Learning Algorithms in the Detection of Unused Functionalities in SOA Systems

Ilona Bluemke, Marcin Tarka

Learning algorithms in the detection of unused functionalities in SOA systems

Ilona Bluemke[1], Marcin Tarka[1]

[1] Institute of Computer Science, Warsaw University of Technology,

Nowowiejska 15/19, 00-665 Warsaw, Poland,

I.Bluemke@ii.pw.edu.pl

Abstract. The objective of this paper is to present an application of learning algorithms to the detection of anomalies in SOA system. As it was not possible to inject errors into the "real" SOA system and to analyze the effect of these errors, a special model of SOA system was designed and implemented. In this system several anomalies were introduced and the effectiveness of algorithms in detecting them were measured. The results of experiments can be used to select efficient algorithm for anomaly detection. Two algorithms: K-means clustering and Kohonen networks were used to detect the unused functionalities and the results of this experiment are discussed.

1 Introduction

With the growth of computer networking, electronic commerce, and web services, security of networking systems has become very important. Many companies now rely on web services as a major source of revenue. Computer hacking poses significant problems to these companies, as distributed attacks can make their systems or services inoperable for some period of time. As this happens often, an entire area of research, called Intrusion Detection, is devoted to detect these activities.

Nowadays many system are based on Service Oriented Architecture (SOA) [1,2] idea. A system based on SOA provides functionalities as a suite of interoperable services that can be used within multiple, separate systems from several business domains. SOA also provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. Service-orientation requires loose coupling of services with operating systems, and other technologies that underly applications. SOA separates functionality into distinct units, or services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications.

The objective of this paper is to present the detection of anomalies in SOA systems by learning algorithms. Related work is presented in section 2 and in section 3, a special model of SOA system which was used in experiments, is presented. In this systems several anomalies were introduced. Four algorithms: Chi-Square statistics, k-

K. Saeed et al. (Eds.): CISIM 2013. LNCS 8104, p. 384-395, 2013.

means clustering, emerging patterns and Kohonen networks were used to detect anomalies. Detection of anomalies by k-means and Kohonen networks is presented in section 4 and some conclusions are given in section 5.

2 Related work

Many anomaly detection algorithms have been proposed in the literature. They differ according to the information used for analysis and according to techniques that are used to detect deviations from normal behavior. Lim and Jones in [3] proposed two types of anomaly detection techniques based on employed techniques: the learning model method and the specification model.

The *learning* approach is based on the application of machine learning techniques, to automatically obtain a representation of normal behaviors from the analysis of system activities. The *specification-based* approach requires that someone manually provides specifications of correct behavior. Approaches that concern the model construction are presented in Fig. 1.

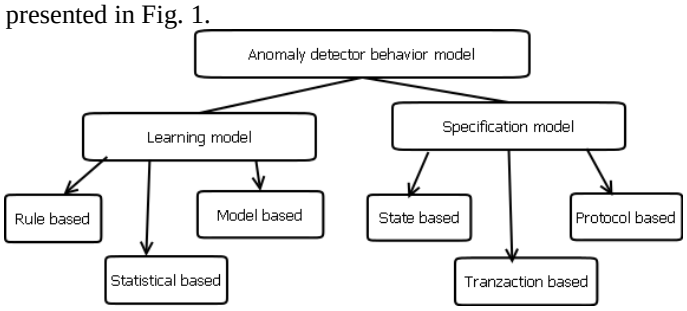

Fig. 1. Taxonomy of anomaly detection behavioral model (based on [3])

The *specification* approach depends more on human observation and expertise than on mathematical models. It was first proposed by C. Ko et. al. [4] and uses a logic based description of expected behavior to construct a base model. This specification-based anomaly detector monitors multiple system elements, ranging from application to network traffic.

In the *protocol based* approach [5] a normal use model is built from the protocol specification e.g. TCP/IP. Lemonnier [5] proposed a protocol anomaly filter able to specifically analyze a protocol and model the normal usage of a specific protocol. This technique can be seen as a filter looking for protocol misuse. The protocol could be interpreted as any official set of rules describing the interaction between the elements of a computer system.

