Neural Network and Artificial Immune Systems for Malware and Network Intrusion Detection

Vladimir Golovko, Sergei Bezobrazov, Pavel Kachurka, and Leanid Vaitsekhovich

Laboratory of Artificial Neural Networks, Brest State Technical University, Moskovskaja str. 267, 224017 Brest, Belarus

Abstract. Neural network techniques and artificial immune systems (AIS) have been successfully applied to many problems in the area of anomaly activity detection and recognition. The existing solutions use mostly static approaches, which are based on collection viruses or intrusion signatures. Therefore the major problem of traditional techniques is detection and recognition of new viruses or attacks. This chapter discusses the use of neural networks and artificial immune systems for intrusion and virus detection. We studied the performance of different intelligent techniques, namely integration of neural networks and AIS for virus and intrusion detection as well as combination of various kinds of neural networks in modular neural system for intrusion detection. This approach has good potential to recognize novel viruses and attacks.

1 Introduction

At present one of the forms of world space globalization is cyber space globalization, because of increasing of a number of computers connected to the Internet. The rapid expansion of network-based computer systems has changed the computing world in the last years. As a result the number of attacks and criminals concerning computer networks are increasing. Due to the increasing computer incidents because of cybercrime, construction effective protecting systems are important for computer systems security. There are many different techniques to build computer security systems [1,2,3]. The traditional approaches use as a rule static models, which are based mostly on signature analysis [4]. It consists of collecting and analyzing of viruses or intrusion signatures. The main problem of signature approach is inability to detect new viruses and attacks. Besides, this approach demands time for signature database updating. The methods of heuristic analysis [5], which were developed for disadvantages removal of traditional approach for malware detection, are still a long way off perfection. The heuristic analyzers are frequently finding malicious code where it absent and vice versa.

To overcome these limitations, the AIS and neural networks can be effectively used to build computer security systems. In order to achieve maximal performance we study different intelligent techniques, namely artificial neural networks and artificial immune systems. In comparison with conventional approaches such technique has ability to detect novel viruses and attacks in real time. Besides, this allows getting more accurate results.

The rest of the chapter is organized as follows. Section 2 presents overview of artificial immune systems (AIS), as well as integration of AIS and neural networks for malicious code detection. Section 3 tackles different neural network techniques for intrusion detection and recognition.

2 Integration of Artificial Immune Systems and Neural Network Techniques for Malicious Code Detection

The actual researches in the information security field are directed to creation on such new methods that will be capable to detect unknown malicious code. The biologically-inspired and ready-built on basic principals of Biological Immune Systems (BIS) [1], Artificial Immune Systems method, thanks to distributed computational power [2, 3], is allow to detect not only known but unknown malware. Combining of two methods of artificial intelligence (Artificial Neural Networks method [6, 7] and Artificial Immune Systems method) let us developed a new technique of detection of malicious code. This technique allows to avoid the main weaknesses of signature analyzers and to detect unknown malware.

2.1 The Biological Immune System Overview

The biological immune system is unique protective mechanism which defends organism from invaders: harmful bacteria and viruses. The BIS capable to detect foreign cells and destroy them, and based on synthesis of special proteins – antibodies, which capable to bind with foreign material. Every day BIS face with a dozens invaders and successfully struggle against them.

The biological immune system is based on capability of antibodies to distinguish between self (cells of own body) and nonself (antigens, foreign substance) [1]. For complete and successful detection of wide variety of antigens the BIS must generate a large variety of detectors (B-lymphocytes and T-lymphocytes). Lymphocytes are formed from the bone marrow stem cells and initially incapable of antigens detect. In order to acquire immunological ability they have to go through maturation process. T-lymphocytes are mature in thymus and B- lymphocytes are mature in lymph nodes. During the maturation process only fittest lymphocytes are survived. Mature lymphocytes have on the own surface specific detectors which able to react on specific antigens. Lymphocytes circulate through the body and perform the function of antigens detection [8,9].

When lymphocyte detects an antigen the process called clonal selection is occurred [10]. The clonal selection process consists in proliferation (cycle of cell division) those lymphocytes who detected a virus. As a result the large population of identical detectors is formed. These generated lymphocytes are reacting on the same antigen and allow to BIS timely eliminating manifestation of disease.

Another important process in the BIS is immune memory [9]. Immune memory keeps information about previous infection and owing to this information defense body against repeated infection. Immune memory consists of detectors which in the past detected antigens. These detectors circulate in the body at long time and form the immune memory. By repeated infection antigens can be detected quickly since the BIS already have lymphocytes which react on this infection. Described processes showed in Figure 1.

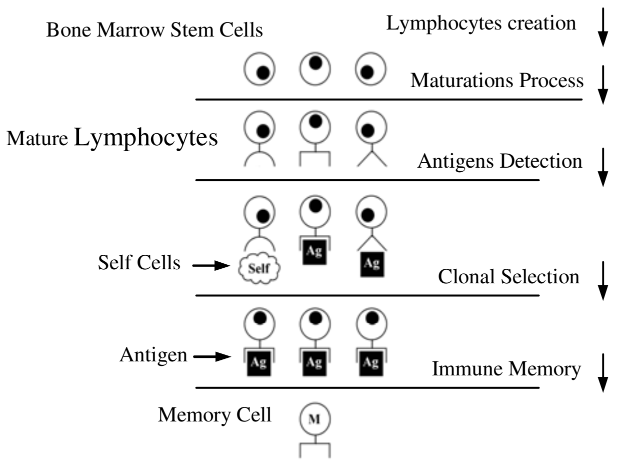


Fig. 1. Basic principles of the biological immune system: stage of lymphocytes evolution

2.2 The Artificial Immune Systems Overview

The AIS is founded on the same processes as BIS: detectors generation, detectors maturation, detection process, detectors cloning and mutation, immune memory creation. Let's view in detail each process (Fig. 2 shows processes as flow block).

Process of detectors creation in computer system represents a random generation of immune detectors population. Each of them can be, for example, as binary string of fixed size [11]. At this stage generated immune detectors have analogy with immature lymphocytes.

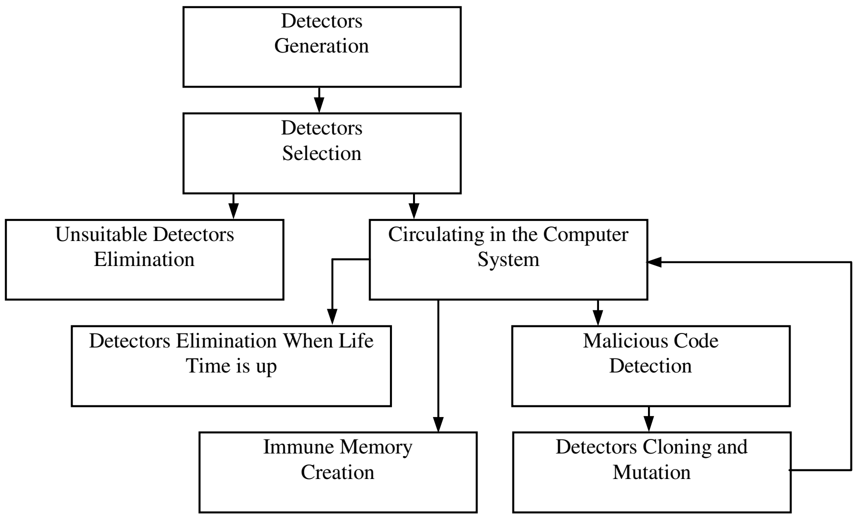

Detectors Generation
Detectors Selection
Unsuitable Detectors Elimination
Circulating in the Computer System
Detectors Elimination When Life Time is up
Malicious Code Detection
Immune Memory Creation
Detectors Cloning and Mutation

Fig. 2. Block-diagram model of artificial immune system: AIS interprocess communication

After generation detectors undergo a selection process. During the selection process detectors are received training in self – nonself recognition. But not all of immune detectors can get ability to correct pattern recognition. Even after training process some of them detect self as nonself and vice versa. These detectors named unsuitable detectors and should be eliminating. As a result of the selection process unsuitable detectors are eliminated and survive only those which able to distinguish between self and nonself. S. Forest at al. [12] proposed negative selection algorithm based on the principles of self – nonself discrimination in the BIS. According to negative selection algorithm immune detectors are compared with set of self pattern. If detector is similar to self pattern, it is reputed as negative and destroyed. Only those detectors survive which are structurally different from self data. For matching between detectors and pattern can be applied different rules: bit-by-bit comparison, r-contiguous matching [13] and r-chunk matching [14]. Mature detectors structurally different from self pattern therefore react only against nonself pattern.

Mature detectors circulate in computer systems. For maintenance of wide variety of structurally different detectors, each immune detector has a lifecycle [1,3]. The lifecycle is a time during immune detector can be found in the computer system. When the life time ends the detector is destroyed but if the detector detected anomaly (in our case it is malicious code) then lifecycle is prolong. Lifecycle mechanism allows the AIS to unload from weak detectors and permanently provide a space for new various immune detectors.

