Neural Network Artificial Immune System for Malicious Code Detection

Vladimir Golovko, Sergei Bezobrazov

Brest State Technical University
Moskovskaja str. 267, Brest, 224017, Republic of Belarus
gva@bstu.by, bescase@gmail.com

Abstract

In this paper we present the intelligent adaptive self-learning and self-organizing system for malicious code detection based on integration of the Artificial Immune Systems and the Artificial Neural Networks. Such a system works according to basic principles of the artificial immune system where immune detectors present neural network and detect a malicious pattern by means of the analysis the structure of the executable code. As a result the proposed system is capable to adapt to the continually changeable computer environment and detect not only known but unknown malicious code, which does not belong to training data set. The purpose of this paper is to present the key ideas and approaches underlying our research in this area.

Keywords. Artificial immune systems, artificial neural networks, malicious code detection, malware, viruses

1 Introduction

Nowadays modern society faces the problem of information security from malicious software. And if already known malware is not so dangerous for the computer users, because the signature analysis method of malware detection successfully cope with such problems, the unknown malicious software may cause a serious security threat. Antivirus experts should detect the unknown malware, investigate it and give the solution for the computer users to secure protection from this malicious code. Though, the computer may be infected with this malware before it has been detected by the antivirus industry. The traditional proactive methods of unknown malware detection based on heuristic analysis don't provide with proper level of computer system defense. The best modern antivirus software obtains 90% of malicious code detection by the methods of reactive defense (already known malware detection) and 70% of malicious code detection by the methods of proactive defense (unknown malware detection). Figure 1 shows the levels of reactive and proactive detection) [1].

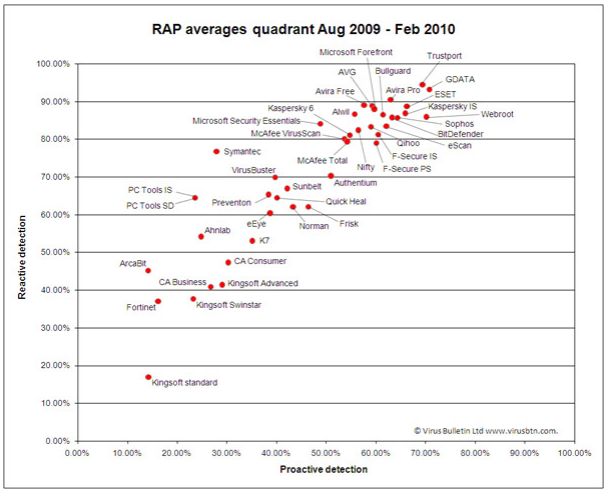

Fig. 1. Reactive and proactive detection.

The test results confirm an existence of malicious code detection problem, and this problem is very important. Recently various techniques of anomaly detection and unknown malware detection have been proposed [2], [3], [4], [5]. Artificial immune systems (AIS) have the particular place among the methods of artificial intelligence for anomaly detection [6], [7]. Unfortunately, many researchers use the AIS only as a metaphor. Different researchers either implement only parts of AIS mechanisms or don't use them at all that reduce the ability such systems to adaptation, self organization and self-learning [8], [9], [10], [4]. Also, most of scientific works don't focus on solving the problem of malicious code detection on the whole. They try to decide only the part of this problem namely viruses (infectors) or worms detection [9], [11], [12].

At present time the most real antivirus software based on signature analysis. Signature-based approach have acceptable detection rate for known virus and relatively low false positives. Unfortunately the ability of signature-based system to detect new viruses is extremely poor.

The artificial immune system (AIS) is an approach inspired by biological immune systems. It can be defined like computational system based on ideas

biological immune system. There exist different models of AIS for malicious code detection (L.N. de Castro 2002, S. Hofmeyr 2000, Janeway 1993). Unfortunately there are some problems to use this approach for malware detection. As a rule AIS models use of binary or real strings structure of detectors. In this case it is difficult to train such detectors for qualitative malware detection. As a result such systems have high computational complexity and nonwell ability for novel malware detection. The other authors propose to combine the neural networks and artificial immune systems for network intrusion or anomaly detection [13]. However they use neural network separately from AIS, as a rule only on final stage like classifier. To overcome these problems we propose an approach the use of neural networks in AIS for malicious code detection.