Many protocol anomaly detectors are built as *state* machines [6]. Each state corresponds to a part of the connection, such as a server waiting for a response from client. The transitions between the states describe the legal and expected changes between states. Besides, Z. Shan et. al. [7] uses network state based model approach to describe intrusion attacks.

In the *transaction* based approach the "positive" behavior is formally described. The desired actions and sequence of actions are specified by the definition of transactions. Such explicit definition makes the transaction an integral part of security policy. In the research proposed by R. Buschkes et. al. [8] the detection of anomalies is based on the definition of correct transactional behavior.

The *learning* model must be trained on the specific network. In the training phase, the behavior of the system is observed and logged and machine learning techniques are used to create a profile of normal behaviors. In the process of creating an effective anomaly detection model: *rule-based, model-based,* and *statistical-based* approaches have been adopted to create the baseline profile.

Rule-based systems used in anomaly detection describe the normal behavior of users, networks and/or computer systems by a set of rules. These predefined rules typically look for the high-level state change patterns observed in the audit data.

SRI International's Next-generation Intrusion Detection Expert System (NIDES) [9] is an innovative statistical algorithm for anomaly detection, and an expert system that encodes known intrusion scenarios. The features considered by NIDES are related to user activity, for instance ,CPU and file utilization. The information collected by NIDES is compared with the long - term profile of analyzed system and used for the learning process based on Chi-squared statistic. Owens et. al. present in [10] an adaptive expert system for intrusion detection based on fuzzy sets.

In the *model based* anomaly detector the intrusions are analyzed at a higher level of abstraction than the audit records of the rule based approach. Execution is restricted to pre-computed patterns of expected behavior. Different types of models are used to characterize the normal behavior of the monitored system like *data mining, neural networks, pattern matching, etc.*

Data mining extracts the behavioral models from a large amount of data. Intrusion detection systems require frequent adaptation to resist new attacks, so data mining techniques that can adaptively build new detection models are very useful. An example of data mining system was proposed by Lee et. al. and is presented in [11]. The key idea is to mine network audit data, then use the patterns to compute inductively learned classifiers that can recognize anomalies and known intrusions.

A *neural network* learns the user profile in the training process. Training may use data more abstract than the audit records. The fraction of incorrectly predicted events constitutes the variance of the user behavior. [12] describes anomalies detection in information system based on Kohonen networks. There are several libraries supporting building neural networks e.g. [13 - 15]. Other neural networks based systems for intrusion detection are also described in [16- 19].

In the *pattern matching* approach, learning is used to build a traffic profile for a given network. Traffic profiles are built using features such as packet loss, link utilization, number of collisions. Normal behavior is captured as templates and tolerance limits are set based on different levels of standard deviation. The usage of emerging patterns in anomaly detection is described in [20].

First *statistical based* anomaly detection was proposed by Denning and Neumann [21] in 1985. The anomaly detector observes subjects and generates profiles for them that represent their behavior. The widely known techniques in statistics can often be

applied; e.g. data points that lie beyond a multiple of the standard deviation on either side of the mean might be considered anomalous. There are many statistical techniques like *Bayesian statistics*, *covariance matrices and Chi-square statistics* [22] for the profiling of anomaly detection. Example of *Chi-square statistic* in anomaly detection is described in [23]. Statistical approaches disadvantage is the insensitivity to the order of occurrence of events. Sequential interrelationships among events should be considered for more accurate detection. It is also difficult to determine a threshold above which an anomaly should be considered.

Commercial products applying rule based, statistical based, model based and neural networks approach to detected anomalies are briefly described in [3].

3 Research model

As we were not allowed to use real SOA system and inject anomalies into it, a special system was implemented. The business idea of this system VTV (Virtual TV) is presented in Fig. 2. The exemplary company is a virtual TV provider. The company does not have its own technical infrastructure and is associating TV digital provider with the telecommunication operator. The receiving equipment is delivered to the client by one of courier companies. The company is also using two applications: Customer Relationship Management (CRM) containing all clients data and a storage management system.

