World Applied Sciences Journal 7 (Special Issue of Computer & IT): 45-53, 2009
ISSN 1818-4952
© IDOSI Publications, 2009

Fast Neural Intrusion Detection System Based on Hidden Weight Optimization Algorithm and Feature Selection

Mansour Sheikhan and Amir Ali Sha'bani

Department of Electrical Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran

Abstract: Intrusion detection is known as an essential component to secure the systems in information and communication technology (ICT). In this paper, two mechanisms are used to achieve a fast intrusion detection system (IDS): 1) the training speed of neural attack classifier is improved by using output weight optimization-hidden weight optimization (OWO-HWO) training algorithm, 2) a feature relevance analysis is performed to decrease the number of input features and size of neural classifier. Experimental results show that the proposed system improves classification rates, especially for remote-to-local (R2L) attack category and is effective in terms of detection rate (DR) and cost per example (CPE). False alarm rate (FAR) of the proposed system is comparable with other intrusion detection systems, as well.

Key words: Intrusion detection . neural model . fast training . feature selection

INTRODUCTION

Intrusion detection is known as an essential component to secure the systems in information and communication technology (ICT). To coordinate and define a common framework for intrusion detection system (IDS), a general architecture is defined by Intrusion Detection Working Group (IDWG). This architecture is based on four functional modules: Event (E), Database (D), Analysis (A) and Response (R) [1].

Information events are acquired by the E blocks and based on the type of information source, IDS may be either host-based or network-based. Host-based IDS examines data held on individual computers that serve as hosts and network-based IDS examines data exchanged between computers. The D blocks store information from E blocks for subsequent processing. The A blocks are processing modules for analyzing events and detecting potential hostile behavior. Based on the type of analysis carried out, IDS may be misuse-based [2, 3] or anomaly-based [4, 5]. The R block executes a response, if any intrusion occurs.

User's activities are compared with the known behaviors of attackers attempting to penetrate a system in misuse-based IDS. The detection techniques that are used in misuse-based IDS can be classified into three main categories: statistical-based [6], knowledge-based [7, 8] and machine learning (e.g. Bayesian networks [9], artificial neural networks (ANNs) [3, 10-13], fuzzy logic [14], genetic algorithms [15], clustering [16], decision trees [17, 18] and hybrid systems [19-21]).

On the other hand, anomaly-based IDS seeks to detect activities that vary from established patterns for users and their detection techniques can also be classified into three main categories: statistical-based [22], knowledge-based [1] and machine learning (e.g. Bayesian networks [23], Markov models [24], ANNs [25, 26], fuzzy logic [14, 27], genetic algorithms [28] and clustering and outlier detection [29]).

In 1999, recorded network traffic from the Defense Advanced Research Project Agency (DARPA) dataset [30] was summarized into network connections with 41 features per connection. This formed the benchmark provided by the international knowledge discovery and data mining group (KDD) [31]. There are four main categories of attacks given in the KDD 99: denial-of-service (DoS), probe, remote-to-local (R2L) and user-to-root (U2R).

In this paper, a network-based IDS is proposed for misuse detection in which two mechanisms are used to improve the training speed and performance of a neural IDS. The first mechanism is output weight optimization-hidden weight optimization (OWO-HWO) algorithm, in which many simple error functions are minimized in the training phase [32, 33]. As the second mechanism, a feature relevance analysis is used to decrease the number of applied input features to neural classifier.

So, by reducing the size of ANN and improving its training speed and convergence, a fast IDS is achieved which is effective in terms of detection rate (DR) and false alarm rate (FAR), as well.

Corresponding Author: Dr. Mansour Sheikhan, P.O. Box 11365/4435,
Post-Graduate Center, South Tehran Branch, Islamic Azad University, Iran

The remainder of this paper is organized as follows. Section 2 provides the KDD dataset details. The foundations of OWO-HWO and the method of feature ranking are described in Section 3. Simulation and experimental results are discussed in Section 4. Conclusions are also drawn in Section 5.

KDD DATASET

The KDD dataset consists of three components: "Whole KDD", "Corrected KDD" and "10% KDD" (Table 1).

There are multiple attack types for each main attack category. Table 2 lists the attack categories along with the attack types in the "10% KDD" dataset.