The key idea of this paper is to integrate of advantages of artificial immune systems and neural networks for creating intelligent adaptive self organizing system for malicious code detection and recognition. Such a system should have ability of novel malware detection and low false positive and false negative rates.

The paper is structured as follows. Section 2 describes the neural network artificial immune system. Section 3 presents the structure of neural network immune detector. In section 4 the performance of immune detector is proposed. In section 5 the experimental results of malicious code detection are given. Section 6 concludes this paper and point out our future work.

2 Neural network artificial immune system

Let consider the main principles of artificial immune system (AIS) development, using neural networks. First of all we will define the environment in which the AIS will exist. The environment includes computer like protected system, which contains as uninfected (self) and virus (nonself) files.

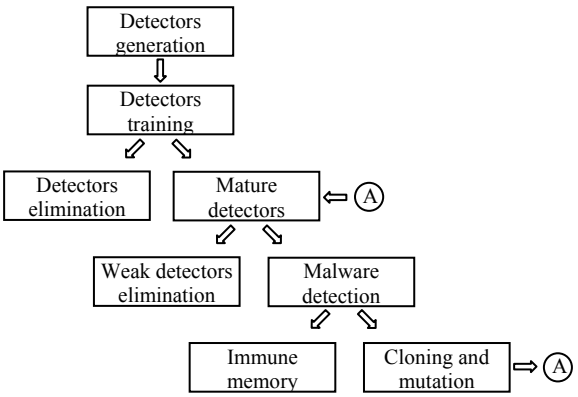

Figure 1: Model of the AIS.

The proposed AIS (Figure 1) consists of the following blocks: block of detectors generation, block of detectors training, block of detectors selection, block of detectors elimination, block of infected files detection, block of detectors cloning and mutation, block of immune memory creation. The immune detectors play a main role in malicious code detection and the architecture of detectors is significant for successful detection. The computer system is changeable environment. The new software is installed and uninstalled continually. Therefore the security system should correct identify legitimate software and detect malicious code both already known and novel.

Let define a detection unit as a component of artificial immune system for malware detecting in a computer system. As a detection unit we will use neural network. Then artificial immune system will consist of certain number of neural networks, which forms the population of immune detectors. Primarily we will present the neural network immune detector as a black box, which has *n-inputs* and two outputs (Figure 2).

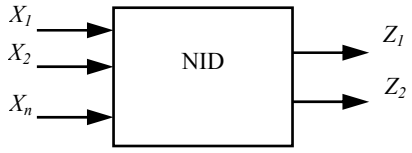

Figure 2: The neuronet immune detector.

The outputs of detector after presentation of all checking data can be obtained in accordance with the following expressions:

$$Z_1 = \begin{cases} 1, & \textit{if legitimate file} \\ 0, & \textit{otherwise.} \end{cases}$$
$$Z_2 = \begin{cases} 1, & \textit{if malicious file} \\ 0, & \textit{otherwise.} \end{cases} \quad (1)$$

The training data set consist of legitimate and malicious files. Of course, the immune detectors will be more diverse, if the more various files are presented at the training data set. It is desirable to have also representatives of all types of malware, namely worms, Trojan programs, macro viruses etc.

The neural network is trained by supervisor. Figure 3 illustrates the input samples for neural network training.

The files are selected from utilities of operating system Microsoft Windows for generation uninfected samples as it is shown in Fig.3 (dwwin.exe, regedit.exe, taskman.exe, autoras.exe). The computer viruses are used for generation malicious samples, for instance *lovesan.vir* in figure 3.

Let T is set of legitimate files and F is set of malicious files. Using these files the set of input samples are generated in a random way, which are applied for training of i th detector (expression 2).

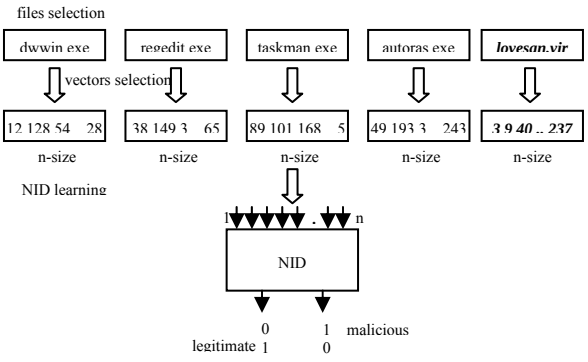

Figure 3: The data set for neural network training.

