Negative Selection with Antigen Feedback in Intrusion Detection

Wanli Ma, Dat Tran, Dharmendra Sharma

Faculty of Information Sciences and Engineering
University of Canberra, Australia
{Wanli.Ma, Dat.Tran, Dharmendra.Sharma}@canberra.edu.au

Abstract. One of the major challenges for negative selection is to efficiently generate effective detectors. The experiment in the past shows that random generation fails to generate useful detectors within acceptable time duration. In this paper, we propose an antigen feedback mechanism for generating the detectors. For an unmatched antigen, we make a copy of the antigen and treat it the same as a newly randomly generated antibody: it goes through the same maturing process and is subject to elimination due to self matching. If it survives and is then activated by more antigens, it becomes a legitimate detector. Our experiment demonstrates that the antigen feedback mechanism provides an efficient way to generate enough effective detectors within a very short period of time. With the antigen feedback mechanism, we achieved 95.21% detection rate on attack strings, with 4.79% false negative rate, and 99.21% detection rate on normal strings, 0.79% false positive. In this paper, we also introduce Arisytis – Artificial Immune System Tool Kits – a project we are undertaking for not only our own experiment, but also the research communities in the same area to avoid the waste on repeatedly developing similar software. Arisytis is available on the public domain. Finally, we also discuss the effectiveness of the r-continuous bits match and its impact on data presentation.

Keywords: Artificial Immune System, Negative Selection, Intrusion Detection System

1 Introduction

Artificial Immune System (AIS) is a branch of computational intelligence, inspired by biologic immune systems. It was first proposed by Forrest et al [1] and has attracted increasing interest from the research communities in the last 20 years [2-5]. Like the other biologically inspired models, such as Artificial Neural Networks, Evolution Algorithms, and Ant Colony etc., AIS is based on the observation of the behaviors and the interaction of antibodies and antigens in a biological system [6, 7]. Negative selection, clonal selection, and immune network theory are the three most popular theories of the current AIS research [3, 8].

Negative selection [9-11] mimics the way a human body detects and destroys harmful antigens. A human body constantly produces lymphocytes, with randomly

mutated surface peptides, from born marrow. A lymphocyte is recognized by its surface peptides, because these peptides are used to match against other cells. All newly generated lymphocytes are sent to thymus to mature. The thymus has almost all types and shapes of self cells. During this period of maturing time, if a lymphocyte matches any cell in the thymus, the lymphocyte is just a copy of a self cell and is then destroyed. Only these which do not match any self cell in the thymus are sent to the body to match, or detect, antigens (also called pathogens), which are invasion cells. The lymphocytes keep trying to match all the cells, including both self cells and antigens, in the body. If a match happens, it basically means that a non-self cell (antigen) is just detected. An alarm might be raised, and immune actions may follow. The lymphocyte which matches the antigen may become a memory lymphocyte and stays in the body to quickly respond to the same invasion in the future. If for a period of time, a lymphocyte does not make any match, it will age and die. New lymphocytes with random peptide mutations are being generated to replace the dead ones. For the detailed explanation on how the immune system works, under the context of AIS, we refer the readers to [12, Chapter 2].

The terms used in AIS literature are yet to be standardized. In this paper, we use the terms *antibody* and *detector* interchangeably. We also use a *memory antibody* (or detector) to mean that the antibody has been successfully activated and also matched the incoming antigens many times. Finally, we view the data to be verified, i.e., to be matched by the antibodies, as a stream of *antigens*. For the purpose of simplicity, we call all data items to be verified as *antigens*, regardless of being attacks or self.

Negative selection, due to its ability of discriminating self and non-self, fits naturally into the area of intrusion detection. There are a few proposals of using negative selection for intrusion detection purpose. The first, and perhaps the most cited system, is LISYS by Hofmeyr etc. [7, 11]. In [13], Balthrop reported comprehensive results of different parameter settings of LISYS. In LISYS, detectors and antigens are represented as the strings of 49 bits long. Gabrielli and Rigodanzo [14] proposed a similar intrusion detection system but restricted their experiment on the HTTP requests to a web server. Gonzalez and Dasgupta proposed a real-valued negative selection (RNS) algorithm [15], where detectors and antigens are represented as real valued vectors. They tested the algorithm on MIT Lincoln Lab DARPA 99 dataset and achieved 95%-98% detection rates (with different false alarm rates). Ji and Dasgupta had a comprehensive survey paper on the development in negative selection [16].