The analysis in this paper is performed on the "10% KDD" dataset. It is reminded that each connection in KDD is characterized by 41 features (listed in Appendix A). These features are grouped into four categories: basic features, content features, time-based traffic features and host-based traffic features.

Basic features can be derived from packet headers without inspecting the payload. In the second category of features, domain knowledge is used to assess the payload of the original transmission control protocol (TCP) packets. Time-based traffic features are designed to capture properties that mature over a two-second temporal window. Host-based traffic features utilize a historical window estimated over the number of connections, instead of time. Therefore, they are designed to assess attacks which span in intervals longer than 2 seconds.

OWO-HWO ALGORITHM AND FEATURE RANKING

A critical problem in multilayer perceptron (MLP) neural networks has been the long training time required. Several fast training techniques, that require the solution of sets of linear equations, have been devised [34, 35].

In output weight optimization-backpropagation (OWO-BP), linear equations are solved to find output weights and backpropagation is used to find hidden weights [36]. Unfortunately, backpropagation is not a very effective method for updating hidden weights [37].

A non-batching approach for finding all the MLP weights, by minimizing separate error functions for each hidden unit, is proposed in [32]. Although this technique is more effective than backpropagation, it does not use OWO to find the output weights optimally. The idea of minimizing a separate error function for each hidden unit is adapted to find the hidden weights and have termed as hidden weight optimization (HWO) [36].

Table 1: Number of samples in KDD 99 datasets

KDD dataset	DoS	Probe	R2L	U2R	Normal
Whole	3883370	41102	1126	52	972780
Corrected	229853	4166	16347	70	60593
10%	391458	4107	1126	52	97277

Table 2: Attack types in "10% KDD" dataset

Category	Type
DoS	smurf, neptune, back, teardrop, pod, land
Probe	satan, ipsweep, portsweep, nmap
R2L	warezclient, guess_passwd, warezmaster, imap, ftp_write, multihop, phf, spy
U2R	buffer_overflow, rootkit, loadmodule, perl

In this paper, OWO-HWO algorithm is used as a superior technique in terms of convergence to standard OWO-BP. In this section, the notations and error functions in a MLP network are reviewed first and then the OWO-HWO algorithm is described.

In a MLP, if the *j*th unit is a hidden unit, then the net input $net_p(j)$ and the output activation $O_p(j)$ for the *p*th training pattern are:

$$\mathrm{net}_p(j) = \sum_i w(j,i).x_p(i) \quad (1)$$

$$O_p(j) = f(\mathrm{net}_p(j)) \quad (2)$$

where the *i*th unit is in any previous layer and $w(j,i)$ denotes the weight connecting the *i*th unit to the *j*th unit. For the *k*th output unit, the net input $net_{op}(k)$ for the *p*th training pattern and the output activation $O_{op}(k)$, with the linear property assumption of the output units, are:

$$\mathrm{net}_{op}(k)=\sum_i w_o(k,i).O_p(i) \quad (3)$$

$$O_{op}(k)=\mathrm{net}_{op}(k) \quad (4)$$

where $w_o(k,i)$ denotes the output weight connecting the *i*th unit to the *k*th output unit.

In order to train a neural network in batch mode, the error for the *k*th output unit is defined as:

$$E(k)=\frac{1}{N_v}\sum_{p=1}^{N_v}[T_p(k)-O_{op}(k)]^2 \quad (5)$$

in which N_v is the number of training patterns $\{(x_p, T_p)\}$.

In this paper, the conjugate gradient approach is used to minimize *E(k)* [36]. For hidden weight changes, it is desirable to optimize the hidden weights by minimizing separate error functions for each hidden unit. By minimizing many simple error functions, instead of a large one, it is hoped that the training speed and convergence can be improved. The desired hidden net function can be approximated by a current net function plus a net change. That is, for *j*th unit and *p*th pattern, a desired net function can be constructed as [32]:

$$net_{pd}(j)=net_p(j)+Zd_p(j) \qquad (6)$$

where *Z* is the learning factor and $d_p(j)$ for output units and hidden units are as follows, respectively:

$$d_p(j)=f'(net_j).[T_p(j)-O_p(j)] \qquad (7)$$

$$d_p(j)=f'(net_j).\sum_n d_p(n)w(n,j) \qquad (8)$$

Similarly, the hidden weights can be updated as:

