Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 740–749
computer communications
www.elsevier.com/locate/comcom

Anomaly detection in TCP/IP networks using immune systems paradigm

Franciszek Seredynski [a,b,*], Pascal Bouvry [c]

[a] Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland
[b] Institute of Computer Science, Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland
[c] Faculty of Sciences, Technology and Communication, Luxembourg University, 6 rue Coudenhove Kalergi, L-1359 Luxembourg-Kirchberg, Luxembourg

Available online 12 September 2006

Abstract

The paper presents an architecture of an anomaly detection system based on the paradigm of artificial immune systems (AISs). Incoming network traffic data are considered by the system as signatures of potential attackers by mapping them into antigens of AISs either using some parameters of network traffic or headers of selected TCP/IP protocols. A number of methods of generation of antibodies (anomaly detectors) were implemented. The way of anomaly detection depends on the method of antibodies generation. The paper presents results of an experimental study performed with use of real data and shows how the performance of the anomaly detection system depends on traffic data coding and methods of generation of detectors.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Artificial immune systems; Anomaly detection; Detectors generation; Traffic data coding; TCP/IP protocols

1. Introduction

Security of information and computational resources is currently one of main research activities in the area of computer networks. Designing intrusion detection systems (IDSs) is one of answers to security issues. Anomaly detection and misuse detection are two major approaches in attempt to solve the intrusion detection problem. Anomaly detection, which is the subject of this study, relies on building models from network data and discovering variations from the model in the observed data.

Currently used commercial IDS systems are usually signature-based (see, e.g. [1]). Signatures in such systems require to be periodically updated and modified to cope with attacks, but this approach does not prevent new kinds of attacks. For this reason other approaches are currently investigated to use them in IDSs. The most common approach is based on building a statistical model of a system behavior (e.g. [2]). Data mining and machine learning methods are also currently applied (see, e.g. [3–5]), as well methods using visualization (see, e.g. [6]). Increasing interest in application of naturally inspired algorithms, such as genetic algorithms (GA) and particle swarm search (see, e.g. [7]) or combining evolutionary algorithms with fuzzy set theory (see, e.g. [8]) can be observed.

Immune systems are naturally existing mechanisms which are responsible for detecting and coping with intruders in living organisms. Artificial immune systems (AISs) are computational models inspired by immune systems and therefore are currently one of the most promising nature-inspired technique used for IDSs [9–14,5] and we follow this approach. The purpose of this project was to study and compare effectiveness of methods of network traffic coding requested by immune systems paradigm and methods of generation of detectors (antibodies), and their work with different matching functions.

The remainder of the paper is organized as follows. The next section presents a background on AISs. Section 3 contains a description of an architecture of AIS-based anomaly detection system including issues of traffic data coding and

* Corresponding author. Tel.: +48 22 648 6045; fax: +48 22 584 4501.
E-mail addresses: sered@ipipan.waw.pl (F. Seredynski), Pascal.Bouvry@uni.lu (P. Bouvry).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.08.016

the choice of matching functions. Section 4 describes methods used for generation of antibodies. Section 5 contains results of experimental study of the system and last section concludes the paper.

2. Artificial immune systems

Artificial immune system (AIS) is a computational technique inspired by ideas coming from immunology and used to develop adaptive systems capable to solve different domain problems. AIS has recently become one of the most popular research tools studied and applied to solve problems in the field of computer security, in particular to detect computer viruses [15] or intruders in computer networks [12]. It also has been applied to solve scheduling problem [16], build decision support systems [17] or solve function optimization and combinatorial optimization problems [18].

While a general computational model inspired by immunology is currently a subject of a research, two basic notions – *antigen* and *antibody* are important and widely applied. Antigens are foreign invaders which attack in some way a considered system. Antibodies are a part of the system which are responsible for detection and elimination of antigens. Antibodies detect antigens by matching them. A number of antibodies is much smaller than the number of antigens, so matching is never perfect but only partial. One of purposes of AIS-based system is to develop and keep relatively small number of antibodies, which are able to detect in reliable way a big number of antigens, including antigens which have never been seen before.

3. AIS-based anomaly detection system

3.1. A general overview of the system

The general scheme of AIS-based anomaly detection system working in a single node of a network is presented in Fig. 1. It is assumed that external users can have an access to resources and services which are opened to them, using network communication system by sending and receiving messages, which constitute *network traffic*. Incoming network traffic is only the source, which contains both an authorized access and attempts to unauthorized access to resources and services.

Incoming network traffic should be parameterized in such a way to be able to define patterns of both malicious behavior corresponding to invaders (antigens) and authorized behavior. A part of network traffic characterized the authorized users behavior (*training data set*) should be available. It is assumed that the system is able to work in two modes: *learning mode* and *normal operating mode*. In the learning mode data from training data set are used to generate a set of antibodies, which model authorized behavior of users. In the normal operating mode *testing data set* – a part of regular incoming network traffic are used to tune the system, detect anomaly and inform the *Control and decision system* about suspicious behavior. In the case of wrong alarm a set of antibodies should be modified.

It is assumed that the node of the system is a part of a TCP/IP network and messages which belong to network traffic consist with a number of packets created and sent according to the TCP/IP protocols. The most popular protocols are TCP, UDP and ICMP (see, Fig. 2a). A packet containing application data *App. data* and sent from a given application to other application is encapsulated, i.e. headers *L4*, *L3*, *L2* corresponding to applied protocol are added to it (see, Fig. 2b).

3.2. Coding antibodies and antigens

The number of packets exchanged between applications located in network nodes are too big to analyze all of them from security point of view. What to analyze in incoming network traffic to be able to detect intruders is an open research issue. In the system, which is studied in this paper, two recently emerged ideas are applied.

