Journal of Engineering and Applied Sciences 11 (Special Issue 2): 3279-3284, 2016
ISSN: 1816-949X

Intrusion Detection System Framework Based on Machine Learning for Cloud Computing

[1]Mohammed Hasan Ali, [2]Mohamad Fadli Zolkipli, [1]Ngahzaifa Binti Ab. Ghani and
[1]Mohammed Abdulameer Mohammed
[1]Faculty of Computer Systems and Software Engineering, School of Computing,
Universiti Malaysia Pahang, Pahang, Gambang, Malaysia
[2]Universiti Utara Malaysia, Pahang, Changlun, Malaysia

Abstract: Security is a rich research area and there are many solutions create to protect the information and make the systems safer. Intrusion detection is one of the powerful solutions in security. Current-day network Intrusion-Detection Systems (IDS) have several flaws such as low detection rates and high rates of false positive alerts and the need for constant human intervention and tuning. This research focus on design intrusion detection system based hybrid Extreme Learning Machine (ELM) and Genetic Algorithm (GA). ELM is randomly generated the parameters because that this research proposes use GA to provide the ELM parameters to find the best classifier that work as IDS. This model will test in Knowledge Discovery and Data Mining Contest 1999 (KDD Cup 99) and Network Security Laboratory-Knowledge Discovery and Data Mining (NSL-KDD) data set. Evaluate the performance of the proposed hybrid by using standard evaluate matrices.

Key words: IDS, proposed, hybrid, ELM, KDD

INTRODUCTION

Cloud computing is a new computing paradigm that makes computing resources be accessible anywhere, anytime and pay only for the services accessed (Vecchiola *et al*., 2009). With the emergence of cloud computing, organizations and individuals can cut down the expenditure on computer resources drastically. Cloud computing to be acceptable for everyone several issues including security, legal and economic factors need to be resolved first. Users and companies are reluctant to move their resources and computing to the cloud as the cloud security system has not been properly addressed yet (Firdhous *et al*., 2011).

Cloud Computing has successfully managed to advertise itself as one of the fastest growing service models on the Internet. Many large scale (IT) providers, such as and IBM company. The National Institute of Standard and Technology (NIST) defines cloud computing as the model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g., Networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interact (Rimal *et al*., 2011). The cloud offers several benefits (Patel *et al*., 2013) like fast deployment, pay-for-use, lower costs, scalability, rapid provisioning, rapid elasticity, ubiquitous network access, greater resiliency, hypervisor protection against network attacks, low-cost disaster recovery and data storage solutions, on demand security controls, real time detection of system tampering and rapid reconstitution of services.

MATERIALS AND METHODS

Issues facing IDS: Extreme Learning Machine (ELM) has set of properties that make itattractive to be adopted for intrusion detection system in cloud environment (Chen *et al*., 2014). Firstly, this classification algorithm has an online version named online sequential extreme learning machine OSELM (Goyal, 2014; Huang *et al*., 2006). This online version enables the system to learn in a progressive way. And this fits with the incremental nature of data in cloud environment. In addition, fits with the needs of intrusion detection to be adaptable with all new nature of attacks that the attacker finds. Secondly, ELM approach has proven itself to be faster than the common support vector machine SVM (Fossaceca *et al*., 2011). Thirdly, this approach behaves well in multi-class problems in which intrusion detection system is regarded as one of them. This is why many researchers have recommended ELM based approaches for this application.

Corresponding Author: Mohammed Hasan Ali, Faculty of Computer Systems and Software Engineering, School of Computing, Universiti Malaysia Pahang, Pahang, Gambang, Malaysia

Despite the variety of approaches that were built on ELM, researchers have agreed that ELM does not provide the optimal solution in the process of making the classification decision (Fossaceca *et al*., 2011; Goyal, 2014).

