# Detectors Tuning in the Intelligent Intrusion Detection Systems

Pavel Kachurka, *Brest State Technical University*
Viachaslau Kachurka, *Brest State Technical University*

**Abstract**

Modern Intrusion Detection and Prevention Systems operate with large amount of data. Most host-based systems can analyze big number of traffic features in real-time mode. But the network-based systems cannot gather and analyze network connections in the same way because of the high network speed and traffic overload. One of the approaches is based on the analysis of intrinsic and statistical traffic features using AI methods including artificial neural network (ANN) algorithms. In this work some aspects of ANN-based detectors' tuning are reviewed. Such tuning is done using genetic algorithms and special fine-tune algorithms. Comparison with non-tuned detectors is given.

## 1. Introduction

An increasing role of network information technologies in human activities leads to a rising level of attention to such technologies from evildoers. The average level of expenses of legitimate users in case of successful attack increases too.

Different systems of detection and prevention of network intrusions are used utilizing several different approaches to data mining. One of the most widely used approaches uses artificial neural networks (ANN) for detection of network intrusions. In our previous works [1-3] different types of neural networks for detectors were considered. As the best network type we have selected recirculation neural network (RNN).

In this paper the necessity of tuning these detectors is considered. Three different approaches are proposed - receiver operating characteristics (ROC)-analysis [4] based selection of best parameter's values, evolutionary programming for fitness-based selection of parameter's values, and fine-tuning iterational algorithm.

The training and testing process of RNN-based detectors was conducted on KDD'99 data base which contains records describing TCP-connections including 41 parameter from processed DARPA 1998 Intrusion detection evaluation database [5]. The given data base includes normal connections, and also the attacks of 23 types belonging to four classes: DOS – «denial-of-service» - refusal in service, for example, a Syn-flood; U2R – not authorized access with root privileges on the given system, for example, various attacks of buffer overflow; R2L – not authorized access from the remote system, for example, password selection; Probe – analysis of the topology of a network, services accessible to attack, carrying out search of vulnerabilities on network hosts.

## 2. RNN-based detectors

Recirculation neural networks (sometimes called replication neural network) differ from others ANNs in the following aspects (Figure 1). The input data vector has to be reconstructed in the same kind into the output vector. RNNs are commonly used in the compression and reconstruction tasks (direct and back propagation of the information in the networks «with a narrow throat») [6], for definition of outliers on a background of the general file of entrance data [7].

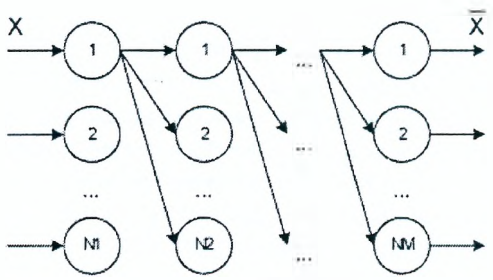


**Fig. 1. *M* layers RNN structure**
***$N_i$* – quantity of neural elements in *i-th* layer, *NM=N1* – quantities of neural elements in entrance and target layers are equal**

Nonlinear RNNs have shown good results as the detector of anomalies [8, 9]. Training RNN can be made in two ways: first, on normal connections so that input vectors on an output were reconstructed in themselves, thus the connection is more similar on normal, the less reconstruction error is:

$$E^k = \sum_j (\bar{X}_j^k - X_j^k)^2, \quad (1)$$

where $X_j^k$ – $j$-th element of $k$-th input vector, $\bar{X}_j^k$ – $j$-th element $k$-th output vector. Whether $E^k > T$, where $T$ – certain threshold for given RNN connection admits anomaly, or attack, differently – normal connection.

The opposite way of utilizing RNN-based intrusion detectors [1] is the following. Every RNN is trained to detect a specific attack, so that if reconstruction error is lower than threshold then connection admits attack; if error is higher (abnormal) then the connection could be normal (the specific attack is not registered). The detectors which use these types of RNNs are called "private detectors".

Thus, one RNN can be applied to definition of an accessory of input vector to one of two classes – to the training class $A$, or to the class of outliers $\bar{A}$:

$$\begin{cases} X^k \in A, & if \quad E^k \le T, \\ X^k \in \bar{A}, & if \quad E^k > T. \end{cases} \quad (2)$$

One of the issues here is the determination of threshold $T$ value which provides the most qualitative detection of attacking connections.