Accordingly, the set of desired output samples can be written in accordance with expression (3), where L is dimension of training set.

The desired output samples for i th detector are formed using expression (4).

$$X_i = \begin{bmatrix} X_i^1 \\ X_i^2 \\ \dots \\ X_i^L \end{bmatrix} = \begin{bmatrix} X_{i1}^1 & X_{i2}^1 & \dots & X_{in}^1 \\ X_{i1}^2 & X_{i2}^2 & \dots & X_{in}^2 \\ \dots & & & \\ X_{i1}^L & X_{i2}^L & \dots & X_{in}^L \end{bmatrix} \quad (2)$$

$$l_i = \begin{bmatrix} l_i^1 \\ l_i^2 \\ \dots \\ l_i^L \end{bmatrix} = \begin{bmatrix} l_{i1}^1 & l_{i2}^1 \\ l_{i1}^2 & l_{i2}^2 \\ \dots & \\ l_{i1}^L & l_{i2}^L \end{bmatrix} \quad (3)$$

$$l_{i1}^k = \begin{cases} 1, & if\ X_i^k \in T \\ 0, & otherwise. \end{cases}$$
$$l_{i2}^k = \begin{cases} 1, & if\ X_i^k \in F \\ 0, & oteherwise. \end{cases} \quad (4)$$

The detectors are trained to classify normal and abnormal patterns. The purpose of the training of each detector is to minimize the total mean square error. The total mean square error for i-th detector is defined as:

$$E_i = \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{2}(Z_{ij}^k - l_{ij}^k)^2, \quad (5)$$

where Z_{ij}^k is j-th output unit of i-th detector for k-th pattern.

The total mean square error characterizes the detector fitness to recognize malicious code. The small value of mean square error means the better fitness of the detector. Therefore the mean square error evaluates the quality of detector and can be used for selection of the best detectors.

The set of trained neural networks form population of immune detectors which circulate (live) in computer system and perform detection of malicious code. Each detector has an assigned lifetime that is decreased at each an iteration of algorithm presented below. If the detector reaches the maturity age, it will be eliminated.

The use of the various files for neural networks training and random generation of input vectors permit to increase diversity of detector population.

The neural network AIS algorithm aims at building such an adaptive system that can detect novel malware patterns for which no signature exists.

The procedure of building and performance of neural network immune system can be represented as follows:

1. Generate an initial population of detectors. It should be noted that each detector represents the neural network with random weights:

$$D = \left\{ D_i, \quad i = \overline{1,r} \right\}, \quad (6)$$

where D_i is i-th neural network immune detector, r is the number of detectors.

2. Train neural network immune detectors. Training data set are generated by random way from legitimate and malicious files or their signatures. The desired output units are obtained in accordance with equation (4). After the training the certain amount of detectors are obtained, which are used in the testing stage.

3. Select the best neural network detectors using test data set. The goal of this process is to eliminate bad (unsuitable) detectors, which have insufficient ability to training and false positive rate. Each detector is verified using test data set, which consists of legitimate files. As a result for each detector the total mean square error E_i is determined in accordance with equation 5. Detectors would be selected with zero mean square error:

$$D_i = \begin{cases} 0, & if\ E_i \neq 0 \\ D_i, & otherwise. \end{cases} \quad (7)$$

where *0* characterizes deletion of detector.

4. Each detector get lifetime and randomly choose the checking file from all files of computer, which detector did not inspect.

5. Scanning by each detector of the chosen file. As a result output values of detectors Z_{i1}, Z_{i2}, where $i=1...r$, are defined.

6. If the i-th detector does not detect malware in scanning file, i.e. $Z_{i1}=1$ и $Z_{i2}=0$, then it choose next file for inspection. If the lifetime of a detector is ended, it is eliminated from the detectors set and new detector is created.

7. If i-th detector detect malware in file, i.e. $Z_{i1}=0$ и

$Z_{i2}=1$, then it activates alarm. In this case cloning and mutation of given detector is performed. As a result the set of clones are generated and each clone is trained by using detected infected file (mutation). Finally we can get the set of clones, which are aimed to detect given virus

$$D_i = (D_{i1}, D_{i2}, \dots, D_{in}). \quad (8)$$

8. Select the best clone detectors, which are most fitness to detect this malicious cod. The mean square error for each clone is calculated, using detected virus file. IF $E_{ij} > E_i$, then detector has passed selection. Here E_{ij} – mean square error for j-th clone of i-th detector.