More specifically, Extreme learning machine has a random nature in the step of initiating the hidden layer neurons (Huang *et al*., 2006).This creates infinite number of degrees of freedom and as a result. It causes range of diversity in the decision making if two different chance seeds were initiated in two extreme learning classifiers with the same exact structure. Some researchers have tried to exploit this fact by creating an ensemble of classifiers and developing a voting strategy under the concept decision fusion (Fossaceca *et al*., 2011). In perspective, ensemble of totally arbitrary ELM classifiers will not perform well unless an assumption of optimal consensus is made. In addition, many researchers use arbitrary strategies in the way of data partitioning into training and testing. Data unbalance is also a problematic nature through the application of the intrusion-detection system due to the fact of low frequency of some attacks and high frequency of others. Thus, reported accuracy of performance is not meaningful unless data is partitioned fairly not arbitrarily as in most approaches (Fossaceca *et al*., 2011). Moreover, classifier training has to ensure its adequate learning before enabling it in testing mode.

Any intrusion-detection system in cloud environment has to be aware of the fact that most attackers can develop new types of attacks continuously. Thus, the detection system has to be built based on learning capability in online mode similar to OSELM core systems. Therefore, improving ELM from the above declared perspectives will result in a significant improve in the overall intrusion detection system.

Extreme Learning Machine Based Genetic Algorithm (ELMGA): Since ELM just randomly chooses the input weights and hidden biases (a_i, b_i) (Zhu *et al*., 2005) much of learning time traditionally spent in tuning these parameters is saved. However, as the output weights are computed based on the prefixed input weights and hidden biases, there may exist a set of non-optimal or unnecessary input weights and hidden biases. ELM may require more unclear neurons than conventional tuning-based learning algorithms in some applications which may make ELM respond well to unknown testing data (Zhu *et al*., 2005). Many well performing machine learning approaches are not scalable to handle larger datasets such as those encountered in network intrusion detection (Fossaceca *et al*., 2011). Some researchers tried to use Incremental Bilateral two-Dimensional Principal Component Analysis (IB2DPCA) for gender classification. DB2 DPCA learns from a small pool of labeled data and then iteratively selects the most informative samples from the unlabeled set to increasingly improve the classifier accuracy. This reearch will let the GA select the inputs of ELM parameters after the initialization of a starting population; also the GA will choose the number of hidden neurons $\tilde{N}$. The main evolution loops each individual generation will compare according to the fitness function that the ELM will provide it. Where: a_i, b_i $i = \{1,\ldots\ldots,\tilde{N}\}$ parameters needs to find the hidden layer output matrix H as Eq. 1. But still the ELM need the inputs data and the target:

$$H\left(W_1,\ldots,W_{\tilde{N}},b_1,\ldots,b_{\tilde{N}},X_1,\ldots X_N\right) = \begin{bmatrix} g\left(W_1\times X_1+b_1\right) & \cdots & g\left(W_{\tilde{N}}\times X_1+b_{\tilde{N}}\right) \\ \vdots & \cdots & \vdots \\ g\left(W_1\times X_N+b_1\right) & \cdots & g\left(W_{\tilde{N}}\times X_N+b_{\tilde{N}}\right) \end{bmatrix}_{N\times\tilde{N}} \quad (1)$$

Where:

N = Numbers of (X_j, T_j)

X = The inputs data as attacks or normal that is use also tofind the hidden layer output matrix H

T = Target

Equation 2 to training data can get from both of the dataset (KDD CUP99, NSL-KDD) to get the output weights β as Eq . 3 and learn the algorithm in the training mode. Mapping chromosomes that provide as parameters mapping, mapping means describes the methods used to identify the locus for them and send to ELM find the fitness function. Where: X_j, T_j $j = \{1,\ldots\ldots,N\}$:

$$H\beta = T \quad (2)$$

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} \quad (3)$$

GA creates generations consecutive and individual, each of these generations providing as solving a problem. The survival of the fittest among the generations depends on the fitness for each of them. GA start working with population that has many chromosomes (solutions), in our framework this space has many of the hidden parameters (a_i, b_i) include the optimal parameters and also the number of the hidden parameters that GA sho uld provide them and send to the ELM. But still some inputs the ELM needs to work with it and give the output. As known the data set has two parts (training and tasting),

during the training mode ELM will get the inputs data and target from the data set to learns the ELM classifier, during the training mode will create virtual testing for the classifier that just training to beensure adequate training of classifier. In this research (Ratio) will divide the data set for 50% for every parts (training, testing) as many researchers work with dataset.