The VTV system simulates a real SOA system and enables to inject typical anomalies into its regular operation. Configurable are frequencies of : single services calls, group of services calls, processes calls, and services in a context. More details can be found in [24].

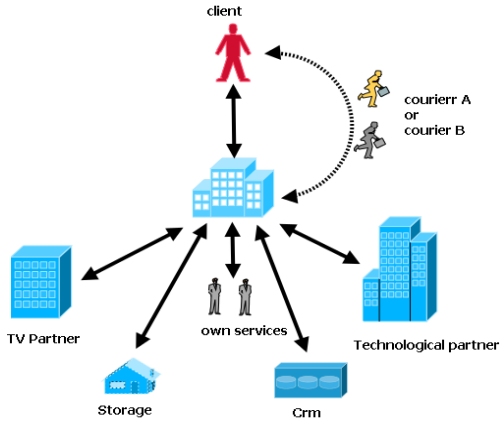

Fig. 2. The business relations of an exemplary company

The architecture of VTV system is presented in Fig. 3. The *request generator* simulates activities of clients. It generates different types of request (e.g. create, modify, deactivate) for available services (e.g. TV or hardware). Depending of the value of

request some requests can be identified as very important. The generated address of a client determines the currier group. The configurable parameters of the request generator enable to simulate real operation and to inject some anomalies: e.g. request for courier, hardware, TV services can be set, ratio of create, modify, deactivate requests in generated requests can be chosen.

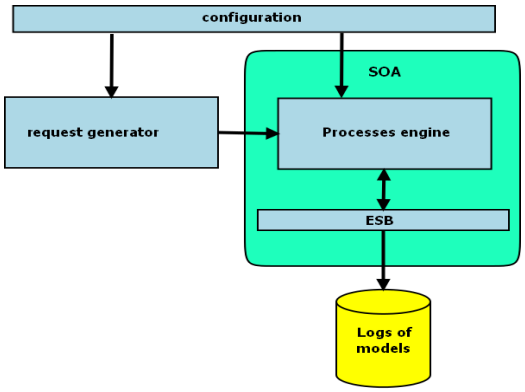

Fig. 3. Architecture of VTV system

The generated request is transferred to the *business processes engine* (Fig.3) which composes processes from services available on Enterprise Service Bus (ESB). Outputs of processes are logs. Logs of model contain information similar to logs from monitoring real SOA systems.

3.1 Environment of experiment

All experiments were conducted on PC with Intel Core 2 Duo T7200, 512 Mb of memory under Fedora Core operating system. The examined algorithms are using text input files prepared from logs of the VTV system (Fig.3). In this log file information like number of requests, name of processes, name of services called, execution time are written. These logs files are then transformed by scripts, implemented in R [25] environment, into transactions or summarized reports which are input to detection algorithms. The flow of data from logs of the model is shown in Fig. 4.

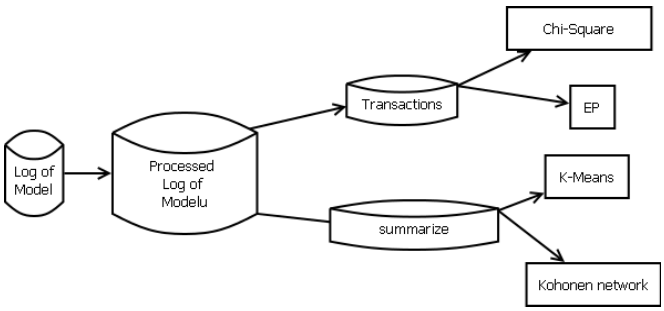

Fig. 4. Flow of data from logs

The implementation of algorithms for anomaly detection *k-means*, *Chi-Square* and *Kohonen* networks algorithms was made in R environment while the *emerging patterns* algorithm, which is more complicated, was implemented in C++.

k-means algorithm was prepared based on [26], for *emerging patterns* algorithm information from [20, 27] were used, *Chi-Square* detector was taken from [23].

4 Experiment

The research system presented in section 3 was used to explore four cases typical for SOA systems i.e.: change in the frequency of service calls, change in the frequency of a group of services, change of the context of services calls and not used functionalities.