9. The set of clones scanning the file system of computer with purpose of given malicious code detection and elimination.

10. Creation of immune memory set. The best neural network detectors are defined, which have shown the perfect results during detection of given computer virus. Detectors of immune memory live in system long time and provide the protection against repeated infection.

Figure 4 shows the performance of the immune detectors based on neural network architecture.

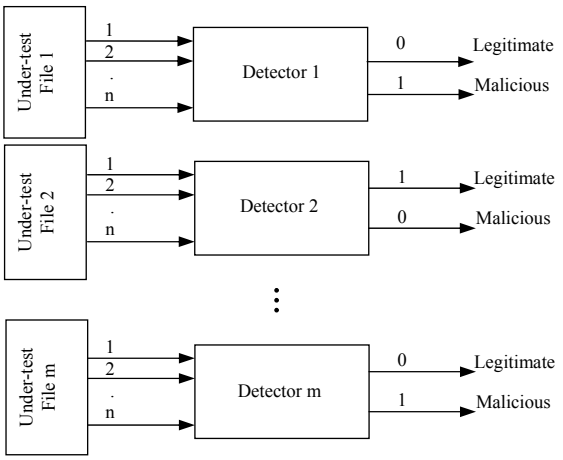

Figure 4: The performance of immune detectors.

Let's note the basic differences of the proposed and known algorithms. In our case each immune detector is completely independent object, i.e. itself chooses the scanning area. For this purpose it receives the list of files stored on a hard disk and randomly chooses a file from this list. After checking of file detector selects randomly next file from the existing list. The procedure is continued until the detector does not detect malicious code or lifetime of the detector is not finished yet. The key advantage of proposed neural network artificial immune system (NAIS) is the ability to detect novel malicious code, for which no signature exists.

3 Neural network immune detector

The choice of neural network immune detectors directly influences on the classification quality of unknown patterns and malware detection. The AIS is characterized by continuous evolution of immune detectors (figure 1). Untrained detectors are incapable of correct classification of legitimate and malicious code. The complexity of learning process depends on the size of training data set. Therefore we should choose the type of neural network that is characterized by minimal size of training set. As a basis of the immune detector (NID) we will use counter-propagation neural network. In comparison to another types of neural networks for instance multi-layer perceptron and multi-recurrent neural networks, counterpropagation neural network is characterized by minimal size of training data set.

Figure 5 shows the structure of immune detector, which consists of three layers and arbiter.

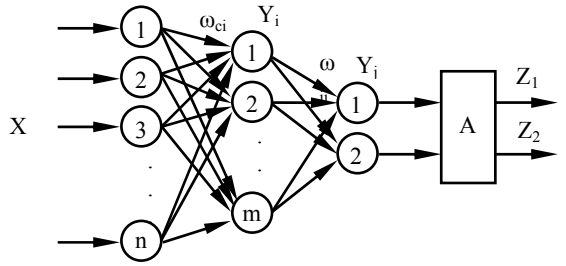

Figure 5: The structure of NID.

The number of neurons of input layers is equal to size of sliding window n (NID scans files by a sliding window method).

The second layer consists of the competitive units. This layer are trained, using the competitive learning rule (winner-takes-all) and performs clusterization of the input patterns. The number of units equals m and

$$m = p + r, \quad (9)$$

where p – the number of the first neurons which correspond to legitimate files; r – the number of last neurons. The activity of these units characterizes the class of malicious files.

The ratio of p to r should be multiple of 4 to 1 (for example $p = 8, r = 2$)

$$\frac{p}{r} = \frac{4}{1}. \quad (10)$$

This ratio is obtained in accordance with the algorithm of training data set generation and showed the best results.

The third layer consists of two linear units. The activity of the first unit characterizes the “clear” legitimate pattern and activity of the second one corresponds to malicious pattern. In general case the

output value of *j*-th neuron of third layer is described in accordance with equation (11).