During the training mode and depend on the Eq.1-3 will find the output of the hidden layer (β) and every (β) will enter to virtual training and testing during the training mode to mitigation the over fitting problem, over fitting occurs when a statically model describes random error or noise instead of underlying relationship. Such as a model have too many parameters relative to the numbers of observations.

During a testing mode will give classifier just the input data (x) from the data set and classifier will give us the target as and testing mode has target, the compare between targets gives the accuracy of the classifiers. Every output from the testing mode in ELM will be fitness for each generation. In each generation during the testing mode, will give new chromosomes (accuracy between the target of our system and the target in testing mode) and set it to part of selection in the GA, this part has been to select two-parents chromosomes from a population according to their fitness, the better fitness is the bigger chance to be selected.

Next step in GA is a crossover and this method used to create second generations (child) depends on the first generation (parents). The main idea is to select the better parents (best survivors) in the hope that the better parents will produce better offspring. There are many Methods of selection of chromosomes for crossover, Fitness proportionate selectionis one of these methods and this work depend on the fitness function to work.

In fitness proportionate selection, as in all selection methods, the fitness function assigns fitness to possible solutions or chromosomes. This fitness level is used to associate a probability of selection with each individual chromosome. If j_j is the fitness of individual i in the population, its probability of being selected is:

$$P_i = \frac{f_i}{\sum_j^N =_1 f_j}$$

where, N is the numbers of individuals fitness in the population. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence, GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The purpose of mutation in gas is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter. A GA usually stops the work when there is no significant improvement in the value of fitness of the population from one generation to the next. And if the algorithms finds a suitably low fitness individual lower than a specified fitness threshold, also there are another factors let's GA to stop the work , the algorithm stops when the number of generation reaches the value of generations, time limits placed on the GA work.

Evaluate the results depend on use a specific table layout that visualization of the performance of an algorithm called confusion matrix. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. All correct guesses are located in the diagonal of the table, so it's easy to visually inspect the table for errors as they will be represented by values outside the diagonal. The table content reports the number of false positives, false negatives, true positives and true negatives. This allows more detailed analysis than mere proportion of correct guesses (accuracy). Where the accuracy is the proportion of true results among the total number of cases examined:

Accuracy = True positive+True negatives/False positives

false negatives+True positives+True negatives

In addition, can evaluate the results by find the recall and precision; recall is the proportion of really positive cases that are correctly predicted positive. Precision denotes the proportion of predicted positive cases that are correctly really positive as shown in Fig. 1 and Equations:

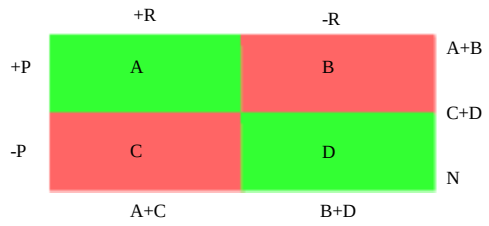

Fig. 1: Color coding indicates correct (green) and incorrect (pink) rates

$$\text{Recall} = A/(A+C)$$

$$\text{Precision} = A/(A+B)$$

This research will try to make the results fairer to reduce the impact of the unbalance in data set, so in this work create the following equation to apply it on the results:

$$M = \sum_{i=1}^{c} \text{Accuracy}_i S_i$$

Where:

i = The index of the subclass of attack which belongs to the main five attacks

Accuracy = The accuracy of classifying the subclass i

S_i = The size of the subclass i

RESULTS AND DISCUSSION

Before using the dataset we need to replace string values with integer values and to remove any duplicated records. Clearly, each field of string fields will have a number of integer values equal to the number of string values, for example the field "protocol type" field will have (1, 2 or 3). We have noted that the dataset approximately contains half million records which makes the comparison between a record and the other records is impossible process. To reduce the complexity of the process we can split the dataset into subsets that do not have any shared records and any record in any subset is impossible to be equal to another record in another subset. Easily we can do that by splitting the data into records that have the same values of "attack type", "protocol type", "service", "flag" and "logged in" fields (note:" logged in" field is a binary field with " field number 12"). After splitting the data, we will note that some subsets have a large number of records, so we still have a problem. In order to solve this problem, we compute the sum of each record (i.e., summation = field1+field2+…..+field42). If we find the number of values of summation of a subset equal to the number of records of this subset, we decide that all records are different from each other. Else, we need to compare only between the records that have equal summation. After removing the duplicated fields, we will find out that the number of records is 145586. The dataset, also is unbalanced, so to know how many records are remained in each classrefer to Table (1-4).

In addition to replacing string values and removing duplicated records, we need to scale the values of each feature to the range [0 1]. During scaling process we have to pay an attention that whatever a record is, all values of feature 20 "num outbound cmds" and all values of feature 21 "is host login" are zeros. Before saving the results, we have added the field 43 which has the value of mapping from attack type (1-23) to attack class (1-5).The resulted data after reading and each preprocessing process and any helpful data are saved in a mat file with name "TheKDDCup99.mat". The following table will show you the most important variables in the mat file:

Table 1: Number of readers in each class

Classes	Values
Normal class	87832
DDOS Attack	54572
Smurf	641
Teardrop	918
Pod	206
back	968
Land	19
Neptune	51820
Probing attack	2131
Ipsweep	651
Portsweep	416
Nmap	158
Satan	906
U2R attack	52
Perl	3
Loadmodule	9
buffer_overflow	30
Rootkit	10
R2L attack	999
ftp_write	8
guess_passwd	53
Imap	12
Multihop	7
Phf	4
Warezmaster	20
Warezclient	893
Spy	2

Table 2: Variables in the mat file

Variables	Function
Train data	A matrix contains the read data
Reduced train data	A matrix contains the data after removing the duplicated records
Scaled train data	A matrix contains the reduced data after scaling to the range [0 1]
Classes	A cell matrix contains the name of all attack types
Protocols	A cell matrix contains the name of all used protocols
Services	A cell matrix contains the name of all services found in the dataset
Flags	A cell matrix contains the name of all flags found in the dataset

Table 3: The training accuracy

Number of neurons in the hidden layer	Basic ELM	Online sequential ELM
150	0. 9912	0. 9883
180	0. 9914	0. 9894
200	0. 9920	0. 9890

Table 4: Testing accuracy

Number of neurons in the hidden layer	Basic ELM	Online Sequential ELM
150	0. 9258	0. 9224
180	0. 9255	0. 9231
200	0. 9283	0. 9224

For basic ELM with 150 neurons in the hidden layer "Training data set"

0.9974	0.0008	0.0003	0.0000	0.0015
0.0121	0.9879	0.0001	0	0.0000
0.1065	0.0066	0.8869	0	0
0.3269	0.0577	0.0192	0.4038	0.1923
0.1111	0.0030	0	0.0010	0.8849

For basic ELM with 180 neurons in the hidden layer "Training data set"

0.9972	0.0010	0.0003	0.0000	0.0014
0.0121	0.9878	0.0000	0	0.0001
0.0924	0.0009	0.9066	0	0
0.3269	0	0	0.4423	0.2308
0.1111	0.0080	0	0	0.8809