For each of the above listed cases it was expected, that anomalies detector provides information useful for the maintenance team. Four learning algorithms (section 2) were used in the anomalies detector: *Chi-square statistic*, *Kohonen network*, *Emerging patterns* and *k-means clustering*.

Each of the above algorithms represents different approach to anomalies detection. The goal of the experiment was to examine advantages and disadvantages of each of these algorithms. Anomalies detection is a kind of clustering with two types of clusters grouping normal and abnormal behaviors. Correctly identified anomaly is an abnormal event which was introduced by purpose by case' scenario and was assigned to the abnormal cluster. Identified as anomalies other events or not recognized anomalies are treated as errors. The values measured in experiments are:

- FP (false positive) – number of incorrectly identified anomalies,
- FN (false negative) - number of not recognized anomalies,
- TP (true positive) - number of correctly identified normal behaviors,
- TN (true negative) - number of incorrectly identified normal behaviors.

Good anomalies detector should generate small number of errors. To compare the quality of detectors often sensitivity and specificity measures are used. The sensitivity and specificity are defined accordingly as:

$$sensitivity = \frac{TP}{TP + FN} \tag{1}$$

and

$$specificity = \frac{TN}{TN + FP} \tag{2}$$

The relation between specificity and sensitivity – ROC (Receiver Operating Characteristics) curve [28] can be used to compare the quality of models. ROC as a technique to analyze data was introduced during the second world war to identify if signal seen on a radar were coming from an enemy, an alliance or if it were noise. Currently ROC curves are used to analyze different types of data e.g. radiological data.

The construction of ROC curves during experiments was as follows:

1. create entities with algorithm' parameters
2. for each entity :
    - perform experiment
    - calculate specificity and sensitivity
    - mark the point on the diagram
3. draw the line connecting points.

For each examined algorithm also the learning time was calculated.

4.1 Plan for experiment

The examination of anomaly detection algorithms was based on four test cases: changes in frequency of service and groups of services calls, change of the context of services calls, and lacking functionalities. Each of these cases simulates one type of anomaly typical for SOA.

The experiment was conducted in following steps:

1. Create data for regular behavior
2. For each of test case' scenario :
    - Create data for abnormal behavior in this scenario
    - For each algorithm execute:
        - Using regular data perform the learning phase
        - Perform detection phase on data for abnormal behavior
        - Evaluate the quality of detection
3. Compare algorithms.

Below the results for the scenario "*not used functionalities*" are presented. The detection of this kind of anomalies is very important in distributed systems especially in SOA systems. Some functions may be not used as a result of errors in routing algorithm or in business logic. If not used services were dedicated to some cluster this cluster could be used for other purposes. In ours research' environment this anomaly was obtained by forcing the request generator not to generate courier delivery services by setting parameter `Courier_share` to zero [24]. In [29] the detection of anomalies in the frequency of service calls by K-means clustering algorithm and emerging patterns are described.

4.2 Results of experiments

The goal of examination was to find high level of detection with minimal number of false alarms. If the ideal detection was not possible preferred were the results with no false alarms. In practice, if system is generating many false alarms the user will neglect any alarm.

***k-means* algorithm**

k-means clustering [26] is a clustering analysis algorithm that groups objects based on their feature values into *k* disjoint clusters. Objects that are classified into the same cluster have similar feature values. *k* is a positive integer number specifying the num-

ber of clusters, and has to be given in advance. All objects are assigned to their closest cluster according to the ordinary Euclidean distance metric.

At the beginning the summarized reports used by the algorithm were created for all logs in the system. The results are given in Table 1. The maximal distance from centroid of cluster for a cluster member is denoted as *maxL*.

