Adaptive Anomaly-Based Intrusion Detection System Using Fuzzy Controller

Farzaneh Geramiraz, Amir Saman Memaripour, and Maghsoud Abbaspour
(Corresponding author: Maghsoud Abbaspour)

Computer Engineering Department, Faculty of Electrical and Computer Engineering,
Shahid Beheshti University, G. C.,Evin, Tehran, Iran.
(Email: maghsoud@sbu.ac.ir)
(Received Mar. 8, 2011; received and accepted Oct. 28, 2011)

Abstract

The main feature of anomaly-based intrusion detection systems is detection of new attacks in the networks, even though numerous false alarms are caused in order to disregard this important feature. Although the previous improved detection models decrease the number of false alarms, but their efficiency due to changes in the normal behavior of the system is not reasonable. In this paper, we present an anomaly-based intrusion detection system to improve the system performance. Fuzzy rule-based modeling and fuzzy controller are used to create a detection model in the training phase and update this model in the test phase respectively. Moreover, the results of system's predictions buffered and presented to the system user later. After that, system user verifies these decisions and fuzzy controller tunes detection model using system user's feedbacks. We evaluated our system using the NCL dataset. Our dataset is a subset of KDD-99 dataset that does not contain any duplicated record. Furthermore, it includes a few difficult records that none of common classification methods in this area is able to classify them correctly. We have also proved that our test results can significantly increase the performance of the system about 20 percent using adaptive IDS. We also conclude that our proposed anomaly based intrusion detection increases the accuracy of the system about 15 percent.

Keywords: Adaptive anomaly-based intrusion detection, fuzzy-rule based modeling, fuzzy control

1 Introduction

Computer networks are one of the most important aspects of today's computer systems and used to transfer precious data between parties. As the importance of transferred data is exponentially increasing, numerous methods are published to protect end-systems and user-data against malicious activities [17, 27]. Although many breathtaking efforts have been done in order to prevent network attacks, such as using firewalls and signature-based intrusion detection systems, but none of them is complete to provide security of the network [9]. We should also note that the cost of improving security is undeniable. One of the methods to improve network security is using anomaly based intrusion detection systems that provide profiles of the target system's normal behavior. After that, they monitor the behavior of the network and compare it with these normal profiles to detect suspicious behavior. In case any abnormal behavior is detected, the intrusion detection system sends an alarm to the administrator. The advantage of using this method is detection of novel and internal attacks. Nevertheless, unlike the traditional methods, it creates a large number of false alarms. Moreover, providing an updated profile of the target system's normal behavior and keeping it up to date are two most important problems of anomaly based intrusion detection systems.

There are a variety of methods to provide profiles of the normal behavior of the target system such as: statistical [1, 15, 24], data mining based [2, 5, 8, 21, 23], and machine learning methods [10, 16, 19]. Here we used fuzzy rule-based modeling in order to model the normal profile of the system. It employs fuzzy rules and represents the output based on input variables. The most important issues in this manner are obtaining rules and fuzzy sets from input variables. For this reason, we have employed either genetic algorithm to find model's rules and FCM to find variables' fuzzy sets. In addition, we have considered a prediction confidence ratio for each taught rule. This parameter is the ratio of correctly detected samples to the total number of samples that have fired this rule.

Due to changes in the normal behavior of the network and appearance of new attacks, static models are not suitable, as they must be updated. In this work, prediction confidence ratio of rules is updated based on verified test results by using fuzzy controller. Using static models, it is not feasible to achieve higher than 55 percent performance, but if we employ adaptive models, we will be able to increase the performance more than 75 percent.

Our proposed adaptive anomaly detection system has some advantages such as using few but efficient parameters to update, online adaptation, tangible improvement in accuracy compared with non-adaptive methods and almost online adaptation. We have tested our proposed architecture

using NCL dataset [28]; which is a selective subset of KDD-99 dataset.

The remainder of this paper is organized as follows: in Section 2, previous works by other authors in the field of intrusion detection systems are discussed. Section 3 focuses on our dataset. In Sections 4 and 5, we describe the details of our proposed architecture for adaptive anomaly-based intrusion detection system. Experimental results obtained through this paper are given in Section 6. Eventually, in Sections 7 and 8, we present the conclusion and future works respectively.