The success of negative selection depends on the success of generating detectors. In [17], Kim and Bentley reported the difficulties in generating useful detectors within an acceptable time window. They concluded that negative selection suffers from scalability problem. Our experiment also repeated their observations; however, we do not share their conclusion. The problem is not on negative selection itself, and the solution is on finding a means to efficiently generate effective detectors.

In this paper, we propose an antigen feedback mechanism to efficiently generate effective detectors. In addition to the randomly generated detectors, for an unmatched antigen, we copy it into the detector space and treat it the same as a randomly generated detector. The detector is called a *feedback detector* (or a *feedback antibody*). This new detector goes through the same maturing process and is subject to elimination if it matches any of the self strings. If it survives, it is used to match

further incoming antigens. If it can be activated, by exceeding the pre-set activation threshold, it becomes a legitimate detector.

The antigen feedback mechanism is justified. The goal of randomly generating detectors is not the randomization, but to generate effective detectors. The ideal situation is that every randomly generated detector matches a type of incoming antigens. Therefore, there is no waste on the generated detectors. This is the ideal situation, which has maximum efficiency, but unachievable in reality. Keeping the goal of randomly generating detectors in mind, it is acceptable to copy an unmatched antigen into the detector space, as it is the same as any one of these many randomly generated detectors. However, this simple antigen feedback mechanism has great impact on the quality of detectors. It makes the detector generating almost close to maximum efficiency.

We employ a similar implementation as LISYS but use KDD CUP 1999 dataset [18] to conduct our experiment on the antigen feedback mechanism. The experiment is conducted on an in-house developed system called Arisytis. Arisytis (Artificial Immune System Tool Kits) is a project we are undertaking in the attempt to implement the up to date AIS algorithms. It does not just provide a test bed for our own research, but is also available in the public domain for the other research communities to avoid the waste of time in repeatedly developing similar software (http://staff.ise.canberra.edu.au/dtran/).

In this paper, we will also discuss the impact of *r-contiguous bits match* method [7, 11] on the presentation format of antibodies and antigens.

The rest of the paper is as follows. In Section 2, we briefly introduce Arisytis. Section 3 provides the details of preparing KDD CUP 1999 dataset for our experiment. Section 4 discusses the generating of detectors. Section 5 gives the experiment results, with the discussion on the r-contiguous bits match method and its impact on data presentation. We conclude the paper with future work in Section 6.

2 Arisytis

Arisytis is a project we are currently undertaking, through which we are trying to integrate the up-to-date AIS algorithms into a single program. Arisytis is designed for research experimenting and educational purposes. It is highly configurable and also provides real time updates on its run-time activities. The real-time updates make it less efficient, but it is justified for its purposes. Changing Arisytis parameters is just a matter of filling forms and ticking boxes. Arisytis is developed on Microsoft C#.NET environment and has an intuitive graphic user interface, Fig 1.

The top region of Arisytis window has 4 groups: *Global Parameters*, *Antibodies*, 4 *buttons* (OpenTrain, OpenTest, Run/Rerun, and Quit), and *Running Environment*. The elements in Global Parameters group are used to change system wide parameters, such as, the number of allowed antibodies, effective length of an antibody, the duration of a clock cycle, and the time to live for a newly generated antibody etc. The elements in Antibodies group have the parameters related to antibodies, for example, the value R for r-continuous bits match and if antigen feedback is turned on etc. The 4 buttons are used to open relevant files and start an experiment run. The elements in

Running Environment group are updated in real time manner to report system activities.

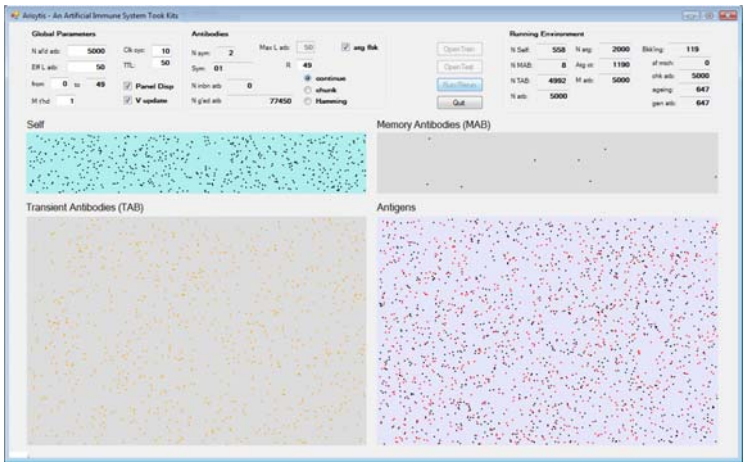

Fig. 1. Arisytis user interface.