Table 1. Results for clustering algorithm.

row	k	maxL	logSize	TP	TN	FP	FN	Sensitivity	Specificity
1	4	1	150	0	1	14	23	1	0.6
2	4	2	150	0	3	12	23	0	0.2
3	4	2.3	150	23	9	6	0	1	0.6
4	4	2.5	150	23	3	12	0	1	0.2
5	5	2	150	23	5	10	0	1	0.33
6	5	2.3	150	23	6	9	0	1	0.4
7	5	2.5	150	23	8	7	0	1	0.53
8	4	1	350	10	4	3	0	1	0.57
9	4	1.5	350	10	5	2	0	1	0.71
10	4	2	350	0	5	2	10	0	0.71
11	4	2.5	350	0	5	2	10	0	0.71
12	4	3	350	0	6	1	10	0	0.86

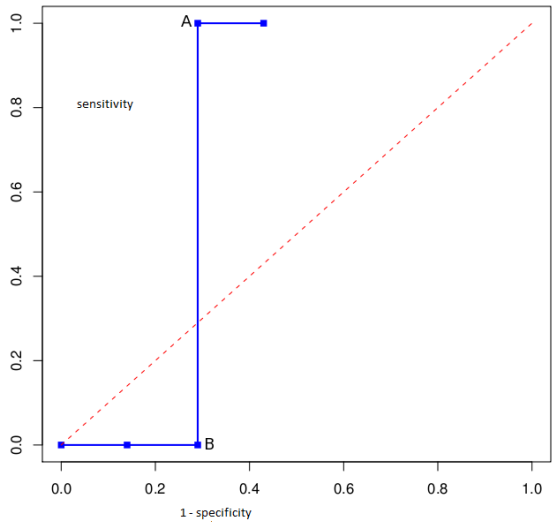

Fig. 5. ROC curve for k-means algorithm with 350 training logs

Initially the experiment was conducted for 150 logs in training set. The anomaly wasn't satisfactorily detected, the numbers of not identified anomalies were high (e.g. 23, first row in Table 1), also the numbers of incorrectly identified anomalies were high (e.g. 14 first row in Table 1). In the training data there were many summarization reports in which the number of services not called in regular data, was close to zero, so the lack of calls of these services was treated as normal behavior. Next, the number of logs was increased to 350. The results improved (rows 8-12 Table 1).

In Fig.5 the ROC curve is shown for 350 training logs. The high change in sensitivity is seen – from zero to one. Point A was obtained by decreasing the value of `maxL` parameter from the value in point B, till the level in which zero was not assigned to the group with the minimal centroid.

The *k-means* clustering algorithm can be used in the detection of not used functionalities if the number of training data is high. In [29] we have shown that this algorithm was not suitable in detecting the change in frequencies of single services calls.

Kohonen networks

A neural network is a set of simple, highly interconnected units called neurons. Neurons transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them, therefore by adjusting the weight the output can be controlled. The process of updating the weights and thresholds is called learning.

Training usually takes the form of presenting the network with set of typical inputs vectors. In unsupervised learning mode (self - organizing map [30] - Kohonen networks), the inputs are presented for training; the adjustments are made so that the network is able to recognize inputs as belonging to a particular profile.

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron whose weight vector is most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance (within the lattice) from the BMU. This process is repeated for each input vector for a number of cycles. The network winds up associating output nodes with groups or patterns in the input data set.

The results for Kohonen network algorithm are shown in Table 2. Column `Lambda` – toleration ration, shows the restraint to the distance from BMU. In rows 1-5 the numbers of not recognized (FN) and of incorrectly identified anomalies (FP) are high. After increasing the number of elements in the training phase the results of detection significantly improved. In rows 6-11 only few anomalies were not recognized.

Table 2. Results for Kohonen network algorithm.

row	N	Lambda	Logsize	TP	TN	FP	FN	Sensitivity	Specificity
1	25	0.5	150	358	331	137	398	0.47	0.707
2	25	1	150	324	366	102	432	0.43	0.782

3	25	1.5	150	254	393	75	502	0.34	0.840
4	25	2	150	163	427	41	593	0.22	0.912
5	25	2.5	150	72	443	25	698	0.1	0.95
6	25	0.2	350	12	77	41	0	1	0.65
7	25	0.5	350	12	137	67	0	1	0.67
8	25	1	350	12	172	32	0	1	0.84
9	25	1.5	350	11	191	13	1	0.92	0.94
10	25	2	350	8	190	14	4	0.67	0.93
11	25	2.5	350	6	198	6	6	0.5	0.97

In Fig. 6 the ROC curve for 350 training logs is shown. Point A is optimal, with the sensitivity= 0.92 and specificity= 0.94 and the costs of false qualification of anomaly and false qualification of normal behavior are equal. These values may be improved by increasing the training data. Point B represents the case in which all anomalies were detected, the specificity was equal to 0.84.