2 Related Works

Two main approaches are used to update the detection model of intrusion detection systems. The first approach adds some sub models to the main model and the second one replaces the main model with a new model [26].

Authors in [12] used the first approach and added a new sub model to the constructed detection model. They have also used a data mining method (RIPPER) to distinguish between abnormal and normal activity. After that, they combined this model with each of the new models as well as some predefined rules and replaced results with the previous model in order to make a decision.

Published architectures in [11, 25, 26] have used second approach. In 2008, Zhenwei et al. introduced an adaptive tuning model [26]. Using SLIPPER, their proposed architecture learns rules with related prediction confidence ratio for each attack type and normal behavior. After that, they update the confidence ratio of the rules using fuzzy controller. They have showed that their adaptive architecture reduces false alarms by about twenty percent. Authors in [25] update the profile of normal behavior of the network in each time interval. They update the network's normal model when they are assured that the network is free from attacks. Time periods are considered in [25] and a separate model of the network is produced for each period. These sequential models are compared in order to see whether attacks have occurred in the network or not. In case the diversity of these models is greater than a threshold, it is considered that the network is not in a normal state and some attacks have happened. If the network recognized free from attacks, previous model would replace the normal model of the network. Otherwise, previous model will be used. Authors in [11] have established some attack and normal behavior clusters using Kernel-ART. In the evaluation stage, if an activity belongs to a cluster, the cluster center will be updated.

Authors in [13, 20] have also proposed architectures that use both methods to update the detection model of the network. Rasoulifard et al. provided an incremental and hybrid architecture. In the first stage, they have used misuse detection module to detect known attack patterns. After that, if this module could not classify an activity in any known attack pattern, it passes the activity to anomaly detection module. This activity is compared to the normal profile of the network and could be a normal activity or an attack. If it was normal, it would be used to update the normal profile; otherwise it will be saved in a database. New attack patterns are added to misuse detection module using this database. Liao [13] used a clustering unsupervised method to create an adaptive model. They have also used two update approaches. First of all, they establish clusters of the normal model. After that, if an activity belongs to a normal cluster, it is used to update the related cluster, otherwise it will belong to the nearest uncertain cluster. After a specified time, if the number of members of an uncertain cluster reached a threshold, that cluster would be added to the normal clusters. Otherwise, all members of uncertain clusters will be labeled as attack and all uncertain clusters will be destroyed. They have used Fuzzy ART and EFuNN as unsupervised clustering learning methods.

3 NCL Dataset

Due to most researchers have used KDD dataset for their works, Tavalaee [22] et al. have reviewed this dataset. They have expressed that one of the most important reasons that leads to the contradiction between the accuracy of research IDSs and commercial IDSs is KDD dataset, which is used in research area. They have two main reasons for this argument: First, there are lots of duplicates in training and testing records. The second one is the lack of difficulty measurement in records. Redundant records in training dataset prevents learning method from learning rare records such as U2R attack and causes wrong results in testing dataset. Lack of difficulty level can wrongly increase accuracy rate. Because of the simplicity of dataset, learning methods can provide high accuracy without any trouble.

In order to solve the first problem, they have detected redundant records in the train and test datasets. They have also found that 78 percent of the training records and 75 percent of the testing records are redundant. To analysis difficulty level of records, they defined a difficulty measurement. They randomly created three smaller subsets of the KDD train set; each contains fifty thousand records. Seven common learning methods are trained over these created train sets. After that, they employed these 21 learned machines to label the records of the entire KDD train and test sets, which provide 21 predicated labels for each record. Further, they annotate each record of the data set with a #successfulPrediction value, which is initialized by zero and if a learned machine classified it correctly, this value would be incremented by one. Presented test results in [22] shows that 98 percent of training records and 86 percent of testing records could be detected correctly by all 21 learned machines.

They have suggested new train and test datasets to solve mentioned problems. First, they removed all the redundant records in both train and test sets. Furthermore, in order to create a subset of the KDD data set, they have randomly sampled records from the #successfulPrediction

groups that are shown in Table 1. Indeed, the number of selected records of each group has an inverse proportion to the percentage of records in the original group. These train and test datasets called KDD-Train+ and KDD-Test+, because they contain a number of records from all groups and create new datasets.