The lower region of Arisytis window has four panels to display the self strings, antibodies, and antigens. *Self* panel displays self strings, one dot per string. *Transient Antibodies Panel* contains generated antibodies, one dot per antibody. A green dot means that the corresponding antibody is newly generated and hasn't gone through the maturing process yet. An orange dot means that the corresponding antibody survives the maturing process and is ready to match against incoming antigens. After the number of the matches exceeds the predefined matching threshold, which can be adjusted in Global Parameters group, this antibody is promoted into a memory antibody and thus moved to *Memory Antibody* panel. *Antigens* panel displays all the antigens, one dot per antigen, as usual. Although we view antigens as a flow of antigens coming from outside, for displaying purpose, we display all antigens in this panel without the order. An orange dot means that the corresponding antigen has been matched by a memory antibody, i.e., is recognized or detected. A black dot means that the corresponding antigen is either not recognizable by the memory antibodies or yet to be processed. While the display of the antigens ignores the order of the antigens in the antigen flow, the program processes the antigens one by one in order through its flow. The order of the antigens being processed is important, especially when the antigen feedback mechanism is turned on.

3 Preparing the Dataset

KDD CUP 1999 dataset was based on MIT Lincoln Lab intrusion detection dataset, also known as DARPA dataset [19]. The data was produced for "*The Third International Knowledge Discovery and Data Mining Tools Competition, which was*

held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining" [18]. The raw network traffic records have already been converted into vector format. Each vector has 41 fields (features). We refer the readers to [18] and [20] for the meanings of these fields. If we ignore the fields with symbolic values, the rest of the fields can be classified into 4 categories:

- *Group I*: fields 0, 4, 5, and 7, these fields are the basic characteristics of a connection. They are the durations, the octets transferred, and wrong fragmentation flags of the connection. We ignore the fields with symbolic values, field 1, 2, 3, and 6, in this paper.
- *Group II*: fields 10-19, these fields are actually not traffic features. The values cannot be obtained by looking at the traffic records alone. The help from host based logs is needed.
- *Group III*: fields 22-30, these fields are time based traffic features. They are the statistics of traffic features in the previous 2 second time window. The calculation is based on the source IP address.
- *Group IV*: fields 31-40: the same as Group III, except that the calculation is destination IP address oriented.

Among the 4 groups, either Group III or Group IV contributes most to the detection rate, and combining the groups won't increase the detection rate [21]. Therefore, we primarily choose Group III fields for our experiment. In addition, we also choose Field 1, 2, and 3. We convert these fields into a string of 50 bits, which has 10 segments as follows:

- *Segment 1*: 1 bit for Field 24. The value range for the field is 0-0.94 with predominately 0s. If the value is 0, we set the segment 0; otherwise 1.
- *Segment 2*: 1 bit for Field 25. The value range for the field is 0-1 with predominately 0s. If the value is 0, we set the segment 0; otherwise 1.
- *Segment 3*: 9 bits for Field 22. The value range for the field is 1-511. We convert the value into its binary format.
- *Segment 4*: 9 bits for Field 23, the same as Segment 3.
- *Segment 5*: 3 bits for Field 1. There are only 3 different values for this field: TCP, UDP, and ICMP; therefore, 100 for TCP, 010 for UDP, and 001 for ICMP.
- *Segment 6*: 7 bits for Field 2. There are 70 different services for this field, for example, auth, ftp, http, and telnet etc. We order the 70 services, in alphabet order, from 1 to 70, e.g. auth ordered as No. 2, ftp No. 17, http No. 22, and telnet No. 59. We then convert the order value into its binary format.
- *Segment 7*: 11 bits for Fields 3. There are 11 distinct flags for the fields. They are OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, and SH. We set 1 bit of the 11 bits to 1 for one of the flags, the same as Segment 5.
- *Segment 8*: 1 bit for Filed 28. The value range for the field is 0-1 with almost all values as 1s. If the value is 1, we set the segment 1; otherwise 0.
- *Segment 9*: 1 bit for Field 29. The value range for the field is 0-1 with almost all values as 0s. If the value is 0, we set the segment 0; otherwise 1.