The other problem is the determination of the best exact architecture of RNN - values of M and N, - number of layers and numbers of neurons in each layer.

## 3. Detectors tuning

### 3.1 ROC analysis

Both problems can be solved using receiver operating characteristics (ROC) analysis.

ROC analysis is a tool of organizing and performance visualizing for any classifier [4]. In our case such organizing can be done for every detector, making threshold values different for every RNN.

ROC analysis uses terms "True positive rate" (TPR) and "false positive rate" (FPR), which both are relative values between 0 and 1:

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P} \quad (3)$$

where TP (true positive) is absolute amount of true positive detects (attacks are detected correctly), FN (false negative) is absolute amount of false negative detects (attacks are not detected);

$$FPR = \frac{FP}{TN + FP} = \frac{FP}{N} \quad (4)$$

where FP (false positive) is absolute amount of false positive detects (normal connection is detected as an attack) and TN (true negative) is absolute amount of true negative detects (normal connections, detected as normal connections).

Each threshold value gives us a pair (TPR; FPR). Testing each threshold value from the set of possible values (from 0 to positive infinity, step 0,001) gives us an ordered array of such (TPR, FPR) pairs, which can be used to plot a so called ROC-curve [4] (Figure 2).

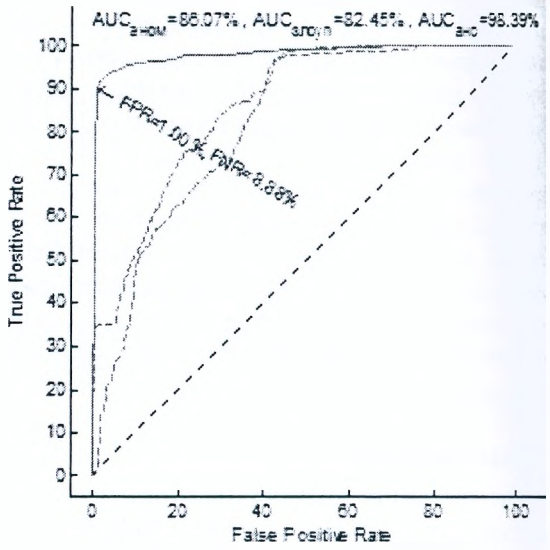


**Fig. 2. Example of ROC-curve for RNN with 41-18-41 architecture, hyperbolic tangent as activation function, ftp_data attack class connections. Three curves show performance of anomaly detector, misuse detector and detector ensemble [1].**

Threshold which gives the best performance can be found using L - distance between the point of ROC-curve and diagonal (0,0)-(1,1) of plot (Figure 3):

$$L_i = \sqrt{FP_i^2 + TP_i^2}\, \sin(\tan^{-1} \frac{TP_i}{FP_i} - \frac{\pi}{4}) \quad (4)$$

The maximal distance is given by the best threshold.

Another parameter, used in ROC-analysis, is "Area under curve" (AUC) [4], which shows performance of classifier.

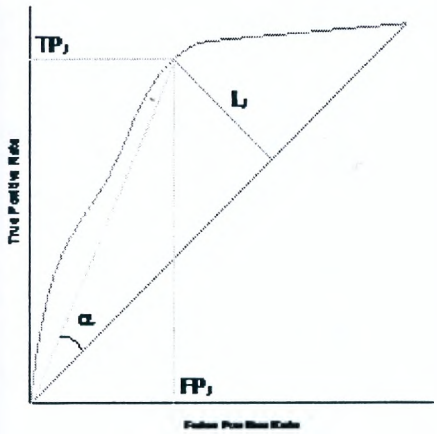


**Fig. 3. Geometrical interpretation of finding best threshold**

Precise AUC calculation is a long algorithm [4], but semantically it is a percentage of area under the ROC-curve in comparison to the whole square area from (0, 0) to (1, 1) points in ROC-coordinate system. A "naive" approach of calculation:

$$AUC = \frac{\sum_{i=2}^{N}\left(TPR_{i-1} + \frac{TPR_i - TPR_{i-1}}{2}\right) * (FPR_i - FPR_{i-1})}{FS} \quad (5)$$

where N is number of experiments, and FS is full square, in which the curve is plotted.

Classifiers can be compared using AUC: the higher AUC, the better classifier. Detectors with different architectures can be compared to select the best architecture.

