A Multiagent Solution to Adaptively Classify SOAP Message and Protect against DoS Attack

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

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

Abstract. SOAP messages use XML code, which makes them vulnerable to denial of service (DoS) attacks and puts the availability of web services at risk. This article presents an adaptive solution for dealing with DoS attacks in web service environments. The solution proposes a distributed hierarchical multiagent architecture that implements a robust mechanism of classification based on an advanced CBR-BDI agent. The agent incorporates a case-based reasoning engine that integrate a Perceptron Multilayer neural network during the re-use phase to classify incoming SOAP messages and reject those that are considered malicious. A prototype of the architecture was developed and the results obtained are presented in this study.

Keywords: SOAP message, XML Security, multiagent systems, CBR, ANN.

1 Introduction

In order to obtain interoperability between web service 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 and are primarily exchanged using HTTP (Hyper Text Transfer Protocol) [1]. 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 type of attack is known as a denial of service (DoS) attack and is perpetrated at the web service level because of the intrinsic relationship it has with the XML standard for coding SOAP messages.

A number of standards for guaranteeing the security of messages have been proposed to date, including WS-Security [2], WS-Policy [3], WS-Trust [4], WS-SecureConversation [5], etc. However, the proposed solutions focus exclusively on the aspects of message integrity and confidentiality, and user authentication and authorization [5].

This article presents a distributed hierarchical multiagent architecture for dealing with DoS attacks in web service environments. The architecture incorporates a CBR-BDI [7] agent with reasoning, learning and adaptation capabilities. The idea of a CBR mechanism is to exploit the experience gained from similar problems in the past and

P. Meseguer, L. Mandow, and R.M. Gasca (Eds.): CAEPIA 2009, LNAI 5988, pp. 181–190, 2010.

then adapt successful solutions to the current problem. The CBR engine initiates what is known as the CBR cycle, which is comprised of 4 phases. The classifier agent uses a Multilayer Perceptron neural network (MLP), which is incorporated into the re-use phase of the CBR cycle. By combining the CBR mechanism and the MLP, the system acquires learning capabilities and is able to adapt to changes in attack patterns for SOAP messages, thus facilitating the classification task when a SOAP message contains a DoS attack.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research work. Section 3 focuses on the details of the multiagent architecture; section 4 explains in detail the classification model integrated within the classifier agent. Finally, section 5 describes how the classifier agent has been tested and presents the results obtained.

2 Web Service Security Problem Description

Attacks usually occur when the SOAP messages either come from a malicious user or are intercepted during the 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 a message with a large payload is parsed within the server.
- Coercive Parsing: An XML parser can analyze a complex format and lead to an attack because the memory and processing resources are being used up.
- Injection XML: The structure of a XML document is modified with a malicious code.
- 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.

All web service security standards focus on strategies independent from DoS attacks [5]. In the following, we will revise those works that focus on denial of web service attacks and will compare to our approach as shown in Table 1.

A “XML Firewall” is proposed by [8]. Messages that are sent to a web service are intercepted and parsed to check the validity and the authenticity of the contents. If the contents of the messages do not conform to the policies that have been set, the messages will be dropped by the firewall. Gruschka and Luttenberger [6] propose an application level gateway system “Checkway”. They focus on a full grammatical validation of messages by Checkway before forwarding them to the server. Checkway generates an XML Schema from a web service description and validates all web service messages against this schema. 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. Agents that act as sensors are used to detect violations to the normal profile using the data mining technique such as clustering, association rules

and sequential association rules. The anomalies are then further analyzed using fuzzy logic to determine genuine attacks so as to reduce false alarms. An approach to countering DDoS and XDoS attacks against web services is presented by [11]. The system carries out request message authentication and validation before the requests are processed by the web services providers. The scheme has two modes: the normal mode and the under-attack mode. In the under-attack mode, the service requests need to be authenticated and validated before being processed. Finally, a recent solution proposed by [12] presents a Service Oriented Traceback Architecture (SOTA) to cooperate with a filter defense system, called XDetector. XDetector, is a Back Propagation Neural Network, trained to detect and filter XDoS attack message. Once an attack has been discovered by SOTA and the attacker's identity known, XDetector can filter out these attack messages.

