Protocol Type Based Intrusion Detection Using RBF Neural Network

Aslıhan Özkaya aozkaya@mevlana.edu.tr
Faculty of Engineering/Computer Engineering Department
Mevlana University
Konya, 42003, TURKEY

Bekir Karlık bkarlik@mevlana.edu.tr
Faculty of Engineering/Computer Engineering Department
Mevlana University
Konya, 42003, TURKEY

Abstract

Intrusion detection systems (IDSs) are very important tools for information and computer security. In IDSs, the publicly available KDD'99, has been the most widely deployed data set used by researchers since 1999. Using a common data set has provided to compare the results of different researches. The aim of this study is to find optimal methods of preprocessing the KDD'99 data set and employ the RBF learning algorithm to apply an Intrusion Detection System.

Keywords: RBF Network, Intrusion Detection, Network Security, KDD Dataset.

1. INTRODUCTION

With the growth in the use of computer and internet, the number of computer and network attacks has increased. Therefore many companies and individuals are looking for solutions and deploying software's and systems such as intrusion detection systems (IDSs) to overcome with the network attacks. Due to the high need of such systems, many researchers' attentions are attracted by IDS [1-4].

KDDCUP'99 is the mostly widely used data set for the evaluation of intrusion detection systems [5-8]. Tavallaee et al. [5] examined and questioned the KDDcup'99 data set, and revised it by deleting the redundant records and applied 7 learners on the new data set. The seven learners are J48 decision tree learning, Naive Bayes, NBTree, Random Forest, Random Tree, Multilayer Perceptron (MLP), and Support Vector Machine (SVM). They also labeled each record with its difficulty and present it publicly on their website. Sabhnani and Serpen [6] evaluated the performance of pattern recognition and machine learning algorithms on KDD'99 data set. In their paper the following algorithms are tested; MLP, Gaussian classifier, K-means, nearest cluster algorithm, incremental RBF, Leader algorithm, Hyper sphere algorithm, Fuzzy ARTMAP and C4.5 decision tree. They mainly focused on comparing the performances of the applied classifiers for the attack categories. Bi et al. [7] picked 1000 records from KDDcup'99. They used Radial Basis Function (RBF) Network on the selected data after preprocessing it. Sagiroglu et al. [8] applied Leverberg Marquardt, Gradient Descent, and Resilient Back-propagation on the KDD'99 data set.

The other machine learning algorithms are also used for intrusion detection. Yu and Hao [9] presented an ensemble approach to intrusion detection based on improved multi-objective genetic algorithm. O. A. Adebayo et al. [10] have presented a method that uses Fuzzy-Bayesian to detect real-time network anomaly attack for discovering malicious activity against computer network. Shanmugavadivu and Nagarajan [11] presented fuzzy decision-making module to build the system more accurate for attack detection using the fuzzy inference approach. Ahmed and Masood [12] proposed a host based intrusion detection architecture using RBF neural network which obtained better detection rate and very low training time as compared to other machine learning algorithms.

In this study, the KDD'99 data set has been pre-processed and divided into three sections according their protocol type; TCP, UDP and ICMP. Conversion of string to numerical value is applied in three different ways and is saved as three different data sets. RBF neural network learning algorithm is used for each data set.

2. DATA SET DESCRIPTION AND PREPROCESSING

2.1. KDD'99 Data Set

In our experiments we have used the KDD'99 data set which has been developed based on the data captured in DARPA'98 [13]. The KDD'99 data set (corrected version) has over 1 million training data and over 300 thousands of test data. Each data consists of 41 attributes and one target (see Figure 1). Targets indicate the attack names. The data set covers over 30 different attack types as outputs which belong to one of four major categories; Denial of Service, User to Root, Remote to Local, and Probing Attacks (see Table 1) [4].

```
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,
1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00
,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00
,1.00,0.00,0.00,39,39,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00
,1.00,0.00,0.00,49,49,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,112,9,1.00,1.00,0.00,0.00,
0.08,0.07,0.00,255,9,0.04,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,18,1.00,1.00,0.00,0.0
0,0.15,0.07,0.00,255,18,0.07,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,130,8,1.00,1.00,0.00,0.00,
0.06,0.06,0.00,255,8,0.03,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.0
0,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf
```

FIGURE 1: Sample data of KDDcup

2.2. Deleting Repeated Data

We used MATLAB on a PC with 4 GB of memory and 2.27 GHz of processing speed. Because of the limited memory and speed of the PC we decided to decrease the number of data of the training sets to around 6,000. Therefore repeated data has been deleted. After this process 614,450 of training and 77,290 of testing data was left.