The compared architectural parameters are number of values, activation functions and numbers of neurons in each layer (Figure 4).

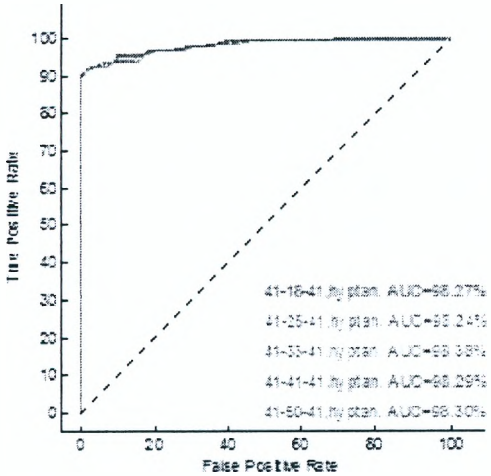


**Fig. 4. Comparison of AUCs for detectors with 18, 25, 33, 41 and 50 neurons in hidden layer**

AUC calculations give very similar values for different architectures, with 1%-deviations (Table 1).

**Tab.1.**

**AUC and accuracy comparison table for 3-layered and 5-layered detector ensembles**

| Num. of neurons in hid. laver | Activation function – tanh | | Activation function – logsig | |
|---|---|---|---|---|
| | ACC, % | AUC, % | ACC, % | AUC, % |
| | *Training set "ALL"* | | | |
| 18 | 97,43 | 99,98 | 97,04 | 99,98 |
| 25 | 97,78 | 99,98 | 97,80 | 99,98 |
| 33 | 97,10 | 99,99 | 96,60 | 99,97 |
| 41 | 97,09 | 99,99 | 97,20 | 99,99 |
| 50 | 97,16 | 99,98 | 97,91 | 99,98 |
| | *New connections set "ALL-NEW"* | | | |
| 18 | 83,88 | 98,27 | 84,41 | 98,34 |
| 25 | 82,80 | 98,24 | 83,88 | 98,25 |
| 33 | 83,72 | 98,38 | 83,55 | 98,31 |
| 41 | 83,73 | 98,29 | 83,49 | 97,68 |
| 50 | 83,80 | 98,3 | 83,41 | 98,33 |
| | *Training set "ALL"* | | | |
| 18/18/18 | 99,00 | 99,92 | 97,28 | 99,97 |
| 33/25/33 | 98,34 | 99,96 | 97,42 | 99,98 |
| 41/33/41 | 98,83 | 99,81 | 97,75 | 99,98 |
| 41/41/41 | 98,69 | 99,99 | 97,88 | 99,83 |
| 25/50/25 | 98,70 | 99,95 | 97,81 | 99,98 |
| | *New connections set "ALL-NEW"* | | | |
| 18/18/18 | 81,81 | 95,49 | 81,78 | 96,61 |
| 33/25/33 | 81,54 | 95,49 | 81,80 | 96,58 |
| 41/33/41 | 81,75 | 97,15 | 81,70 | 95,32 |
| 41/41/41 | 81,69 | 97,06 | 81,99 | 96,66 |
| 25/50/25 | 81,74 | 96,1 | 81,83 | 97,73 |

Values shown in Table 1 were calculated using two sets of data - "ALL-train", set with all types of connections (normal and all attacks), on which detectors were trained; and "ALL-new", all connections set, which is unknown for detectors.

As it is shown in Table 1, deviations for AUC and accuracy parameters are very small, so almost every given architecture can be used; however, the performance is slightly better for 3-layered **tanh**-activated networks with 25 or 33 neurons in hidden layer.

### 3.2 Evolutionary programming

Evolutionary programming (EP) is one of the four major evolutionary algorithm paradigms [10]. It is similar to genetic programming [11], but the structure of the program to be optimized is fixed, while its numerical parameters are allowed to evolve.

In case of detectors tuning, the parameter to evolve is threshold.

Chromosome consists of N values of thresholds (each one for every private detector, each private detector for one of N classes of attacks [1]). Fitness function is summary cost of detection errors, which we try to minimize:

$$Fitness = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\text{cost}(FP_{ij}) + \text{cost}(FN_{ij})\right) \rightarrow \min \quad (6)$$

where $FP_{ij}$ is the amount of $i$-th detector's false positive detections of $j$-th attack type, $FN_{ij}$ is the amount of $i$-th detector's false negative detections of $j$-th attack type; N - number of detectors (and attack type), 5 or 23 [1].