Table 1 presents a comparison of our approach with the current approaches aimed at detecting DoS attacks in web services environments. Those parameters that couldn't be evaluated are marked with a hyphen.

Table 1. Comparison of selected models approaches vs. our approach

	XML Firewall	CheckWay Gateway	ID/IP framework	DDoS and XDoS	SOTA & XDetector	Our Approach
Distributed Approach	No	No	No	No	Yes	Yes
Learning ability	No	No	Yes	No	Yes	Yes
Adaptive ability	No	No	Yes	No	No	Yes
Balances the Workload	No	No	-	No	Yes	Yes
Tolerance to Failure	No	Yes	-	-	-	Yes
Scalability	Yes	-	Yes	Yes	Yes	Yes
Ubiquity	No	No	No	No	No	Yes

According to the results shown in Table 1, our approach outperforms the existing models with respect to:

a.) Distributed Approach: our approach is based on a multiagent architecture that can execute tasks derived from the classification process in a distributed way. b) Adaptive ability: our approach includes one type of intelligent agents that was designed to learn and adapt to changes in attack patterns and new attacks c.) Balances the Workload: our approach was designed to distribute the classification task load throughout the various layers of the hierarchical architecture. d.) Tolerance to Failure: our approach has a hierarchical design that can facilitate error recovery through the instantiation of new agents. e.) Scalability: our approach is capable of growing (by means of the instantiation of new agents) according to the needs of its environment. f.) Ubiquity: our approach provides a ubiquitous alert mechanism to notify security personnel in the event of an attack.

Our approach presents novel characteristics that have not heretofore been considered in previous approaches. The next section presents the architecture in greater detail.

3 Multiagent Architecture

Agents are characterized by their autonomy; which gives them the ability to work independently in real-time environments [13]. Furthermore, when they are integrated within a multiagent system they can also offer collaborative and distributed assistance in carrying out tasks [14].

One of the main characteristics of the multiagent architecture proposed in this paper is the incorporation of CBR-BDI [7] 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 structure and content of the SOAP messages and the messages processing tasks.

Our proposal is a distributed hierarchical multiagent architecture integrated for 3 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. The architecture presented in figure 1 shows the three levels with BDI agents organized according to their roles.

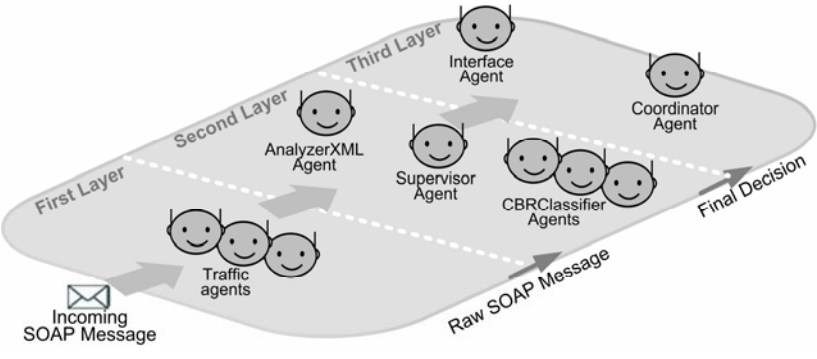

Fig. 1. Design of the multiagent architecture proposed

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

- Traffic agents (layer 1): Capture any traffic directed towards the server. JPCAP [15] is the API used to identify and capture any traffic that contains SOAP message packets. The captured SOAP messages are sent to the next layer in order to carry out the classification process. In addition to these tasks, Traffic 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.