When malware enters into the computer system it often infects a large quantity of files. For quick reacting and eliminating virus manifestation we need a great number of similar detectors, which react on the same malicious code. For large quantity of similar detectors generations in AIS a cloning process is exists. During cloning process immune detector which found malicious code undergoes a cloning mechanism. Cloning means a large quantity of similar detectors creation. This mechanism allows the AIS infection elimination in a short space of time. Along with cloning a mutation mechanism is used [15]. Mutation process means introduce small random changes in detectors structure (for example, inverting of several bits in binary string) thereby as much as possible similar structure to finding virus acquires.

When the malicious code is eliminated then most of cloning detectors die because of lifecycle. However the fittest of them are kept as memory detectors. A set of such detectors are formed an immune memory. The immune memory keeps information about all malicious code which a computer system infects. The same as BIS the immune memory allows the AIS to quickly react on repeated infection and to fight against it.

2.3 Application of Neural Networks in Artificial Immune System to Malicious Code Detection

A quality of malicious code detections depends on the structure of immune detectors. We considered the immune detector as a binary string. This structure is comfortable, as it corresponds with data presentation in computer systems, and allows implement simple matching rules. However binary structure applies some restrictions. As it is well known bit-by-bit comparison is one of the slowest operations and needs heavy computational power. We propose the artificial neural networks (ANN) applying for the immune detectors formation. This approach for the detectors generation should remove weaknesses of the binary string structure and should increase a rate of the malicious code detection.

The ANN for vector quantization was proposed by T. Kohonen in 1982 and named as learning vector quantization (LVQ) [16]. The LVQ uses for classification and image segmentation problems. The LVQ is a feedforward artificial neural network with an input layer, a single hidden competitive Kohonen layer and an output layer (see Fig.3).

The output layer has as many elements as there are classes. Processing elements of the hidden (Kohonen) layer are grouped for each of these classes. Each class can be represents as a number of cells of the input space of samples. The centre of each cell corresponds to a codebook vector. One codebook vector represents one class only. The main idea of vector quantization is to cover the input space of samples with codebook vectors. A codebook vector can be seen as a hidden (Kohonen) neuron or a weight vector of the weights between all input neurons and the regarded Kohonen neuron respectively [17].

The Learning consists in modifying weights in accordance with adapting rules and, therefore, changing the position of a code vector in the input space. Many methods of training of the LVQ are exists [18]. We used the competitive training with one winner.

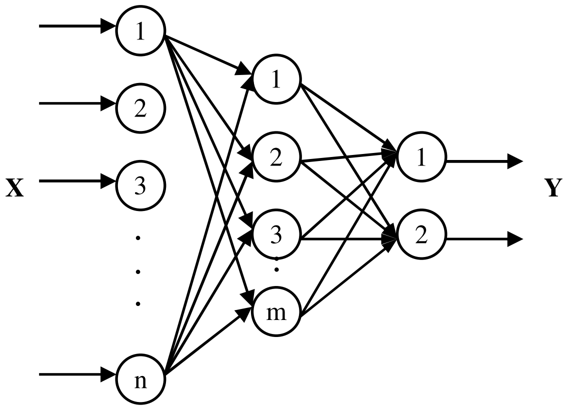


Fig. 3. LVQ one hidden competitive layer of neurons fully connected with the input layer, and the linear output layer consists of a number of neurons equal of a number of classes

Let's examine the process of detectors generation based on the LVQ. First an initial population of detectors is created. Each detector represents one LVQ. Further we will determine a set of self files consisting of different utilities of operating system, various software files etc, and one or a few malicious code (or signature of malicious code). Both self files and malicious virus will be used for LVQ learning. It is necessary to be sure that files from the set of self's are noninfected (without malicious code). Presence of malicious code or its signature in a learning sample allows a mature detector to tell the difference between self and nonself. Of course the more there are diverse files in the learning sample the more structurally different detectors are got. It is desirable to have all kind of malicious cod (worms, Trojans, file infectors etc.) in the learning sample. However it is not compulsory condition. As stated above there are differences between malicious software and noninfected files, which influence on the decision of a mature detector. By learning we denote to the LVQ where data from noninfected files and where data from malicious code (learning by instruction). A set of mature LVQ form a population of detectors which circulate into the computer system. In process of checking of a file the LVQ identifies unknown pattern and determines its proximity to one or another sample vector. Depending on this the LVQ takes a decision about the nature of files – self or malicious code.

General algorithm of neural immune detectors activity can be represents as next iterations:

1. Neuronet immune detectors generations. Each immune detector represents one neural network.
2. Detectors learning. The training set of self and nonself files is formed.
3. Unsuitable detectors eliminations.
4. Circulation of neuronet immune detectors in the computer system. On this stage detectors during scanning different files perform the function of malicious code detection.
5. Neuronet immune detectors eliminations by lifecycle.
6. Detection of malicious code.
7. Detectors cloning and mutation. On this iteration the AIS is formed a large quantity of similar detectors which react on the same malicious code.
8. Immune memory creation. Detectors of immune memory keep information about previous infections.

2.4 Description of Experimental Model of the AIS Security System

We used next structure of the LVQ for detectors formation – 128 neurons of the input layer, 10 neurons of the hidden layer and 2 neurons of the output layer (such detector is illustrated in Fig. 3, where $n = 128$, $m = 10$). A learning sample for one detector is formed as follows:

- four noninfected files from self's and one malicious code are selected randomly;
- from each selected file in fives fragments (binary string with length equal 128 bits) are randomly chosen. Then these fragments step by step will be inputted to the LVQ.

Competitive learning with one winner is used for the LVQ training. It is learning by instruction that is we indicate during training to the neural network where data from noninfected files is and where data from malicious code is. As a result of learning we get 10 code vectors in the hidden layer and they correspond with two output classes. The first class consists from 8 code vectors (noninfected files). The second class consists from 2 code vectors (malicious code).

As a result we will have a set of structurally different mature detectors since random process for files selecting is used for detectors learning. These detectors will be used for file identifications and decision making – is it self file or malicious virus? Experimental results in next section are described.

An immature detector compares any input pattern (independently of malicious code or noninfected file) to the first class (noninfected files) with probability 80% and to the second class (malicious code) with probability 20% since we divide the input space of samples in proportions 8 to 2 (see above). A mature detector (after the LVQ learning) will correlate an input pattern from a noninfected file with the first class with an expectancy of hitting more then 80%. Accordingly, the mature detector will correlate an input pattern from malicious code with the second class with expectancy of hitting more then 20%. The detector divides the under test file into pieces of 128 bytes apiece, examines them for malicious code in series and calculates total expectancy of hitting in one or another class:

$$P = X/N * 100\% , \tag{1}$$

where X is a number of pieces running in one of a class, N is a total number of pieces of an under test file.

Let's review an example:

The file diskcopy.com (utility of operation system): file size is 7168 byte – 56 pieces of 128 bytes. A detector correlated 49 pieces with the first class (self) that was $P_S = 49/56 \cdot 100\% = 87{,}5\%$ expectancy of hitting. Accordingly an expectancy of hitting in the second class (malicious code) was $P_M = 7/56 \cdot 100\% = 12{,}5\%$. Detector's decision was noninfected file.

Experimental model of AIS for malicious code detection showed in Figure 4.

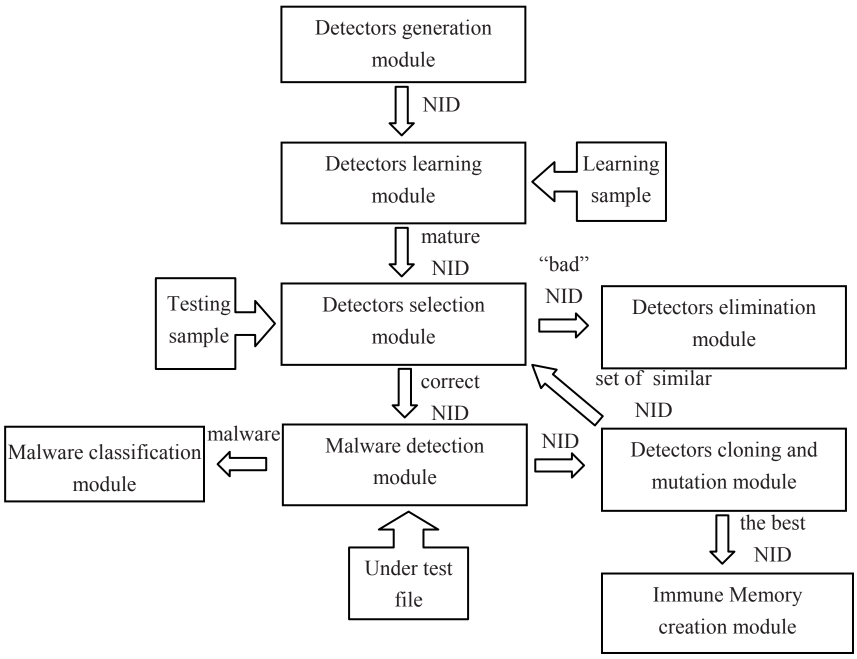

Detectors generation module
NID
Detectors learning module
Learning sample
mature NID
"bad" NID
Testing sample
Detectors selection module
Detectors elimination module
correct NID
set of similar NID
malware
Malware classification module
Malware detection module
NID
Detectors cloning and mutation module
the best NID
Under test file
Immune Memory creation module

Fig. 4. Model of AIS for malicious code detection: NID – neuronet immune detector

2.5 Experimental Results

For our experiments we choose "wild" malwares, which were in the top 10 of the most prevalence at January and February in many countries. The latest malware using new algorithms and methods are chosen for test in order to observe the ability of neuronet immune detectors to find unknown malware.