$$w(j,i)\longleftarrow w(j,i)+Z.e(j,i) \qquad (9)$$

where *e(j,i)* is the weight change and serves the same purpose as the negative gradient in backpropagation. By defining an objective function in terms of mean squared error (MSE) for the *j*th unit as:

$$E_d(j)=\sum_{p=1}^{N_v}[d_p(j)-\sum_i e(j,i).O_p(i)]^2 \qquad (10)$$

and taking the gradient of $E_\delta(j)$ with respect to the weight changes and setting it to zero, the following linear equations are achieved:

$$\sum_i e(j,i).R_{oo}(i,m)=\frac{-\partial E}{\partial w(j,m)} \qquad (11)$$

where

$$R_{oo}(i,m)=\sum_{p=1}^{N_v} O_p(i).O_p(m) \qquad (12)$$

The steps of OWO-HWO algorithm are listed in Fig. 1.

```
1.  Initialize all weights and thresholds.
2.  Increase n by 1 and stop if $n>N_{it}$. % $N_{it}$=Number of iterations
3.  Apply training pattern and calculate the output activation.
4.  Use the conjugate gradient approach to minimize error.
5.  If MSE(n) > MSE(n-1)
        $Z \leftarrow Z\downarrow$ % Reduce the value of Z (Learning factor)
        Reload the previous best hidden weights
        Go to step 9
6.  If MSE (n) ≤ MSE (n-1)
        Accumulate the cross-correlation $R_{\delta o}(m)$ and auto-correlation $R_{oo}(m)$ for hidden units:
        $R_{do}(m)=\sum_{p=1}^{N_v} d_p(j).O_p(m)$
        $R_{oo}(m)=\sum_{p=1}^{N_v} O_p(i).O_p(m)$
7.  Solve linear equations for hidden weight changes:
        $\sum_i e(j,i).R_{oo}(i,m)=R_{do}(m)$
8.  Calculate the learning factor as
        $Z=\frac{-0.05E}{\sum_j\left[\sum_i \frac{\partial E}{\partial w(j,i)}.e(j,i)\right]}$
9.  Update the hidden weights as:
        $w(j,i)\longleftarrow w(j,i)+Z.e(j,i)$
10. Go to step 2
```

Fig. 1: OWO-HWO algorithm

Feature ranking is an important issue in intrusion detection, as well. Elimination of less significant features lowers the size of ANN and speeds up the computations. Logistic regression was used to rank the features based on the Chi-square values for different subsets selected using best subset selection model [38].

The higher the Chi-square value, the higher is the ranking. In Table 3, the ranking results of the Chi-square test on KDD dataset are listed for the 25 most significant features.

SIMULATION AND EXPERIMENTAL RESULTS

80,507 records from KDD 99 dataset were chosen for our experiments. 49,403 of them built the training set and the rest (31,104 samples) built the test set. This dataset has the same distribution of attacks as "10% KDD" dataset (Table 4).

Features in the KDD datasets have different forms: discrete, continuous and symbolic, with significantly

Table 3: Chi-square values of the 25 most significant features with respect to the attack class

Feature	DoS	Probe	R2L	U2R
dst_host_diff_srv_rate	1334.8	3686.3	1114.1	2532.0
rerror_rate	1016.3	2734.5	1016.5	613.4
dst_host_srv_rerror_rate	967.9	2707.7	586.2	301.1
srv_rerror_rate	805.5	2515.7	583.3	244.9
dst_ host_rerror_rate	732.8	2252.0	560.6	207.8
diff_srv_rate	551.7	1228.3	350.1	39.9
dst_host_same_srv_rate	449.2	793.3	311.1	39.2
service	438.8	588.7	249.5	36.7
dst_host_srv_count	433.0	546.1	239.2	32.6
logged_in	363.6	427.2	141.8	25.1
dst_host_srv_diff_host_rate	353.5	422.3	141.3	25.0
srv_count	344.9	123.4	141.2	15.5
same_srv_rate	336.9	91.8	126.1	15.3
protocol_type	328.7	84.6	125.0	10.7
num_compromised	308.4	70.4	116.0	10.3
wrong_fragment	275.6	68.6	99.8	6.4
dst_host_same_src_port_rate	274.0	65.4	78.3	6.3
hot	240.3	33.9	53.1	6.2
srv_serror_rate	188.9	20.3	46.8	6.2
dst_host_srv_serror_rate	129.1	19.6	45.5	6.2
is_guest_login	121.4	18.2	37.1	3.8
serror_rate	102.2	17.7	33.9	3.4
src_bytes	101.5	8.3	27.7	3.4
duration	52.4	7.6	26.1	2.9
dst_host_serror_rate	45.4	7.4	26.0	2.7