- CBRClassifier agents (layer 2): These CBR-BDI agents carry out the classification process. 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. Once the data have been extracted from the message, a CBR mechanism is initiated by using a Multilayer Perceptron (MLP) neural network in the re-use phase.
- Supervisor Agent (layer 2): This agent supervises the AnalyzerXML agent since there still exists the possibility of an attack during the syntactic processing of the SOAP message.
- AnalyzerXML Agent (layer 2): 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 CBRClassifier agents.
- Coordinator Agent (layer 3): This agent is in charge of supervising the correct overall functioning of the architecture. Additionally, it oversees the classification process. 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 manages the actions required for responding to the attack.
- Interface Agent (layer 3): This agent was designed to function in different devices (PDA, Laptop, and 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.

4 Mechanism for the Classification of Malicious SOAP Messages

The application of agents and multiagent systems facilitates taking advantage of the inherent capabilities of the agents. Nevertheless, it is possible to increase the reasoning and learning capabilities by incorporating a case-based reasoning (CBR) mechanism into the agents. In the case at hand, a CBR classifier agent (CBR-BDI) is responsible for classifying the incoming SOAP messages. In the BDI (Beliefs Desires Intentions) model, the internal structure of an agent and its capacity to choose is based on mental aptitudes: agent behaviour is composed of beliefs, desires, and intentions [7]. A BDI architecture has the advantage of being intuitive and capable of rather simply identifying the process of decision-making and how to perform it.

Case-based Reasoning is a type of reasoning based on the use of past experiences [16]. The purpose of case-based reasoning systems is to solve new problems by adapting solutions that have been used to solve similar problems in the past. The fundamental concept when working with case-based reasoning is the concept of case. A case can be defined as a past experience, and is composed of three elements: A problem description which describes the initial problem, a solution which provides the sequence of actions carried out in order to solve the problem, and the final state which describes the state achieved once the solution was applied. The way in which cases

are managed is known as the case-based reasoning cycle. This CBR cycle consists of four sequential steps: retrieve, reuse, revise and retain [16]. A CBR engine 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 web service environment, which permits it to handle each incoming message based on the particular characteristics of each web service available. Each new SOAP message sent to the architecture is classified as a new case study object. Based on the structure and content of the SOAP messages and the message processing tasks, we can obtain a series of descriptive fields. These fields are extracted from the SOAP message and provide the case description for the CBRClassifier agent. Table 2 presents the fields used in describing the case for the CBR in this layer.

Table 2. Case Description - CBR

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*

Applying the nomenclature shown in the table above, each case description is given by the following tuple:

$$c = \left(i, m, s, n, l, u, h, b, d, t, f, a, c, k, P / c_{.im}, x^p, x^r\right) \tag{1}$$

For each incoming message received by the agent and requiring classification, we will consider both the class that the agent predicts and the class to which the message actually belongs. x^p represents the class predicted by the CBRClassifier agent belonging to the group. $x^p \in X = \{a, g, u\}$; *a*, *g*, *u* represent the values attack, good and undefined respectively; and x^r is the class to which the attack actually belongs $x^r \in X = \{a, g, u\}$, $P / c_{.im}$ is the solution provided by the neural network MLP associated to service *i* and subnet mask *m*.

The reasoning memory used by the agent is defined by the following expression: $P = \{p_1, ..., p_n\}$ and is implemented by means of a MLP neural network. Each P_i is a reasoning memory related to a group of cases dependent of the service and subnet

mask of the client. The Multilayer Perceptron (MLP) is the most widely applied and researched artificial neural network (ANN) model. MLP networks implement mappings from input space to output space and are normally applied to supervised learning tasks [17]. The Sigmoidal function was selected as the MLP activation function, with a range of values in the interval [0, 1]. It is used to detect if the SOAP message 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 given by

$$f(x) = \frac{1}{1+e^{-ax}} \tag{2}$$

The CBR mechanism executes the following phases:

Retrieve: Retrieves the cases that are most similar to the current problem, considering both the type of web service to which the message belongs and the network mask that contains the message.