The first one [13] assumes to use a set of general traffic parameters such as *a number of bytes per second* (*Bps*), *a number of packets per second Pps*, and *a number of ICMP packets per second*. These values are calculated and

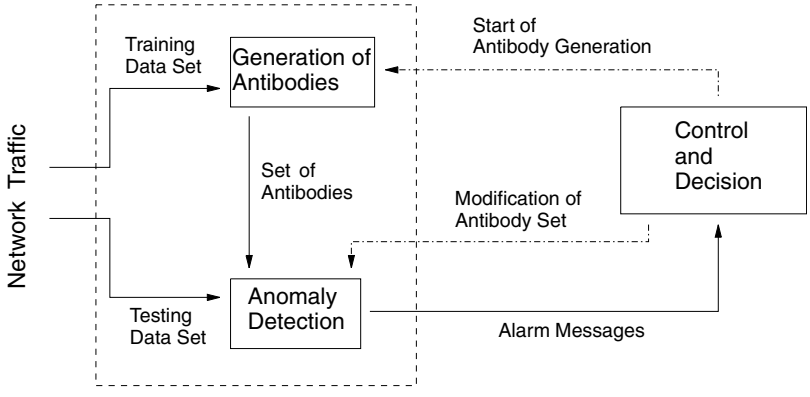

Fig. 1. General architecture of anomaly detection system.

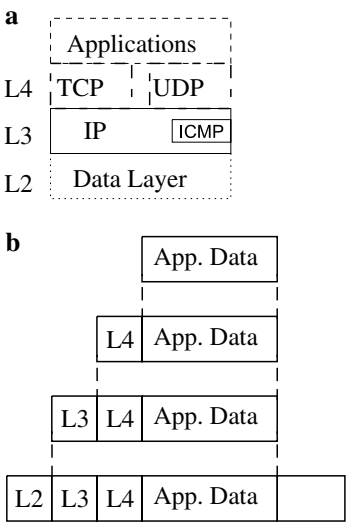

Fig. 2. TCP/IP network: TCP/IP communication protocols (a), and a structure of a network packet (b).

averaged using small time intervals, using the idea of a moving window. Because in the network traffic data used in experiments (see, Section 5) ICMP packets were not present, therefore, only the two first parameters *Bps* and *Pps* were used. It means that both antibodies and antigens will be represented in the form of real number vectors $x = \{x_1, x_2\}$ in the 2-dimensional space.

The second idea assumes using [14] full headers of TCP/IP protocols. In the implementation of the system this idea was reduced to the analysis of only TCP SYN packets (see, Fig. 3), i.e. TCP headers with flag *S* set. Such packets are sent to establish TCP communication, and can be considered as good representatives of TCP packets. Under this approach both antibodies and antigens will be represented by binary arrays corresponding to headers of TCP SYN packets.

3.3. Similarity measures between antibodies and antigens

A decision about the degree of matching antibodies and antigens can be taken in particularly on the base of evaluation of a distance between corresponding antibodies and antigens. Euclidean or Hamming distances will be used as measures of similarities between antibodies and antigens for vector or binary coding of antibodies and antigens, respectively. These measures can be defined in the following way:

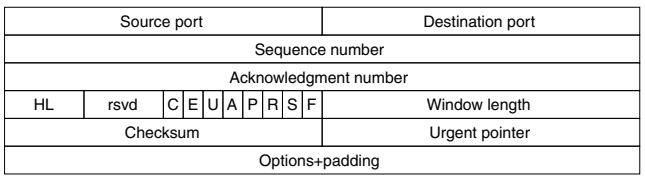

Fig. 3. Header of TCP protocol. Only packets TCP SYN (with the flag S set) are analyzed.

- Euclidean distance $d(x,y)$. For two vectors $x = \{x_1, \ldots, x_n\}$ and $y = \{y_1, \ldots, y_n\}$ from n-dimensional space, the Euclidean distance between them can be calculated as

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}, \quad d(x,y) \in \langle 0, \infty)$$

- normalized Hamming distance $f_H(X,Y)$. For two binary arrays X and Y $(X, Y \in \{0,1\}^N)$ the normalized Hamming distance can be evaluated as $f_H(X,Y) = \frac{1}{N}\sum_{i=1}^{N} X_i \oplus Y_i, f_H \in \langle 0, 1\rangle$, where N – a number of bits in each array, and $\oplus$ is *XOR* operation.

If a distance calculated for a given pair of antigen and antibody exceeds some threshold, the incoming network traffic corresponding to the antigen is considered as a traffic generated by an intruder.

4. Generating antibodies

An important issue for any AIS-based anomaly detection system is a generation of a set of antibodies, which will represent a legal network traffic. Such a set should be enough small, and each antibody should cover relatively large part of a space corresponding to the legal traffic. The space covered by the set of antibodies can not be a source of any false alarm. Three methods of generation of antibodies have been implemented in the system: *positive characterization* technique, *random generation of antibodies*, and *negative characterization* technique.

4.1. Positive characterization technique

The *positive characterization* (*PC*) technique [13] assumes that information about the legal traffic is directly used to create detectors (antibodies). A number of antibodies is equal to the number of information units describing the legal traffic. The method is simple and effective in detecting anomalies, but increasing the number of detectors may cause the increase of the computational costs of detecting. Clustering algorithms can be applied to decrease a number of detectors.

Algorithm 1. Random generation of antibodies

```
ab = generateAntibody()
autoaggressive = true
for each selfExample → se from selfExampleSet
   dist = distance(ab, se)
   if dist < threshold
      antibodySet.addAntibody(ab)
      autoaggressive = false
      break
   if autoaggressive
      ab.destroy()
```

4.2. Random generation of antibodies

In this method [14] information about the legal traffic is used only as a test set to build a population of detectors, which are created in a random way (see, Algorithm 1). A candidate **ab** for antibody is created randomly and next compared with each representative **se** of the legal traffic from `selfExampleSet`. If the candidate covers at least one example of the legal traffic, i.e. the distance **dist** between it and **se** is lower than some `threshold` than it is included into the created set `antibodySet` of antibodies. The final set `antibodySet` is usually smaller than the set obtained using *PC*.

4.3. Negative characterization technique

Negative characterization (*NC*) technique [13] proposes to consider antibodies as rules of the following form:

$$R^k: \textbf{if } x_1 \in \langle min_1^k, max_1^k \rangle \textbf{ and } \ldots \textbf{ and } x_n \in \langle min_n^k, max_n^k \rangle \textbf{ then } \textbf{\textit{anomaly}},$$

where R^k is kth rule which can be interpreted as n-dimensional hypercube with min_i^k and max_i^k as boundaries of each ith dimension, and $\{x_1, \ldots, x_n\}$ are parameters of a suspected traffic activity (antigen).