2.3. Dividing Data into Three Sections

As shown in Figure 1, one of the attributes is the protocol type which is TCP, UDP or ICMP. We divided both training and testing data into these three protocol types in order to train and test our data separately.

Table 2 shows the number of remaining data after repeated data has been deleted. The number of training data for TCP and UDP is still large. Therefore some number of data was deleted randomly. The data to be deleted were chosen mostly from "normal" labeled data.

There were also some attacks in testing data set that were not in the training data set. Since RBF is a supervised learning technique, we had to train the network for all attacks which are going to be tested. Therefore we copied some of these attacks into the training data set. But the testing data sets were untouched (see Table 3).

Category	Attack Name	TEST	TRAIN
Normal	normal.	595,797	60,593
Remote to Local (R2L)	ftp_write.	8	3
	guess_passwd.	53	4,367
	httptunnel.	0	158
	imap.	12	1
	multihop.	6	18
	named.	0	17
	phf.	3	2
	sendmail.	0	17
	snmpgetattack.	0	7,741
	snmpguess.	0	2,406
	warezmaster.	20	1,602
	worm.	0	2
	xlock.	0	9
	xsnoop.	0	4
Probing	ipsweep.	7,579	306
	mscan.	0	1,053
	nmap.	2,316	84
	portsweep.	2,782	354
	saint.	0	736
	satan.	5,393	1,633
Denial of Service (DoS)	apache2.	0	794
	back.	2,002	1,098
	land.	17	9
	mailbomb.	0	5
	neptune.	204,815	58,001
	pod.	40	87
	processtable.	0	759
	smurf.	227,524	164,091
	teardrop.	199	12
	udpstorm.	0	2
User to Root (U2R)	buffer_overflow.	5	22
	loadmodule.	2	2
	perl.	2	2
	ps.	0	16
	rootkit.	0	13
	sqlattack.	0	2
	xterm.	0	13
	TOTAL	1,048,575	311,029

TABLE 1: Attack categories for test and training data sets

Protocol Name:	TCP		UDP		ICMP	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
Normal	529,517	43,908	28,435	3,770	1,324	233
Attack	50,499	27,214	866	922	3,809	1,242
Total	**580,016**	**71,122**	**29,301**	**4,692**	**5,133**	**1,475**

TABLE 2: Data information after separating it into three different protocol types.

Protocol Name:	TCP		UDP		ICMP	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
Normal	2,698	43,908	5,134	3,770	1,325	233
Attack	3,302	27,214	942	922	3,838	1,242
Total	**6,000**	**71,122**	**6,076**	**4,692**	**5,163**	**1,475**

TABLE 3: Data information after deleting some data randomly and copy some attacks from test to train data set

2.4. Normalization

In order to normalize the data, we need to make sure that all values are in numerical formats. There are three inputs (attributes) and one output given in string formats. One of these attributes is the protocol name. Since the data is divided according their protocol names, there is no need to convert the protocol types to numeric values. We deleted the column which belongs to the protocol name, since one set of data has always the same protocol name. The output has been converted to 1 if it is an attack and to 0 (zero) if it is a normal communication data. The other two attributes are the service and flag name. They are converted to numerical values with respect to their frequency in the test set. We applied three different conversion techniques. We differentiated these techniques by naming them as Type-A, Type-B and Type-C.

Flag	Frequency	Type-A	Type-B	Type-C
SF	6765	1	11	10
S0	3986	2	10	6
REJ	1488	3	9	2
RSTR	633	4	8	5
RSTO	307	5	7	3
S3	272	6	6	9
SH	182	7	5	11
S1	58	8	4	7
S2	29	9	3	8
RSTOS0	25	10	2	4
OTH	4	11	1	1

TABLE 4: Conversions of Flag Names to Numerical Values (for TCP Data set)