Table 4: Size of the training and test datasets

Class	Number of training samples	Number of test samples
DoS	39146	22985
Probe	411	417
R2L	113	1619
U2R	6	24
Normal	9727	6059

Table 5: Performance of MLP-based IDS with 20 input features and N_h nodes in hidden layer after 50 training epochs

Number of hidden nodes	MSE-training samples	MSE-test samples	Detection rate (%)
10	0.0044	0.0447	97.42
15	0.0038	0.0406	97.75
20	0.0031	0.0407	97.96
25	0.0020	0.0350	99.20
30	0.0020	0.0284	99.58
35	0.0018	0.0291	99.52

varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, preprocessing is required.

Symbolic-valued features, such as protocol_type (3 different symbols), service (70 different symbols) and flag (11 different symbols) are mapped to integer values ranging from *0* to *S-1*, where *S* is the number of symbols. Continuous features having smaller integer value ranges like wrong_fragment [0,3], urgent [0,14], hot [0,101], num_failed_logins [0,5], num_compromised [0,9], num_root [0,7468], num_file_creations [0,100], num_shells [0,5], num_access files [0,9], count [0,511], srv_count [0,511], dst_host_count [0,255] and dst_host_srv_count [0,255] are also scaled linearly to the range [0,1].

Logarithmic scaling (base 10) is applied to three features spanned over a very large integer range, namely duration [0,58329], src_bytes [0,1.3billion] and dst_bytes [0,1.3billion], to reduce the ranges to [0,4.77] and [0,9.11], respectively. Other features are either Boolean, like logged_in, having binary values, or continuous, like diff_srv_rate, in the range of [0,1] and no scaling is needed for these features. So, each of the mapped features are linearly scaled to the range [0,1].

Before discussing about the results of experiments, it seems necessary to mention the standard metrics that have been developed for evaluating IDS. Detection rate (DR) and false alarm rate (FAR) are the two most common metrics. DR is computed as the ratio between the number of correctly detected attacks and the total number of attacks, while FAR is computed as the ratio between the number of normal connections that is incorrectly misclassified as attacks and the total number of normal connections.

Based on the results of feature ranking, four experiments are performed by selecting 41, 25, 20 and 15 features as the input vector of MLP, respectively. The MLP in each of these experiments has five linear output neurons (representing 4 attack categories and 1 normal category). The number of MLP hidden nodes in each of these experiments is selected as 35, 33, 30 and 25, respectively. This selection is based on monitoring the MSE on test data for different values of hidden nodes, e.g. as shown in Table 5 for the case of 20 input features.

The convergence of OWO-HWO algorithm is fast and the DR of neural classifier is more than 99%, as shown in Table 6 for the case 41 input features.

The effect of the input feature-vector size reduction on the performance of MLP, when using OWO-HWO as training algorithm, is shown in Fig. 2 for each of the mentioned experiments. The structure of MLP is shown as [x y z] in the legend of figure, representing number of input, hidden and output nodes, respectively.

