System for detecting network anomalies using a hybrid of an uncontrolled and controlled neural network

Galina Kirichek[0000-0002-0405-7122], Vladyslav Harkusha[0000-0001-5980-4802], Artur Timenko[0000-0002-7871-4543] and Nataliia Kulykovska[0000-0003-4691-5102]

National University "Zaporizhzhya Polytechnic",
64, Zhukovsky Str., Zaporizhzhia, 69063, Ukraine
{kirgal08, garkusha2580, artureuro, natalya.gontar}@gmail.com

Abstract. In this article realization method of attacks and anomalies detection with the use of training of ordinary and attacking packages, respectively. The method that was used to teach an attack on is a combination of an uncontrollable and controlled neural network. In an uncontrolled network, attacks are classified in smaller categories, taking into account their features and using the self-organized map. To manage clusters, a neural network based on back-propagation method used. We use PyBrain as the main framework for designing, developing and learning perceptron data. This framework has a sufficient number of solutions and algorithms for training, designing and testing various types of neural networks. Software architecture is presented using a procedural-object approach. Because there is no need to save intermediate result of the program (after learning entire perceptron is stored in the file), all the progress of learning is stored in the normal files on hard disk.

Keywords: neural network, learning, intrusion, anomalies detection, SOM.

1 Introduction

The probability of threats in computer networks increases every year and is a rather serious issue, so the use of intrusion detection technologies is an important issue in providing network and computer security. The process of detecting an attack is implemented as a monitoring of events in the system or computer network, and allows you to determine, with the indicated probability, an intrusion or not.

Modern filters of network traffic, detection systems and counteraction interventions become ever less effective when dealing with large volumes of traffic in high-speed networks and also unsuitable for recognizing new types and methods of attacks on computer systems and networks. Inductive methods provide the opportunity to obtain accurate identification or prediction of various complex processes in the case of short or noisy input data. This is relevant for network traffic recognition based on protocol classification because most of the normal network thread meets the RFC standards set by the developers, and the anomalies most often manifest themselves in non-standard behavior and packet status.

The system of intrusion detection verifies the network traffic that is being investigated for suspicious activity and also alerts the system or system administrator of possible attacks. The main purpose of intrusion detection system is to protect the availability, confidentiality and integrity of critical network information systems. Two main approaches to the system of intrusion detection are used: the detection of abuses and abnormalities [5].

Detection of misuse is based on the description of known dangerous actions. This description is often modeled as a set of rules that are referred to as signature attacks. The Anomaly Detection ID looks for a threat and applies the rules or predefined terms: the normal and abnormal activity of the system. In the future, we use it to detect the difference in threats from the normal system behavior, to monitor the report, or to block the threats when they arise. Different methods of artificial intelligence are used in intrusion detection system (IDS) anomalies, such as machine learning [11; 14], intelligent data analysis, image recognition and neural networks [18].

To identify abnormalities, it is more rational to develop an interactive intrusion detection system than regular rules and programs that work under the normal principle of detecting and responding to anomalies in the network. Therefore, it makes sense to integrate the classical approaches of IDS and approaches to data analysis using neural networks, which is considered a more flexible approach to the analysis and data classification [6; 9; 15].

2 Formulation of problem

The aim of this work is to develop an IDS system prototype based on a hybrid neural network to detect anomalies and threats from the network, based on the principle of self-organizing maps and the error backpropagation of neural network (learning with teacher). The object of research is the implementation process of modules for detecting threats and anomalies in network. The subject is to formulate model and implementation methods of system prototype.

Research has made it possible to determine that the software should perform analysis and separate the usual and dangerous data based on the input data, in this case, on the basis of network packets. But after revealing the dangerous data, he still needs to carry out the classification of the threat type.

The approach to using neural networks (perceptrons) is chosen as the basis for fulfilling the tasks. These are neural networks based on self-organizing maps used to analyze data and to detect ordinary packets on the network when filtering traffic. After analyzing the data in the first neural network, potentially dangerous data is transmitted to the next neural network to detect the threat type based on the reverse error propagation. In this case, both perceptrons need to be trained to distinguish between suspicious packages and types of threats, respectively. The data used in the study of perceptrons is a dataset of the Lincoln Laboratory of Massachusetts University of Technology. This set is designed to evaluate DARPA intrusion detection systems and is considered to be a benchmark for IDS research [1; 7; 17].