- Expression 3 is used to select cases from the case memory based on the type of web service and the network mask.

$$c_{.im} = f_s(C) = \{c_j \in C / c_{j.i} = c_{n+1 \cdot i}, c_{j \cdot m} = c_{n+1 \cdot m}\} \tag{3}$$

- Once the similar cases have been recovered, the neural network MLP $P / c_{.im}$ associated to service *i* and mask *m* is then recovered.

Reuse: The classification of the message is begun in this phase, based on the network and the recovered cases. It is only necessary to retrain the neural network when it does not have previous training. The entries for the neural network correspond to the case elements *s, n, l, u, h, b, d, t, f, a, c, k*. Because the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between [0, 1], the incoming variables are redefined so that their range falls between [0.2-0.8]. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of [0.2, 0.8] with the value 0.2 corresponding to a non-attack and the value 0.8 corresponding to an attack. The training for the network is carried out by the error Backpropagation Algorithm [18]. The weights and biases for the neurons at the exit layer are updated by following equations:

$$w_{kj}^p(t+1) = w_{kj}^p(t) + \eta(d_k^p - y_k^p)(1 - y_k^p) y_k^p y_j^p + \mu(w_{kj}^p(t) - w_{kj}^p(t-1)) \tag{4}$$

$$\theta_k^p(t+1) = \theta_k^p(t) + \eta(d_k^p - y_k^p)(1 - y_k^p) y_k^p + \mu(\theta_k^p(t) - \theta_k^p(t-1)) \tag{5}$$

The neurons at the intermediate layer are updated by following a procedure similar to the previous case using the following equations:

$$w_{ji}^p(t+1) = w_{ji}^p(t) + \eta(1 - y_j^p) y_j^p \left(\sum_{k=1}^{M} (d_k^p - y_k^p)(1 - y_k^p) y_k^p w_{kj}\right) x_i^p + \mu(w_{ji}^p(t) - w_{ji}^p(t-1)) \tag{6}$$

$$\theta_j^p(t+1)=\theta_j^p(t)+\eta(1-y_j^p)y_j^p(\sum_{k=1}^{M}(d_k^p-y_k^p)(1-y_k^p)y_k^p w_{kj})+\mu(\theta_j^p(t)-\theta_j^p(t-1)) \quad (7)$$

where w_{kj}^p represents the weight that joins neuron *j* from the intermediate layer with neuron *k* from the exit layer, *t* the moment of time and *p* the pattern in question. d_k^p represents the desired value, y_k^p the value obtained for neuron *k* from the exit layer, y_j^p the value obtained for neuron *j* from the intermediate layer, η the learning rate and μ the momentum. θ_k^p represents the bia value *k* from the exit layer. The variables for the intermediate layer are defined analogously, keeping in mind that *i* represents the neuron from the entrance level, *j* is the neuron from the intermediate level, *M* is the number of neurons from the exit layer.

When a previously trained network is already available, the message classification process is carried out in the revise phase. If a previously trained network is not available, the training is carried out following the entire procedure beginning with the cases related to the service and mask, as shown in equation (8).

$$p_r = MLP^t(c_{\cdot im}) \quad (8)$$

Revise: This phase reviews the classification performed in the previous phase. The value obtained by exiting the network $y = P_r^e(c_{n+1})$ may yield the following situations:

- If $y > \mu_1$ then it is considered an attack
- Otherwise, if $y < \mu_2$, then the message is considered a non-attack or legal
- Otherwise, the message is marked as suspicious and is filtered for subsequent revision by a human expert. To facilitate the revision, an analysis of the neural network sensibility is shown so that the relevance of the entrances can be determined with respect to the predicted value.

Retain: If the result of the classification is suspicious or if the administrator identifies the classification as erroneous, then the network $P/c_{\cdot im}$ repeats the training by incorporating a new case and following the BackPropagation training algorithm.

$$p_r = MLP^t(c_{\cdot im} \cup c_{n+1}) \quad (9)$$

5 Conclusions and Results

Traditional security mechanisms for guaranteeing the integrity and confidentiality of SOAP messages do not offer protection against DoS attacks in web service environments.