In the first experiment we train detectors using well-known, not new different malware. They were owned to various classes: network worms (*Email-Worm.Win32. Eyeveg.m*, *Net-Worm.Win32.Bozori.a*), Trojans (*Trojan-Downloader.Win32.Adload.a*), classic viruses (*Virus.Win32.Hidrag.d*). Malware are classed according to Kaspersky classification [19]. Table 1 shows the results of malware detection. In all tables we used next parameters: P_S is expectancy that the under test file is noninfected (self) and P_M is expectancy that the under test file is malicious code. If $P_S > 0.8$ then detector takes under test file as self. If $P_S < 0.8$ then detector takes under test file as malware.

Table 1. The results of malware detection

Malware	Detector 1	Detector 2	Detector 3	Detector 4
	P_S / P_M	P_S / P_M	P_S / P_M	P_S / P_M
Backdoor.Agobot	0.79/**0.21**	0.72/**0.28**	0.72/**0.28**	0.**85**/0.15
E-Worm.Bagle	0.69/**0.31**	0.51/**0.49**	**0.99**/0.01	0.74/**0.26**
E-Worm.Brontok	0.74/**0.26**	0.60/**0.40**	**0.98**/0.02	0.78/**0.22**
E-Worm.LovGate	0.72/**0.28**	0.53/**0.47**	**0.99**/0.01	0.74/**0.26**
E-WormMydoom	0.74/**0.26**	0.66/**0.34**	**0.81**/0.19	**0.83**/0.17
E-Worm.NetSky	0.77/**0.23**	0.70/**0.30**	0.77/**0.23**	0.75/**0.25**
E-Worm.Nyxem	**0.81**/0.19	0.76/**0.24**	0.72/**0.28**	**0.87**/0.13
E-Worm.Rays	**0.93**/0.07	**0.86**/0.14	0.79/**0.21**	**0.88**/0.12
E-Worm.Scano	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00
Net-Worm.Mytob	0.69/**0.31**	0.54/**0.46**	**0.97**/0.03	0.75/**0.25**
Trojan.KillWin	**1.00**/0.00	**1.00**/0.00	**0.95**/0.05	**1.00**/0.00
Trojan.Dialer	**0.82**/0.18	0.76/**0.24**	0.80/0.20	**0.87**/0.13
Trojan.VB	**0.91**/0.09	**0.86**/0.14	0.69/**0.31**	**0.91**/0.09
Trojan-D.Small	0.79/**0.21**	0.75/**0.25**	0.68/**0.32**	**0.84**/0.16
Trojan-D.Zlob	**0.87**/0.13	0.74/**0.26**	**0.90**/0.10	0.80/0.20

The detector 1 is trained on *Email-Worm.Win32.Eyeveg.m* and able to detect 53% of all amount malware. The detector 2 is trained on *Net-Worm.Win32.Bozori.a* and able to detect 73% of all amount malware. The detector 3 is trained on *Trojan-Downloader.Win32.Adload.a* and able to detect % of all amount malware. The detector 4 is trained on *Virus.Win32.Hidrag.d* and able to detect 33% of all amount malware. In the result four detectors cover almost whole space of malware with the exception *Email-Worm.Win32.Scano.gen* and *Trojan.BAT.KillWin.c*.

In the second experiment we had subset all collection of the newest malware in to their classes, then we chosen a typical sample of every class and trained detectors on selected malware. The goal of this experiment is to research how different neuronet immune detectors react to unknown malware. Table 2 shows the results of the second experiment.

The detector 1 is trained on *Email-Worm.Win32.NetSky.c*, the detector 2 is trained on *Email-Worm.Win32.Nyxem.e*, the detector 3 is trained on *Net-Worm.Win32.Mytob.w* and the detector 4 is trained on *Trojan-Downloader.Win32.Zlob.jd.* As follows from results the first, second and third detectors find email worms and net worms very well as representatives of this class were included in learning sample for these detectors. Detection of relating to another class malware (in our case they are Trojans) is already not so well. The picture of malware detection by the firth detector is directly opposite. The detector 4 finds Trojans very well and net worms not so well. As a result all four detectors cover the whole space of malware (except *Email-Worm.Win32.Scano.gen* and *Trojan.BAT.KillWin.c*).

Table 2. The results of malware detection

Malware	Detector 1	Detector 2	Detector 3	Detector 4
	P_S / P_M	P_S / P_M	P_S / P_M	P_S / P_M
Backdoor.Agobot	0.72/**0.28**	0.68/**0.32**	0.79/**0.21**	**0.87**/0.13
E-Worm.Bagle	0.69/**0.31**	0.73/**0.27**	0.61/**0.39**	0.60/**0.40**
E-Worm.Brontok	0.73/**0.27**	0.77/**0.23**	0.68/**0.32**	0.66/**0.34**
E-Worm.LovGate	0.70/**0.30**	0.76/**0.24**	0.63/**0.37**	0.60/**0.40**
E-WormMydoom	0.70/**0.30**	0.64/**0.36**	0.74/**0.26**	**0.82**/0.18
E-Worm.NetSky	0.71/**0.29**	0.66/**0.34**	0.77/**0.23**	**0.84**/0.16
E-Worm.Nyxem	0.75/**0.25**	0.70/**0.30**	**0.82**/0.18	**0.89**/0.11
E-Worm.Rays	**0.90**/0.10	**0.93**/0.07	**0.91**/0.09	0.79/**0.21**
E-Worm.Scano	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00
Net-Worm.Mytob	0.68/**0.32**	0.71/**0.29**	0.63/**0.37**	0.63/**0.37**
Trojan.KillWin	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00	**1.00**/0.00
Trojan.Dialer	0.77/**0.23**	0.74/**0.26**	**0.81**/0.19	0.73/**0.27**
Trojan.VB	**0.82**/0.18	0.79/**0.21**	**0.91**/0.09	0.75/**0.25**
Trojan-D.Small	0.75/**0.25**	0.72/**0.28**	0.79/**0.21**	0.70/**0.30**
Trojan-D.Zlob	**0.87**/0.13	**0.93**/0.07	**0.85**/0.15	0.71/**0.29**

In the third experiment we compare malware detection results by heuristic analyzer of ESET NOD32 antivirus software [20] and by our system. The results of experiment are displayed in the table 3.

Both *Trojan.BAT.KillWin.c* and *Email-Worm.Win32.Scano.gen* stay undetectable for NOD32 and AIS (we consider reason of this above). In addition NOD32 misses two malware (*Net-Worm.Win32.Mytob.q* and *Trojan.Win32.VB.at*), while AIS detects them.

Table 3. The comparative analysis of malware detection results

Malware	NOD32	AIS
Backdoor.Win32.Agobot.gen	Virus	Virus
Email-Worm.Win32.Bagle.gen	Virus	Virus
Email-Worm.Win32.Brontok.q	Virus	Virus
Email-Worm.Win32.LovGate.w	Virus	Virus
Email-Worm.Win32.Mydoom.l	Virus	Virus
Email-Worm.Win32.NetSky.aa	Virus	Virus
Email-Worm.Win32.Nyxem.e	Virus	Virus
Email-Worm.Win32.Rays	Virus	Virus
Email-Worm.Win32.Scano.gen	Ok	Ok
Net-Worm.Win32.Mytob.q	Ok	Virus
Trojan.BAT.KillWin.c	Ok	Ok
Trojan.Win32.Dialer.z	Ok	Virus
Trojan.Win32.VB.at	Ok	Virus
Trojan-Downloader.Win32.Small.to	Virus	Virus
Trojan-Downloader.Win32.Zlob.jd	Virus	Virus

Thus, as was shown the AIS for malicious code detection is able to discern between noninfected files of operation system and malicious code. The feature of the AIS consists in capability for unknown malicious code detection. Application of the ANN for detectors generation allows us to create the powerful detectors. Undesirable detectors are destroyed during the selection process which allows avoiding false detection appearance. Uniqueness of detectors consists in capability to detect several malicious viruses. That is detector can detect viruses analogous with that malicious code on which training are realized. In that way we significant increased probability of unknown malicious code detection. As experiments show it is necessary to large population of detectors creation. Presence of random probability by detectors generation enables to create different detectors. However it is significant that detectors ability depends on files on which they are trained. It is desirable for training process a various noninfected files and all types of malicious code to have. If your computer system with outdated antivirus bases can be unprotected in the face of new malicious code attack then the AIS gives you a high probability detect it. Applying of the AIS for malicious code detection will expand the potentialities of existing antivirus software and will increase level of computer systems security.