- *Segment 10*: 7 bits for Field 30. The value range for the field is 0-1 with even distribution. We first times the value with 100 (range: 0-100) and then convert the product into its binary format.

We use the file "kddcup.data" to generate self strings. We pick up all the vectors with the label "normal" and then convert them into the bit strings as described. The file contains 972,781 vectors with the "normal" label. After being converted into bit strings, we obtain 23,587 unique strings. With these 23,587 unique strings, most of them are just a one off string, and only 558 strings have each individual string being repeated for more than 100 times. We keep these 558 strings as self strings.

We use the file "corrected" to generate the testing strings. The file contains 311,029 vectors, among which 60,593 are labeled as "normal", and the rest 250,436 are labeled with verities of attacks. For our experiment purpose, we pick up the first 10,000 of "normal" labeled vectors and attack labeled vectors respectively and convert them into bit strings.

4 Generating Detectors

The efficiency and the accuracy of the detection are decided by how well the detectors are generated. We encountered the same difficulty as report by Kim and Bentley [17] – we cannot generate even a single useful detector in a period of time which is even far beyond the time required to process all antigens. A simple analysis reveals that the difficulty is actually expected.

We use a 50 bits string to represent an antibody or an antigen. It means that the number of all possible strings is 2^{50}, which is about 10^{15}. As discussed before, from the 250,436 vectors with attack labels, we only obtain 12,351 unique strings. If we generate the detectors completely randomly, and assume that the random numbers are evenly distributed, the chance for us to generate a useful detector is $\frac{12,351}{10^{15}} \approx 10^{-11}$, which is fundamentally impossible.

To overcome the problem, we introduce the antigen feedback mechanism. For any unmatched antigen, we copy it into the antibody repository. It is then treated the same as a randomly generated detector and is subject to the same maturing, eliminating, and activating processes. If it survives, it becomes a legitimate detector.

5 Experiment Results and Discussions

With the data described in Section 3 and the antigen feedback mechanism, we run a number of experiments on the 10,000 attack strings and 10,000 normal strings, with the different combinations of R for r-continuous bits match and T for the number of matches to activate an antibody. Table 1 lists some sample results. In the table, R is the value of r-continuous bits match, T is the activation threshold, "D Rate" means

detection rate, “M ATB” lists the number of memory antibodies obtained during this run.

Table 1. Some sample results of different combinations of R and T.

R	T	Attacks		Normal	
		D Rate	M ATB	D Rate	M ATB
49	3	94.96%	8	98.50%	3
	5	93.95%	4	98.92%	2
	10	90.67%	3	99.45%	1
40	3	94.99%	8	98.51%	2
	5	93.99%	4	98.81%	2
	10	90.43%	3	99.60%	1
32	3	65.65%	41	98.41%	5
	5	65.59%	37	98.73%	4
	10	61.31%	25	99.60%	1

For the attack strings, we have 3 types of results. When *R* is small ($R \le 32$), the results are random, as the match is too general to produce any meaningful results. The larger the value *R* is, the more specific the match is. When $R = l$, where l is the full length of a string, any two strings have to be exactly the same to make a match; while at the other extreme $R = 0$ means that any two strings will always match. When *R* is close to the full string length ($R \ge 35$), the detection rates are around 94%, when the activation matching threshold was set at 5. The error rates 6% (from the detection rate 94%) were mainly caused by the initial learning process. The system has to learn, from the feedback antigens, and then activated the detectors. The higher the activation matching threshold we set, the higher the error rate. If we keep the memory detectors of the last run and insert them into the next run, we can achieve almost 100% detection rate. The insertion of memory detectors can be viewed as immunization injection. When $R = 33$ or $R = 34$, we achieved the best results, 95.19% and 95.21%, respectively, when the activation threshold was set at 5. In these 2 cases, *R* is large enough to avoid mistakenly matching the self strings and also is not yet specific enough (i.e., even larger) to exclude similar attacking strings. For all the experiments we conducted, the useful detectors were actually all obtained from antigen feedback, and none of them were generated randomly.

