Detection of network attacks by deep learning method

T Tatarnikova¹, P Bogdanov¹, E Kraeva¹, S Stepanov², A Sidorenko²

¹Department of information technologies and security systems, Russian State Hydrometeorological University, 79 Voronezhskaya Street, St. Petersburg 192007, Russia
²Department of applied computer science, Russian State Hydrometeorological University, 79 Voronezhskaya Street, St. Petersburg 192007, Russia

4Corresponding email: stepanov.sergey.y@gmail.com

Abstract. In this paper, we discuss the possibility of using a neural network approach in solving the problem of detecting network attacks. The neural network is designed to solve the problem of classifying the transmitted traffic into not containing an attack and containing an attack. The difficulties of using a neural network in this problem are discussed. Difficulties are associated with the choice of significant information features for the formation of the date set and having as many training examples as possible for each type of attack. Solutions for the selection of significant information features are proposed. It consists in ranking the features in order of importance. A method and rules for ranking features are proposed. In the future, it is proposed to use only important features to train the neural network. The problem of uneven number of training examples for each type of attack is considered. It is proposed to preserve significant examples represented by small sample sizes by assigning weights to them. Experiments show the effectiveness of the proposals discussed in the paper.

1. Introduction

Attack detection systems (ADS) are the basic means of protecting corporate information resources. Attack detection involves first collecting data and then analyzing it using ADS. In particular, you can analyze information about data packets transmitted over the network, the performance of software and hardware – the computational load on network nodes, the load of RAM, the speed of application software, access to certain files, etc. [1]

The operation of ADS is based on special methods for detecting attacks, usually signature and / or behavioral, while it is known that the use of the signature method does not allow detecting new types of attacks, and its modifications. It is known, that it is difficult to create an accurate model of normal operation of the network, and sometimes even impossible [2].

Due to the growth in the amount of digital data, research related to the use of deep learning methods to detect the network attacks – has become relevant. The attractiveness of methods based on neural networks lies in the possibility of their training on the proven data set, retraining when additional parameters appear, and self-learning when new results that did not appear before. A well-known disadvantage of artificial neural networks is the need for a data set and low computation speed with a large number of input parameters.
2. Methods and Materials

During the use of neural networks in the tasks of detecting network attacks, several data sets have been formed, for example, Overview, Aposemat IoT-23, Normal and others. The work uses the data set NSL-KDD of traffic transmitted via TCP, UDP and ICMP protocols. Each entry in NSL-KDD is a chain of network packets, captured at a time interval and sent from source to destination in accordance with the IP addresses indicated in the packet header. Each record includes a set of parameters $\{x_i\}$, where $i=1,42$, in which $x_1$, ..., $x_{41}$ are informational signs, and the last parameter $x_{42}$ is a label of the class "attack" and "not attack". Analysis of works [3, 4] showed that not all parameters are equally important when detecting network attacks – a reduction in dimension $\Pi$ is required to increase the speed of the ADS.

Dataset NSL-KDD contains 36 types of attacks, representing 4 categories [5]:

1. Denial of Service (Dos) – attacks that restrict access of verified users to a specific service through a specific protocol (Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm);

2. Remote to Local (r2l) – attacks aimed at gaining access to the user's local machine from the external environment (Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httpunnel, Sendmail, Named);

3. User to Root (u2r) – attacks aimed at obtaining privileged access rights to the victim's machine (Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps);

4. Probe - attacks aimed at obtaining information about the user's infrastructure (Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint).

In total, NSL-KDD contains 125973 records for training and 22,544 records for testing. The histogram of the distribution of types of network attacks in dataset NSL-KDD is shown in Figure 1.

![Figure 1. Distribution of attack types in the NSL-KDD data set](image-url)
Figure 1 that the number of training examples by attack type is unevenly distributed, which affects the detection accuracy and, accordingly, the quality of the ADS. Therefore, another urgent task is to minimize the impact of uneven distribution of the sample data set on the quality of training [7].

The problem of reducing the dimension of the vector $\Pi$ was solved by a linear method, which made it possible to rank the parameters in order of importance into "useful", "secondary" and "useless". Thus, from the vector $\Pi = \{x_1, x_2, \ldots, x_{41}\}^T$ it is necessary to obtain a new vector $\Pi' = \{k_1, k_2, \ldots, k_c\}^T$, where $c < 41$ and

$$k_i = w_{i,1} x_1 + \ldots + w_{i,41} x_{41}, \quad i = 1, 2, \ldots, c,$$

$$K = W \Pi,$$

where $W$ is the weights matrix of linear transformations.