In opposite to *PC* and *random generation* methods rules-antibodies are generated in such a way to cover a part of n-dimensional space not belonging to the legal traffic. Examples of the legal traffic are n-tuples with a hyperradius v (considered as a threshold), and can be seen as hyperspheres in the n-dimensional space. Because not all points of the legal traffic in this n-dimensional space are known, different hyperradiuses v of self-examples (the legal traffic) are used and rules-hypercubes are generated with different boundaries. They can intersect and create multilevel subspaces corresponding to different levels of anomalies (see, Section 5).

A standard GA [19] with tournament selection, one-point crossover and a single bit mutation is used to generate rules. A single run of GA results in producing a single rule $R = GA(S, v, C, ruleSet)$, where v-assumed threshold of hyperspheres corresponding to self-examples from set S, *ruleSet*-a set of currently produced rules, and C is a parameter of a fitness function.

The fitness function `Fit(R)` has three components:

(1) a volume of the space covered by a rule `R:volume(R)` $= \prod_{i=1}^{n}(max_i^k - min_i^k)$, where min_i^k and max_i^k are searched boundary values of the ith dimension of the hypercube corresponding to the rule
(2) a penalty – a number of self-examples covered by the rule:`falsePositives(R)` $= \{x \in S | x \in R\}$, where x is a self-example from the set `S` covered by the rule `R`
(3) a penalty for sharing a volume of the common space by a new rule `R` and a rule R^j from the set `ruleSet` of rules already generated: $overlap(R) = \sum_{R^j \in \texttt{ruleSet}} volume(R \cap R^j)$.

Finally, the fitness function of a searched rule has the following form: $Fit(R) = volume(R) - C{\cdot}falsePositives(R) - overlap(R)$, where C is a user-defined coefficient.

5. Experimental results

5.1. Data used in experiments

Five experiments have been conducted [21] using the AIS-based anomaly detection system and their results are reported below. The main purpose of the experiments was to study the performance of different methods of generating the detectors for corresponding coding the network traffic. The first four experiments were conducted using intrusion detection data from MIT Lincoln Laboratory [22] and collected in the network configuration replicated in experiments as it shown in Fig. 4. In the experiments 240 min of recorded network traffic were used. The first 120 min of the traffic were used as training data, and the next 120 min were used as testing data. Fig. 5 shows these data averaged over 60 second time windows, normalized to the range $(0, 1)$, and expressed in packets per second (Pps), and in bytes per second (Bps), respectively, with the size of the window equal to 1. Because these data contain only legal traffic, data containing attacks of the type *port scanning* were created using the program `nmap` and added later to the testing data.

The fifth experiment was conducted using data collected from the computer network of Institute of Computer Science PAS, Warsaw (IPI PAN) and obtained using the program `tcpdump`. The experiments were conducted in offline mode of the system using a Pentium III 933 MHz, 256MB RAM, Linux, JDK 1.4.2.

5.2. Experiment 1

In the first experiment traffic data interpreted as the 2-dimensional vectors $\{Pps, Bps\}$ were used. Also the Euclidean measure of distance and the *PC* method for generating antibodies were applied. During the learning process and using the training data (the first 120 min) 32 different detectors (see Table 1) have been created.

Fig. 6a shows Euclidean distance between vectors of the network traffic (antigens) arriving to the system and the nearest vector-detector from the set of detectors. One can see that during the first 120 min the distance is equal to 0 – all detectors correspond to training data set. In the remaining 120 min distances between vectors of still legal traffic and detectors are not greater than 0.25. This value can be considered as the value of threshold which guarantees the zero level of false alarms for the testing set of traffic data (see, Fig. 6b). Fig. 7a shows how found set of detectors and threshold equal to 0.25 create circles covering the space of the whole (training and testing sets) legal traffic.

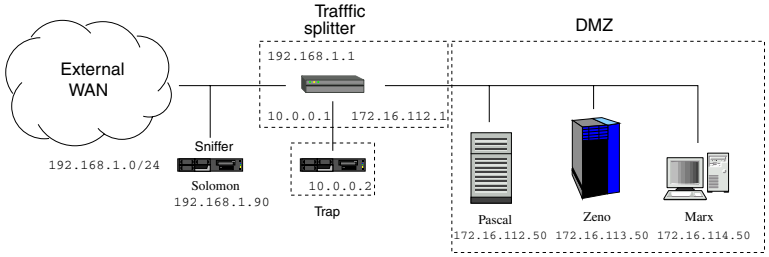

Fig. 4. Network configuration used in experiments 1–4.

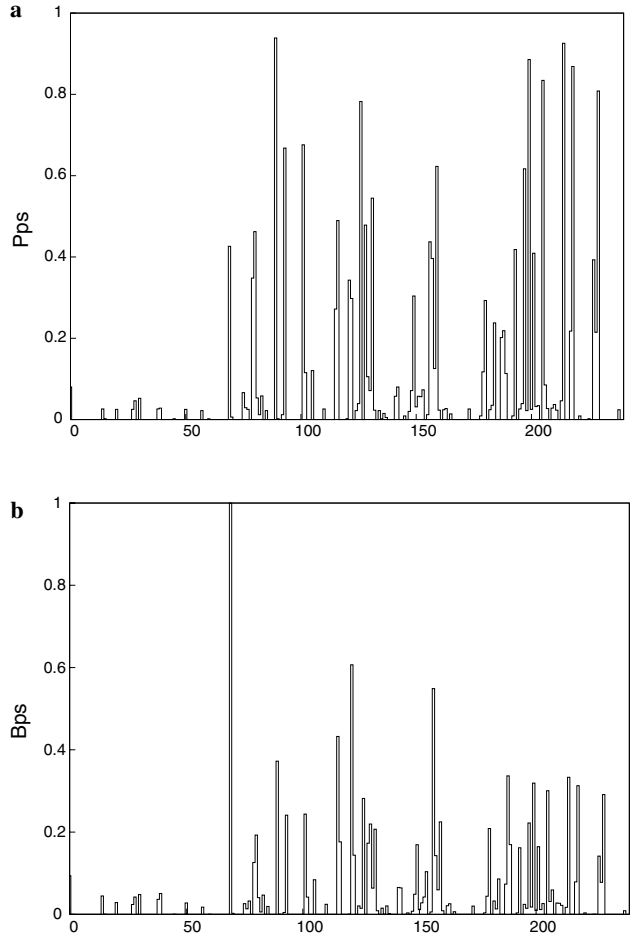

Fig. 5. Normalized traffic in router *Traffic splitter* of the network configuration during 240 min: in packets/s. (Pps) (a), and in bytes/s. (Bps) (b).