For normal strings, when R is small ($R \le 31$), we achieved 100% detection rate persistently. This is understandable. The randomly generated detectors rarely match the incoming antigens, while any feedback antibody always matches the self strings and thus is eliminated. Therefore, no match against the incoming antigens can be made. When R is large enough ($R \ge 32$), the match against self strings becomes more specific, and some feedback antibodies cannot match the self strings. As the result, some detectors were built up. They made some matches against the incoming antigens, and the detection rates dropped to around 99%. The best result was achieved when $R = 35$, the detection rate was 99.21%, with error rate 0.79%.

Combining both attack strings and normal strings together, the best results were under the setting of either $R = 33$ or $R = 34$, when the activation matching threshold was set at 5, Fig 2.

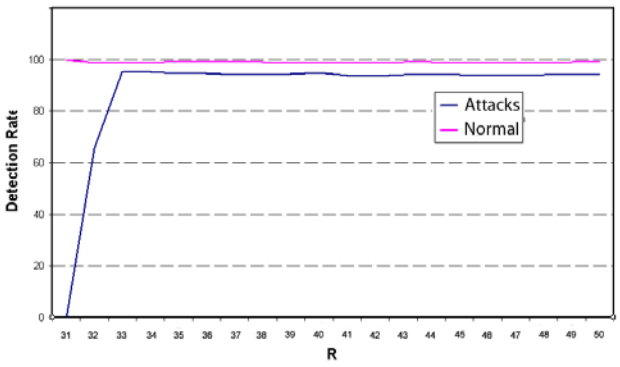

Fig. 2. The detection rates for R from 31 to 50 with the activation matching threshold 5.

These results are surprisingly good for the antigen feedback mechanism. They prove its effectiveness. With the feedback, the system can quickly establish useful detectors and then use them to discover similar patterns from further incoming antigens.

The matching method, r-continuous bits match, was inspired by real immune systems [7, 11]. We believe that it can accurately measure the similarity of two strings, because it measures the degree of *resembleness* instead of distance, such as Euclidean distance or Hamming distance. For example, the Euclidean distance distances for the two pairs (i) "10101" and "00100" and (ii) "10101" and "11111" are the same ($\sqrt{2}$), but the two strings in the first pair resembles each better than the second pair, as the 3 continuous bits in the middle are the same (010).

Given the good detection rates we achieved under the antigen feedback mechanism, we still have some concerns on using segmented string format to represent antibodies and antigens for r-continuous bits match method. Suppose that the length of antibodies and antigens is l, and a string has *n* segments from s_1 to s_n. If $R = l$, we have full matching, where a detector can only detect the antigens which are exactly the same as itself. If $\frac{l}{2} < R < l$, r-continuous bits match just ignores the leading or the ending, or both, segments, e.g., s_1, s_2, s_{n-1}, and s_n etc., depending on the value of R and the lengths of the leading and the ending segments. Therefore, the method makes the segments in the middle weight more than these at the both ends. If a match happens, these in the middle have to have exact match. In other words, a match involves all the segments in the middle and some from either end of the string. Therefore, a match is achieved by completely and also consistently ignored

some segments at either ends. This is the reason responsible for our string format as described in Section 3.

This effect is not desirable. However, the problem is not the matching method, but the way we present the data. We are seeking a better way to represent antibodies and antigens by homogenizing the information carried by each segment.

6 Conclusion and Future Work

This paper presents our experiment on negative selection in intrusion detection by using KDD CUP 1999 dataset. In order to solve the problem of efficiently generating effective detectors, we propose an antigen feedback mechanism. Under the mechanism, we can achieve 95.21% detection rate for attack strings, with false negative rate 4.79%, and 99.21% detection rate for normal strings, with false positive rate 0.79%.

The surprisingly good results by such a simple mechanism were achieved on the KDD CUP 1999 dataset. More experiment on other datasets, and perhaps the datasets from different domains, is needed to verify if the results are just coincident or persistent. We are arranging more experiment. Given the good results achieved by the antigen feedback mechanism and the fact that randomly generated antibodies hardly match the incoming antigens, we have a bold conjecture that negative selection may not need randomly generated antibodies at all but just rely on the feedback antigens. If the conjecture can be proven to be true in general cases, we avoid the scaling problem of negative selection.