In conducting the experiment for classifying network traffic models, in accordance with the taxonomy of the five templates, we use a data set consisting of five classes of packages, which include: ordinary packages, packages for sensing and scanning infrastructure, packets that caused denial of service equipment, packages that have increased user privileges to super user and external threat packages.

3 Software tools and solutions

It was decided to use two interceptors for a more convenient scaling and use software with independent modules. This simplifies not only the ability to scale but also reduces the concentration of responsibility on each of the software modules. That is to teach and arrange two smaller perceptrons is much easier than one big one. Hybrid network approach based on a neural network without a teacher (first module) and a neural network with a teacher (second module) is used when developing a network threats analyzer. As a network without a teacher, use self-organizing maps.

One of the main approaches to solving cluster analysis problems is a self-organizing maps (SOM) [10; 12]. They are adapted for using learning without teacher, that is, without the end result.

The method of back propagation is a learning method that is controlled by the training of artificial neural networks [16]. The purpose of back propagation is to prepare the network to achieve a balance between the ability to respond correctly to the input models used for learning (memorization) and the ability to give intelligent input responses, as in the training.

The process of learning without a teacher in SOM can be briefly described in three stages. In the first stage, weights of the connection are assigned small random numbers and the choice of the speed learning parameter is made.

At the second stage, the best matching block is fixed, with determination of neuron with the greatest weight in the layer of neural network, Euclidean square is used to measure the distance between the input vector and the weight vector, and also the unit chosen whose weight vector has the smallest Euclidean distance from the input vector is selected.

At the last stage, weights are updated according to the rule of training of Kohonen network according to formula (1):

$$\omega_{ij}^{\text{new}} = \omega_{ij}^{\text{old}} + \alpha(x_i - \omega_{ij}^{\text{old}}), \tag{1}$$

where x_i is the i-th input vector, ω_{ij} is the j-th column of the weight matrix, and α, the learning rate, decreases as learning proceeds. Updating neuronal weight in the network occurs only for active output neurons. It is allowed to teach a unit whose weighted vector is closest to the input vector [2].

Learning process itself continues until all input vectors are processed. Criterion of convergence in neural networks is an epoch. This is one iteration in the learning process, which includes representation of all examples from training set, as well as verification of quality training in a controlled set. Epoch determines how many times all input vectors must be submitted to the SOM for learning.

This algorithm is also called the gradient descent algorithm because the strategy of selecting such an important parameter as the weight for each neuron of a multi-layered network is based on the gradient method. Continuous target function as a measure of network success in the general case is defined as quadratic amount difference between an actual result and expected output value. Algorithm for the reverse distribution of a learning error uses two extensions of the network - direct and reverse.

At the very beginning of the algorithm there is a direct passage where input data in the form of a vector implement distribution among the layers, from the original to the last. As a result of direct pass, a set of output signals is generated, which determines response of the network to the input data. During a straight pass, all synaptic weights of network are fixed. The second stage of algorithm is return pass where parameters (all synaptic weights) are adjusted according to the error correction rules. The essence of the rule is as follows: expected output values subtract resulting (resulting) value of an actual output and error signal is generated as a result of such an operation. Error signal extends like an echo in opposite synaptic bonds, so the algorithm got this name. And synaptic scales, in turn, are adapted to maximize expected output of network's output signal.

Scales are adjusted to reduce the error by distributing original error back through the network. Training kit is supplied several times to the network, and the weight values are corrected until overall error exceeds the specified one. Developed system uses identification process of abnormal and normal packets in a computer network. Whole process of system development can be divided into 2 stages. The first is the stage of training in which the SOM neural networks and reverse error propagation have been trained for a certain amount of time (epoch), it is shown in Fig. 1.

Next step is to detect threats or testing yourself. Model of method functioning is shown in Fig. 2. Since the usual packet-transfer analysis operations are specified and they display the expected behavior, we can initiate knowledge-based definitions (improper use), whereas the non-typical packet activity (the invasion is likely to indicate the non-typical behavior of the packet) is constantly being developed and can't be regarded as defined an attack, so identifying IDS abnormalities is performed on attacks.