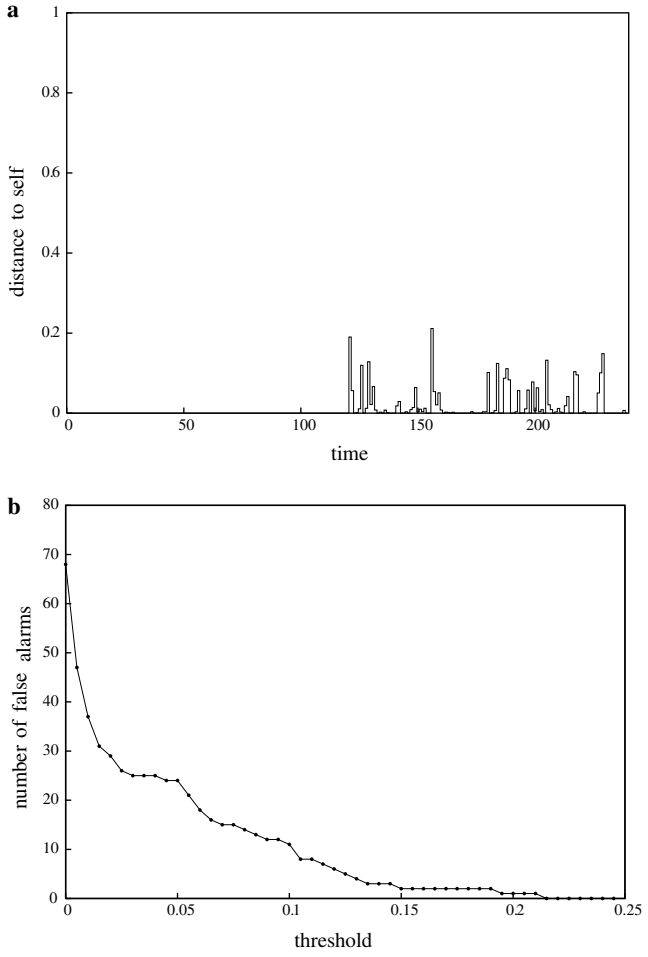

Fig. 6. Matching antibodies to the legal traffic (a), and the level of false alarms as a function of threshold (b).

Table 1
Experiment 1: set of detectors obtained from training set using *PC*

{0.0000,0.0000}	{0.0021,0.0007}	{0.0064,0.0023}	{0.0128,0.0043}
{0.0128,0.0055}	{0.0235,0.0174}	{0.0235,0.0192}	{0.0267,0.0237}
{0.0267,0.0276}	{0.0267,0.0284}	{0.0267,0.0324}	{0.0277,0.0243}
{0.0277,0.0364}	{0.0277,0.0446}	{0.0299,0.0504}	{0.0309,0.0132}
{0.0491,0.0419}	{0.0555,0.0479}	{0.0566,0.0404}	{0.0619,0.0468}
{0.0705,0.0259}	{0.0854,0.0939}	{0.1228,0.0418}	{0.1282,0.0841}
{0.2895,0.4323}	{0.3707,0.1259}	{0.4540,1.0000}	{0.4925,0.1926}
{0.5213,0.1761}	{0.7115,0.2411}	{0.7200,0.2435}	{1.0000,0.3724}

Fig. 7b shows the behavior of the system in the case when two attacks of the type of *port scanning* were added to the legal traffic. Two attacks were performed using the program `nmap`. The first one conducted in the moment of time equal to 60 when a scanning method `connect` was applied, and the next one in time equal to 130 when a scanning method `SYNstealth` was used. Corresponding Euclidean distances are equal to 4.41 and 3.42, respectively. Both distances are greater than threshold what results in an alarm.

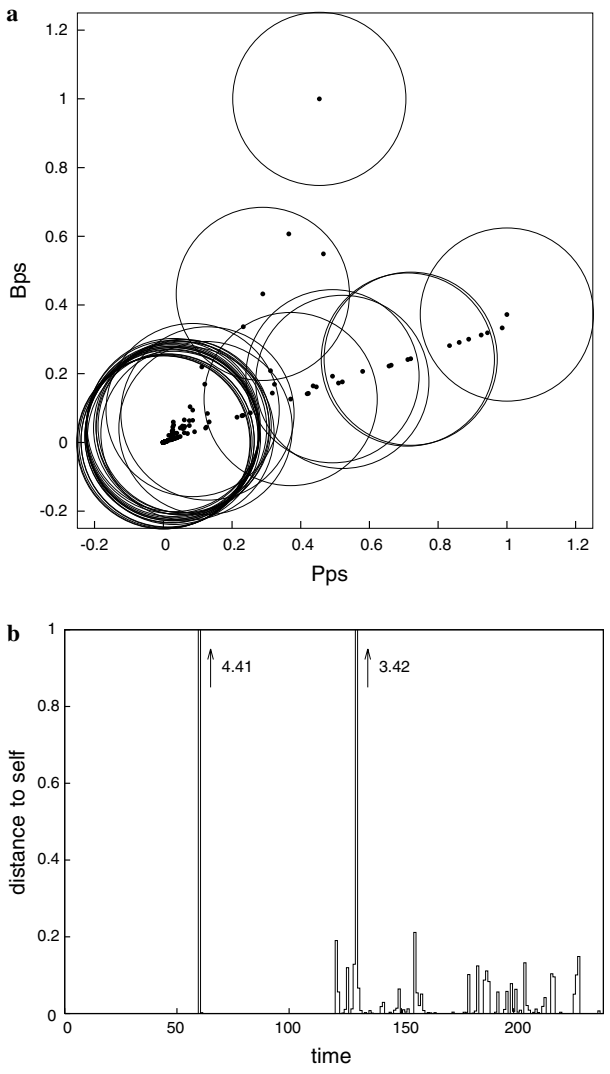

Fig. 7. Antibodies with corresponding threshold cover the legal traffic (a), and detection of anomalies (b).