3 Neural Network Techniques for Intrusion Detection

The goal of Intrusion Detection Systems (IDS) is to protect computer networks from attacks. An IDS has been widely studied in recent years. There exist two main intrusion detection methods: misuse detection and anomaly detection. Misuse detection is based on the known signatures of intrusions or vulnerabilities. The main disadvantage of this approach is that it cannot detect novel or unknown attacks that were not previously defined. There are examples of misuse detection models: IDIOT [22], STAT [23] and Snort [24]. Anomaly detection defines normal behavior and assumes that an intrusion is any unacceptable deviation from normal behavior. The main advantage of anomaly detection model is the ability to detect unknown attacks. There are examples of anomaly detection models: IDES [25] and EMERALD [26].

Different defense approaches exist in order to protect the computer networks, namely, neural networks, data mining, statistical approach.

The principal component classifier is examined in [27, 28]. The data mining techniques were presented in [29, 30]. The other authors proposed a geometric framework for unsupervised anomaly detection and three algorithms: cluster, k-Nearest Neighbor (k-NN) and Support Vector Machine (SVM) [31, 32]. The different neural networks can be used for intrusion detection [33, 34]: Self Organizing Maps (SOM), MLP, Radial Basis Function (RBF) network.

The major problem of existing models is recognition of new attacks, low accuracy, detection time and system adaptability. The current anomaly detection systems are not adequate for real-time effective intrusion prevention [32]. Therefore processing a large amount of audit data in real time is very important for practical implementation IDS.

We use the KDD-99 data set [35] for training and testing of our approach. The data set contains approximately 5 000 000 connection records. Each record in the data set is a network connection pattern, which is defined as a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol.

Every record is described by 41 features and labeled either as an attack or non-attack. Every connection record consists of about 100 bytes. Among these features, 34 are numeric and 7 are symbolic. For instance, the first one is the duration of connection time, the second is protocol type, and the third is service name, and so on.

The goal of IDS is to detect and recognize attacks. There are 22 types of attacks in KDD-99 data set. All the attacks fall into four main classes: DoS – denial of service attack. This attack leads to overloading or crashing of networks; U2R – unauthorized access to local super user privileges; R2L – unauthorized access from remote user; Probe – scanning and probing for getting confidential data.

Every class consists of different attack types (Smurf, Neptune, Buffer Overflow, etc.)

3.1 Intrusion Detection Based on Recirculation Neural Networks

In the following sections, the recirculation neural network (RNN) based detectors to construct intrusion detection systems are discussed. The fusion classifier built up of these detectors is introduced to perform detection and recognition of network attacks.

3.1.1 RNN Based Detectors

The Anomaly Detector

Recirculation neural networks (Figure 5) differ from others ANNs that on the input information in the same kind is reconstructed on an output. They are applied to compression and restoration of the information (direct and return distribution of the information in the networks «with a narrow throat») [36], for definition of outliers on a background of the general file of entrance data [37].

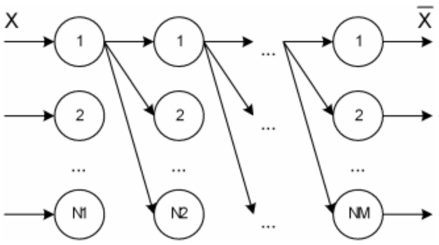


Fig. 5. *M* layers RNN structure N_i – quantity of neural elements in *i-th* layer, *NM=N1* – quantities of neural elements in entrance and target layers are equal

Nonlinear RNNs have shown good results as the detector of anomalies [38, 39]: training RNN is made 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:

$$E^k = \sum_j (\overline{X}_j^k - X_j^k)^2, \tag{2}$$

where X_j^k – j-th element of k-th input vector, $\overline{X}_j^k$ – 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. Thus there is a problem of a threshold T value determination, providing the most qualitative detection of abnormal connections. It is possible to get threshold value minimizing the sum of false positive (FP) and false negative (FN) errors, basing on cost characteristics of the given errors – FN error seems to be more expensive, than FP error, and its cost should be higher [39].

Private Classifiers

The described technique of definition of an input vector accessory to one of two classes – "normal" or "attacks", that is "not-normal" – it is possible to use in opposite way. If at training the detector of anomalies we used normal vectors which were restored in itself, and the conclusion about their accessory to a class "normal" was made, training the detector on vectors-attacks which should be restored in itself, it is possible to do a conclusion about their accessory to a class of "attack". Thus, if during functioning of this detector the reconstruction error (3) exceeds the certain threshold, given connection it is possible to carry to a class "not-attacks", that is normal connections. As training is conducted on vectors-attacks the given approach realizes technology of misuse detection, and its use together with previous technique is righteous.

Thus, one RNN can be applied to definition of an accessory of input vector to one of two classes – to on what it was trained (class A), or to the second (class $\overline{A}$), to which correspond outliers:

$$\begin{cases} X^k \in A, & if \quad E^k \le T, \\ X^k \in \overline{A}, & if \quad E^k > T. \end{cases} \tag{3}$$

Worth to note that it is possible to train RNN in the special way [39] on connections of both classes so that to raise quality of detection on conditions (4).

As already it was mentioned above, database KDD includes normal connections and also attacks of four classes which considerably differ from each other. Therefore it is advisable to train detectors for each of five classes separately, not uniting all classes of attacks in a single whole.

Here again there is a problem of a choice of a threshold *T* for each concrete detector. If for the anomaly detector it was possible to speak at once, that cost of FN error is higher, than cost of FP error, in case of the detector for a class of attacks R2L it is hard to tell what will be worse – FP error (that is to name "R2L" connection to this class not concerning – attack of other class or normal connection) or FN detection of the given attack (on the contrary).

Many researchers [40] use a cost matrix for definition of cost of errors F (Table 4). Average values of FP and FN errors (Table 5) for each class can be calculated as follows:

$$F_i^{FP} = \frac{\sum\limits_{j,\ j \neq i} F_{ji}}{N-1}, \quad F_i^{FN} = \frac{\sum\limits_{j,\ i \neq j} F_{ij}}{N-1}, \tag{4}$$

where *N* – quantity of classes *(N=5)*.

Table 4. The cost matrix F of incorrect classification of attacks and average costs of errors of detectors of each class

Real class	Cost of false prediction					Av. cost	
	normal	dos	probe	r2l	u2r	F_i^{FP}	F_i^{FN}
1 normal	0	2	1	2	2	2,5	1,75
2 dos	2	0	1	2	2	2	1,75
3 probe	1	2	0	2	2	1,5	1,75
4 r2l	4	2	2	0	2	2	2,5
5 u2r	3	2	2	2	0	2	2,25

On the basis of the given costs it is possible to choose value of a threshold which minimizes a total average error on training or validation data base.

Experimental Results

For an estimation of efficiency of the offered approach a number of experiments is lead. Private detectors for each class are trained, and all over again the training set got out of all base KDD, then from connections on concrete services – HTTP, FTP_DATA, TELNET. Nonlinear RNNs were used with one hidden layer with function of activation a hyperbolic tangent and logical sigmoid function of activation in a target layer. Quantity of neural elements in input and target layers according to quantity of parameters of input data – 41, in the hidden layer – 50. The training dataset contained 350 vectors of normalized values for each class. The RNNs were trained with layer-by-layer learning [36].

After each detector was trained the validation on training samples was conducted with the purpose of a finding of value of threshold *T* at which average cost of an error is minimal. In the further the testing of trained detectors was made on test samples with threshold values received before (Table 5). 10% of KDD database was used for testing purposes.

Table 5. Results of detectors testing

Service	Threshold	FP, %	FN, %	Av. cost	Service	Threshold	FP, %	FN, %	Av. cost
ALL					HTTP				
normal	0,00070	12,56	6,68	0,1844	normal	0,00123	6,8	2,72	0,0841
dos	0,00214	4,33	1,09	0,0542	dos	0,00340	0	0	0
probe	0,00120	7,79	14,21	0,1675	probe	0,00132	0	0	0
r2l	0,00116	2,87	5,38	0,0947	r2l	0,00114	5,17	0,25	0,0463
u2r	0,00112	7,07	5,54	0,1323	u2r	0,00126	0	0,07	0,0009
HTTP					TELNET				
normal	0,00620	2,4	0,17	0,0214	normal	0,00036	44,4	1,31	0,2394
dos	0,00290	1,5	0	0,0098	dos	0,00650	0	0	0
probe	0,00114	0	0	0	probe	0,00162	0	0	0
r2l	0,00110	0	0	0	r2l	0,00136	3,33	0	0,0294
					u2r	0,00076	5,91	2	0,0907

3.1.2 Fusion of Private Classifiers

Joint Functioning

As it was told above the best classification results can be achieved using several independent classifiers of the identical nature, because construction of the general estimation from private can be made by greater number of methods. We shall unite the private detectors trained in the previous section in one general (Figure 6).