An uncontrolled neural network based on a self-organizing map (SOM) divides classification of threats into smaller categories, taking into account their similar features, and then, clustering of threats is performed based on the error of nonpropagation of the neural network.

The SOM training is implemented on the basis of data from KDD-99 (knowledge discovery in databases), which is a set of data used during the second international competition on open knowledge and data mining.

Connections in KDD-99 are presented in the form of functions, each of which is located in significantly different ranges, in one of the continuous, discrete, and symbolic forms. Functions in this set are protocol type, service type and respectively. The protocol type value can match tcp, udp or icmp; the service type may be one of the different network services, such as http, smtp, etc.; the checkbox corresponds to one of 11 values, such as SF or S2. Other parameters in these connections are the length of connection; number of bytes of data from the beginning to destination and vice versa;

number of connections to the same end node as the current connection in the last two seconds, etc. The full list of attributes set for the connection records is given in corresponding sources of information [3; 7].

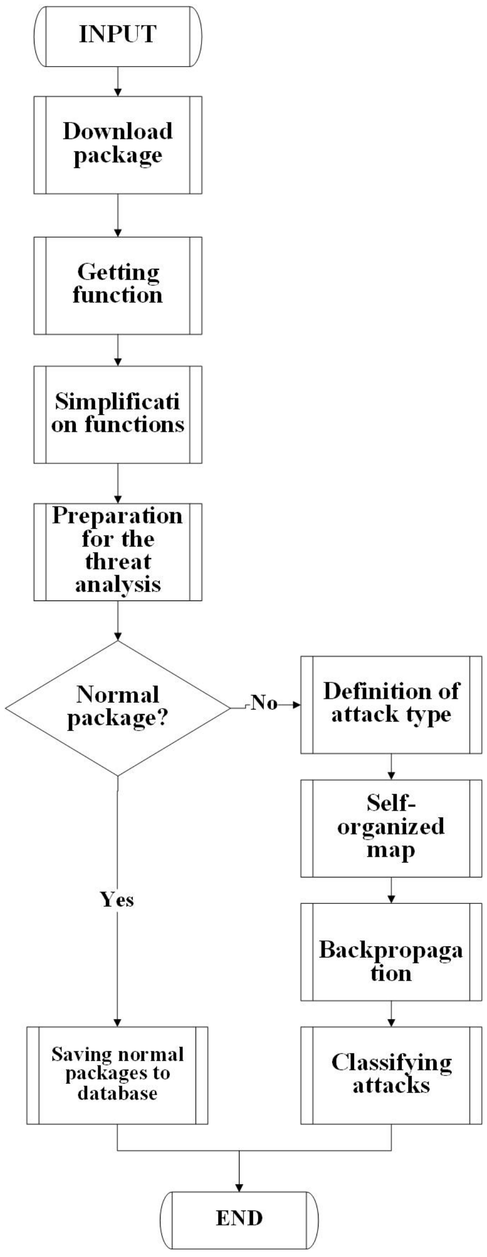

Fig. 1. Phase training algorithm.