New train and test datasets include 20% of KDD-Train+ and KDD-Test+ datasets without any record with #successfulPrediction equal to 21. Tables 1 and 2 show statistics of randomly selected records for train and test datasets.

They have compared the performance of the selected learning machines on three train and test datasets and showed that using original KDD test dataset can increase performance about 30 percent. In addition, KDD-Test+ dataset can increase performance about 15 percent spuriously. They also proposed that learning machines can obtain accuracy of more than 86 percent by using KDD+ dataset, but it is not possible to achieve the same accuracy percentage with KDD-Test-21. In this case, the performance was bounded to 65 percent. Note that we have used the last suggested dataset in our work. Review results of Tavalaee et al. are available in Table 3. (Training model is created using KDD-Train+).

4 Architecture Overview

Figure 1 describes our proposed architecture and its components in detail. The architecture is composed of four main components: a Detection Model Generator, an IDS Engine, a Fuzzy Model Tuner and a Buffer. Detection Model Generator is responsible for creating a detection model. The model consists of a number of fuzzy rules that each one has a prediction confidence ratio. IDS Engine classifies test records by using this model. After that**,** test classification results and parameters that are required for updating the detection model are stored in the Buffer. System user verifies test results that have a predefined delay and sends these verified results to the fuzzy model tuner**.** Fuzzy model tuner employs parameters needed for updating and verified results to update the confidence prediction ratio of effective rules in test sample's classification. Fix delay means if a test record was arrived at t, the model is updated at t+delay using fuzzy controller.

Table 1: Statistics of randomly selected records for train data set

	Distinct Records	***Percent***	***KDD-Train+***	***20%KDD-Train+***
0-5	407	0.04	407	81
6-10	768	0.07	767	173
11-15	6,525	0.61	6,485	1,336
16-20	58,995	5.49	55,757	11,107
21	1,008,297	93.8	62,557	12,495
Total	1,074,992	100	125,973	25,192

Table 2: Statistics of randomly selected records for test data set

	Distinct Records	**Percent**	**KDD-Train+**	**20%KDD-Train+**
0-5	589	0.76	585	585
6-10	847	1.1	838	838
11-15	3,540	4.58	3,378	3,378
16-20	7,845	10.15	7,049	7,049
21	64,468	83.41	10,694	0
Total	77,289	100	22,544	11,850

Table 3: The performance of selected learning methods by Tavalaee et al. using mentioned datasets

Learning Method	**KDD-Test**	**KDD-Test+**	**KDD-Tes-21**
J48	93.82	81.05	63.97
Naïve Bayesian	81.66	76056	55.77
NB Tree	93.51	82.02	66.16
Random forest	92.79	80.67	63.26
Random Tree	92.53	81.59	58.51
M-Layer Perceptron	92.26	77.41	57.34
SVM	65.01	69.52	42.29

5 Detailed Proposed Architecture

As mentioned in the previous section, proposed architecture has 4 main components. We describe its components in detail bellow.

5.1 Detection Model Generator

This module, as its name implies, creates the detection model using the training dataset. As mentioned before, static models are not suitable for anomaly detection, so the model must be adaptive. Using rules in adaptive architectures is one of the most popular methods, because a rule can be changed without affecting the remainder of the model and its results are remarkable [26]. On the other hand, employing fuzzy logic into IDS is recommended