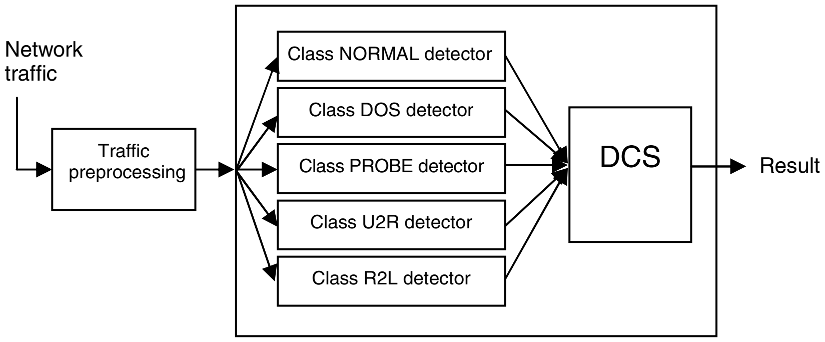


Fig. 6. Fusion of independent private classifiers in one general

The basic problem in construction of such a classifier becomes definition of a cumulative estimation proceeding from estimations of private detectors. In works of various researchers (for example [41]) the set of methods, such as a finding of average value for each class on the basis of indications of all classifiers, the sum of votes for each class, methods of an estimation «a priori» and «a posteriori» is considered. These methods mean that each classifier states a private estimation concerning an opportunity of an accessory of input image to at once several classes, and these classes are identical to all classifiers. However in our case classes, about an accessory to which each classifier judges, first, are various, secondly, are crossed. Therefore all the methods listed above are not applicable.

Dynamic Classifier Selection

The general classifier consists from *N=5* private detectors, each of which has a threshold T_i. Values of thresholds got out proceeding from minimization of average cost of errors. To make estimation values comparable it is enough to scale reconstruction error on a threshold. Then (4) will be:

$$\begin{cases} X^k \in A_i, & if \quad \delta_i^k \le 1, \\ X^k \in \overline{A}_i, & if \quad \delta_i^k > 1, \end{cases} \tag{5}$$

where $\delta_i^k = \frac{E_i^k}{T_i}$ - a relative reconstruction error. Thus, than less δ_i^k, the probability of accessory of an input image X^k to a class A_i is higher. Therefore it is possible to allocate the method of determination of a cumulative estimation – by the minimal relative reconstruction error:

$$\begin{cases} X^k \in A_m, \\ \delta_m^k = \min\limits_i \delta_i^k. \end{cases} \tag{6}$$

As the purpose of improvement of efficiency of classification is the minimization of erroneous classification expressed in minimization of average cost of classification, in construction of a cumulative estimation it is possible to act the same as at the choice of a threshold in private detectors – to consider cost of erroneous classification. If δ_i^k - a characteristic of probability of error of classification on *i-th* detector the estimation of possible average cost of error on each of detectors will be equal:

$$\Omega_i^k = \frac{\sum\limits_{j,j\neq i} \delta_i^k F_{ji}}{N-1}. \tag{7}$$

The estimation (8) shows, what ability of loss in cost if we shall name a vector belonging to *j-th* class by a vector of *i-th* class, i. e. *i-th* classifier instead of *j-th* will be chosen.

On the basis of the given estimation we shall allocate the second method of a cumulative estimation determination – on the minimal possible cost of false classification:

$$\begin{cases} X^k \in A_m, \\ \Omega_m^k = \min_i \Omega_i^k. \end{cases} \tag{8}$$

Besides it is possible to consider mutual influence of possible errors – to add up an estimation Ω_i^k and an estimation of a prize in cost if *i-th* classifier instead of wrong *j-th* will be chosen:

$$\Psi_i^k = -\frac{\sum_{j,j\neq i} (\delta_j^k - \delta_i^k) F_{ij}}{N-1}. \tag{9}$$

Then on the basis of estimations (8) and (10) it is possible to allocate the third rule of winner detector selection – on the minimal possible mutual cost of false classification:

$$\begin{cases} X^k \in A_m, \\ \Omega_m^k + \Psi_m^k = \min_i (\Omega_i^k + \Psi_i^k). \end{cases} \tag{10}$$

Experimental Results

Efficiency of the general classifier functioning we shall check up experimentally using the private detectors trained in section 3.1.1. Results are presented in Tables 6-8.

Table 6. Results of attack detection and recognition by fusion of classifiers with minimal relative reconstruction error DCS

Service	FP, %	FN, %	MC, %	Recognized, %			
				dos	probe	r2l	u2r
ALL	10,80	2,34	3,76	98,17	96,55	91,88	100
HTTP	0	0,08	0,25	99,75	100	100	–
FTP_DATA	0,66	1,09	1,45	100	100	96,66	100

Table 7. Results of attack detection and recognition by fusion of classifiers with the minimal possible cost of false classification DCS

Service	FP, %	FN, %	MC, %	Recognized, %			
				dos	probe	r2l	u2r
ALL	30,80	0,9	30,80	97,8	99,3	92,5	100
HTTP	0	0,08	0	99,8	100	0	–
FTP_DATA	0,70	1,06	0,70	100	100	96,7	100

Table 8. Results of attack detection and recognition by fusion of classifiers with the minimal possible mutual cost of false classification DCS

Service	FP, %	FN, %	MC, %	Recognized, %			
				dos	probe	r2l	u2r
ALL	18,8	0,7	18,8	98,3	98,0	93,1	98,2
HTTP	0	0,08	0	99,8	100	100	–
FTP_DATA	27,3	0,4	27,3	100	77,6	98,7	100

Apparently from results, the unequivocal answer to a question – which method is better – is not present. The method of a choice of a final class with use of mutual cost can minimize a error, but with substantial growth of quantity of false detection, methods with minimal relative reconstruction error and possible cost give basically comparable results, on some service one is better, on some – another.

3.2 Modular Neural Network Detectors

In the following sections, several modular neural network detectors to construct Intrusion Detection Systems (IDS) are discussed. They are based on the integration of different artificial neural networks each of which performs complex classification task. Each neural network is intended for carrying out a specific function in the system. The proposed approaches are results of evolution from a single neural network detectors to multi-agent systems [42, 43, 44].

3.2.1 Basic Element of Intrusion Detection System

Let's examine the basic neural element to construct Intrusion Detection Systems (Fig.7). As input data, the 41 features from KDD-99 dataset will be used, which contain the TCP-connection information. The main goal of IDS is to detect and recognize the type of attack. Therefore, 5-dimensional vectors will be used for output data, because the number of attack classes plus normal connection is five. We propose to use the integration of PCA (principal component analysis neural network) and MLP (multilayer perceptron) as for basic element of IDS. We will name it the first variant of IDS.

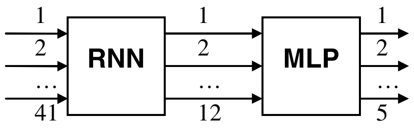


Fig. 7. The first variant of IDS (Model 1)

The PCA network, which is also called a recirculation network (RNN), transforms 41-dimensional input vectors into 12-dimensional output vectors. The MLP processes those given compressed data to recognize type of attacks or normal transactions.

In this section we present two neural networks based on principal component analyses techniques, namely linear and nonlinear RNN networks.

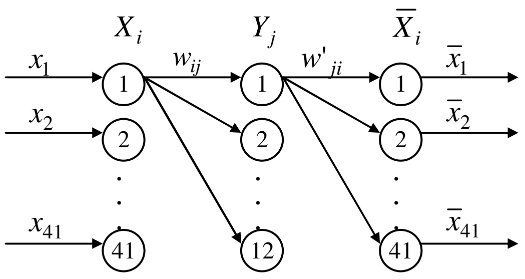


Fig. 8. RNN architecture

Let's consider an auto-encoder, which is also called a recirculation neural network (see Fig. 8). It is represented by MLP, which performs the linear or nonlinear compression of the dataset through a bottleneck in the hidden layer. As shown in the figure, the nodes are partitioned into three layers. The bottleneck layer performs the compression of the input dataset. The output of the *j*-th hidden unit is given by

$$y_j = F(S_j), \tag{11}$$

$$S_j = \sum_{i=1}^{41} w_{ij} \cdot x_i, \tag{12}$$

where F is activation function; S_j is weighted sum of the output from *j*-th neuron; w_{ij} is the weight from the *i*-th input unit to the *j*-th hidden unit; x_i is the input to the *i*-th unit.

The output from the *i*-th unit is given by

$$x_i = F(S_i), \tag{13}$$

$$S_i = \sum_{i=1}^{12} w'_{ji} \cdot y_j. \tag{14}$$

We use two algorithms for RNN training. One is the linear Oja rule and the other is the backpropagation algorithm for nonlinear RNN.