If the winning neuron of competitive layer has number *k* then the output value of *j*-th neuron is calculated by equation (12).

$$Y_j = \sum_{i=1}^{m} \omega_{ij} \cdot Y_i, \tag{11}$$

where ω_{ij} – weighting coefficient between *i*-th neuron of competitive layer and *j*-th neuron of linear layer; *m* - the number of units of competitive layer.

$$Y_j = \omega_{kj} \cdot Y_k. \tag{12}$$

Eventually, for the correct mapping of input patterns into two classes the matrix of weights of third layer should be obtained as follows:

$$\begin{aligned} &\omega_{kj} = 1 \text{ if } k = 1,2,..,p \text{ and } j=1, \text{ or } k=p+1,..,r \text{ and } j=2 \\ &\omega_{kj} = 0 \text{ if } k = 1,2,..,p \text{ and } j=2, \text{ or } k=p+1,..,r \text{ and } j=1 \end{aligned} \tag{13}$$

For example, if $p = 8$ and $r = 2$ then the matrix of weights will be the following:

$$W^T = \begin{bmatrix} 1111111100 \\ 0000000011 \end{bmatrix} \tag{14}$$

The arbiter performs final decision about the class (legitimate or malicious) of the under test file. The output values of detector are obtained after analysis of the checking file

$$\begin{aligned} Z_1 &= \begin{cases} 1, & \text{if legitimate file} \\ 0, & \text{otherwise.} \end{cases} \\ Z_2 &= \begin{cases} 1, & \text{if malicious file} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{15}$$

4 Performance of the immune detector

The training data set consists of malicious and legitimate files. Legitimate and malware patterns contain accordingly 80% and 20% of learning samples from whole training data set. This ratio 4/1 affects to the proportion of "self" and "nonself" units in competitive layer.

The training patterns for each detector are formed by the following way:

1. Four legitimate files and one malicious file are randomly chosen from training data set.
2. *A* fragments (vectors) of *n* size (*n* = size of sliding window) are randomly chosen from the each selected files. As a result are obtained learning set. The number of training samples is equal $L = 5 \cdot A$.

The learned and selected NID scans the file memory and performs the function of malicious code detection. It should be noted that every NID is an independent autonomous agent which chose the target files for scanning by oneself.

The process of file scanning is performed by sliding window technique. The size of window can be varied within 128 and 512. These values are taken from the traditional method of malware detection based on signature analysis where similar sizes of signatures guaranties exact malicious code detection. The procedure of performance of neural network immune detector can be represented as follows:

1. The initial values of units are set to 0

$$\begin{aligned} \overline{Y_1}(k-1) &= 0, \\ \overline{Y_2}(k-1) &= 0. \end{aligned} \tag{16}$$

2. The input patterns *(k = 1...L*, where *L* – the number of patterns) from a checked file sequentially enter on the immune detector and for each pattern the following calculations are performed:

a. Euclidean distance between input pattern and weights of competitive layer units;

b. the winning neuron with index *k*

$$D_k = \min_j D_j. \tag{17}$$

c. the output values of linear units (equation 12);
d. the number of legitimate and malicious fragments of under-test file

$$\begin{aligned} \overline{Y_1}(k) &= \overline{Y_1}(k-1) + Y_1^k, \\ \overline{Y_2}(k) &= \overline{Y_2}(k-1) + Y_2^k. \end{aligned} \tag{18}$$

3. The membership probability of under-test file into legitimate or malicious class is calculates

$$\begin{aligned} P_T &= \frac{\overline{Y_1}}{L} \cdot 100\%, \\ P_F &= 1 - P_T = \frac{\overline{Y_2}}{L} \cdot 100\%, \end{aligned} \tag{19}$$

where P_T – the probability of legitimate file; P_F – the probability of malware.

5. The final decision is performed as follows:

$$\begin{aligned} Z_1 &= \begin{cases} 1, & \text{if } P_T > 80\% \\ 0, & \text{otherwise.} \end{cases} \\ Z_2 &= \begin{cases} 1, & \text{if } P_F > 20\% \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{20}$$

The space of output values of the arbiter can be represented in the following form (table 1).

Table 1: The arbiter output values space.

Z_1	Z_2	Class
1	0	Clear
0	1	Malicious
0	0	Undefined

If $Z_1 = 0$ and $Z_2 = 0$ then another NID for scanning this file is assigned.