For basic ELM with 200 neurons in the hidden layer "Training data set"

0.9976	0.0009	0.0002	0.0000	0.0012
0.0108	0.9891	0.0000	0	0.0000
0.0953	0.0070	0.8977	0	0
0.2885	0.0385	0	0.4423	0.2308
0.1041	0.0040	0.0010	0.0020	0.888

For Online Sequential ELM with 150 neurons in the hidden layer "Training data set"

0.9971	0.0013	0.0003	0.0000	0.0013
0.0170	0.9822	0.0008	0	0.0001
0.1122	0.0136	0.8742	0	0
0.3462	0	0.0577	0.4231	0.1731
0.1732	0.0040	0.0010	0.0020	0.8198

For Online Sequential ELM with 180 neurons in the hidden layer "Training data set"

0.9971	0.0012	0.0004	0.0001	0.0013
0.0157	0.9841	0.0000	0	0.0001
0.1178	0.0023	0.8785	0	0.0014
0.5000	0	0	0.3077	0.1923
0.1211	0.0050	0.0030	0.0020	0.8689

For Online Sequential ELM with 200 neurons in the hidden layer "Training data set"

0.9972	0.0010	0.0003	0.0001	0.0014
0.0161	0.9832	0.0006	0	0.0001
0.1164	0.0174	0.8663	0	0
0.4423	0	0	0.3269	0.2308
0.1091	0.0040	0	0.0010	0.8859

For basic ELM with 150 neurons in the hidden layer "Testing data set"

0.9956	0.0015	0.0022	0.0001	0.0006
0.0557	0.9408	0.0035	0	0.0000
0.2241	0.1566	0.6137	0	0.0056
0.7857	0	0.0429	0.1429	0.0286
0.9388	0.0330	0.0199	0.0007	0.0075

For basic ELM with 180 neurons in the hidden layer "Testing data set"

0.9929	0.0038	0.0026	0.0000	0.0007
0.0523	0.9369	0.0107	0	0.0001
0.2054	0.1089	0.6853	0	0.0004
0.8000	0.0143	0	0.1714	0.0143
0.9722	0.0043	0.0134	0.0003	0.0098

For basic ELM with 200 neurons in the hidden layer "Testing data set"

0.9952	0.0016	0.0027	0.0001	0.0004
0.0554	0.9391	0.0055	0	0
0.2409	0.0884	0.6641	0	0.0067
0.5857	0.0571	0.0286	0.2571	0.0714
0.9258	0.0036	0.0268	0.0007	0.0432

For Online Sequential ELM with 150 neurons in the hidden layer "Testing data set"

0.9925	0.0046	0.0023	0.0000	0.0004
0.0539	0.9403	0.0057	0	0
0.2539	0.1641	0.5817	0	0.0004
0.6429	0	0.1857	0.1571	0.0143
0.9454	0.0072	0.0455	0.0007	0.0013

For Online Sequential ELM with 180 neurons in the hidden layer "Testing data set"

0.9866	0.0113	0.0018	0	0.0004
0.0536	0.9432	0.0030	0	0.0002
0.2278	0.0940	0.6782	0	0
0.8571	0.0143	0	0.0429	0.0857
0.9716	0.0059	0.0144	0	0.0082

For Online Sequential ELM with 200 neurons in the hidden layer "Testing data set"

0.9864	0.0107	0.0021	0.0001	0.0008
0.0470	0.9439	0.0062	0	0.0029
0.2539	0.0831	0.6626	0	0.0004
0.9429	0.0143	0	0.0286	0.0143
0.9863	0.0023	0.0088	0	0.0026

Fig. 2: Demonstrating the estimated and target values "Testing data set"