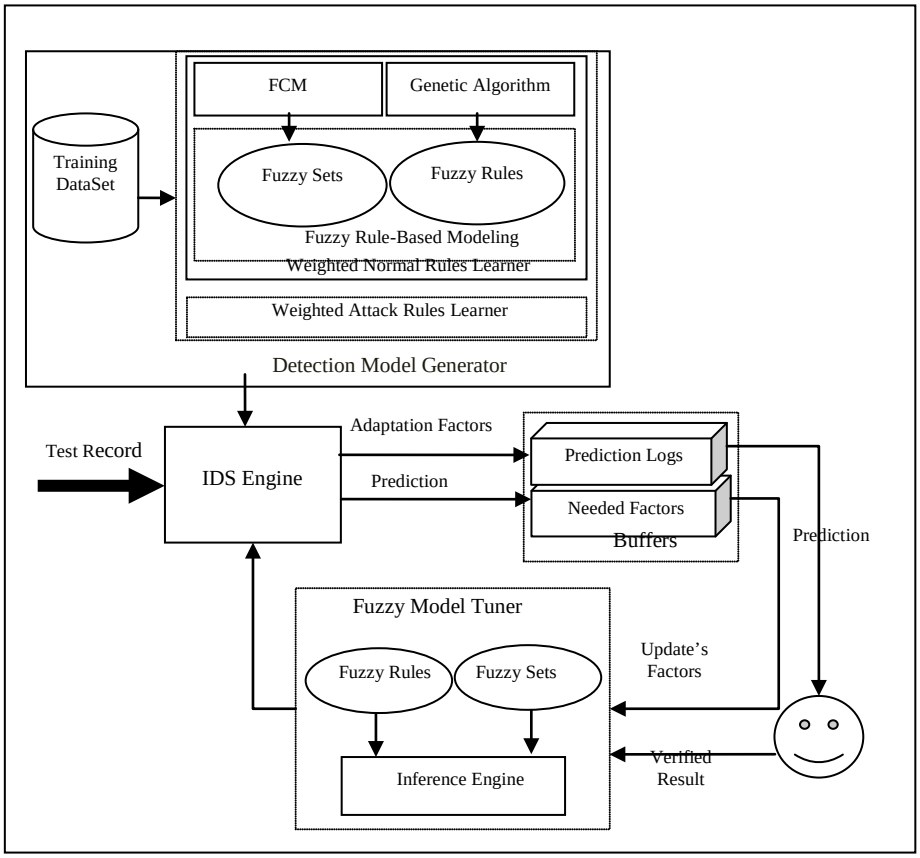

Figure 1: Proposed Architecture

for two important reasons [4]. Firstly, it uses several quantitative parameters in IDSs, that can potentially be viewed as fuzzy variables and the second is secure nature of fuzzy. Since we are not able to correctly differ between an attack and a normal behavior, fuzziness can help us to smooth out the abrupt separation of normal and attack behavior. For these reasons, fuzzy rule-base is used to generate detection model. In order to utilize this model, identifying fuzzy set of each input feature and trust rules are requested.

The first problem is determining fuzzy sets. Some of the features have numeric values such as duration and src-bytes while others have symbolic values such as service and protocol. FCM is used to obtain fuzzy sets of features with numeric values. In addition, the number of clusters is considered to be equal to six. In order to ensure that the number of clusters is sufficient, we used subtractive clustering method, which is a fast one-pass algorithm for estimating the number of clusters and cluster centers. The number of estimated clusters for each numeric feature is less than six by this algorithm, so we have used number six as the number of clusters for each numeric feature. Total possible values for features with symbolic values are considered as fuzzy sets. Each value is belongs to only one fuzzy set. In fact its membership value in that fuzzy set is one, while in the other fuzzy sets this value is zero.

Finding the best and most confident rules is the second problem. A genetic algorithm approach is used to find the best and trustable rules. In this area, there are two learning approaches called Pittsburgh and Michigan. In this paper, the second approach is used. Inside of each rule, five fuzzy terms are appeared and "is" or "is not" can be included in each term. Genetic algorithm is ran separately for each normal and attack class. Meanwhile learning normal rules, normal class is the considered class and attack class is the opposite one and vice versa.

We followed the laws with the highest prediction confidence ratio in addition we are able to classify the greatest number of considered class correctly by adjusting following function as the fitness function (Formula 1).

$$FitFunc = \frac{\#CCI_C}{\#CCI} * \frac{\#SC}{\#SC + 1.5*\#WC} \qquad (1)$$

CCI is the abbreviation for Considered Class's Instances and SC is Successfully Classified and WC is the abbreviation for Wrong Classified.

The first term is used to classify maximum percentage of considered class's instances correctly (CCIC). The

second term is employed in order to achieve rules with highest prediction confidence ratio for this purpose; we multiplied wrong classified by 1.5. This action leads to finding more confident rules.