It is obvious that reducing the dimension of the vector $\Pi$ by eliminating useless parameters will speed up the computations performed by the neural network – the number of neurons in the input layer is reduced, thereby increasing the accuracy of detecting network attacks due to the concentration of training the neural network only on significant parameters.

The assessment of the importance of the parameters was determined empirically: first, one parameter was deleted at a time, and on the obtained data set, training and, accordingly, testing of the neural network was performed. During testing, the quality indicators of the neural network were evaluated according to the following metrics:

Classification accuracy $P$ (precision):

$$P = \frac{TP}{TP + FP},$$

where $TP$ is the number of true-positive records, $FP$ is the number of false positive entries.

Detection rate $DR$:

$$DR = \frac{TP}{TP + FN},$$

where $FN$ is the number of false negative entries.

Thus, in assessing the importance of each parameter, three performance criteria were involved: classification accuracy – $A$, training time $T_{tr}$, and testing time $T_{test}$. The parameters were ranked in accordance with the decision tree shown in Figure 2.

Further, the learning process was carried out only on “useful” parameters, of which, as a result of the ranking, 10 remained, that is, it was possible to reduce the number of input parameters by 4 times.

The second task is to minimize the influence of the uneven distribution of the sample data set on the quality of training, it is solved by modifying the training algorithm. The essence of the modification is reduced to the adaptive assignment of weighting coefficients to training examples, represented by small samples.

It is known that in the process of training a neural network, the weights of synapses are corrected after submitting the entire training sample according to the average value of the gradient $E$ of the objective function:

$$E = \frac{1}{n} \sum_{i=1}^{n} (y_i - y'_i)^2,$$

where $y_i$ is the received value of the $i$-th neuron of the output layer; $y'_i$ is the expected value of the $i$-th neuron of the output layer; $n$ is the number of neurons in the output layer.
Figure 2. Decision tree defining the parameter ranking rules

The error found from examples of small samples in the training process can be lost in the total value of $E$ over the entire data set, and as a result these examples can be lost. To minimize the occurrence of such a situation, it is proposed to assign weight coefficients to training examples represented by small samples, which in the physical sense enhances the contribution of these examples to the value of error $E$. The essence of modification of the learning algorithm demonstrated in Figure 3.
Figure 3. Explanations for the modification of the learning algorithm

Suppose that at the beginning of training the neural network, the error $E$ throughout the data set does not exceed the value $E_2$, and $E_1$ is the admissible error value. In the interval [$E_1$, $E_2$], each example at each of $N$ iterations of the learning process introduces its own weight coefficients into the synaptic map of the neural network. In most examples, the $E$ value begins to decline, but in some examples, the $E$ value does not change or does change insignificantly. As soon as the error of the training example becomes $E < E_1$, then the example receives a unit coefficient, which indicates its significant contribution to the training of the neural network. If during further training for the same example becomes $E \geq E_1$, then its weight coefficient will not change, however, as soon as $E > E_2$, then the example does not participate in further training [8].

3. Results and Discussion

The structure of the neural network for solving the problems posed is a multilayer perceptron with ten neurons of the input layer, one hidden layer with twelve neurons and six neurons of the output layer, all neurons of the sigmoidal type. Table 1 shows the values of the neural network quality metrics.

| Attack number | Attack type       | Neural network quality metrics |
|---------------|-------------------|--------------------------------|
| 1             | neptune           | $DR, \%$ | $P, \%$  |
| 2             | saint             | 98,5     | 100,0   |
| 3             | mscan             | 92,7     | 98,1    |
| 4             | guess_passwd      | 66,4     | 97,0    |
| 5             | smurf             | 95,2     | 99,5    |
| 6             | apacher2          | 97,8     | 99,7    |
| 7             | satan             | 90,7     | 81,8    |
| 8             | buffer_overflow   | 0        | 0       |
| 9             | back              | 96,1     | 97,7    |
| 10            | warezmaster       | 16,1     | 98,1    |
| 11            | snmpgetattack     | 88,7     | 99,9    |
| 12            | processtable      | 85,8     | 98,4    |
| 13            | pod               | 82,9     | 70,8    |
| 14            | httptunnel        | 98,5     | 100,0   |
| 15            | nmap              | 79,5     | 90,6    |
Checking the quality of the neural network for detecting attacks presented in the data set NSL-KDD showed an average classification accuracy \( P \) equal to 59.2%. After training with the modified algorithm, the average \( P \) value became equal to 92.5%. The \( P \) values for the neural network model trained by the modified learning algorithm are given in Table 2.