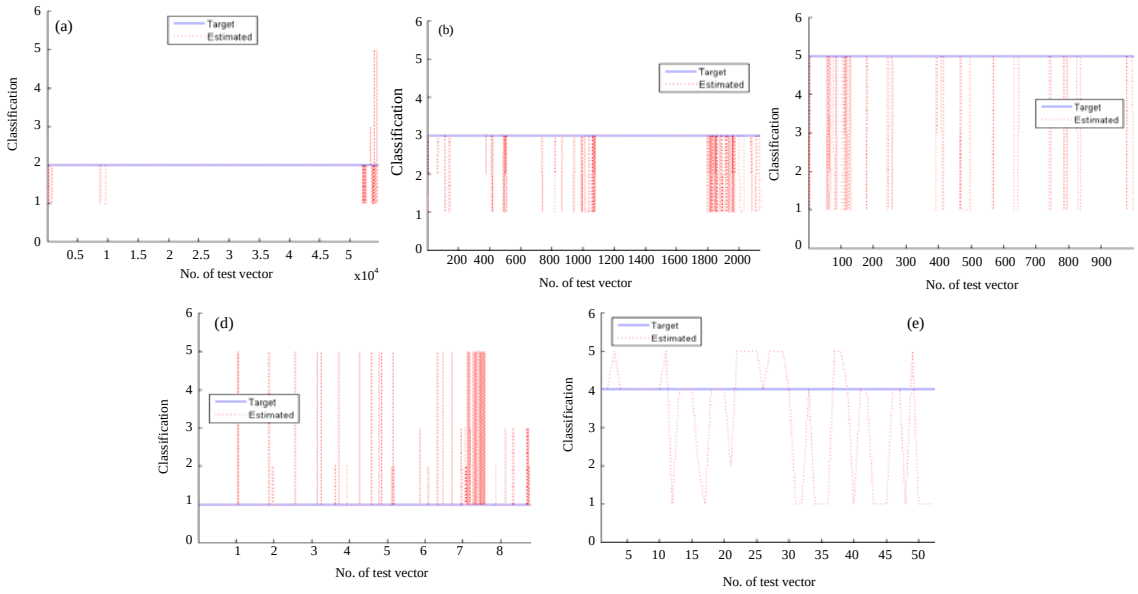

Fig. 3: For basic ELM with 200 neurons in the hidden layer training data set: a) Class 1 *Training*; b) Class 2 *Training; c) Class 3 *Training*; d) Class 4 *Training*; e) Class 5 *Training*

Tests: There are three result files for testing basic ELM and three result files for testing online sequential ELM. I have used "The KDD Cup 99" 10 % dataset for training and "corrected" dataset which you can download from the same website for testing. The following table will show you the training accuracy of each method. The following table will show you the testing accuracy of each method. We have known that there are five classes, so it is better to insert the confusion matrix for each case: To understand these values well, we will visualize them by using 0.1 and 2 demonstrating the estimated and target values. For basic ELM with 200 neurons in the hidden layer "Training data set". For basic ELM with 200 neurons in the hidden layer "Testing data set" (Fig. 2-4).