Considered instances of classes with compatibility (Formula 2) higher than 0.5 and instances of opposite class with compatibility less than 0.5 are successfully classified (SC) samples and those of opposite class with compatibility higher than 0.5 are wrong classified samples. Compatibility of each training sample x=(x1,x2,…xn) with the rule r is calculated by Formula 2. S is the feature set that is available in the rule, F is the fuzzy set of each feature and μ is the membership function of each fuzzy set.

$$Com(x,r_i)=\min_{n=1}^{n=5}(\mu_{F[n]}(x_{S[n]})) \quad (2)$$

Prediction confidence ratio (PCR) of each rule is calculated using Formula 3. A suggested rule can be inserted in the rule set, if and only if it has a confidence ratio higher than 50 percent.

$$PCR(r_i)=\frac{\#SC}{\#SC+\#WC} \quad (3)$$

Using Michigan approach, after each iteration instances that are covered by the taught rule are removed from the training dataset. Removing training instances gradually reduces the degree of credibility of rules, because deleted instances could not be measured by subsequent rules. So we have determined more difficult conditions to remove one instance from training set. Only considered class's instances with compatibility higher than 0.5 and opposite class's instances with compatibility less than 0.3 with the taught rule are removed from training set.

5.2 IDS Engine

The IDS engine employs the detection model to classify test samples. Each test sample is given to normal and attack rules. Formula 4 calculates the instance membership value for each category. Finally, the test sample belongs to the category with the highest membership value. We have also considered equal number of rules for each category. In the following formula, C is a normal or attack classes and n is the number of learned rules in each class.

$$M_C(x)=\sum_{i=1}^{i=5} Com(x,r_i)*CF_i \quad (4)$$

Since the number of rules in each normal and attack class is equal, M_c could be used without worrying about correctness of this decision formula.

5.3 Buffers

Prediction logs and compatibility of test samples with each rule are buffered. The system administrator monitors the prediction class of each test record with a predefined delay. He verifies this prediction and reports to the fuzzy model tuner module. We should also consider that the related record is deleted after employing each tuning.

5.4 Fuzzy Model Tuner (Fuzzy Controller)

Without updating the static detection model, it is not feasible to reach the total accuracy of higher than 55 percent. For this reason and due to existing new attacks in the test dataset, the learned model is tuned using a fuzzy controller. Moreover, fuzzy controller determines adaptation intensity. In order to decide about the class of a test sample, we have employed the results of available rules for both normal and attack classes.

Fuzzy rules of fuzzy controller are presented in Table 10 and Table 11 in the Appendix I. Input variables of fuzzy rules are compatibility degree and confidence ratio of test samples for each rule. Test results buffered and presented to the administrator and verified by him after a while. The delay is considered constant and predefined.

For two input variables, two same fuzzy sets are considered. Output has three fuzzy sets and SOM (Smallest Of Maximum) is used for defuzzification. Membership functions of input variables and general view of output's membership functions are shown in Figures 2 and 3.