It is worth to notice that computational time which is necessary to generate detectors using *PC* depends linearly on the size of training set.

5.3. Experiment 2

In the experiment the random generation method of creating antibodies is used, and remaining assumptions are the same as in the previous experiment. The purpose of the experiment was to see whether a reduction of a number of detectors is possible without the loss of anomaly detection quality.

If randomly generated antibodies are in the close proximity to points of the training set they replace them. Using this method it was possible to reduce the number of antibodies from 32 to 16, i.e. on 50%. The set of new detectors is shown in Table 2.

The detectors were found by setting a small threshold equal to 0.05, which was used as radius of circles around

Table 2
Experiment 2: set of detectors obtained using *random generation*

{0.0031,0.0382}	{0.0036,0.0442}	{0.0801,0.0266}	{0.0905,0.0678}
{0.1585,0.1081}	{0.2528,0.4147}	{0.3019,0.4687}	{0.3978,0.0873}
{0.4141,0.9784}	{0.4933,0.1581}	{0.4960,0.1605}	{0.4995,0.1823}
{0.5384,0.1407}	{0.5500,0.1940}	{0.7192,0.2602}	{0.9587,0.3621}

points of the training set (see, Fig. 8a). The circles cover all 32 detectors found in the previous experiment. Randomly generated detectors are accepted if they fall into circles, and next they replace all previously found detectors in corresponding circles. Using data from the testing set one can find a final threshold, which reduces the number of

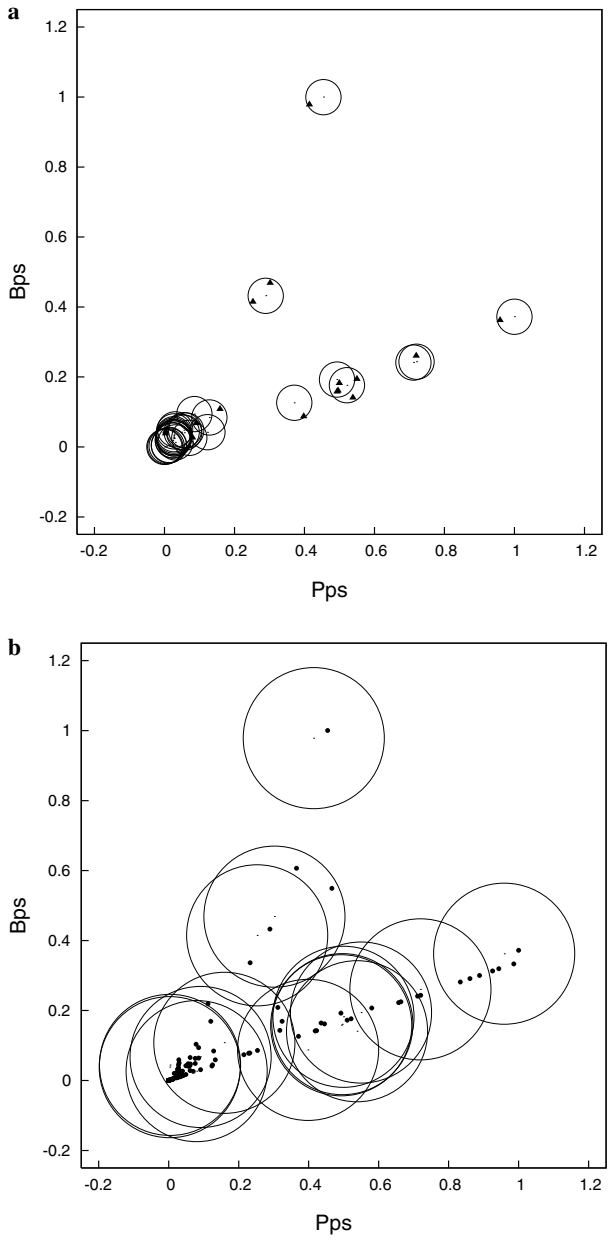

Fig. 8. Generation of detectors using *random generation* method (a), and antibodies with corresponding threshold covering the legal traffic (b).

false alarms to 0. The final value of the threshold for found set of detectors is equal to 0.2, what means that now circles better cover the space of self-examples.

The behavior of the system when attacks (the same as in the previous experiment) are conducted is similar to that observed in the previous experiment. Computational time which is necessary for detection is proportional to the size of detectors, but in opposite to the *PC* method this time does not depend on the size of a training set. The considered method of generation of detectors may give a reasonable reduction of detection cost if a big number of points in the training set is close to each other.

5.4. Experiment 3

In this experiment *NC* method of generation of antibodies is used. Remaining conditions are the same as in the previous experiments. It was assumed that the two level system of rules will be searched: one system of rules for threshold (a radius of spheres of the training set points) $v = 0.10$, and the second one for $v = 0.25$. Searching a single rule was conducted using a GA with the following parameters: a population size of rules was $N = 100$, a number of generation $L = 200$, the probabilities of crossover and mutation were, respectively, $p_k = 0.75$, $p_m = 0.003$. Maximal number of rules to be searched was equal to 16.

Fig. 9 shows changing the fitness function values and its components (with $C = 2$) corresponding to the best individual (detector) in each generation during a run of GA searching a single rule. One can notice that the best detector (with the best shape) is found after about 100 generations. Two sets of rules corresponding to two levels of the threshold v were found. The level 1 set contains 12 rules and the level 2 set contains 4 rules. They are shown graphically (rectangles) in Fig. 10 and are listed in Tables 3 and 4, respectively.