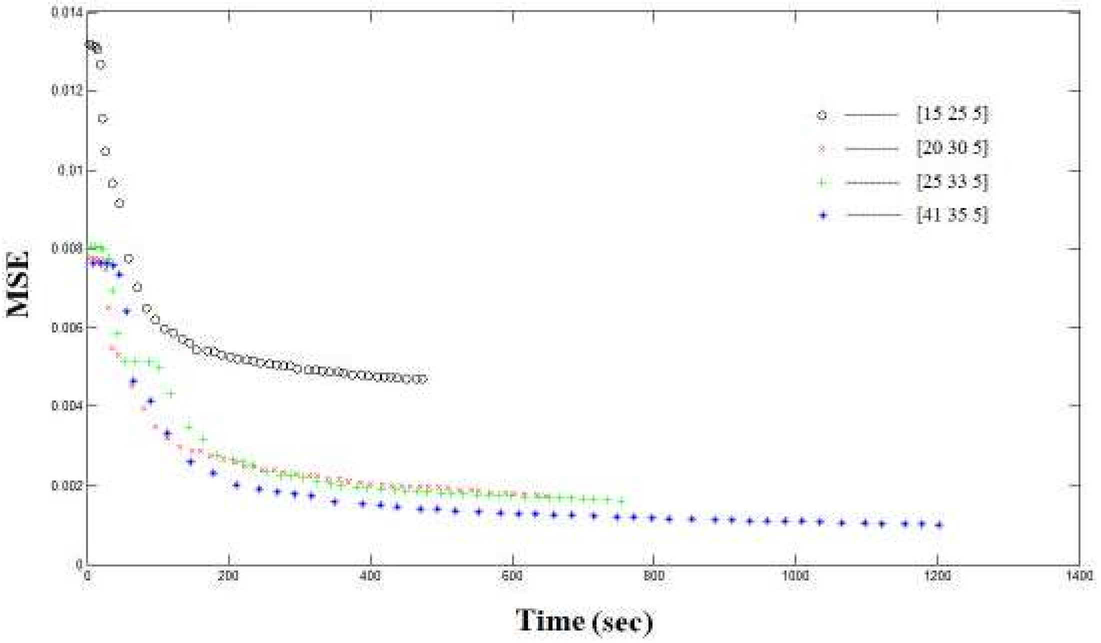

Fig. 2: Performance of MLP-based IDS, using feature-selection and OWO-HWO training algorithm

Table 6: Performance of MLP-based IDS, using OWO-HWO training algorithm with 41 input features

Number of epochs (sec)	MSE-training samples	MSE-test samples	Detection rate (%)	Training time
25	0.0014	0.0202	99.68	410
50	0.0010	0.0186	99.79	820
75	0.0008	0.0188	99.78	1250
100	0.0008	0.0188	99.78	1750
150	0.0007	0.0193	99.76	2700
200	0.0007	0.0186	99.79	3650

Table 7: Cost matrix values for KDD dataset

	Predicted				
Actual	DoS	Probe	R2L	U2R	Normal
DoS	0	1	2	2	2
Probe	2	0	2	2	1
R2L	2	2	0	2	4
U2R	2	2	2	0	3
Normal	2	1	2	2	0

As shown in Fig. 2, selection of only 25 or even 20 of the most important features does not degrade the performance, noticeably.

For the purpose of classifier algorithm evaluation, another comparative measure is cost per example (CPE) [39]. CPE is calculated using the following formula:

$$CPE=\frac{1}{T}\sum_{i=1}^{m}\sum_{j=1}^{m}CM(i,j).C(i,j) \qquad (13)$$

Table 8: Confusion matrix of MLP-based IDS with 41 input features

	Predicted				
Actual	DoS	Probe	R2L	U2R	Normal
DoS	22976	2	5	0	2
Probe	10	403	1	0	3
R2L	5	1	1602	0	11
U2R	1	0	10	12	1
Normal	5	8	2	0	6044

Table 9: Confusion matrix of MLP-based IDS with 25 input features

	Predicted				
Actual	DoS	Probe	R2L	U2R	Normal
DoS	22964	1	8	0	12
Probe	32	374	4	0	7
R2L	4	2	1598	0	15
U2R	3	10	5	2	4
Normal	10	8	14	0	6027

Table 10:Confusion matrix of MLP-based IDS with 20 input features

	Predicted				
Actual	DoS	Probe	R2L	U2R	Normal
DoS	22956	5	19	0	5
Probe	30	371	8	0	8
R2L	15	2	1585	0	17
U2R	1	10	9	1	3
Normal	16	7	21	0	6015

Table 11: Confusion matrix of MLP-based IDS with 15 input features

	Predicted				
Actual	DoS	Probe	R2L	U2R	Normal
DoS	22899	8	70	0	8
Probe	21	362	25	0	9
R2L	25	0	1589	0	5
U2R	1	0	19	1	3
Normal	7	10	1	0	6041

where *CM* and *C* are confusion matrix and cost matrix, respectively. In Eq. (13), *T* represents the total number of test instances and *m* is the number of classes in classification. *CM* is a square matrix in which each column corresponds to the predicted class, while rows correspond to the actual classes. An entry at row *i* and column *j*, *CM(i,j)*, represents the number of misclassified instances that originally belong to class *i*, although incorrectly identified as a member of class *j*. The entries of the primary diagonal, *CM(i,i)*, stand for the number of properly detected instances. Cost matrix is similarly defined, as well and entry *C(i,j)* represents the cost penalty for misclassifying an instance belonging to class *i* into class *j*. Cost matrix values employed for the KDD 99 classifier learning contest are shown in Table 7 [31].