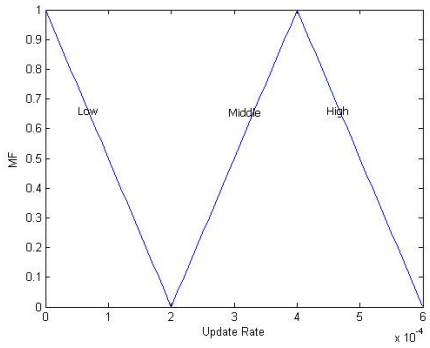

Figure 2: Membership functions of output variable

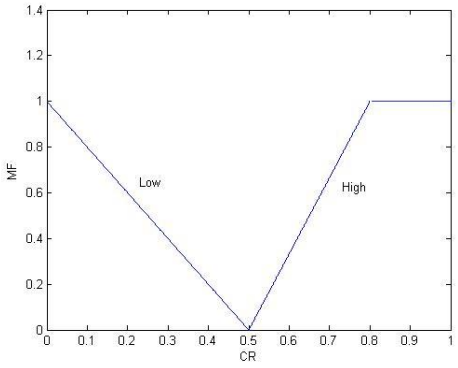

Figure 3: Membership functions of input variables

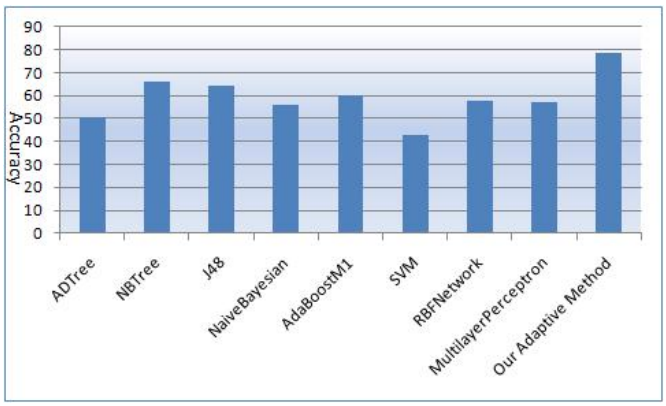

Figure 4: The performance of our method vs. some common methods

Table 4: Comparison between proposed architecture and the system proposed in [23]

Train Size	True Positive		True Negative		False Positive		False Negative		Total Accuracy	
1000	48.94		93.30		6.69		51.05		57.00	
2000	53.52		90.28		9.71		46.47		60.20	
3000	53.09		76.95		23.04		46.90		57.42	
5000	52.51		74.67		25.32		47.48		56.54	
7000	58.16		68.21		31.78		41.83		59.99	
9000	53.09		66.12		33.87		46.90		55.45	
10000	49.91		89.63		10.36		50.08		57.13	
AVG	52.75	86.71	79.88	42.29	20.11	57.71	47.24	13.29	57.67	78.60
11000	51.88		71.51		28.48		48.11		55.45	
11000	53.75		73.93		26.06		46.24		57.41	
11000	61.01		64.31		35.68		38.98		61.61	
11000	57.54		67.84		32.15		42.45		59.41	
11000	55.42		69.05		30.94		44.57		57.89	
AVG	55.92	86.71	69.33	42.29	30.66	57.71	44.07	13.29	58.35	78.60
							[23] system			Our system

6 Experimental Results

As mentioned before, due to changes in normal behavior of the network and appearance of new attacks, using the static model for intrusion detection systems is not relevant. Here we have improved the performance of detection by updating the detection model substantially. Results that are shown in Figure 4 prove this claim.

As you can see, the accuracy of our adaptive model is about 15 percent higher than other common machine learning methods.

Our presented architecture is compared against two fuzzy systems [7, 23]. Proposed system in [23] includes two layers. There are five ANFIS modules in the first layer, one for modeling normal behavior and the others for intrusions, which are trained using train dataset. Each module provides an output, which specifies the relativity degree of the data to the specific class. An output equal to 1 shows total membership while -1 is used otherwise. In the second layer, they used a fuzzy inference module to make the final decision in order to recognize if the input is normal or intrusive. The output value of each ANFIS classifier has two fuzzy sets (Low and High) and provides an input for the fuzzy inference module. Genetic algorithm is used to optimize these sets, while Fuzzy rules are fixed and predefined. Simply these five ANFIS modules are trained using a subset of the train dataset at the first stage. After that, genetic algorithm is used to optimize fuzzy sets of FIS inputs using verification train dataset. Table 5 shows some experimental results using new datasets [22]. Each average cell has two columns, first one is the average of upper rows and the second is the result of our proposed architecture as shown in Table 4.

Furthermore, our proposed architecture is compared against a fuzzy decision tree [7]. The main problem in designing a binary tree classifier is to determine what features and thresholds to use at each non-terminal node based on a set of training data. In this fuzzy decision tree, each of the internal nodes includes two fuzzy sets, Greater and Less than or equal (Figure 5).