In this two level detection system there is not one single threshold, and a kind of a decision about an alarm depends on the degree of matching of an antigen to detectors. For

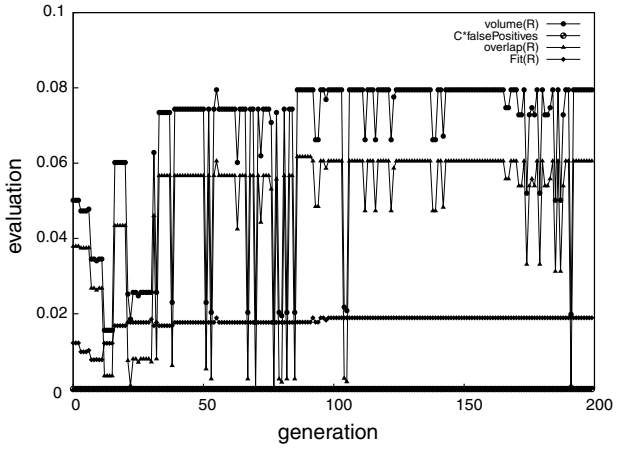

Fig. 9. Evolving a single rule using GA.

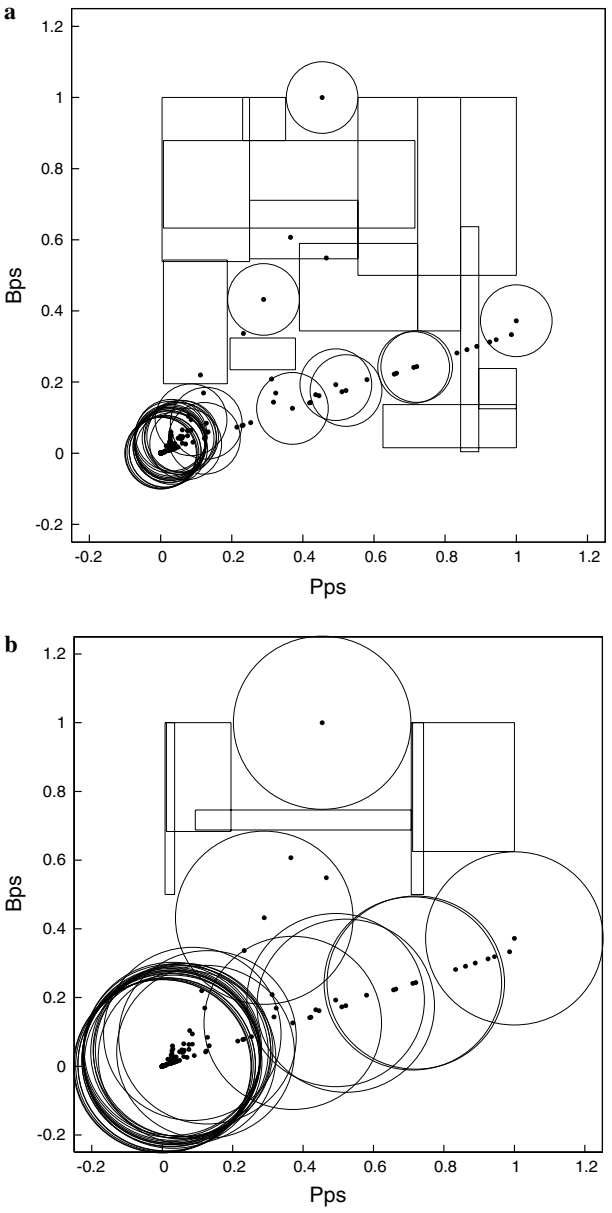

Fig. 10. Set of detectors for: $v = 0.10$ (a), and $v = 0.25$ (b).

Table 3
Experiment 3: set of detectors (level 1) obtained using *NC*

{0.5546 ÷ 1.0000,0.5000 ÷ 1.0000}	{0.0078 ÷ 0.7148,0.6328 ÷ 0.8789}
{0.0039 ÷ 0.2500,0.5390 ÷ 1.0000}	{0.6250 ÷ 1.0000,0.0156 ÷ 0.1367}
{0.0078 ÷ 0.1875,0.1953 ÷ 0.5429}	{0.8437 ÷ 0.8945,0.0039 ÷ 0.6367}
{0.7226 ÷ 0.8437,0.3437 ÷ 1.0000}	{0.2304 ÷ 0.3515,0.8789 ÷ 1.0000}
{0.3906 ÷ 0.7226,0.3437 ÷ 0.5898}	{0.8945 ÷ 1.0000,0.1250 ÷ 0.2382}
{0.2500 ÷ 0.5546,0.5468 ÷ 0.7109}	{0.1953 ÷ 0.3789,0.2343 ÷ 0.3242}

Table 4
Experiment 3: set of detectors (level 2) obtained using *NC*

{0.7109 ÷ 1.0000,0.6250 ÷ 1.0000}	{0.0117 ÷ 0.1953,0.6835 ÷ 1.0000}
{0.7070 ÷ 0.7421,0.5000 ÷ 1.0000}	{0.0937 ÷ 0.7070,0.6875 ÷ 0.7460}
{0.0078 ÷ 0.0351,0.5000 ÷ 1.0000}	

the testing set five false alarms of the level 1 will appear, and no alarms of the level 2. Scanning attacks discussed earlier will have caused alarms of the level 2.

Computational cost of generating detectors is much higher than it was for the previous methods, mainly because of multiple run of GA. Time of detection is proportional to the number of created detectors.

5.5. Experiment 4

In this experiment binary coding of TCP SYN packets from traffic data, and the Hamming distance f_H as a measure of distance is used. *PC* method of generation of antibodies is applied. Detectors are created using 73 TCP SYN packets, which are in the training set. An example of a detector in the form of a sequence of bytes is shown in Table 5.

Fig. 11a shows that a value of the threshold providing zero level of false alarms in the testing data is equal to 0.165. Fig. 11b shows how the system copes with the scanning type of attacks described earlier. This type of attacks is characterized by a huge number of TCP SYN packets which are sent. The figure shows only the end of the `connect` attack, corresponding to antigens with labels 14–2350, and the beginning of the attack `SYNstealth` – antigens with labels 2422–4341. Antigens 2351–2409 are a part of the training set, and antigens 2410–2421 are only in the testing set.