Table 2. Results after learning by the modified algorithm

| Attack type         | Quality examples | \( P \), % |
|---------------------|------------------|------------|
| buffer_overflow     | 4                | 100,0      |
| pod                 | 17               | 100,0      |
| ps                  | 2                | 100,0      |
| multihop            | 2                | 100,0      |
| named               | 2                | 100,0      |
| sendmail            | 14               | 71,1       |
| loadmodule          | 2                | 100,0      |
| xterm               | 13               | 77,5       |
| worm                | 2                | 100,0      |
| teardrop            | 12               | 83,3       |
| rootkit             | 13               | 84,2       |
| xlock               | 9                | 89,1       |
| perl                | 2                | 100,0      |
| land                | 7                | 86,2       |
| xsnoop              | 4                | 75,3       |
| sqlattack           | 2                | 100,0      |
| ftp_write           | 3                | 67,1       |
| udpstorm            | 2                | 100,0      |
| phf                 | 2                | 100,0      |
4. Summary
Due to the growth of digital data [9, 10], deep learning models based on neural networks have gained popularity in solving the problem of detecting network attacks. A well-known drawback of the signature-based neural network model is the possibility of false positives and gaps of real attacks, as well as the need for a high-quality data set for training. Eliminating these shortcomings or minimizing them is associated, firstly, with the choice of significant information signs that allow classifying network traffic into normal – not containing an attack and anomalous - containing an attack, and secondly, saving significant training examples.

It is proposed to solve the reduction of the dimension of the vector of information features by a linear method with ranking the features according to the degree of importance and in the future to use only “important” features to train the neural network.

It is proposed to solve the preservation of significant training examples represented by a small sample size by modifying the learning algorithm, the essence of which is reduced to adaptive assignment of weighting coefficients to such examples.

The experiments carried out indicate the effectiveness of the proposed method and neural network training algorithm for detecting network attacks - the classification accuracy has increased from 59.2% to 92.5%. The use of a neural network model in conjunction with the signature approach will obviously increase the efficiency of the ADS, including when detecting new network attacks and modifications of existing ones.

References
[1] Rowayda A, Sadek M, Sami Soliman and Elsayed H.S 2013 Effective Anomaly Intrusion Detection System based on Neural Network with Indicator Variable and Rough set Reduction. International Journal of Computer Science Issues vol. 10 issue 2 no 2 227–233
[2] Guojie L, Jianbiao Z 2020 Research of Network Intrusion Detection Based on Convolutional Neural Network Discrete Dynamics in Nature and Society 1-11. DOI: 10.1155/2020/4705982.
[3] Bhattacharjee P, Fujail A, Begum S. 2017 Intrusion Detection System for NSL-KDD Data Set using Vectorised Fitness Function in Genetic Algorithm Advances in Computational Sciences and Technology vol 10 235–246
[4] Ingre B, Yadav A, Soni A.K. 2017 Decision Tree Based Intrusion Detection System for NSL-KDD Dataset Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems Cham: Springer vol. 2 207–218
[5] NSL-KDD dataset. URL: https://www.unb.ca/cic/datasets/nsl.html
[6] Chockwanich N, Visoottiviseth V. 2019 Intrusion Detection by Deep Learning with TensorFlow 21st International Conference on Advanced Communication Technology 654-659. DOI: 10.23919/ICACT.2019.8701969
[7] Tatarnikova T. and Kutuzov O. 2018 Model of a self-similar traffic generator and evaluation of buffer storage for classical and fractal queuing systems Moscow Workshop on Electronic and Networking Technologies (Moscow, Russia) 2018 1-3 DOI: 10.1109/MWENT.2018.8337306
[8] Tatarnikova T Statistical methods for studying network traffic Informatsionno-Upravliaiushchie Sistemy Vol. 2018 Issue 5 2018 35-43 DOI: 10.31799/1684-8853-2018-5-35-43
[9] Tatarnikova T., Stepanov S., Petrov Y., Sidorenko A. and Martyn I. A conceptual model for geodata processing for sustainable forest management. IOP Conference Series: Earth and Environmental Science. vol. 507. pp. 012029. DOI:10.1088/1755-1315/507/1/012029
[10] Tatarnikova T, Stepanov S, Petrov Y and Sidorenko A. Development of a software module for the analysis of the state of the forest using remote sensing data of the Earth. IOP Conference Series: Earth and Environmental Science. vol. 574. pp. 012080. DOI:10.1088/1755-1315/574/1/012080