The costs of false positive and false negative errors are defined as cost matrix in [12].

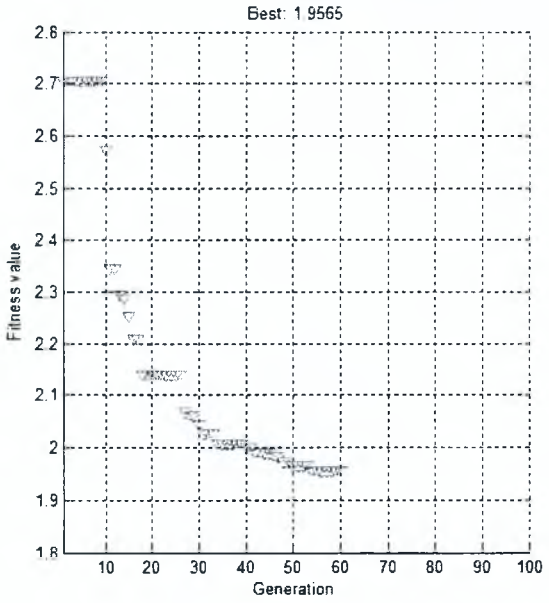


**Fig. 5. 60 iterations of EP tuning of thresholds**

Usage of EP, unlike ROC-analysis, can be effective for detector tuning on unknown input data.

### 3.3 Fine-tuning algorithm

Another approach is somehow similar to back-propagation learning algorithm.

After training of neurodetectors of all types it is needed to tune the thresholds $T_i$ to minimize the classification error. The following algorithm can be used.

1. Matrix of classification results $C$ is calculated, where $C_{ij}$ – amount of vectors of class $A_i$, detected as vectors of class $A_j$; if $i \neq j$, this number shows the amount of incorrectly classified vectors. To minimize this number, we should for every $k$-th vector minimize $\delta_i^k$ and maximize $\delta_j^k$, where $\delta_j^k$ is a relative error of reconstruction for detecting $k$-th vector of class $A_j$:

$$\delta_j^k = \frac{E_j^k}{T_j}, \tag{7}$$

where $T_j$ is a threshold of $i$-th detector; initially $T_i = mean\delta_i^k$. The less is $\delta_i^k$, the more is possibility of input $k$-th vector being of class $A_i$.

To do the minimization (maximization) of relative reconstruction error, we have to compute summary relative errors for every vector, which actually is of class $A_i$, but has been detected as a vector of class $A_j$:

$$ER_{ij} = \sum_{k=1}^{C_{ij}} \left(\frac{E_i^k}{E_j^k} T_j - T_i\right) \tag{8}$$

for $\forall i = 1..N, j = 1..N, i \neq j$.

2. For $\forall i = 1..N$ new threshold is calculated:

$$T_i = T_i + \alpha \frac{\sum_{j=1}^{N, i \neq j} ER_{ij}}{\sum_{j=1}^{N, i \neq j} C_{ij}} - \beta \frac{\sum_{j=1}^{N, i \neq j} ER_{ji}}{\sum_{j=1}^{N, i \neq j} C_{ji}} \tag{9}$$

where $a=\beta=0,01$ (empiric values).

Formula 9 decreases $T_i$, increasing possibility of correct classification of vectors of $i$-th class (not as a vector of $j$-th class), at the same time increasing this threshold, decreasing possibility of incorrect classification as of $i$-th type for vectors of $j$-th type. Depending on the values of this errors, $T_i$ will be either increased or decreased on a small value.

3. If max number of iterations is not reached, go to p. 1.

Figures 6 and 7 show graphs of summary changes of thresholds and summary classification error $MC$.