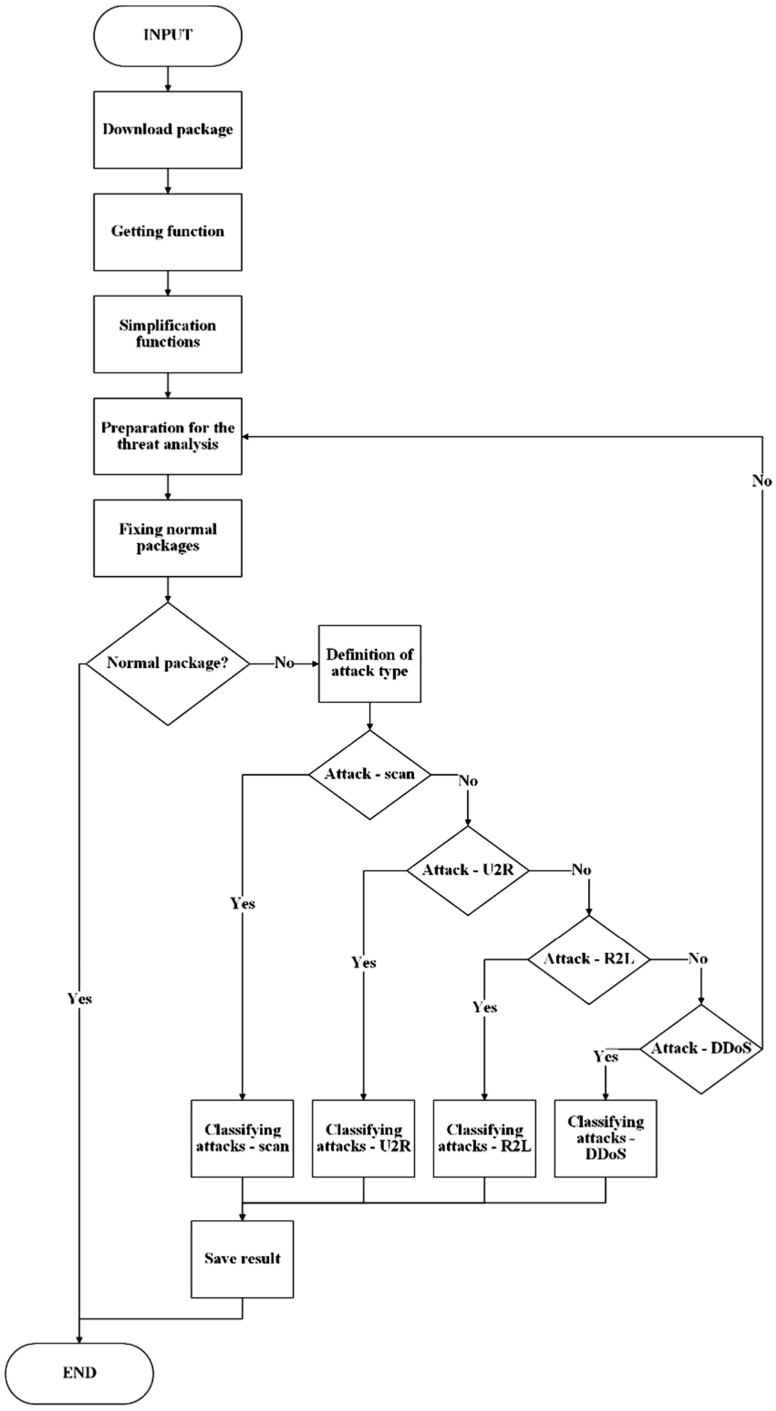

Fig. 2. Detection phase algorithm.

4 System for detecting network anomalies

In the process of designing and system software implementation, authors developed software (utility) that allows you to fulfill the purpose of the work – detection and classification of threats in network traffic (packets).

As a software method, Python has been selected as an interpreted object-oriented programming language that supports module packs and several programming paradigms: object-oriented, procedural, functional, and aspect-oriented [4; 8]. The choice has been influenced by: Python's support of object-oriented approach, simplicity of syntax, and availability of built-in functions and data structures. Also, in this language, a large number of ready-made solutions and documentation in the field of development and training of perceptrons is realized.

We use PyBrain as the main framework for designing, developing and learning perceptron data. This framework has a sufficient number of solutions and algorithms for training, designing and testing various types of neural networks. Software architecture is presented using a procedural-object approach. Because there is no need to save intermediate result of the program (after learning entire perceptron is stored in the file), all the progress of learning is stored in the normal files on hard disk.

We use setuptools as the main software, when creating this utility, to simplify the construction of the main framework of the system.