The weights of the linear RNN are updated iteratively in accordance with the Oja rule [45]:

$$w'_{ji}(t+1) = w'_{ji}(t) + \alpha \cdot y_j \cdot (x_i - \overline{x}_i),$$
$$w_{ij} = w'_{ji}. \tag{15}$$

Such a RNN is known to perform a linear dimensionality reduction. In this procedure, the input space is rotated in such a way that the output values are as uncorrelated as possible and the energy or variances of the data is mainly concentrated in a few first principal components.

As already mentioned, the backpropagation approach is used for training nonlinear RNN. The weights are updated iteratively in accordance with the following rule:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot F'(S_j) \cdot x_i, \tag{16}$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \cdot y_j \cdot F'(S_i)(\overline{x_i} - x_i) \tag{17}$$

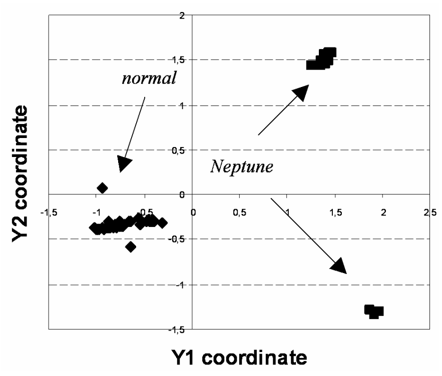
where γ_j is error of *j*-th neuron:

$$\gamma_j = \sum_{i=1}^{n} (\overline{x_i} - x_i) \cdot F'(S_i) \cdot w'_{ji}. \tag{18}$$

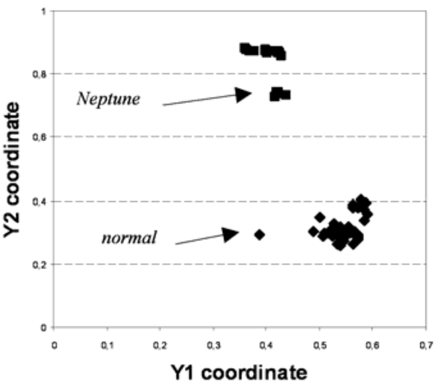
The weights data in the hidden layer must be re-orthonormalized by using the Gram-Schmidt procedure [44].

Let's consider the mapping of input space data for the normal state and Neptune type of attack on the plane of the two first principal components. As we can see in Fig. 9(a), the data which belong to one type of attack can be located in different areas. The visualization of such data obtained by using only linear RNN will not be satisfactory because of complex relationship between the features. One of the ways to solve this problem is to use the nonlinear RNN network.

As we can see in Fig. 9(b), the nonlinear RNN performs better in visualizing dataset in comparison with a linear RNN.



a) linear RNN



b) nonlinear RNN

Fig. 9. Data processed with: a – linear RNN, b – nonlinear RNN

There is a problem in Principal Components Analysis (PCA). We do not know the number of principal components.

Table 9. Recognition Rates for Some Set of Samples Depending on Number of Principal Components

Number of principal components	2	4	5	7	10	12	15	20	41
Recognition rates	39,24%	47,15%	71,84%	78,16%	95,25%	95,70%	96,84%	96,52%	96,84%

We have tried several neural network classifiers with different number of principal components, and analyzed the results of recognition by choosing the number of principal components which gave the best performance in efficiency in each of those classification model.

Our experiments (see Table 9) show that the optimal number of principal components lies near 12.

As already mentioned, the MLP is intended to classify attacks on the basis of components, which are obtained by using RNN. The number of output units depends on number of attack classes. The backpropagation algorithm is used for training MLP. After training of neural networks they are combined together for an intrusion detection system.

3.2.2 Generation of Different Intrusion Detection Structures

Using the results presented in the section 3.2.1, we can suggest several neural network classification models for development of intrusion detection systems.

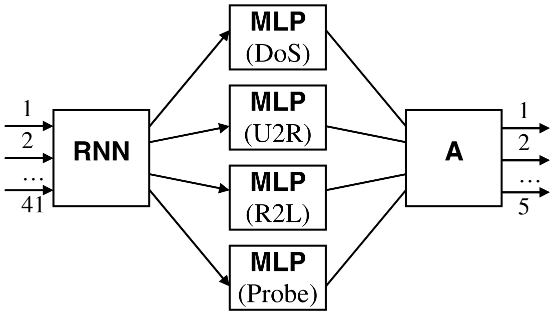


Fig. 10. The second variant of IDS (Model 2)

The second variant of IDS structure is shown in Fig. 10. It consists of four MLP networks. As we can see, every MLP network is intended to recognize the class of attack, that is, DoS, U2R, R2L or Probe. The output data from 4 multilayer perceptrons enter the Arbiter, which accepts the final decision according to the class of attack. A one-layer perceptron can be used as the Arbiter. The training of the Arbiter is performed after leaning of RNN and MLP neural networks. This approach enables to make a hierarchical classification of attacks. In this case, the Arbiter can distinguish one of the 5 attack classes by the corresponding MLP.

Complex computational problems can be solved by dividing them into a number of small and simple tasks. Then the results of each task are integrated for a general conclusion. An appropriate simplicity is achieved by distributing those training tasks to several *experts*. The combination of such experts is known as *Committee Machine*. This integrated knowledge has priority over the opinion taken separately from each expert. We have prepared two modular neural networks for the purpose of intrusion detection.

The third variant of IDS is based on this idea, and is shown in Fig. 11. Expert is represented by a single classification system. We use basic intrusion detection system as an expert (see Fig. 7). Training data sets for each expert are not the same with each other. They are self-organizing during the training process as a result of classification performed by the previous experts. The rule that was chosen for this purpose is Boosting by filtering algorithm [46]. After training, the neural networks have an ability to detect intrusions. In testing mode, every expert is intended for processing the original 41-demensional vectors. The Arbiter performs vote functions and accepts the final joint resolution of three experts. Arbiter is represented by the two-layer perceptron.

1. Train a first expert network using some training set;
2. A training set for a second expert is obtained in the following manner:
   - Toss a fair coin to select a 50% from NEW training set and add this data to the training set for the second expert network;
   - Train the second expert;
3. A third expert is obtained in the following way:
   - pass NEW data through the first two expert networks. If the two experts disagree, then add this data to the training set for the third expert:
   - Train the third expert network.
4. Vote to select output.

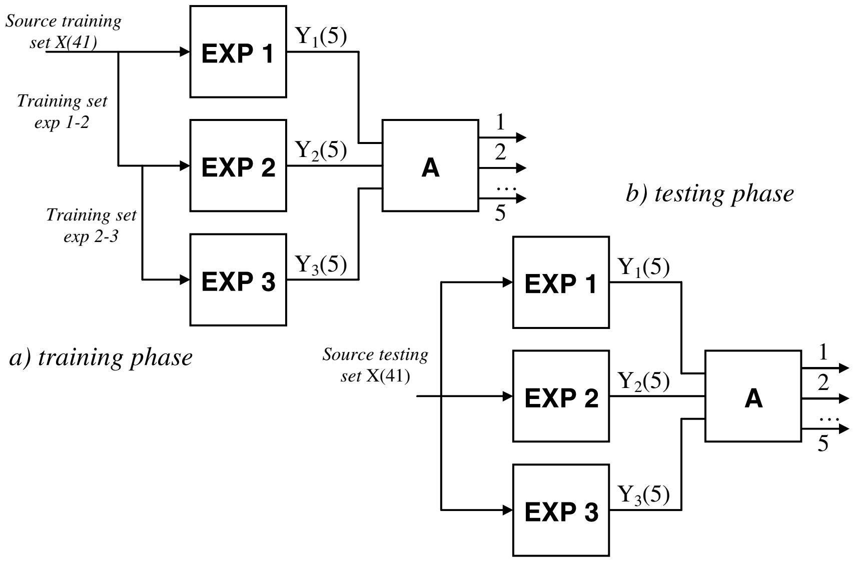


Fig. 11. The third variant of IDS, based on boosting by filtering algorithm (Model 3)

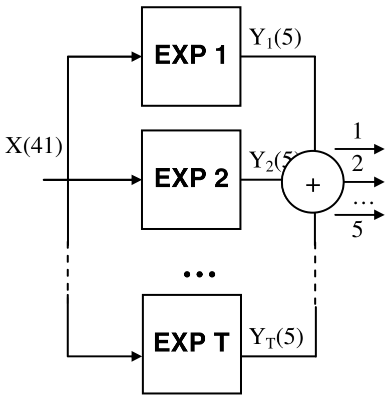


Fig. 12. The fourth variant of IDS, based on AdaBoost algorithm (Model 4)

In the case of AdaBoost algorithm [47] (Fig. 12), Summator performs the functions of the Arbiter. This analog of the Arbiter generates the result of the voting by summarizing private decisions.