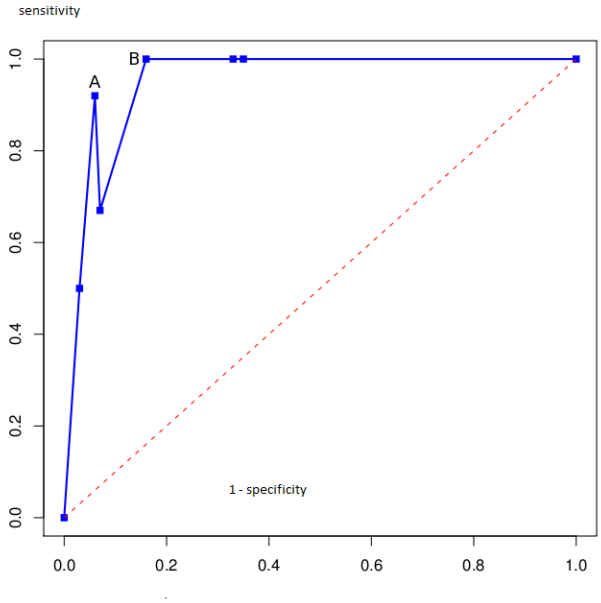

Fig. 6. ROC curve for Kohonen networks algorithm with 350 training logs

The above presented data show that Kohonen networks can be used to detect the unused functionalities but large training data must be provided.

5 Conclusions

In this paper some results of the detection of one anomaly – unused functionalities in SOA system are presented. An experiment was conducted in an environment designed to introduce several types of anomalies. This environment, described in section 3, enables the detection of anomalies by four learning algorithms, from different types (section 2): *emerging patterns*, *k-means* clustering, *Kohonen* networks and statistical *Chi-Square* . In this paper the results of only two: *k-means* clustering and *Kohonen* networks are presented. Both algorithms were able to satisfactorily detect unused functionalities while two others i.e. *Chi-Square* and *emerging patterns* appeared to be inappropriate in the detection of unused functionalities.

In [29] the results of detecting other anomaly - the change in frequencies of service calls was described. The least accurate in the detection of this kind of anomaly was the *k-means* clustering algorithm and the best was *emerging pattern* algorithm. Kohonen algorithm also produced quite good results.

The results presented in this paper and in [29] show that in anomalies detection in SOA systems different algorithms may appear most suitable for different type of anomaly so further research should be conducted. The exemplary SOA system (section 2) enables to conduct other experiments examining the suitability of learning algorithms in the detection of other anomalies. The results of these experiments will be available soon.

Acknowledgments. We are very grateful to the reviewers for many valuable remarks.

References

1. BPEL Standard, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, (access July 2011)
2. SOA manifesto: http://www.soa-manifesto.org (access July 2011)
3. Lim S. Y., Jones A.: Network Anomaly Detection System: The State of Art of Network Behavior Analysis. In Proc. of the Int. Conf. on Convergence and Hybrid Information Technology 2008. DOI 10.1109/ICHIT2008.249, pp. 459-465 (2008)
4. Ko C., Ruschitzka M., Levitt K.: Execution monitoring of security-critical programs in distributed systems: a specification-based approach. In Proc. of IEEE Symposium on Security and Privacy. Oakland, CA, USA (1997)
5. Lemonnier E.: Protocol Anomaly Detection in Network-based IDSs. Defcom white paper (2001)
6. Sekar R., Gupta A., Frullo J., Shanbag T., Tiwari A., Yang H., Zhou S.: Specification-based anomaly detection: A New Approach for Detecting Network Intrusions. In ACM Computer and Communication Security Conference. Washington, DC, USA (2002)
7. Shan Z., Chen P., Xu Y., Xu K.: A Network State Based Intrusion Detection Model. In Proc. of the 2001 Int. Conf. on Computer Networks and Mobile Computing (ICCNMC'01) (2001)
8. Buschkes R., Borning M., Kesdogan D.: Transaction-based Anomaly Detection. In Proc. of the Workshop on Intrusion Detection and Network Monitoring. Santa Clara, California, USA (1999)