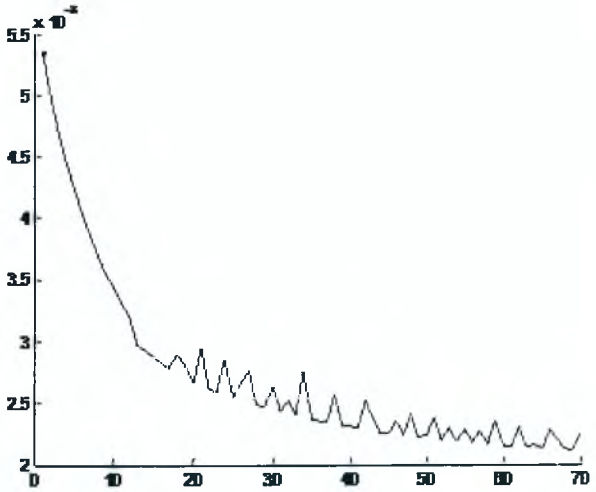


**Fig. 6. Summary changes of thresholds in 70 iterations**

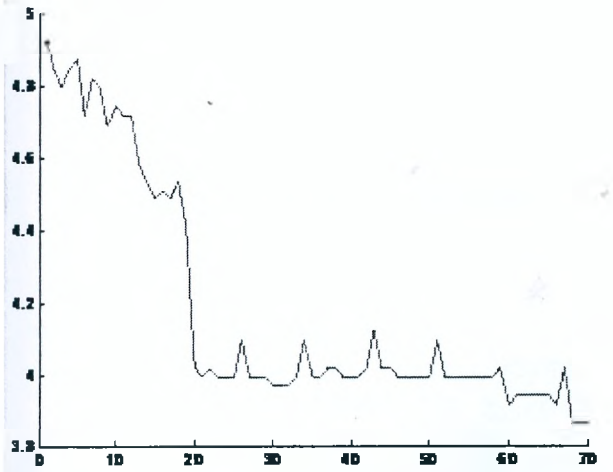


**Fig. 7. Summary classification error MC**

## 4. Experimental results

In [1] it is shown that 2 approaches of detections can be used: using fusion of private detectors, where every detector is trained for one exact type of attack (one of 23 types); and using fusion of attack class detectors - where one detector is trained not for one attack, but for one of connection classes (one of 5 classes - DoS, Probe, U2R, R2L, normal connection).

Table 2 shows FPR (false positive rate), FNR (false negative rate) and ACC (accuracy of detection) for all 5 classes of connections, using different types of tuning (ROC-tuning, EP-tuning and fine-tuning) for different approaches of detecting (5 or 23 detectors) - after threshold tuning.

**Tab.2.**

**Detection results after threshold tuning**

| Class | $FPR_i$, % (5 det.) | | | |
|---|---|---|---|---|
| | None | ROC | EP | Fine |
| DoS | 1,27 | 1,01 | 1,14 | 1,04 |
| Probe | 3,41 | 1,52 | 1,85 | 1,54 |
| U2R | 5,77 | 5,17 | 4,75 | 4,4 |
| R2L | 6,50 | 5,42 | 4,77 | 4,22 |
| Normal | 12,75 | 10,4 | 10,81 | 11,8 |
| Class | $FNR_i$, % (5 det.) | | | |
| | None | ROC | EP | Fine |
| DoS | 0,28 | 0,01 | 0,14 | 0,1 |
| Probe | 1,87 | 0,71 | 1,42 | 1,2 |
| U2R | 0,72 | 0,13 | 0,66 | 0,55 |
| R2L | 0,66 | 0,11 | 0,32 | 0,23 |
| Normal | 0,10 | 0,02 | 0,01 | 0,01 |
| Class | $ACC_i$, % (5 det.) | | | |
| | None | ROC | EP | Fine |
| DoS | 98,94 | 99,22 | 99,1 | 99,32 |
| Probe | 96,89 | 98,78 | 98,74 | 99,05 |
| U2R | 95,24 | 94,85 | 96,74 | 94,93 |
| R2L | 94,67 | 95,5 | 96,25 | 98,7 |
| Normal | 89,78 | 91,14 | 90,4 | 90,21 |
| Class | $FPR_i$, % (23 det.) | | | |
| | None | ROC | EP | Fine |
| DoS | 1,91 | 1,77 | 1,99 | 3,64 |
| Probe | 3,33 | 1,52 | 2,14 | 3,3 |
| U2R | 6,98 | 5,17 | 4,75 | 6,89 |
| R2L | 5,53 | 5,42 | 4,77 | 2,75 |
| Normal | 12,75 | 10,4 | 10,81 | 10,84 |
| Class | $FNR_i$, % (23 det.) | | | |
| | None | ROC | EP | Fine |
| DoS | 0,18 | 0,21 | 0,2 | 0,39 |
| Probe | 0,03 | 0,02 | 0,02 | 0,02 |
| U2R | 2,54 | 1,91 | 1,89 | 1,49 |
| R2L | 3,37 | 2,51 | 2,99 | 0,37 |
| Normal | 0,1 | 0,04 | 0,05 | 0,02 |
| Class | $ACC_i$, % (23 det.) | | | |
| | None | ROC | EP | Fine |
| DoS | 97,91 | 98,09 | 96,57 | 94,86 |
| Probe | 96,64 | 98,11 | 97,62 | 96,93 |
| U2R | 96,48 | 97,22 | 98,12 | 96,73 |
| R2L | 91,1 | 93,51 | 96,58 | 99,08 |
| Normal | 87,15 | 90,51 | 89,11 | 90,14 |