Service Name	Frequency	Type-A	Type-B	Type-C
private	3156	1	57	40
http	3012	2	56	17
telnet	1669	3	55	50
ftp	910	4	54	13
other	864	5	53	35
ftp_data	821	6	52	14
smtp	765	7	51	44
finger	507	8	50	12
pop_3	401	9	49	38
imap4	227	10	48	19
auth	177	11	47	1
sunrpc	113	12	46	47
IRC	110	13	45	20
time	88	14	44	51
domain	52	15	43	8
remote_jc	40	16	42	41
sql_net	39	17	40	45
ssh	39	18	41	46
X11	32	19	39	56
discard	29	20	36	7
echo	29	21	37	9
systat	29	22	38	49
gopher	28	23	34	15
link	28	24	35	25
iso_tsap	26	25	28	21
mtp	26	26	29	27
netbios_n	26	27	30	30
netstat	26	28	31	32
pop_2	26	29	32	37

Service Name	Frequency	Type-A	Type-B	Type-C
rje	26	30	33	42
daytime	25	31	24	6
netbios_dgm	25	32	25	29
supdup	25	33	26	48
uucp_path	25	34	27	53
bgp	24	35	20	2
ctf	24	36	21	5
netbios_ssn	24	37	22	31
whois	24	38	23	55
csnet_ns	23	39	17	4
name	23	40	18	28
vmnet	23	41	19	54
hostnames	22	42	15	16
Z39_50	22	43	16	57
nntp	18	44	13	34
pm_dump	18	45	14	36
ldap	15	46	12	24
uucp	10	47	11	52
login	9	48	10	26
nnsp	7	49	7	33
printer	7	50	8	39
shell	7	51	9	43
kshell	6	52	6	23
courier	5	53	3	3
exec	5	54	4	11
http_443	5	55	5	18
efs	4	56	2	10
klogin	3	57	1	22

TABLE 5: Conversions of Service Names to Numerical Values (for TCP Data)

Service Name	Frequency	Type A	Typle B	Type C
domain_u	3679	1	5	1
ntp_u	1373	2	4	3
private	795	3	3	5
other	152	4	2	2
tftp_u	0	5	1	4

TABLE 6: Conversions of Service Names to Numerical Values (for UDP Data)

In Type-A, we gave the highest number to the attribute with most frequency and 1 with less frequency. We did this in the opposite way for Type-B, and random numerical values were given in Type-C.

Service Name	Frequency	Type A	Typle B	Type C
eco_i	2990	5	1	1
ecr_i	1727	4	2	5
urp_i	270	3	3	2
urh_i	146	2	4	4
tim_i	0	1	5	3

TABLE 7: Conversions of Service Names to Numerical Values (for ICMP Data)

There is only on Flag name in ICMP and UDP data sets; therefore the columns belong to the flag names are deleted for both ICMP and UDP. There were also some other columns with only one value. These columns (inputs) are also deleted because they have no influence on the outputs. The final number of inputs and outputs of the data sets can be seen in Table 8.

Protocol Name:	TCP	UDP	ICMP
Input #	31	20	18
Output #	1	1	1

TABLE 8: Number of Output and Input after preprocessing the data sets

After converting text to integer and deleting columns with same data, the data sets are normalized.

3. RBF NETWORK

Radial Basis Function (RBF) Network is a type of Artificial Neural Network for supervised learning [14]. It uses RBF as a function which is usually Gaussian and the outputs are inversely proportional to the distance from the center of the neuron [15]. The traditional RBF function network can be seen in Figure 2. MATLAB provides functions to implement RBF Network within their Neural Network Toolbox. The training function newrb() and simulation function sim() is used to train and test the network [15-16].

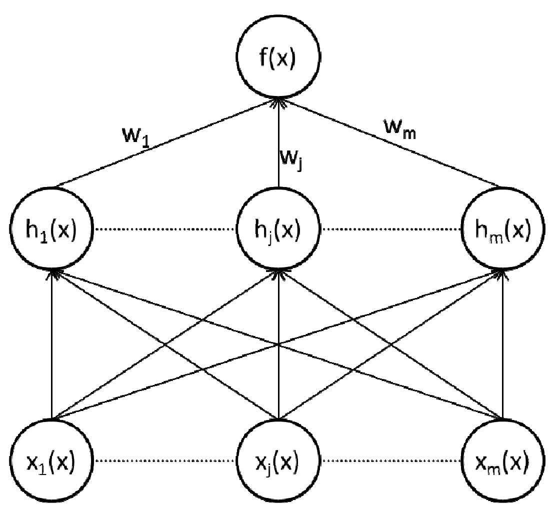

FIGURE 2: A single layer radial basis function network

4. EXPERIMENTS

The experiments are applied for all three types of string to integer conversations to see if there is any difference. For all trainings the maximum number of neurons is set as 1000.