This article has presented a distributed hierarchical multiagent 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 proposed mechanism of classification is based on a multiagent architecture whose core is a CBR-BDI agent that incorporates a case-based reasoning engine in conjunction with a neural network of the type Multilayer Perceptron. It is a novel mechanism that had never been used in this type of problems. While the capability of CBR systems is limited to adapting and learning, neural networks can offer predictions, thus turning the solution into a robust classification mechanism with capabilities spanning from automatic learning to adaptation, and the ability to approach new changes that appear in DoS attacks patterns contained in SOAP messages. A prototype of our proposed solution was based on a classification mechanism and developed in order to evaluate its effectiveness. The tests of the simulation were carried out within a small web application developed with Java Server Page and hosted in a Apache Tomcat 6.0.18 Server by using as web service engine, Apache Axis2 1.4.1. The tests were organized within 6 blocks with a specific number of requests (50, 100, 150, 200, 250 and 300) that allowed evaluating the effectiveness of the classifier agent in accordance with the gained experience. Within the blocks were included legal and illegal requests. Figure 3 shows the results obtained.

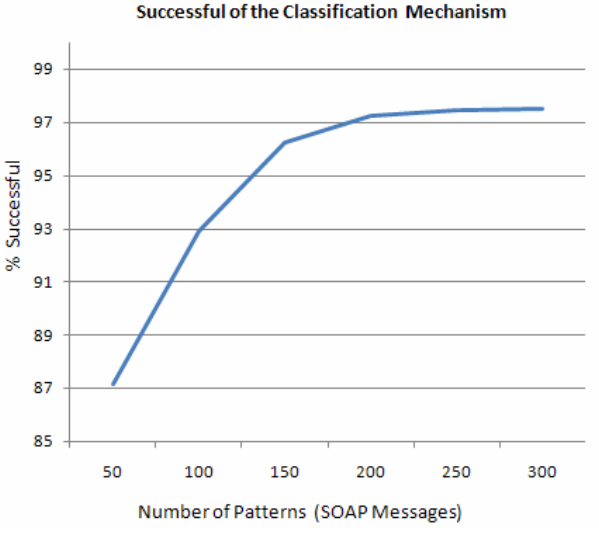

Fig. 3. Effectiveness of the classification mechanism integrated within the CBRClassifier agent according to the number of patterns

Figure 3 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. With the advantage of a distributed process for classification tasks, it would be possible to evaluate both the effectiveness of the classification mechanism, and the response time.

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. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR, Englewood Cliffs (2005)
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. Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B.: Web Services Secure Conversation Language (WS-SecureConversation) Version 1.1 (2004)
6. Gruschka, N., Luttenberger, N.: Protecting Web Services from DoS Attacks by SOAP Message Validation. Security and Privacy in Dynamic Environments 201, 171–182 (2006)
7. 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)
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 '07, pp. 35–44 (2007)
11. Ye, X.: Countering DDoS and XDoS Attacks against Web Services. In: IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 346–352 (2008)
12. Chonka, A., Zhou, W., Xiang, Y.: Defending Grid Web Services from XDoS Attacks by SOTA. In: IEEE International Conference on Pervasive Computing and Communications, pp. 1–6 (2009)
13. 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)
14. Corchado, J.M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences. IEEE Intelligent Systems 23, 19–25 (2008)
15. Fujii, K.: Jpcap - A network packet capture library for applications written in Java (2000), `http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html`
16. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. 7, 39–59 (1994)
17. Gallagher, M., Downs, T.: Visualization of learning in multilayer perceptron networks using principal component analysis. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 33(1), 28–34 (2003)
18. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient BackProp. In: Neural Networks - Tricks of the Trade, p. 546. Springer, Heidelberg (1998)