules, we set up a final classification of the defined groups and the percentage of erroneously classified attacks.
- Revision: Based on the set of previously established rules, we can classify the new SOAP message in such a way that if the final group is "suspicious" or the percentage of errors for the rules is greater than a pre-determined limit, we proceed to the classification phase. Once the classification has been achieved in this

first phase, or if there has been a classification error indicated by an expert, we apply a new decision tree to obtain the rules for the knowledge obtained.

- Retain: In this phase, the decision rules derived from previous phases are stored and associated with the new case based on the IP class and the similar bytes.

Once a solution to the first phase of classification has been obtained, we will determine the need for executing the second phase of the process. This phase is carried out in cases where the message was classified as "Suspicious" or when the percentage of error classification exceeds the defined limit. The second phase involves a much more complex process with an exhaustive syntactic analysis of the SOAP message and the execution of a new CBR cycle. During the execution of the second phase, certain control policies have been established that continuously validate the process overload. The values of the control policies is established by defined variables such as: TimeParsing, CPUTimeParsing, MemoryCostParsing, ValueNestingDepth

Finally, in order to complete the second phase of classification, a CBR mechanism is carried out using a multi-layer perceptron (MLP) in the Reuse phase. This CBR mechanism requires the memory of cases to be specifically defined for each service. The definition of the case is detailed as follows:

Table 2. Case Description – CBR (Second Phase of the Mechanism of Classification)

NumberEle-mentsBody	NumberHeader Blocks	ValueNesting Depth	NumberLeft Nodes	TagXML RepeatedBody
Int	Int	Int	Int	Int

The CBR mechanism executes the following tasks:

- Retrieve: If there happens to be a neural network that is trained for the web service identified in the message, it will be retrieved to perform the classification. If none exists, all of the stored cases corresponding to the web service will be retrieved.
- Reuse: If it has not already received previous training, the neural network will be trained beginning with the retrieved cases. The initial information corresponds to the fields described in table 2, which are transformed into values between 0 and 1 inclusive. Upon completing the training, the new message is classified as either "Malicious" or "Legal".

  The MLP uses a Sigmoidal function with a range of possible values at the interval [0,1]. It is used to detect if the request is classified as an attack or not. The value 0 represents a legal message (non attack) and 1 a malicious message (attack). The Sigmoidal activation function is the activation function most used for classifications between two groups.

$$f(x) = \frac{1}{1+e^{-ax}} \tag{1}$$

  The number of neurons in the output layer for the Multilayer Perceptrons is 1, and is responsible for deciding whether or not there is an attack.
- Revise: If the estimated value does not exceed a certain threshold near zero, the message is assumed to be valid; contrarily, it is up to an expert to validate it.

- Retain: The results from the previous stage are stored in the event that the classification of the SOAP message has been either successful or indicated as such by an expert. Finally, in the event that the number of cases related to the network has increased by a percentage, a retraining of the network will be carried out.

5 Conclusions

This research has presented the nucleus of a novel solution that focuses on the protection of web services. The focus incorporates case-based reasoning methods, decision trees, fuzzy logic rules, neural networks, and intelligent agent technology that allows us to approach the problem of web security from a perspective based on learning, adaptability and flexibility.

The solution was designed to be carried out in two phases. In the first phase, a CBR mechanism incorporates decision trees; fuzzy logic rules generate a preliminary robust solution regarding the condition of the message, without sacrificing application performance. If the obtained solution is classified as suspicious, we then proceed to the second phase of the process. This phase does involve a more complex process, with a greater need for resources, and where a second CBR mechanism embeds within a neural network to generate a final result. The proposed solution will continue in the investigation and development for its application in various environments where its performance can be evaluated and real results obtained.

Acknowledgments. This development has been partially supported by the Spanish Ministry of Science project TIN2006-14630-C03-03.

References

1. Rahaman, M.A., Schaad, A., Rits, M.: Towards secure SOAP message exchange in a SOA. In: 3rd workshop on Secure web services, pp. 77–84. ACM, New York (2006)
2. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Security: SOAP Message Security 1.1 (WS-Security (2004), http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAP MessageSecurity.pdf
3. Bajaj, S., et al.: Web Services Policy Framework, WSPolicy (2004), http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf
4. Loh, Y., Yau, W., Wong, C., Ho, W.: Design and Implementation of an XML Firewall. Computational Intelligence and Security 2, 1147–1150 (2006)
5. 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)
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, vol. 3040, pp. 96–106. Springer, Heidelberg (2004)
7. Rao, A., Georgeff, M.: Modeling Rational Agents within a BDI-Architecture. In: KR, pp. 473–484 (1991)

8. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. 7, 39–59 (1994)
9. Bittencourt, H., Clarke, R.: Use of classification and regression trees (CART) to classify remotely-sensed digital images. In: Geoscience and Remote Sensing Symposium, IEEE International, vol. 6, pp. 3751–3753 (2003)
10. Shun, J., Malki, H.: Network Intrusion Detection System Using Neural Networks. In: Fourth International Conference on Natural Computation, vol. 5, pp. 242–246 (2008)
11. Snell, J., Tidwell, D., Kulchenko, P.: Programming Web Services with SOAP. O'Reilly, Sebastopol (2001)
12. 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)
13. 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)
14. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multi-agent architecture as a real-time problem-solving model. Expert Syst. Appl. 34, 2–17 (2008)
15. Abraham, A., Jain, R., Thomas, J., Han, S.Y.: D-SCIDS: distributed soft computing intrusion detection system. J. Netw. Comput. Appl. 30, 81–98 (2007)
16. Corchado, J.M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences. Intelligent Systems, IEEE 23, 19–25 (2008)