Let's consider an example performance of NID, using two files *write.exe* and *Virus.Win32.VB.d*. The NID has next architecture: $n = 256$, $m = 10$, $b = 2$, where n – the number of input units; m – the number of competitive units; b – the number of output units. The next four legitimate file – *forcedos.exe, rspndr.exe, share.exe, lpq.exe* and one malicious file – *Net-Worm.Win32.Bozori.k* are used for the NID training. Let's examine the inspection of *write.exe.* File. The number of patterns (NID scans files using sliding window technique) is computed as

$$L = S - n + 1 = 2402 - 256 + 1 = 2147, \quad (21)$$

where S – file size; n – window size.

As a result the NID classifies *1084* fragments to legitimate class and *343* fragments to malicious class. Therefore probabilities of legitimate and malicious class is

$$P_T = \frac{\overline{Y_1}}{L} \cdot 100\% = \frac{1804}{2147} \cdot 100\% = 84\%,$$
$$P_F = 1 - P_T = \frac{\overline{Y_2}}{L} \cdot 100\% = \frac{343}{2147} \cdot 100\% = 16\%. \quad (22)$$

Finally the arbiter performs decision as regards maliciousness of the file:

$$Z_1 = 1, \text{ since } P_T > 80\%,$$
$$Z_2 = 0, \text{ since } P_F < 20\%. \quad (23)$$

Therefore *Write.exe* belongs to legitimate class.

In case of *Virus.Win32.VB.d* the number of windows is equal to

$$L = S - n + 1 = 33330 - 256 + 1 = 33075, \quad (24)$$

As a result the NID classifies *21499* fragments to legitimate class and *11576* fragments to malicious class. The probabilities of legitimate and malicious classes is

$$P_T = \frac{\overline{Y_1}}{L} \cdot 100\% = \frac{21499}{33075} \cdot 100\% = 65\%,$$
$$P_F = 1 - P_T = \frac{\overline{Y_2}}{L} \cdot 100\% = \frac{11576}{33075} \cdot 100\% = 35\%. \quad (25)$$

And finally the arbiter makes a decision about maliciousness of the file:

$$Z_1 = 0, \text{ since } P_T < 80\%,$$
$$Z_2 = 1, \text{ since } P_F > 20\%. \quad (26)$$

Therefore *Virus.Win32.VB.d* belongs to malicious class.

This example shows that the proposed algorithm permits to detect new malicious code.

5 Experimental results

We performed series of experiments for testing of NAIS and comparison results of malicious code detection with known anti-viruses software. Table 2 shows these results.

Table 2. The results of malware detection.

Malware	Kaspersk anti-virus	Eset NOD32	Dr.Web	NAIS 500 NID
Worm.Brontok.q	worm	win32/brontok	worm	malicious
Worm.NetSky.q	worm	win32/netsky	worm	malicious
Worm.Rays	worm	legitimate	legitimate	malicious
Worm.Zafi.d	legitimate	newheur_pe	legitimate	malicious
Worm.Zafi.f	legitimate	newheur_pe	legitimate	malicious
Worm.Bozori.a	legitimate	win32/bozori	legitimate	malicious
Worm.Bozori.k	legitimate	win32/bozori	legitimate	malicious
Worm.Lovesan.a	worm	win32/lovesan	worm	malicious
Worm.Maslan.a	worm	win32/maslan	worm	malicious
Worm.Mytob.a	legitimate	win32/mytob	legitimate	malicious
Worm.Sasser.a	worm	legitimate	legitimate	malicious
Packed.Tibs	legitimate	legitimate	legitimate	malicious
Trojan.Dialer.eb	trojan	legitimate	legitimate	malicious
Trojan.Small.kj	trojan	legitimate	legitimate	malicious
Trojan.Psyme.y	legitimate	legitimate	legitimate	malicious
Trojan.Adload.a	trojan	legitimate	legitimate	malicious
Trojan.Bagle.f	legitimate	win32/bagle	legitimate	malicious
Trojan.INS.bl	trojan	win32/trojan	trojan	malicious
Trojan.INS.gi	trojan	win32/trojan	trojan	malicious
Trojan.Ladder.a	trojan	win32/trojan	trojan	malicious
Trojan.Small.da	trojan	win32/trojan	trojan	malicious
Trojan.Small.dde	trojan	win32/trojan	trojan	malicious
Trojan.Small.dg	trojan	win32/trojan	trojan	malicious
Trojan.Agent.y	trojan	legitimate	legitimate	legitimate
Trojan.Daemon.a	trojan	legitimate	trojan	malicious
Trojan.Lager.d	trojan	win32/trojan	trojan	malicious
Trojan.Mitglied.o	trojan	win32/trojan	trojan	malicious
Trojan.Small.a	trojan	win32/trojan	trojan	malicious
Trojan.Antigen.a	trojan	legitimate	legitimate	malicious
Trojan.Fantast.3	legitimate	legitimate	legitimate	malicious
Trojan.Hooker.2	trojan	legitimate	legitimate	malicious
Trojan.LdPinch.f	trojan	win32/psw	trojan	malicious
Virus.Bee	virus	win32/virus	virus	malicious
Virus.Hidrag.d	virus	win32/virus	virus	malicious
Virus.Neshta.a	virus	win32/virus	virus	malicious
Virus.VB.d	virus	win32/virus	virus	malicious