4.1. Training Results

Training results can be seen in Table 9, 10 and 11. The results are shown as mean squared error (MSE) which represents the performance (or accuracy).

The best training results are Type-C for TCP, Type-A for UDP and Type-B for ICMP.

TCP TRAINING			
# of Neurons	Type-A	Type-B	Type-C
50	0.02702	0.02718	0.02985
100	0.01540	0.01575	0.01648
150	0.01127	0.01097	0.01275
200	0.00900	0.00869	0.00927
250	0.00772	0.00722	0.00680
500	0.00321	0.00335	0.00295
750	0.00165	0.00157	0.00151
1000	0.00101	0.00097	0.00089

TABLE 9: Training results (MSE) of the TCP data set

UDP TRAINING			
# of Neurons	Type-A	Type-B	Type-C
50	0.00410	0.00410	0.00259
100	0.00175	0.00175	0.00138
150	0.00108	0.00108	0.00085
200	0.00079	0.00079	0.00065
250	0.00062	0.00062	0.00057
500	0.00033	0.00034	0.00032
750	0.00023	0.00022	0.00022
1000	0.00018	0.00020	0.00021

TABLE 10: Training results (MSE) of the UDP data set

ICMP TRAINING			
# of Neurons	Type-A	Type-B	Type-C
50	0.00617	0.00617	0.00625
100	0.00382	0.00382	0.00264
150	0.00183	0.00184	0.00211
200	0.00183	0.00182	0.00210
250	0.00183	0.00182	0.00208
500	0.00099	0.00053	0.00080
750	0.00087	0.00036	0.00052
1000	0.00073	0.00030	0.00043

TABLE 11: Training results (MSE) of the ICMP data set

The training performances are plotted to set the results of one type of conversion against the other types of conversion (see Figure 3, 4, and 5). It can be seen that the learning performances for each type is very close to each other.

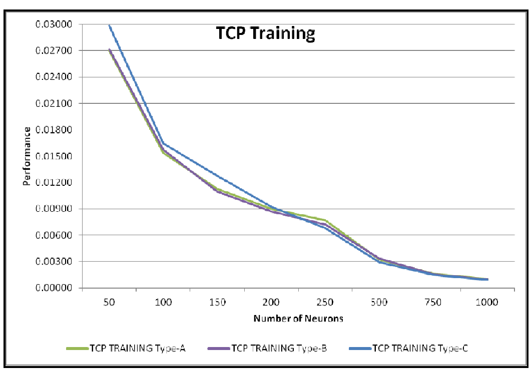

FIGURE 3: Graphical training results of the TCP data set

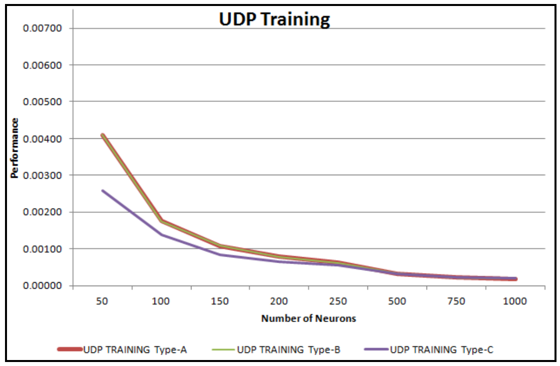

FIGURE 4: Graphical training results of the UDP data set

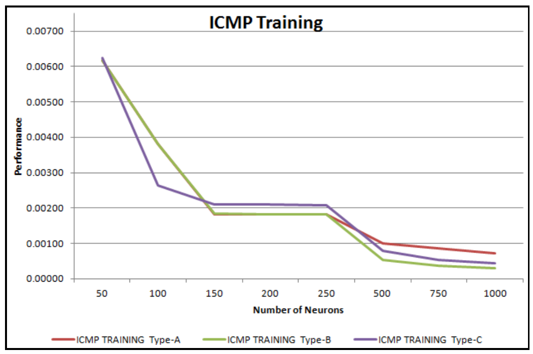

FIGURE 5: Graphical training results of the ICMP data set

4.2. Testing Results

The best performance is obtained with Type-C conversion of all three data sets. The MSE and FAR values are 95.65%, 79.39%, 62.96% and 2.6%, 4.72%, 7.85% for TCP, UDP and ICMP respectively.