This paper also introduces Arisytis, the test bed we used for our experiment. It is available to the other researchers in the field to avoid the waste of the effort to write similar program. Arisytis can also be used as an educational tool.

Finally, we believe that negative selection with the antigen feedback mechanism has the ability to quickly discover unknown patterns. The ability can be applied to other domains for pattern detection than just self/non-self discrimination, for example, spam email recognition, faulty parts detection, and finance fraud discovery etc. Another future task of ours is to find out a way to homogenously blend the information carried by the characters of the antibody and antigen strings and then study the differences of r-continuous bits match on the new string format. Last but not least, we will further expand Arisytis with other up to date AIS algorithms.

References

1. Forrest, S., S.A. Hofmeyr, et al. A sense of self for Unix processes. in IEEE Symposium on Security and Privacy. 1996. Oakland, CA, USA.
2. Timmis, J., Artificial immune systems - today and tomorrow. Natural Computing: an international journal, 2007. 6(1): p. 1-18.
3. Dasgupta, D., Advances in artificial immune systems. IEEE Computational Intelligence Magazine, 2006. 1(4): p. 40 - 49.
4. Garrett, S.M., How Do We Evaluate Artificial Immune Systems? Evolutionary Computation, 2005. 13(2): p. 145 - 177.

5. Dasgupta, D., Z. Ji, and F. Gonzalez. Artificial immune system (AIS) research in the last five years. in The 2003 Congress on Evolutionary Computation (CEC-03). 2003: IEEE Press.
6. Hofmeyr, S.A. and S. Forrest. Immunity by Design: An Artificial Immune System. in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999). 1999. Orlando, Florida, USA: Morgan Kaufmann.
7. Hofmeyr, S.A. and S. Forrest, Architecture for an Artificial Immune System. Evolutionary Computation, 2000. 8(4): p. 443-473.
8. Hart, E. and J. Timmis. Application Areas of AIS: The Past, The Present and The Future. in Proceedings of Artificial Immune Systems: 4th International Conference, ICARIS 2005. 2005. Banff, Alberta, Canada: Springer.
9. Forrest, S., A.S. Perelson, et al. Self-Nonself Discrimination in a Computer. in Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy. 1994. Oakland, CA, USA: IEEE Computer Society Press.
10. Hofmeyr, S.A., S. Forrest, and A. Somayaji, Intrusion Detection Using Sequences of System Calls. Journal of Computer Security, 1998. 6: p. 151-180.
11. Hofmeyr, S., An Immunology Model of Distributed Detection and Its Application to Computer Security, in Department of Computer Science. 1999, University of New Mexico, USA.
12. Castro, L.N.D. and J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach. 2002: Springer.
13. Balthrop, J., S. Forrest, and M.R. Glickman. Revisiting LISYS: Parameters and normal behavior. in Proceedings of the Congress on Evolutionary Computing (CEC-2002). 2002.
14. Gabrielli, N. and M. Rigodanzo. An Artificial Immune System for Network Intrusion. Detection on a Web Server: First Results. in Proceedings of the 2nd Italian Workshop on Evolutionary Computation (GSICE 2006). 2006.
15. Gonzalez, F.A. and D. Dasgupta, Anomaly Detection Using Real-Valued Negative Selection. Genetic Programming and Evolvable Machines, 2003. 4(4): p. 383-403.
16. Ji, Z. and D. Dasgupta, Revisiting Negative Selection Algorithms. Evolutionary Computation, 2007. 15(2): p. 223-251.
17. Kim, J. and P. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. in Proceedings of GECCO-2001. 2001.
18. ACM. KDD CUP 1999 data. [cited 12 January 2007]; Available from: http://kdd.ics.uci.edu//databases/kddcup99/kddcup99.html.
19. DARPA. DARPA Intrusion Detection Evaluation Data Sets. 1999 [cited 2006 15 October 2006]; Available from: http://www.ll.mit.edu/IST/ideval/data/data_index.html.
20. Stolfo, S.J., W. Fan, et al. Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection: Results from the JAM Project. in Proceedings of 2000 DARPA Information Survivability Conference and Exposition. 2000.
21. Ma, W., D. Tran, and D. Sharma. A Study on the Feature Selection of Network Traffic for Intrusion Detection Purpose. in the Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI 2008). 2008. (To be published).