Program consists of the main function that is called when the program is started and after class initialization [13].

```
from __future__ import __all__
from scipy import random
from scipy.ndimage import minimum_position
from scipy import mgrid, zeros, tile, array, floor, sum
from module import Module

class Kohonen_SOM_Map(Module):
  learn_rate = 0.01
  neighbourd = 0.9999
  outdim, winner, neurons_num, dist_matrix, inputs_num, diff,
neurons, outputFullMap = None, None, None, None, None, None

  def __init__(self, dim, neurons_num, name=None,
output_full_map=False):
    outdim = 2 if output_full_map else neurons_num ** 2
    Module.__init__(self, dim, outdim, name)
    Kohonen_SOM_Map.outputFullMap = output_full_map
    Kohonen_SOM_Map.neurons = random.random((neurons_num,
neurons_num, dim))
    Kohonen_SOM_Map.winner = zeros(2)
    Kohonen_SOM_Map.diff = zeros(self.neurn.shape)
    Kohonen_SOM_Map.inputs_num = dim
    Kohonen_SOM_Map.neurons_num = neurons_num
    Kohonen_SOM_Map.neighbours = neurons_num

    # Init matrix of predicates
    Kohonen_SOM_Map.__dist_matrix_create()

  def _forward_err_implement(self, inbuf, outbuf):
    Kohonen_SOM_Map.diff = Kohonen_SOM_Map.neurons - tile(inbuf,
(Kohonen_SOM_Map.neurons_num, Kohonen_SOM_Map.neurons_num, 1))
    error = sum(Kohonen_SOM_Map.diff ** 2, 2)
    Kohonen_SOM_Map.winner = array(minimum_position(error))
    if not Kohonen_SOM_Map.outputFullMap:
      outbuf[:] = Kohonen_SOM_Map.winner

  @classmethod
  def _backward_err_implement(cls):
    n = floor(cls.neighbours)
    cls.neighbours *= cls.neighbourdecay
    tl = (cls.winner - n)
    br = (cls.winner + n + 1)
    tl[tl < 0] = 0
    br[br > cls.neurons_num + 1] = cls.neurons_num + 1

    # calculate distance matrix
    tempm = 1 - sum(abs(cls.dist_matrix - cls.winner.reshape(1, 1,
2)), 2) / cls.neurons_num
    tempm[tempm < 0] = 0
    distm = zeros((cls.neurons_num, cls.neurons_num, cls.nInput))
    for i in range(cls.nInput):
      distm[:, :, i] = tempm
      distm[:, :, i] = tempm
    cls.neurons[tl[0]:br[0], tl[1]:br[1]] -= cls.learningrate *
cls.diff[tl[0]:br[0], tl[1]:br[1]] * distm[tl[0]:br[0],
tl[1]:br[1]]

  @classmethod
  def __dist_matrix_create(cls):
    if not cls.neurons_num:
      print ("Kohonen_map: not setted neural layers")
    distx, disty = mgrid[0:cls.neurons_num, 0:cls.neurons_num]
    cls.dist_matrix = zeros((cls.neurons_num, cls.neurons_num, 2))
    cls.dist_matrix[:, :, 0] = distx
    cls.dist_matrix[:, :, 1] = disty
```

The KohonenMap _forward_err_implement method assigns one of the neurons to input dates in the input buffer and fixed coordinates of neurons in the output buffer, and also performs calculation of the largest neuron, calculating data with the slightest error using square of difference.

The KohonenMap _backward_err_implement method training the Kohonen map in an uncontrolled mode, moving the closest neuron and neurons adjacent to it closer to the input template [4; 13].

The main function performs initialization of an instance of the class, namely, it creates the KohonenMap object and assigns variable to given object. After that, in the input buffer, training data is asked in order to conduct training of this object. Learning result is stored on the hard disk after training for several cycles.

The obtained results confirm that the quality of the classification of packages depends on the number of standards of separate classes in the educational voter. If the number is small, then the detection rate of the attacks is high and the number of detected intrusions by class is improved. This indicates that the method works in real-time with high performance.

5 Conclusion

The purpose of this work is to develop a hybrid neural network (perceptron) based on 2 other neural networks, namely, the Kohonen neural network and the neural network with back propagation. Data set from the Lincoln Laboratories of Massachusetts Technology University from United Stateswas used as learning data sets. This data set includes type of package, its useful data and metadata. The Python language and PyBrain framework are selected as the software component.

When developing software based on the idea of hybridization of neural networks, the problem was solved to ensure protection of internal network from external threats using packet filtering for threats such as denial of service and unauthorized increase of user privileges. Effectiveness of methods to protect computer networks from harmful traffic has been increased using prior analysis of packets risk. Also, in this neural network, the so called boosting is applied - an increase in the efficiency of the neural network at the expense of another neural network, which delivers already filtered information to the inputs.