Figure 6 and Figure 7 show the comparison of the performances and False Alarm Rates for TCP, UDP and ICMP testing data sets with their three different Type of conversions (Type-A, Type-B and Type-C).

		Type-A	Type-B	Type-C
TCP	Performance	90.86%	94.28%	95.65%
	False Alarm	3.45%	3.38%	2.60%
UDP	Performance	61.42%	65.09%	63.96%
	False Alarm	8.78%	10.29%	7.85%
ICMP	Performance	88.95%	83.46%	79.39%
	False Alarm	16.31%	15.88%	4.72%

TABLE 12: Testing Results for TCP, UDP and ICMP data sets.

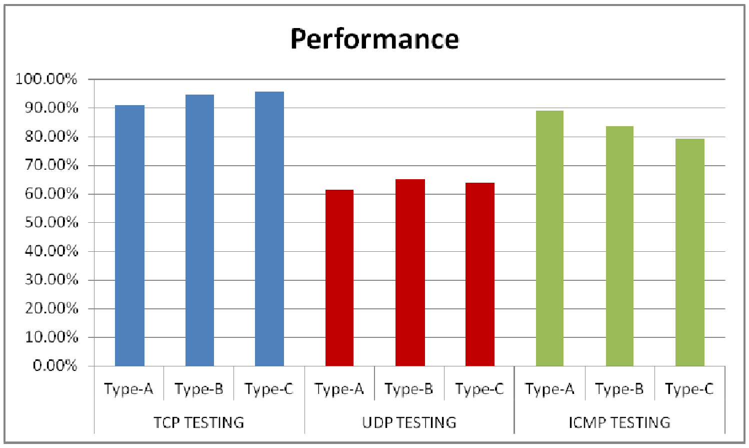

FIGURE 6: Testing result of Performances.

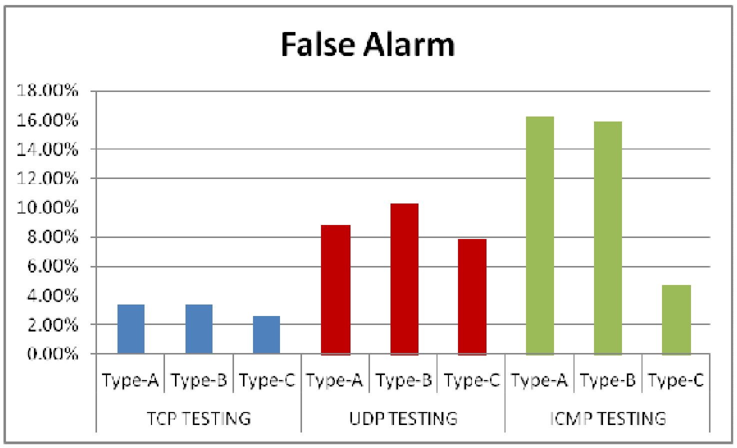

FIGURE 7: Testing results of False Alarm Rates (FARs)

False alarm rates of all type of conversions have been observed similar for both TCP and UDP testing datasets. The FAR results for ICMP testing dataset have an appreciable amount of differences. It is observed that FARs are over 15% for type-A and type-B while it is less than 5% for type-C.

According to experimental results, false alarms are always the highest percentage for type-A and type-B data sets. This shows that converting strings to numbers with respect to their frequency may not be a good solution.

Learning and testing the TCP dataset gives good results and can still be improved, while the results for UDP and ICMP datasets are very poor. More training data or more attributes may improve the results.

In this paper the overall MSE and FAR values are calculated as 93.42% and 2.95% respectively. These results are better than the results in some other papers where different methods have been applied. For instance in [5] the performance values are 81.66%, 92.79%, 92.59%, 92.26% and 65.01% with Naïve Bayes, Random Forest, Random Tree, Multi-Layer Perceptron, and SVM respectively. Again in the same paper the performance values of some other methods (J48 and NB Tree) are very close to our overall results which are 93.82% and 93.51% respectively. In [7] the performance is 89% and FAR is 11% with RBF neural network.