One can notice a small difference between distances corresponding to the legal traffic from the testing set and the anomaly traffic. Small differences between the legal traffic and the attack are the result of features of both IP and TCP headers. Very frequently some parts (e.g. an address of a destination) of both headers are the same, what decreases the differences between a legal traffic and attack.

5.6. Experiment 5

This experiment has been conducted using network traffic data collected during 9 days in a new installed server in IPI PAN. The data contained about 4.4 millions of packets, including about 2.5 millions of IP packets. The training set, a part of the legal traffic was selected on the base of an interview with the network administrator.

The purpose of the study was to define periods of averaging and methods of data selection, which could provide the best inside to the most frequently observed attacks: attempts to connect with ports 135, 139, 145 and 901. The problem of data selection was solved by choosing for the analysis only the TCP SYN packets. These packets were next processed using the *PC* method and the Euclidean distance.

Table 5
Experiment 4: an example of a detector as a sequence of bytes

45	10	00	2c	01	6e	00	00	40	06	a0	6a	c2	07	f8	99
ac	10	72	32	04	00	00	17	f0	88	b3	9e	00	00	00	00
60	02	02	00	15	04	00	00	02	04	05	b4				

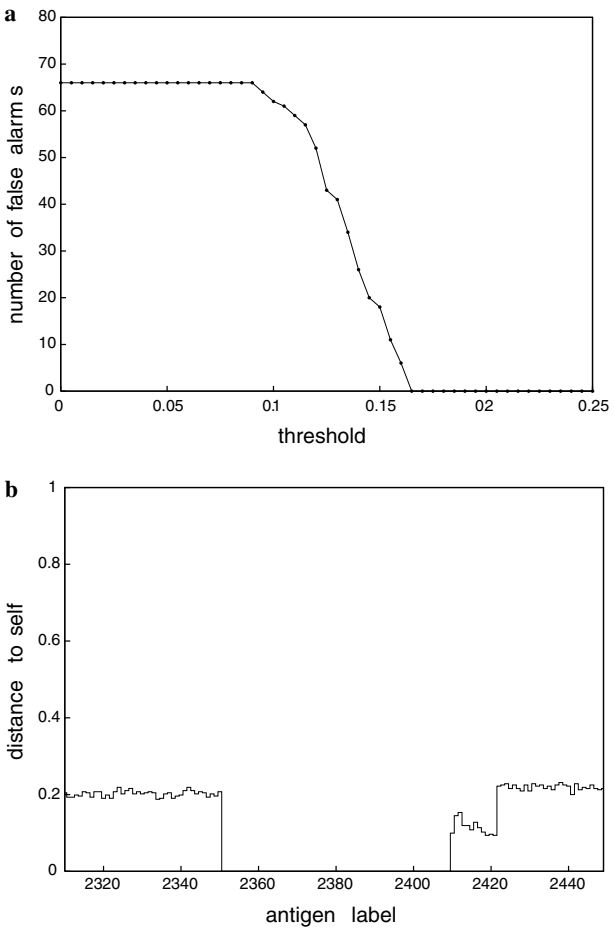

Fig. 11. The level of false alarms (a), and discovery of anomalies (b).

The most clear results were obtained using 10 min period analysis. The distances between parameters of the traffic considered as legal (11 detectors) and the whole traffic (1198 antigens) are shown in Fig. 12. The frequency of

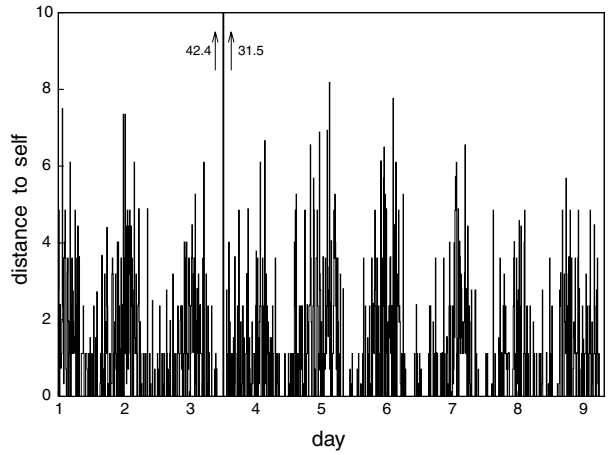

Fig. 12. Discovery of anomalies in IPIPAN.

Table 6
Time of antibodies generation and anomalies detection

	Exp. #1	Exp. #2	Exp. #3	Exp. #4	Exp. #5
Generation time	113 ms	121 ms	8750 ms	159 ms	32 ms
Antibodies number	32	16	12 + 5	73	11
Detection time	373	360	360	2360	196
Antigens number	240	240	240	4393	1198
Detection time/ antigen	1.554	1.500	1.500	0.537	0.164

attacks is high. There are visible periods of activities in middays, and two attacks conducted from a private computer (the distance equal to 31.5 and 42.4 in two subsequent 10 min periods of time). Analysis of all attacks suggests to set the value of the alarm threshold equal to 4.

6. Conclusions

In this paper, the architecture of the anomaly detection system based on the paradigm of artificial immune systems has been presented. A number of methods of traffic data coding and generation of detectors were studied from the point of view of their efficiency. The results of experimental study have shown that while the system is able to detect anomalies in all considered cases the cost of running the system highly depends on a method of generation of antibodies. The results summarizing the time of generation of antibodies and detection are presented in Table 6.

One can see that the faster method of antibodies generation is the ***PC*** method, which directly converts a training set into the set of detectors. The method of random generation requires more computational costs. The most expensive is the ***NC*** method.

From point of view of detection time the best method of generation of detectors is the random generation method, which provides the smallest number of detectors. The time of detection using *NC* method is similar, but detectors require much more storage memory. The *PC* method seems to be the weakest. It requires the same memory as *NC* method, but the time of detection is much larger.