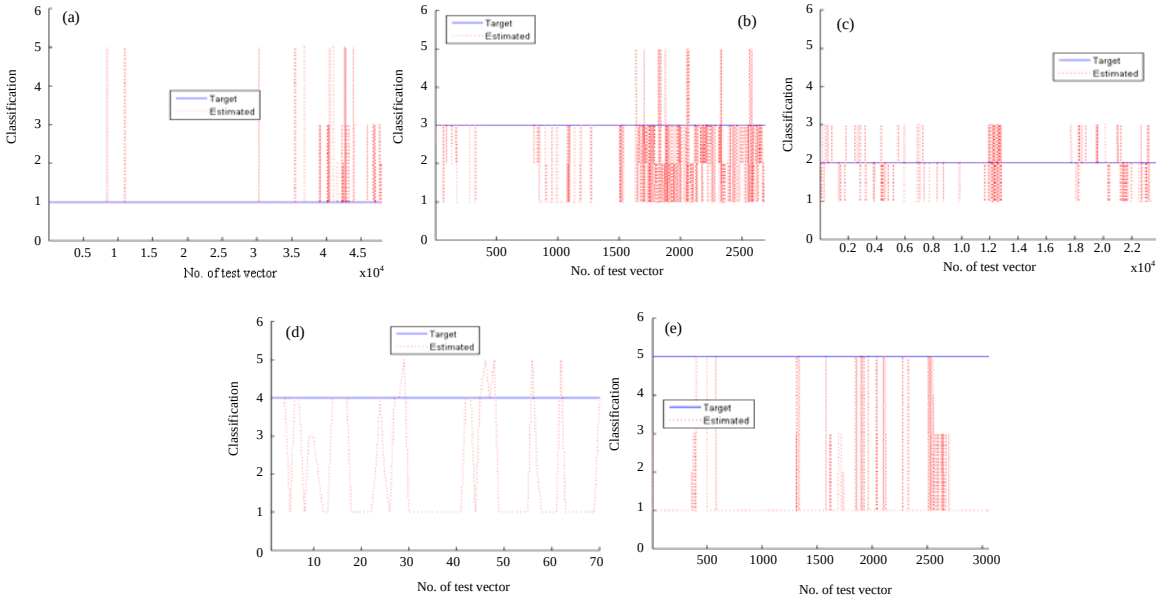

Fig. 4: For basic ELM with 200 neurons in the hidden layer "Testing data set": a) Class 1 *Testing*; b) Class 2 *Testing*; c) Class 3 *Testing*; d) Class 4 *Testing*; e) Class 5 *Testing*

CONCLUSION

To improve the accuracy of detection rate and faster in detect attacks our research used ELMGA algorithm based intrusion-detection system. Create randomly ELM parameters, until now, still open problem. This researcher used GA to provide these parameters for ELM to find the best classifier. The initial results shown the ELM performance as a single algorithm when work as IDS has some problems to detect some attacks when apply it with KDD99 as data set.

REFERENCES

Chen, C.C., L. Yuan, A. Greenberg, C.N. Chuah and P. Mohapatra, 2014. Routing-as-a-Service (RaaS): A framework for tenant-directed route control in data center. IEEE./ACM. Trans. Netw., 22: 1401-1414.

Firdhous, M., O. Ghazali, S. Hassan, N.Z. Harun and A. Abas, 2011. Honey bee based trust management system for cloud computing. Proceedings of the 3rd International Conference on Computing and Informatics (ICOCI 2011), June 8-9, 2011, ICOCI, Bandung, Indonesia, pp: 327-332.

Fossaceca, J.M., T.A. Mazzuchi and S. Sarkani, 2011. MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. Expert Syst. Appl., 42: 4062-4080.

Goyal, S., 2014. Public vs. private vs. hybrid vs. community-cloud computing: A critical review. Int. J. Comput. Netw. Inf. Secur., 6: 20-29.

Huang, G.B., Q.Y. Zhu and C.K. Siew, 2006. Extreme learning machine: Theory and applications. Neurocomputing, 70: 489-501.

Patel, A., M. Taghavi, K. Bakhtiyari and J.C. Junior, 2013. An intrusion detection and prevention system in cloud computing: A systematic review. J. Netw. Comput. Appl.,36: 25-41.

Rimal, B.P., A. Jukan, D. Katsaros and Y. Goeleven, 2011. Architectural requirements for cloud computing systems: An enterprise cloud approach. J. Grid Comput., 9: 3-26.

Vecchiola, C., S. Pandey and R. Buyya, 2009. High-performance cloud computing: A view of scientific applications. Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms and Networks, December 14-16, 2009, Kaohsiung, Taiwan, pp: 4-16.

Zhu, Q.Y., A.K. Qin, P.N. Suganthan and G.B. Huang, 2005. Evolutionary extreme learning machine. Pattern Recognition, 38: 1759-1763.