Table 5: Comparison between proposed architecture and fuzzy decision tree with fixed p [7]

True Positive		True Negative		False Positive		False Negative		Total Accuracy	
59.77		86.94		13.05		40.22		64.71	
59.94		84.66		15.33		40.05		64.43	
51.73		86.10		13.89		48.26		57.97	
65.91		68.72		31.27		34.08		66.42	
51.73		87.87		12.12		48.26		58.30	
63.10		80.80		19.19		36.89		66.32	
56.72		86.11		13.89		43.27		62.06	
63.19		84.66		15.33		36.80		67.10	
54.155		86.01		13.98		45.84		59.94	
58.20	**86.71**	**83.75**	**42.29**	**16.24**	**57.71**	**41.79**	**13.29**	**62.80**	**78.60**
	Our system			[7] system					

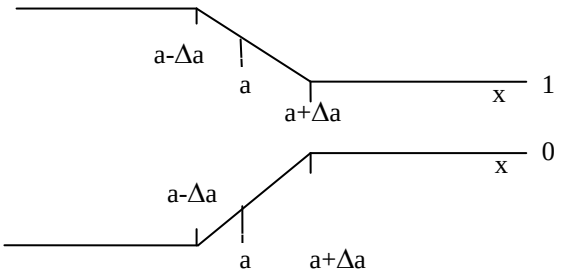

Figure 5: Fuzzy logic membership functions for x < a and x >= a

The left branch of a node has weight equal to *wl*, given by the membership function $x < a$ associated with it, and the right branch of the node has weight equal to *wr*, provided by the membership function $x >= a$ associated with it. In case the absolute value of distance between x and a is greater than Δa, the weight value will be either 1 or 0. The value of Δa is determined by a single user-defined parameter p, which is called the *fuzzy percent*. This parameter is the percentage of selected feature's domain size in a particular node, which is used to evaluate the value of Δa in that node. For example, if feature x_i has values ranging from x_{ij}^- to x_{ij}^+ in node j, then value of Δa is provided using Formula 5:

$$\Delta a = p(x_{ij}^+ - x_{ij}^-)/100 \qquad (5)$$

The fittest feature and the value of a in each node are determined using genetic algorithms.

Table 5 is a comparison between our proposed architecture and this fuzzy decision tree. In this experiment, the value of p is fixed and equal to 0.5. We have done another experiment while the value of p is variable and obtained using verification train dataset. Table 6 shows these results.

The fuzzy model tuner must tune confidence ratio of detection rule set. It has two input variables and one output. The compatibility ratio of test samples with each rule and the rules' confidence prediction ratio are inputs and the adaption intensity is considered as output for fuzzy model.

As expected, changes in the output membership functions (intensity of adaptation) can cause tangible changes in the performance. We have changed the slope of output's membership functions and reviewed the accuracy. We have also studied the effect of punishment and encouragement on improving accuracy in two separate experiments. Six experiments have done in order to show the impact of these parameters on total accuracy. Test results are available on Table 3. Total number of normal samples in the test set is 2152, while the number of attack instances is 9698.

As it can be seen in Table 7, non-adaptive model provides lowest accuracy. Results of these experiments show that the static model classifies more samples as normal behavior. 89 percent TN and 53 percent FN prove this claim.

Table 6: Comparison between proposed architecture and fuzzy decision tree with variable p [7]

True Positive		True Negative		False Positive		False Negative		Total Accuracy	
60.24		84.89		15.10		39.75		64.72	
59.97		85.13		14.86		40.02		64.54	
58.96		86.57		13.43		41.03		63.97	
61.42		86.84		13.15		38.57		66.04	
55.81		84.43		15.56		44.18		61.01	
57.67		69.93		30.06		42.32		59.89	
55.06		65.00		34.99		44.93		56.86	
56.38		85.22		14.77		43.61		61.62	
57.66		87.59		12.40		42.33		63.09	
58.568		87.17		12.82		41.43		63.76	
58.17	**86.71**	**82.28**	**42.29**	**17.71**	**57.71**	**41.82**	**13.29**	**62.55**	**78.60**

Table 7: Impact of Slope and presence or absence of Encourage on each reviewed parameter