The confusion matrices for each of neural classifiers, with the mentioned architectures, are reported in Table 8-11, respectively.

The performance of the proposed classifiers, with reduced number of input features and also fast OWO-HWO training algorithm, has been compared with some other machine learning methods, tested on the KDD dataset, as well (Table 12).

It should be noted that most of the machine learning algorithms offered an acceptable level of classification rate for DoS and Probe attack categories and demonstrated poor performance on the R2L and U2R categories [41].

As shown in Table 12, the proposed classifiers demonstrate better performance in R2L category. Classification rate for U2R attack category is the best, when using 41 input features. DR and CPE of the proposed classifiers are better than mentioned models, too. FAR of the proposed classifier with 41 input features is the best among mentioned models, as well.

CONCLUSIONS

In this paper, two mechanisms were used concurrently to achieve fast IDS. As the first mechanism, the training speed of a neural attack classifier was improved by using OWO-HWO algorithm. As the second mechanism, a feature relevance analysis was performed to decrease the number of input features and size of neural classifier.

Experimentations showed that the resulting reduced-size neural classifiers have improved classification rates, especially for R2L attack category, as compared to other machine learning algorithms. The proposed approach was evaluated effective in terms of detection rate (DR) and cost per example (CPE), and its false alarm rate (FAR) was comparable to other machine learning methods, as well.

Table 12: Performance comparison of different models for intrusion detection

	Classification rate							
Model	DoS	Probe	R2L	U2R	Normal	DR	FAR	CPE
MLP-41 features-HWO	99.96	96.64	98.95	50.0	99.75	99.79	0.25	0.0046
MLP-25 features-HWO	99.91	89.69	98.70	8.33	99.47	99.57	0.53	0.0095
MLP-20 features-HWO	99.87	88.97	97.90	4.17	99.27	99.47	0.73	0.0119
MLP-15 features-HWO	99.63	86.81	98.15	4.17	99.70	99.23	0.30	0.0132
Winner of KDD in 2000 [17]	97.10	83.30	8.40	13.20	99.50	91.80	0.60	0.2331
Runner up of KDD in 2000 [18]	97.50	84.50	7.30	11.80	99.40	91.50	0.60	0.2356
PNrule [39]	96.90	73.20	10.70	6.60	99.50	91.10	0.40	0.2371
ESC-IDS [40]	99.50	84.10	31.50	14.10	98.20	95.30	1.90	0.1579

Appendix A: Description and type of 41 features in KDD dataset

Feature	Description	Type
duration	Duration of the connection (in seconds)	continuous
protocol_type	Type of the connection protocol	discrete
service	Service on the destination	discrete
flag	Status flag of the connection	discrete
src_bytes	Number of bytes sent from source to destination	continuous
dst_bytes	Number of bytes sent from destination to source	continuous
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	Number of wrong fragments	continuous
urgent	Number of urgent packets	continuous
hot	Number of "hot" indicators	continuous
num_failed_logins	Number of failed logins	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	Number of "compromised" conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
num_root	Number of "root" accesses	continuous
num_file_creations	Number of file creation operations	continuous
num_shells	Number of shell prompts	continuous
num_access_files	Number of operations on access control files	continuous
num_outbound_cmds	Number of outbound commands in a FTP session	continuous
is_host_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a "guest" login; 0 otherwise	discrete
count	Number of connections to the same host as the current connection in the past two seconds	continuous
srv_count	Number of connections to the same service as the current connection in the past two seconds	continuous
serror_rate	Percent of connections that have "SYN" errors (same-host connections)	continuous
srv_serror_rate	Percent of connections that have "SYN" errors (same-service connections)	continuous
rerror_rate	Percent of connections that have "REJ" errors (same-host connections)	continuous
srv_rerror_rate	Percent of connections that have "REJ" errors (same-service connections)	continuous
same_srv_rate	Percent of connections to the same service	continuous
diff_srv_rate	Percent of connections to different services	continuous
srv_diff_host_rate	Percent of connections to different hosts	continuous
dst_host_count	Number of connections having the same destination host	continuous
dst_host_srv_count	Number of connections having the same destination host and using the same service	continuous
dst_host_same_srv_rate	Percent of connections having the same destination host and using the same service	continuous
dst_host_diff_srv_rate	Percent of different services on the current host	continuous
dst_host_same_src_port_rate	Percent of connections to the current host having the same src port	continuous
dst_host_srv_diff_host_rate	Percent of connections to the same service coming from different hosts	continuous
dst_host_serror_rate	Percent of connections to the current host that have an S0 error	continuous
dst_host_srv_serror_rate	Percent of connections to the current host and specified service that have an S0 error	continuous
dst_host_rerror_rate	Percent of connections to the current host that have an RST error	continuous
dst_host_srv_rerror_rate	Percent of connections to the current host and specified service that have an RST error	continuous