5. CONCLUSION AND DISCUSSION

In this study, the most widely used data set (KDD'99) is pre-processed. Some duplicated data is deleted then training and testing data is divided into three sections according the protocol types. Afterwards strings in the data sets are converted to numerical values using three different techniques as Type-A, Type-B and Type-C. All preprocessed data sets are trained and tested with RBF network using MATLAB toolbox. It is experimented that the preprocessing phase plays an important role on the performance of the learning system.

It is also observed that applying learning algorithms on divided data (with respect to their protocol types) enables better performance.

As mentioned in the testing results section, the accuracy of testing results is more satisfied than the literature studies. However this proposed learning algorithm and alternative string to integer converting techniques need more research to find optimal solutions.

6. REFERENCES

[1] K. Ilgun, R.A Kemonerer and P.A Porras, "State Transition Analysis: A Rule Based Intrusion Detection Approach", IEEE Transaction on Software Engineering, Vol.21(3), March 1995, pp.181-199.

[2] S. Capkun. Levente Buttyan. "Self-Organized Public Key Management For Mobile Ad Hoc Networks", IEEE Transactions on Mobile Computing, Vol. 2(1), January -March 2003, pp. 52-64.

[3] Yao, J. T., S.L. Zhao, and L.V. Saxton, "A Study On Fuzzy Intrusion Detection", In Proceedings of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, SPIE, Vol. 5812,pp. 23-30, Orlando, Florida, USA, 2005.

[4] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D.McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation," in Proc. DARPA Inf. Survivability Confer. Exposition (DISCEX), Vol. 2, 2000, pp. 12–26.

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in Proc. 2009 IEEE International Conference on Computational Intelligence for Security and Defense Applications. pp. 53-58.

[6] M. Sabhnani and G. Serpen, "Application of Machine Learning Algorithms to KDD 1999 Cup Intrusion Detection Dataset within Misuse Detection Context", International Conference on Machine Learning, Models, Technologies and Applications Proceedings, Las Vegas, Nevada, June 2003, pp. 209-215.

[7] J. Bi, K. Zhang, X. Cheng, "Intrusion Detection Based on RBF Neural Network", Information Engineering and Electronic Commerce, 2009, pp. 357 - 360

[8] Ş.Sagiroglu, E. N. Yolacan, U. Yavanoglu, "Designing and Developing an Intelligent Intrusion Detection System", Journal of the Faculty of Engineering and Architecture of Gazi University, Vol. 26 (2), June 2011, pp. 325-340.

[9] Yu Y, and Huang Hao, "An Ensemble Approach to Intrusion Detection Based on Improved Multi-Objective Genetic Algorithm", Journal of Software, Vol.18 (6), June 2007, pp.1369-1378.

[10] O. Adetunmbi Adebayo, Zhiwei Shi, Zhongzhi Shi, Olumide S. Adewale, "Network Anomalous Intrusion Detection using Fuzzy-Bayes", IFIP International Federation for Information Processing, Vol. 228, 2007, pp. 525-530.

[11] R. Shanmugavadivu and N. Nagarajan, "An Anomaly-Based Network Intrusion Detection System Using Fuzzy Logic", International Journal of Computer Science and Information Security, Vol. 8 (8), November 2010, pp. 185-193.

[12] U. Ahmed and A. Masood, "Host Based Intrusion Detection Using RBF Neural Networks", Emerging Technologies, ICET 2009, 19-20 Oct. 2009, pp. 48-51.

[13] The UCI KDD Archive, University of California, KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 28, 1999, [Feb 2012].

[14] J. Mark and L. Orr, "Introduction to Radial Basis Function Networks", Technical Report, April 1996.

[15] Z. Caiqing, Q. Ruonan, and Q. Zhiwen, "Comparing BP and RBF Neural Network for Forecasting the Resident Consumer Level by MATLAB," International Conference on Computer and Electrical Engineering, 2008 (ICCEE 2008), 20-22 Dec. 2008, pp.169-172.

[16] A. Iseri and B. Karlık, "An Artificial Neural Networks Approach on Automobile Pricing", Expert Systems with Applications, Vol. 36 (2), March 2010, pp. 2155-2160.