Results of this study clearly point out that the choice of a method of detectors generation for a given system is a matter of a compromise between a requested speed of the system and available memory. Current research is directed to find more effective methods of generating detectors. Promising alternative for searching e.g. rule-detectors is applying coevolutionary algorithms [20] instead of GA.

References

[1] M. Roesch, Snort: lightweight intrusion detection for networks, in: Proceedings of the 13th Conference on Systems Administration (LISA-99), Seatlle, WA, November 7–12, USENIX, 1999, pp. 229–238.
[2] E. Eskin, Anomaly detection over noisy data using learned probability distributions, in: Proceedings of the 17th International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, 2000, pp. 255–262.
[3] W. Lee, S.J. Stolfo, K.W. Mok, Mining in a data-flow environment: experience in network intrusion detection, in: KDD, 1999, pp. 114–124.
[4] R. Agrawal, M. Yoshi, Pnrule: a new framework for learning classifier models in data mining a case study in network intrusion detection, Technical Report RC-21719, IBM Research Division, 2000.
[5] Y. Bouzida, S. Gombault, Eigenconnections to intrusion detection, in: Security and Protection in Information Processing Systems, IFIP 18th WorldComputer Congress, TC11 19th International Information Security Conference, 22–27 August 2004, Toulouse, France, 2004, pp. 241–258.
[6] S. Axelsson, Visualising intrusions: watching the webserver, in: Security and Protection in Information Processing Systems, IFIP 18th WorldComputer Congress, TC11 19th International Information Security Conference, 22–27 August 2004, Toulouse, France, 2004, pp. 229–274.
[7] G.V. Dozier, D. Brown, J. Hurley, K. Cain, Vulnerability analysis of ais-based intrusion detection systems via genetic and particle swarm red teams, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, IEEE Press, Portland, Oregon, 2004, pp. 111–116.
[8] E. Leon, O. Nasraouli, J. Gomez, Anomaly detection based on unsupervised niche clustering with application to network intrusion detection, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, IEEE Press, Portland, Oregon, 2004, pp. 502–508.
[9] F. Sergio de Paula, L. Nunes de Castro, P. Licio de Geus, An intrusion detection system using ideas from the immune system, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, IEEE Press, Portland, Oregon, 2004, pp. 1059–1066.
[10] J.D. Farmer, N.H. Packard, A.S. Perelson, The immune system, adaptation, and machine learning, Physica D 22 (1986) 187–204.
[11] S. Forrest, R.E. Smith, B. Javornik, A.S. Perelson, Using genetic algorithms to explore pattern recognition in the immune system, Evolutionary Computation 1 (3) (1993) 191–211.
[12] S.A. Hofmeyr, S. Forrest, Architecture for an artificial immune system, Evolutionary Computation 8 (4) (2000) 443–473.
[13] D. Dasgupta, F. Gonzales, An immunity-based technique to characterize intrusions in computer networks, IEEE Transactions on Evolutionary Computation 6 (3) (2000) 281–291.
[14] P.K. Harmer, P.D. Wiliams, G.H. Gunsch, G.B. Lamont, Artificial immune system architecture for computer security applications, IEEE Transactions on Evolutionary Computation 6 (3) (2002) 252–279.
[15] D. Dasgupta, An Overview of Artificial Immune Systems and Their Applications, Springer, 1999, pp. 3–23.
[16] E. Hart, P. Ross, J. Nelson, Producing robust schedules via an artificial immune system, in: Proceedings of the 1998 IEEE Congress on Evolutionary Computation, 1998, pp. 464–469.
[17] D. Dasgupta, An artificial immune system as a multi-agent decision support system, in: IEEE International Conferences on Systems, Man and Cybernetics, 1998, pp. 3816–3820
[18] L. de Castro, F. Von Zuben, Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation 6 (3) (2002) 239–251.
[19] Z. Michalewicz, Genetic algorithms + data structures = evolution programs, Second extended ed., Springer-Verlag, New York, Inc., New York, NY, USA, 1994.
[20] M. Ostaszewski, F. Seredynski, P. Bouvry, A Nonself Space Approach to Network Anomaly Detection, in: Proceedings of NIDISC'06, held in conjuction with IPDPS 2006, Rhodes Island, Greece.
[21] D. Rutkowski, Detecting intruders in computer networks using immune system (in polish), Master's thesis, Warsaw University of Technology, 2004.
[22] M. Zissman, Darpa intrusion detection evaluation [online], http://www.ll.mit.edu/ist/ideval, Tech. Rep., MIT, 1999.

Dr. Seredynski received his M.S. and Ph.D. degrees in Computer Science from the State Electrotechnical University - St. Petersburg in 1978 and 1973, respectively, and the habilitation from the Institute of Computer Science, Polish Academy of Sciences in 1998.

He is a professor of Computer Science at the Polish-Japanese Institute of Information Technology, Warsaw, Poland and an associate professor at the Institute of Computer Science, Polish Academy of Sciences.

His research interests concern naturally inspired computational paradigms such as evolutionary algorithms, artificial immune systems and cellular automata and their application for computer security, management of mobile and ad hoc networks, multiprocessor scheduling and grid computing.

He has published more than 100 papers in journals, books and conference proceedings. He was a visiting researcher at Luxembourg University (2004-2006), Centre for Mathematics and Computer Science, Amsterdam, The Netherlands (1996), International Computer Science Institute, Berkeley, USA (1995), University of Edinburgh (1993), Institut National Polytechnique de Grenoble, France (1991-1992), and a visiting associate professor at the University of Missouri, St. Louis, USA (1998-1999). He has served on programme committees for a number of international conferences in the areas of evolutionary computation, parallel, distributed and grid computing and multi-agent systems.

He is a member of Editorial Advisory Board of the International Journal of Pervasive Computing and Communications.

Pascal Bouvry earned his Ph.D. degree ('94) in computer science at the University of Grenoble, France. He is now Professor at the Faculty of Sciences, Technology and Communication of the University of Luxembourg and heading the Computer Science and Communication research unit (http://wiki.uni.lu/csc). Pascal Bouvry is specialized in parallel and evolutionary computing. His current interest concerns the application of nature-inspired computing for solving security issues.