REFERENCES

1. Garcia-Teodoro, P., J. Diaz-Verdejo, G. Macia-Fernandez and E. Vazquez, 2009. Anomaly-Base Network Intrusion Detection: Techniques, Systems and Challenges. Journal of Computers and Security, 28: 18-28.
2. Ilgun, K., R.A. Kemmerer and P.A. Porras, 1995. State Transition Analysis: A Rule-Based Intrusion Detection Approach. IEEE Transactions on Software Engineering, 21: 181-199.
3. Beghdad, R., 2007. Training All the KDD Dataset to Classify and Detect Attacks. Journal of Neural Network World, 17: 81-91.
4. Chen, Z., H. Wang, B. Yang, L. Wang and R. Sun, 2007. A FDRS-Based Data Classification Method Used for Abnormal Network Intrusion Detection. In the Proceedings of the IEEE International Conference on Natural Computation, 2: 375-380.

5. Ma, R., Y. Liu and X. Lin, 2007. Hybrid QPSO Based Wavelet Neural Networks for Network Anomaly Detection. In the Proceedings of the IEEE Workshop on Digital Media and its Application in Museum and Heritage, pp: 442-447.
6. Biermann, E., E. Cloeteand and L.M. Venter, 2001. A Comparison of Intrusion Detection Systems. Journal of Computers and Security, 20: 676-683.
7. Han, S.J. and S.B. Cho, 2003. Detecting Intrusion with Rule-Based Integration of Multiple Models. Journal of Computers and Security, 22: 613-623.
8. Novikov, D., R.V. Yampolskiy and L. Reznik, 2006. Artificial Intelligence Approaches for Intrusion Detection. In the Proceedings of the IEEE Conference on Systems, Applications and Technology, pp: 1-8.
9. Joshi, M.V., R.C. Agrawal and V. Kumar, 2001. Mining Needless in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction. In the Proceedings of the ACM SIGMOD Conference on Management of Data, pp: 91-102.
10. Debar, H. and B. Dorizzi, 1992. An Application of Recurrent Network to an Intrusion Detection System. In the Proceedings of the International Joint Conference on Neural Networks, pp: 478-483.
11. Kayacik, G., N. Zincir-Heywood and M. Heywood, 2003. On the Capability of an SOM-Based Intrusion Detection System. In the Proceedings of the International Joint Conference on Neural Networks, pp: 1808-1813.
12. Golovko, V., L. Vaitsekhovich, P. Kochurko and U. Rubanau, 2007. Dimensionality Reduction and Attack Recognition Using Neural Network Approaches. In the Proceedings of the International Joint Conference on Neural Networks, pp: 2734-2739.
13. Beghdad, R., 2008. Critical Study of Neural Networks in Detecting Intrusions. Journal of Computers and Security, 27: 168-175.
14. Dickerson, J.E., 2000. Fuzzy Network Profiling for Intrusion Detection. In the Proceedings of the North American Fuzzy Information Processing Society (NAFIPS) International Conference, pp: 301-306.
15. Lin, Y., K. Chen and X. Liao, 2004. A Genetic Clustering Method for Intrusion Detection. Journal of Pattern Recognition, 37: 924-927.
16. Denning, D.E., 1987. An Intrusion-Detection Model. IEEE Transactions on Software Engineering, 13: 222-232.
17. Pfahringer, B., 2000. Winning the KDD 99 Classification Cup: Bagged Boosting. Journal of SIGKDD Explorations, 1: 65-66.
18. Levin, I., 2000. KDD Classifier Learning Contest: LLSoft's Results Overview. Journal of SIGKDD Explorations, 1: 67-75.
19. Mukkamala, S., G. Janoski and A.H. Sung, 2002. Intrusion Detection Using Neural Networks and Support Vector Machines. In the Proceedings of the International Joint Conference on Neural Networks, pp: 1702-1707.
20. Abadeh, M.S., J. Habibi and C. Lucas, 2005. Intrusion Detection Using a Fuzzy Genetic-Based Learning Algorithm. Journal of Network and Computer Application, 30: 414-428.
21. Tajbakhsh, A., M. Rahmati and A. Mirzaei, 2009. Intrusion Detection Using Fuzzy Association Rules. Journal of Applied Soft Computing, 9: 462-469.
22. Ye, N., S.M. Emran, Q. Chen and S. Vilbert, 2002. Multivariate Statistical Analysis of Audit Trials for Host-Based Intrusion Detection. IEEE Transactions on Computers, 51: 810-820.
23. Kruegel, C., D. Mutz, W. Robertson and F. Valeur, 2003. Bayesian Event Classification for Intrusion Detection. In the Proceedings of the Annual Computer Security Applications Conference, pp: 14-23.
24. Yeung, D.Y. and Y. Ding, 2003. Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models. Journal of Pattern Recognition, 36: 229-243.
25. Cansian, A.M., E. Moreira, A. Carvalho and J.M. Bonifacio, 1997. Network Intrusion Detection Using Neural Networks. In the Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, pp: 276-280.
26. Ramadas, M., S. Ostermann and B. Tjaden, 2003. Detecting Anomalous Network Traffic with Self-Organizing Maps. Recent Advances in Intrusion Detection, RAID, Lecture Notes in Computer Science (LNCS), 2820: 36-54.
27. Gomez, J. and D. Dasgupta, 2002. Evolving Fuzzy Classifiers for Intrusion Detection. In the Proceedings of the IEEE Workshop on Information Assurance, pp: 68-75.
28. Song, D., M.I. Heywood and A.N. Zincir-Heywood, 2005. Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection. IEEE Transactions on Evolutionary Computation, 9: 225-239.
29. Sequeira, K. and M. Zaki, 2002. ADMIT: Anomaly-Based Data Mining for Intrusions. In the Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp: 386-395.