In the following, the possibility of using this software on operating system for such routers as OpenWrt is considered. This integration will not only increase an efficiency of this system while protecting the network, but also will increase an accuracy of the perceptron through the adoption of a large number of network traffic with self-study of neural networks.

References

1. Akbar, S., Rao, K.N., Chandulal, J.A.: Intrusion detection system methodologies based on data analysis. International Journal of Computer Applications **5**(2), 10–20 (2010). doi:10.5120/892-1266

2. Bahrololum, M., Salahi, E., Khaleghi, M.: An improved intrusion detection technique based on two strategies using decision tree and neural network. Journal of Convergence Information Technology **4**(4), 96–101 (2009)
3. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature deduction and ensemble design of intrusion detection systems. Computers & Security **24**(4), 295–307 (2005). doi:10.1016/j.cose.2004.09.008
4. Dierbach Ch.: Python as a first programming language. Journal of Computing Sciences in Colleges **29**(6), 153–154 (2014)
5. García-Teodoro, P., Díaz-Verdejoa, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security **28**(1–2), 18–28 (2009). doi:10.1016/j.cose.2008.08.003
6. Imamverdiyev, Y.N., Sukhostat, L.V.: Obnaruzhenie anomalii v setevom trafike na osnove informativnykh priznakov (Network traffic anomalies detection based on informative features). Radio electronics, computer science, control 3, 113–120 (2017) doi:10.15588/1607-3274-2017-3-13
7. KDD Cup 1998 Data. http://kdd.ics.uci.edu//databases/kddcup98/kddcup98.html (1999). Accessed 21 Mar 2019
8. Kirichek, G., Kurai, V.: Implementation quadtree method for comparison of images. In: 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), pp. 129–132. IEEE (2018) doi:10.1109/TCSET.2018.8336171
9. Kirichek, G., Tymoshenko, V., Rudkovskyi, O., Hrushko, S.: Decentralized System for Run Services. CEUR Workshop Proceedings **2353**, 860–872 (2019)
10. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin, Heidelberg (2001). doi:10.1007/978-3-642-56927-2
11. Mukkamala, S., Janoski, G., Sung, A.: Intrusion detection using neural networks and support vector machines. In: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02. Honolulu, HI, USA, pp. 1702–1707 (2002). doi:10.1109/IJCNN.2002.1007774
12. Ritter, H., Martinetz, T., Schulten, K., Barsky, D., Tesch, M., Kates, R.: Neural Computation and Self-Organizing Maps: An Introduction. Addison-Wesley, Reading (1992)
13. Rueckstiess T.: Python PyBrain package v0.3, pybrain.structure.modules.kohonen module source code :: PyDoc.net. http://pydoc.net/PyBrain/0.3/pybrain.structure.modules.kohonen (2009). Accessed 17 Aug 2019
14. Sabhnani, M., Serpen, G.: Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. In: Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications. MLMTA'03, June 23–26, 2003, Las Vegas, Nevada, USA, pp. 209–215. CSREA Press (2003)
15. Salnik, S.V., Salnyk, V.V., Symonenko, O.A., Sova, O.Ya.: Metod vyiavlennia vtorhnen v mobilni radiomerezhi na osnovi neironnykh merezh (Method of intrusion detection in mobile radio networks on the basis of neurals networks). Science and Technology the Air Force of Ukraine 4(21), 82–90 (2015)
16. Semerikov, S.O., Teplytskyi, I.O., Yechkalo, Yu.V., Kiv, A.E.: Computer Simulation of Neural Networks Using Spreadsheets: The Dawn of the Age of Camelot. In: Kiv, A.E., Soloviev, V.N. (eds.) Proceedings of the 1st International Workshop on Augmented Reality in Education (AREdu 2018), Kryvyi Rih, Ukraine, October 2, 2018. CEUR Workshop Proceedings **2257**, 122–147. http://ceur-ws.org/Vol-2257/paper14.pdf (2018). Accessed 30 Nov 2018

17. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. IEEE (2009). doi:10.1109/CISDA.2009.5356528
18. Zhang, Z., Manikopoulos, C.: Neural networks in statistical anomaly intrusion detection. Neural network world **11**(3), 305–316 (2001)