9. Anderson D., Frivold T., Valdes: A Next-generation Intrusion Detection Expert System (NIDES) Summary. Computer Science Laboratory, SRI-CSL-95-07, May 1995 (1995)
10. Owens S., Levary R.: An adaptive expert system approach for intrusion detection. International Journal of Security and Networks, vol. 1, no 3-4 (2006)
11. Lee W., Stolfo S. J.: Data mining approaches for intrusion detection. In Proc. of the 7th USENIX Security Symposium (1998)
12. Bivens A., Palagrini Ch., Smith R., Szymański B., Embrechts M.: Network-based intrusion detection using neural networks. In Proc. Intelligent Eng. Systems through Neural Networks ANNIE 2002. St. Louis, MO vol.12, ASME Press, NY, pp. 579-584 (2002)
13. C Neural network library, http://franck.fleurey.free.fr/NeuralNetwork/
14. NeuroBox, http://www.cdrnet.net/projects/neuro/
15. Fast Artificial Neural Network Library, http://sourceforge.net/projects/fann/
16. Ryan J., Lin M., Miikkulainen M.: Intrusion Detection with Neural Networks. Advances in Neural Information Processing Systems, vol. 10 (1998)
17. Ghosh A. K., Schwartzbard A.: A Study in Using Neural Networks for Anomaly and Misuse Detection. In Proc. of the 8th USENIX Security Symposium. Washington D.C., USA (1999)
18. Sang-Jun Han, Sung-Bae Cho: Evolutionary Neural Networks for Anomaly Detection Based on the Behaviour of a Program. IEEE Transactions on Systems, Man and Cybernetics (2006)
19. Bivens A. et al.: Network-based intrusion detection using neural networks. In Proc. of Intelligent Engineering Systems through Artificial Neural Networks ANNIE-2002. St.Luis, MO, vol. 12, pp. 579-584, ASME press, New York (2002)
20. Ceci M., Appice A., Caruso C., Malerba D.: Discovering Emerging Patterns for Anomaly Detection in Network Connection Data, LNAI vol. 4994, pp. 179–188, Springer (2008)
21. Denning D., Neumann P.: Requirements and Model for IDES-A Real-Time Intrusion-Detection Expert System. SRI Project 6169, SRI International, Menlo Park, CA (1985)
22. Masum S., Ye E M, Chen Q., Noh K.: Chi-square statistical profiling for anomaly detection. In Proceedings of the 2000 IEEE Workshop on Information Assurance and Security (2000)
23. Ye N., Chen Q.: An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Qual. Reliab. Eng. Int, vol. 17, pp.105–112 (2001)
24. Tarka M.: Anomaly detection in SOA systems. Msc Thesis, Institute of Computer Science, Warsaw University of Technology (in polish) (2011)
25. The R Project for Statistical Computing: http://gcc.gnu.org/ (access September 2011)
26. Munz G., Li S., Carle G.: Traffic Anomaly Detection Using K-Means Clustering, Wilhelm Schickard Institute for Computer Science, University of Tuebingen (2007)
27. Guozhu D., Jinyan L.: Efficient Mining of Emerging Patterns: Discovering Trends and Differences. Wright State University, The University of Melbourne (2007)
28. Hanley J. A.: Receiver operating characteristic (ROC) methodology: the state of the art. Crit Rev Diagn Imaging (1989)
29. Bluemke I., Tarka M.: Detection of anomalies in SOA system by learning algorithms. In Complex Systems and Dependability. Zamojski W. et al. (eds.), Advances in Intelligent and Soft Computing, vol. 170, pp. 69-85, Springer, DOI: 10.1007/978-3-642-30662-4_5 (2012)
30. Kohonen T.: The self-organizing map. Proc. IEEE, vol. 78, no. 9, pp. 1464-1480, Sept. (1990)