As it is shown in Table 2, false detection rates (FPR and FNR) are decreased, and accuracy is increased after using any tuning algorithm. All three algorithms show comparable results.

As it is said in [13], a detector is effective if FPR and FNR are less than 5% - such effectiveness threshold is reached with EP-tuning and fine-tuning algorithms for 5-detector IDS.

## Conclusions

As it is shown in p.4, non-tuned detectors are showing worse effectiveness, than tuned ones. The fusion of 3 tuning algorithms (choosing the best algorithm for each attack class) gives better results, than using only one algorithm for all attack classes.

Usage of such tuning is obligatory in creating fully functional intrusion detection system.

## Bibliography

[1] Kachurka Pavel: *Neural Network Approach to Real-Time Intrusion Detection*, proceedings of OWD'2008, 18-21 October 2008, Wisla

[2] Golovko Vladimir, Bezobrazov Sergei, Kachurka Pavel, Vaitsekhovich Leanid: *Neural Network and Artificial Immune Systems for Malware and Network Intrusion Detection*, "Studies in Computational Intelligence", v. 263, Springer, 2010; pp. 485-515

[3] Kachurka Pavel, Golovko Vladimir: *Neural Network Approach to Real-Time Network Intrusion Detection and Recognition*, proceedings of IDAACS'2011, v.1, pp. 393-398

[4] Fawcett Tom: *An introduction to ROC analysis*, Pattern Recognition Letters, 2006. №27. – pp. 861–874

[5] KDD Cup'99 Competition, 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[6] V. Golovko, O. Ignatiuk, Yu. Savitsky, T. Laopoulos, A. Sachenko, L. Grandinetti:

*Unsupervised learning for dimensionality reduction.* Proc. of Second Int. ICSC Symposium on Engineering of Intelligent Systems EIS'2000, University of Paisley, Scotland, June 2000. Canada / Switzerland: ICSS Academic Press, pp. 140 – 144, 2000

[7] S. Hawkins, H. He, G. Williams, R. Baxter: *Outlier Detection Using Replicator Neural Networks.* In Proc. of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK02) Lecture Notes in computer Science, Vol. 2454, Springer, Pages 170-180, ISBN 3-540-44123-9, 2002

[8] Kachurka Pavel, Golovko Vladimir: *Neural Network Approach to Anomaly Detection Improvement.* In Proc. of 8th International Conference on Pattern Recognition and Information Processing (PRIP'05), May, 18-20, Minsk, Belarus, 2005 – pp. 416-419.

[9] Golovko Vladimir, Kachurka Pavel: *Intrusion recognition using neural networks,* International Scientific Journal of Computing, vol.4, issue 3, 2005, p.37-42

[10] Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*, Springer, 2003. ISBN 3-540-40184-9

[11] Langdon, W. B.: *Genetic Programming and Data Structures*, Springer, 1998. ISBN 0-7923-8135-1

[12] Elkan C.: *Results of the KDD'99 Classifier Learning*, SIGKDD Explorations, ACM SIGKDD, Jan 2000

[13] Lukacki, A.: *Attack Detection*, BXB-Petersburg, St. Petersburg (2001) (in Russian)

**Authors:**

Pavel Kachurka
Brest State Technical University
Moskovskaya str., 267
224017 Brest
tel. +375 29 7202122
email: pakochurko@bstu.by

Viachaslau Kachurka
Brest State Technical University
Moskovskaya str., 267
224017 Brest
tel. +375 29 2045651
email:
viachaslau.kachurka@gmail.com