30. The 1998 Intrusion Detection Off-Line Evaluation Plan, MIT Lincoln Lab., Information Systems Technology Group, 25 Mar. 1998 (Available on http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt).
31. 1999 KDD Cup Competition (Available on http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html).
32. Scalero, R.S. and N. Tepedelenlioglu, 1992. A Fast New Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Signal Processing, 40: 202-210.
33. Yu, C., M.T. Manry, J. Li and P.L. Narasimha, 2006. An Efficient Hidden Layer Training Method for Multilayer Perceptron. Neurocomputing, 70: 525-535.
34. Sartori, M.A. and P.J. Antsaklis, 1991. A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks. IEEE Transactions on Neural Networks, 2: 467-471.
35. Rohani, K., M.S. Chen and M.T. Manry, 1992. Neural Subnet Design by Direct Polynomial Mapping. IEEE Transactions on Neural Networks, 3: 1024-1026.
36. Chen, H.H., M.T. Manry and H. Chandrasekaran, 1996. A Neural Network Training Algorithm Utilizing Multiple Sets of Linear Equations. In the Conference Record of the 30th Asilomar Conference on Signals, Systems and Computers, pp: 1166-1170.
37. Werbos, P., 1988. Backpropagation: Past and Future. In the Proceedings of the International Conference on Neural Networks, pp: 343-353.
38. Tamilarasan, A., S. Mukkamala, A.H. Sung and K. Yendrapalli, 2006. Feature Ranking and Selection for Intrusion Detection Using Artificial Neural Networks and Statistical Methods. In the Proceedings of the International Joint Conference on Neural Networks, pp: 4754-4761.
39. Agrawal, R. and M.V. Joshi, 2000. PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection). IBM Research Division, Report No. RC-21719.
40. Nadjaran Toosi, A. and M. Kahani, 2007. A Novel Soft Computing Model Using Adaptive Neuro-Fuzzy Inference System for Intrusion Detection. In the Proceedings of the IEEE International Conference on Networking, Sensing and Control, pp: 834-839.
41. Sabhnani, M. and G. Serpen, 2004. Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set. Journal of Intelligent Data Analysis, 6: 1-13.