3.2.3 Experimental Results

To assess the effectiveness of proposed intrusion detection approaches, a series of experiments were performed. The KDD99 cup network data set was used for training and testing different neural network models, because it is one of only a few publicly available data set of intrusion detection that attracts the researchers' attention due to its well-defined nature.

So we used 10% data selected from KDD dataset (almost 500000 records) to generate a subset for training and testing afterwards. To be more specific, we used 6186 samples for training neural networks, and used all records for testing the system (see Table 10).

Table 10. Training and Testing Samples

	DoS	U2R	R2L	Probe	Normal	Total count
training samples	3571	37	278	800	1500	6186
testing samples	391458	52	1126	4107	97277	494020

The same data sets were applied for model 1 and model 2 as well, so that we can compare the performance of those proposed models here. The approaches proposed are designed to detect 5 classes of attacks from this dataset which includes DoS, U2R, R2L, Probe and Normal.

To evaluate our system, we used three major indicators, that is, the detection rate and recognition rate for each attack class and false positive rate. The detection rate (true attack alarms) is defined as the number of intrusion instances detected by the system divided by the total number of intrusion instances in the test set. The recognition rate is defined in a similar manner. The false positive rate (false attack alarms) represents the total number of normal instances that were classified as intrusions divided by the total number of normal instances.

Let’s examine the recognition of attacks using the model 1. This model is quite simple. Table 11 shows statistics of recognition depending on attack class.

Table 11. Attack Classification with Model 1

class	count	detected	recognized
DoS	391458	391441 (99.99%)	370741 (94.71%)
U2R	52	48 (92.31%)	42 (80.77%)
R2L	1126	1113 (98.85%)	658 (58.44%)
Probe	4107	4094 (99.68%)	4081 (99.37%)
Normal	97277	---	50831 (52.25%)

The above results show that the best detection rate and recognition rates were achieved for attacks by DoS and Probe connection. U2R and R2L attack instances were detected a bit worse (80.77% and 58.44%, respectively). Besides, the bottom row in Table 11 shows that some normal instances were (incorrectly) classified as intrusions.

The number of false positives emerged from the first model is considerable. This can be corrected by the second model described above. As shown in table 12, the second model performed quite well in terms of false positives. This is due to the four single multilayer perceptrons corresponding to each of the four attack classes.

Table 12. Attack Classification with Model 2

class	count	detected	recognized
DoS	391458	391063 (99.90%)	370544 (94.66%)
U2R	52	49 (94.23%)	37 (71.15%)
R2L	1126	1088 (96.63%)	1075 (95.47%)
Probe	4107	3749 (91.28%)	3735 (90.94%)
Normal	97277	---	83879 (86.22%)

As mentioned above, each expert in Model 3 and Model 4 is represented by a single classification system. We use model 1 as an expert in the experiments here as shown in Table 13 and 14. But every subsequent expert influences the outputs of other performing aggregated opinions of the several neural networks.

Table 13. Attack Classification with Model 3

class	count	detected	recognized
DoS	391458	391443 (99.99%)	370663 (94.69%)
U2R	52	50 (96.15%)	42 (80.76%)
R2L	1126	1102 (97.87%)	1086 (96.45%)
Probe	4107	3954 (96.27%)	3939 (95.91%)
Normal	97277	---	84728 (87.09%)

Table 14. Attack Classification with Model 4

class	count	detected	recognized
DoS	391458	389917 (99.61%)	369088 (94.29%)
U2R	52	51 (98.08%)	44 (84.62%)
R2L	1126	1119 (99.37%)	636 (56.48%)
Probe	4107	3908 (95.15%)	3668 (89.31%)
Normal	97277	---	77212 (79.37%)

The total results of the detection rates and false positive rates related with each model are shown in Table 15.

Table 15. Total Results for each Model

model	True attack alarms	False attack alarms	Recognized correctly	Total recognized %
Model 1	396696 (99.98%)	46446 (47.75%)	375522 (94,65%)	86.30%
Model 2	395949 (99.80%)	13398 (13.77%)	375391 (94.61%)	92.97%
Model 3	396549 (99.95%)	12549 (12.90%)	375730 (94.70%)	93.21%
Model 4	394995 (99.56%)	20065 (20.62%)	373436 (94.13%)	91.22%

In general, model 3 is shown to achieve the lowest false positive rate and the highest accuracy (93.21%). In fact, it is more accurate than the model 2 (92.97%) and the model 4 (91.22%). So, the three last models can be effectively used for the classification of huge input data set with a complicated structure.

3.2.4 Multiagent Neural Networks

Multiagent neural networks use several detectors that specialize different fields of knowledge.

In our work artificial immune system has been exploited for a development of multiagent IDS. Several important questions that strongly influence the efficiency of the model arise in the course of designing multiagent structures: obtaining of the generalized decision on the basis of the set of detector opinions, selection of detectors, cloning and mutation, destruction of bad and/or irrelevant detectors.

First of all, it is necessary to define what we will use as a detector to classify attacks. As shown in Fig. 13, we offer the model slightly modified the model proposed in the previous sections. See section 3.2.1 for more in detail.

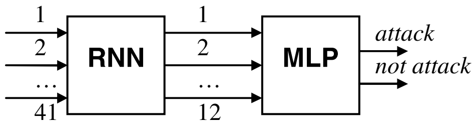


Fig. 13. Modified detector for immune system construction

Each detector is represented by artificial neural network consisted of recirculation neural network and multilayer perceptron, which functions were already discussed above. Such a detector specializes certain type of attack. There are two output values "yes" (when the entrance pattern relates to the given type of attack) and "no" (when the entrance pattern is not attack of the considered type).

The detectors, which represent the same type of attack, are combined in groups from 3 to 10. Generally, all the detectors in the group give the diverse conclusions which is the results of casual processes during the training. Theoretically, the number of detectors in the system is not limited and their number can be easily varied, but in real world problems with computational resources such as operative memory, speed etc..., arise.

Recognition process of an entrance pattern consists of the following sequence of steps:

1. Input pattern is transmitted to the multiagent system.
2. Each detector gives a conclusion about entrance activity.
3. So-called *factor of reliability on each group of the detectors* is formed. This factor reflects percent of voices in the group, given for the type of attack the group is specialized in.
4. The analysis of factors of reliability, obtained from each group, is carried out. A decision of the group with the maximum value of the factor is considered to be the final decision.

The obvious advantage of such an approach is, (i) Training process is made comparatively easily; (ii) Detectors are trained on a smaller number of samples than models considered in the previous sections; (iii) It allows to increase quality of their training and to considerably reduce time spent for preparation of the next detector.

Let's consider how such a multiagent system work from an example of a population of detectors. The population consists of 110 detectors (5 detectors in a group for each attack type from the KDD99 dataset). The results were prepared in the same way as the models in the previous sections (Table 16) so that we can compare them. As we can see, the results are similar to each other.

Table 16. Attack Classification with the Multiagent Neural Network

class	count	detected	Recognized
DoS	391458	383953 (98.08%)	368779 (94.21%)
U2R	52	47 (90.39%)	46 (88.46%)
R2L	1126	1122 (99.67%)	359 (31.88%)
Probe	4107	4105 (99.95%)	2369 (57.68%)
Normal	97277	---	75538 (77.65%)

The second experiment is related with the recognition of new attacks. For this purpose, we prepared a special set of samples for testing and training. The testing samples consist of network connection records that represent some of the most popular network services taken from the KDD99 dataset (http, ftp, ftp data, smtp, pop3, telnet). As dataset for testing, we generated a considerably reducing number of samples for each attack type. Also what is necessary to draw attention is that the records of some scanty attack types were entirely excluded from the training set. Therefore, only 9 types of attacks have been selected here. Accordingly, 9 groups (5 detectors in each) have been generated. So, the quantity of the population has made up 45 detectors.

Table 17. New Attack Detection with the Multiagent Neural Network

type	count	detected	type	count	detected
Normal	75952	71338 (93.93%)	Multihop*	7	7 (100.00%)
Land*	1	1 (100.00%)	Phf*	4	0 (0.00%)
Neptune	901	901 (100.00%)	Spy*	2	1 (50.00%)
Buffer_overflow	30	30 (100.00%)	Warezclient	1015	1003 (98.82%)
Loadmodule	9	9 (100.00%)	Warezmaster	20	20 (100.00%)
Perl*	3	1 (33.33%)	Ipsweep	9	9 (100.00%)
Rootkit*	7	3 (42.86%)	Nmap*	2	2 (100.00%)
ftp_write*	6	6 (100.00%)	Portsweep	15	15 (100.00%)
guess_passwd	53	53 (100.00%)	Satan	10	9 (90.00%)

* - *the attacks that were absent in the training set.*

The results shown in Table 17 show a lot of records corresponding to new attacks were detected and classified as an “attack”. It means that multiagent systems are capable of detecting new attacks and have high generalization capacity.