This test presents the ability of conventional methods for malware detection that use various anti-viruses to detect novel malicious code. Therefore, in our tests all used anti-viruses were with outdated signature bases and most of presented malware were novel, "unknown" as well as for NAIS. The presented malware cover various classes such as viruses, Trojans, worms etc.

Let's analyze the results. Different anti-viruses showed poor results: 75% detection from Kaspersky antivirus, 67% detection from NOD32 and 53% from Dr.Web. Of course, small size of test sample cannot give exact results. However if we address to figure 1 then we can see the similar results. Thus, NAIS provide the best results in task of novel malicious code detection. No doubt, and we are sure that the increasing of number of malware lead up to decreasing of rate of malware detection by NAIS. But obtained results permit us to conclude that NAIS provides better defense against novel malicious code.

6 Conclusions

In this paper we present the neural network artificial immune system for malware detection. Such system consists of certain number of neural network detectors and has ability to novel malware detection with low false positive and false negative rates. Applying of the NAIS for malicious code detection permits to expand the possibilities of existing antivirus software and will increase level of computer security.

References

1. Virus bulletin, http://www.virusbtn.com
2. Shafig, M., Tabish, S., Farooq, M.: On the Appropriateness of Evolutionary Rule Learning Algorithms for Malware Detection. In: 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, pp. 2609--2616. ACM Press, New York (2009).
3. Edge, K., Lamont, G., Raines, R.: A Retrovirus Inspired Algorithm for Virus Detection & Optimization. In: 8th Annual Conference Companion on Genetic and Evolutionary Computation Conference, pp. 103--110. ACM Press, New York (2006).
4. Lee, H., Kim, W., Hong, M.: Artificial Immune System Against Viral Attack. In: Bubak, M. (eds.) ICCS 2004. LNCS, vol. 3037, pp. 499--506. Springer, Heidelberg (2004)
5. Harmer, P., Williams, P., Gunsch, G., Lamont, G.: An Artificial Immune System Architecture for Computer Security Applications. IEEE Transactions on Evolutionary Computation. vol. 6(3), pp. 252--280 (2002)
6. De Castro, L., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (2002)
7. Dasgupta, D.: Artificial Immune Systems and Their Applications. Springer, Berlin (1999)
8. Marhusin, M., Cornforth, D., Larkin, H.: Malicious Code Detection Architecture Inspired by Human Immune System. In: 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 312--317. IEEE Press, Phuket (2008)
9. Daoud, E.: Metamorphic Viruses Detection Using Artificial Immune System. In: International Conference on Communication Software and Networks, pp. 168--172. IEEE Press, Macau (2009)
10. Wang, W., Zhang, P., Tan, Y., He, X.: A Hierarchical Artificial Immune Model for Virus Detection. In: International Conference on Computational Intelligence and Security, pp. 1--5. IEEE Press, Beijing (2009)
11. Tesauro, G., Kephart, J., Sorkin, G.: Neural Networks for Computer Viruses Recognition. IEEE Expert. vol 11(4), 5--6 (1996)
12. Doumas, A., Mavroudakis, K.: Design of a Neural Network for Recognition and Classification of Computer Viruses. Computers & Security. vol. 14(5), 435-448 (1995)
13. Powers, S., He, J.: A Hybrid Artificial Immune System and Self Organising Map for Network Intrusion Detection. Information Sciences. vol. 178, 3024 – 3042 (2008)