We have discussed only the prototype of one population. Nevertheless, the results are promising due to the fact that many unknown records were detected. Extension of the proposed approach based on multiagent neural networks with the basic mechanisms of immune system (which exploits selection, mutation, cloning, etc.) will allow us to build a real time intrusion detection system.

4 Conclusion

In this chapter the artificial immune systems and neural network techniques for computer viruses and intrusion detection have been addressed. The AIS allow detecting unknown computer viruses. Integration of AIS and neural networks permits to increase performance of the security system. The IDS structure is based on integration of the different neural networks. As a result fusion classifier, modular neural networks and multiagent systems were proposed. The KDD-99 dataset was used for experiments performing. Experimental results show that the neural intrusion detection system has possibilities for detection and recognition computer attacks.

Proposed techniques have been shown powerful tools with respect to conventional approaches.

References

[1] de Castro, L.N., Timmis, J.I.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)

[2] Janeway, C.A.: How the Immune System Recognizers Invaders. Scientific American 269(3), 72–79 (1993)

[3] Dasgupta, D.: Artificial immune systems and their applications. Springer, New York (1999)

[4] Computer virus, `http://en.wikipedia.org/wiki/Computer_virus`

[5] Traditional antivirus solutions – are they effective against today's threats? (2008), `http://www.viruslist.com`

[6] Proactive protection: a panacea for Viruses? (2008), `http://www.viruslist.com`

[7] de Castro, L.N., Timmis, J.I.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)

[8] Janeway, C.A.: How the Immune System Recognizers Invaders. Scientific American 269(3), 72–79 (1993)

[9] Handbook of neural network processing. CRC Press LLC, Boca Raton (2002)

[10] Ezhov, A., Shumsky, S.: Neurocomputing and its application in economics and business, Moscow, MIPHI (1998)

[11] Ayara, M., Timmis, J., de Lemos, L., de Castro, R., Duncan, R.: Negative selection: How to generate detectors. In: Timmis, J., Bentley, P.J. (eds.) Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS), pp. 89–98. University of Kent at Canterbury Printing Unit, Canterbury (2002)

[12] Forrest, S., Hofmeyr, S.A.: Immunology as information processing. In: Segel, L.A., Cohen, I. (eds.) Design principles for the immune system and other distributed autonomous systems, Oxford University Press, New York (2000)

[13] Jerne, N.K.: Clonal Selection in a Lymphocyte Network, pp. 39–48. Raven Press (1974)

[14] Bezobrazov, S., Golovko, V.: Neural Networks for Artificial Immune Systems: LVQ for Detectors Construction. In: Proceedings of the IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2007), Dortmund, Germany (2007)

[15] Forest, S., Perelson, F., Allen, L., Cherukuri, R.: Self-Nonself Discrimination in a Computer. In: Proceedings IEEE Symposium on Research in Security and Privacy, pp. 202–212. IEEE Computer Society Press, Los Alamitos (1994)

[16] Balthrop, J., Esponda, F., Forrest, S., Glickman, M.: Coverage and Generalization in an Artificial Immune System. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 3–10. Morgan Kaufmann Publishers, San Francisco (2002)
[17] Hofmeyr, S., Forrest, S.: Architecture for an artificial immune system. EvolutionaryComputation 8(4), 443–473 (2000)
[18] Hofmeyr, S.A.: An interpretative introduction to the immune system. In: Cohen, I., Segel, L. (eds.) Design principles for the immune system and other distributed autonomous systems, Oxford University Press, New York (2000)
[19] Kohonen, T.: Self-organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 43, 59–69 (1982)
[20] Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design, 1st edn. PWS Pub. Co. (1995)
[21] Golovko, V.: Neural networks: training, organization and application, Moscow, IPRZHR (2001)
[22] Kaspersky Lab: Antivirus software (2008), `http://www.kaspersky.com`
[23] ESET NOD32 antivirus software (2008), `http://www.eset.com`
[24] Kumar, S., Spafford, E.H.: A Software architecture to support misuse intrusion detection. In: Proceedings of the 18th National Information Security Conference, pp. 194–204 (1995)
[25] Ilgun, K., Kemmerer, R.A., Porras, P.A.: State transition analysis: A rule-based intrusion detection approach. IEEE Transaction on Software Engineering 21(3), 181–199 (1995)
[26] SNORT, `http://www.snort.org`
[27] Lunt, T., Tamaru, A., Gilham, F., et al.: A Real-time Intrusion Detection Expert System (IDES) – final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California (February 1992)
[28] Porras, P.A., Neumann, P.G.: EMERALD: Event monitoring enabling responses to anomalous live disturbances. In: Proceedings of National Information Systems Security Conference, Baltimore, MD (October 1997)
[29] Denning, D.E.: An intrusion-detection model. IEEE Transaction on Software Engineering 13(2), 222–232 (1987)
[30] Lee, W., Stolfo, S., Mok, K.: A data mining framework for adaptive intrusion detection. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy, Los Alamos, CA, pp. 120–132 (1999)
[31] Lee, W., Stolfo, S.: A Framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security 3(4), 227–261 (2000)
[32] Liu, Y., Chen, K., Liao, X., et al.: A genetic clustering method for intrusion detection. Pattern Recognition 37(5), 927–934 (2004)
[33] Eskin, E., Rnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A Geometric framework for unsupervised anomaly detection. In: Applications of Data Mining in Computer Security. Kluwer Academics, Dordrecht (2002)
[34] Shyu, M., Chen, S., Sarinnapakorn, K., Chang, L.: A Novel Anomaly Detection Scheme Based on Principal Component Classifier. In: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM 2003), pp. 172–179 (2003)
[35] Kayacik, H., Zincir-Heywood, A., Heywood, M.: On the capability of an SOM based intrusion detection system. In: Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN 2003), pp. 1808–1813 (2003)

[36] Zhang, Z., Li, J., Manikopoulos, C.N., Jorgenson, J., Ucles, J.: HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In: Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, pp. 85–90 (2001)

[37] 1999 KDD Cup Competition, `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

[38] Golovko, V., Ignatiuk, O., Savitsky, Y., Laopoulos, T., Sachenko, A., Grandinetti, L.: Unsupervised learning for dimensionality reduction. In: Proc. of Second Int. ICSC Symposium on Engineering of Intelligent Systems EIS 2000, University of Paisley, Scotland, pp. 140–144. ICSS Academic Press, Canada (2000)

[39] Hawkins, S., He, H., Williams, G., Baxter, R.: Outlier Detection Using Replicator Neural Networks. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 170–180. Springer, Heidelberg (2002)

[40] Golovko, V., Kochurko, P.: Some Aspects of Neural Network: Approach for Intrusion Detection. In: Kowalik, Janusz, S., Gorski, J., Sachenko, A. (eds.) Cyberspace Security and Defense: Research Issues. NATO Science Series II: Mathematics, Physics and Chemistry, vol. 196, pp. 367–382. Springer, Heidelberg (2005); VIII, p. 382

[41] Kochurko, P., Golovko, V.: Neural Network Approach to Anomaly Detection Improvement. In: Proc. of 8th International Conference on Pattern Recognition and Information Processing (PRIP 2005), Minsk, Belarus, May18-20, pp. 416–419 (2005)

[42] Giacinto, G., Roli, F., Didaci, L.: Fusion of multiple classifiers for intrusion detection in computer networks. Pattern Recognition Letters 24, 1795–1803 (2003)

[43] Giacinto, G., Roli, F., Fumera, G.: Selection of image classifier. Electron 26(5), 420–422 (2000)

[44] Golovko, V., Vaitsekhovich, L.: Neural Network Techniques for Intrusion Detection. In: Proceedings of the International Conference on Neural Networks and Artificial Intelligence (ICNNAI 2006), Brest State Technical University - Brest, pp. 65–69 (2006)

[45] Golovko, V., Kachurka, P., Vaitsekhovich, L.: Neural Network Ensembles for Intrusion Detection. In: Proceedings of the 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2007), Research Institute of Intelligent Computer Systems, Ternopil National Economic University and University of Applied Sciences Fachhochschule Dortmund - Dortmund, Germany, pp. 578–583 (2007)

[46] Golovko, V., Vaitsekhovich, L., Kochurko, P., Rubanau, U.: Dimensionality Reduction and Attack Recognition using Neural Network Approaches. In: Proceedings of the Joint Conference on Neural Networks (IJCNN 2007), Orlando, FL, USA, pp. 2734–2739. IEEE Computer Society, Los Alamitos (2007)

[47] Oja, E.: Principal components, minor components and linear networks. Neural Networks 5, 927–935 (1992)

[48] Drucker, H., Schapire, R., Simard, P.: Improving performance in neural networks using a boosting algorithm. In: Hanson, S.J., Cowan, J.D., Giles, C.L. (eds.) Advanced in Neural Information Processing Systems, Denver, CO, vol. 5, pp. 42–49. Morgan Kaufmann, San Mateo (1993)

[49] Freund, Y., Schapire, R.E.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence 14(5), 771–780 (1999)