	Static Model	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
		1/1000		1/3000		1/5000		1/8000		1/1000	
True Positive	0.4687	0.7431	0.7737	0.8671	0.7794	0.8517	0.7531	0.8204	0.7205	0 .8009	0.7024
True Negative	0.8982	0.4308	0.3676	0.4229	0.5404	0.4698	0.5818	0.5214	0.6436	0.5497	0.6617
False Negative	0.5313	0.2569	0.2263	0.1329	0.2206	0.1483	0.2469	0.1796	0.2795	0.1991	0.2976
False Positive	0.1018	0.5692	0.6324	0.5771	0.4596	0.5302	0.4182	0.4786	0.3564	0.4503	0.3383
Total Accuracy	**0.5466**	**0.6864**	**0.6991**	**0.786**	**0.736**	**0.7823**	**0.722**	**0.766**	**0.7064**	**0.7552**	**0.695**

Table 8: Impact of delay on accuracy

Delay	10	20	30	40	50	70	90	100
Accuracy	0.7824	0.7755	0.7739	0.7683	0.7636	0.7598	0.7558	0.7538

These results show that encourage reduces the total accuracy significantly. It can be inferred that using encourage makes the model more similar to the initial model. Whereas initial models tend to classify more samples as normal, using encourage will increase TN while decrease TP. As more instances of attack samples determined as normal in this way, system achieves less total accuracy. Using punishment solely makes the model more accurate than the initial model and this could increase TP and decrease TN that leads to improvement in total accuracy. Results of experiments as shown in Table 7 sustain this thought. The best results are highlighted in Table 7.

Another important factor to increase accuracy is reducing the amount of delay between evaluation and adaptation. Less delay could improve the accuracy rate. We have done an experiment on the effect of delay to the accuracy rate. Table 8 shows the results of this experiment. The first row is the number of samples between verification and applying adaptation to the model and the second row is accuracy rate. Less delay makes the model update faster that leads to classify more samples correctly and improves total accuracy.

As noted, each sample has a label, which is called #successfulPrediction. For each #successfulPrediction value, Table 9 shows the total number of instances of the dataset with this label in the second column, while #successfulPrediction values are shown in the first column. Moreover the first, second and third columns of each category in Table 9 are the results of evaluating static model, adaptation with punishment model and punishment and encourage model respectively. The second and third columns of each category belong to the fourth column of Table 9, so the output membership function's slope that used is 1/5000. Adaptive models can correctly classify most of the samples with #successfulPrediction values equal to zero.

Table 9: Detailed performance of static model, adaptation with punishment model and punishment and encourage model with respect to labeled instances

	Total	**TP**			**TN**			**FN**			**FP**		
0	123	7	91	65	3	0	0	109	25	51	4	7	7
1	87	4	64	48	6	0	0	72	12	28	5	11	11
2	55	8	40	27	6	1	2	40	8	21	1	6	5
3	116	24	54	43	15	6	7	37	7	18	40	49	48
4	101	16	58	43	17	2	3	51	9	24	17	32	31
5	103	20	60	50	12	2	2	53	13	23	18	28	28
6	157	26	102	81	19	14	13	96	20	41	16	21	22
7	249	38	182	149	9	3	3	186	42	75	16	22	22
8	131	35	106	96	4	0	2	85	14	24	7	11	9
9	106	35	81	72	5	2	3	59	13	22	7	10	9
10	195	51	147	126	23	9	12	119	23	44	2	16	13
11	461	99	391	321	14	3	4	344	52	122	4	15	14
12	486	233	421	391	16	6	8	232	44	74	5	15	13
13	519	278	448	406	27	20	21	206	36	78	8	15	14
14	736	416	649	577	39	24	29	276	43	115	5	20	15
15	1176	534	1022	891	37	23	30	597	109	240	8	22	15
16	681	205	557	480	42	21	25	427	75	152	7	28	24
17	1168	265	589	503	439	338	388	458	134	220	6	107	57
18	2967	822	1635	1396	735	416	474	1391	578	817	19	338	280
19	890	562	670	648	94	21	45	228	120	142	6	79	55
20	1343	867	893	891	371	100	181	87	61	63	18	289	208

7 Conclusion

Anomaly based intrusion detection systems are provided in order to protect computer networks against novel attacks and improve network security. These systems perform intrusion detection by comparing current network traffic with a behavioral model of normal network activity. As the pattern of network traffic changes over time, static models are not appropriate to monitor malicious activities. As the static models could be tuned with respect to changes in traffic pattern, adaptive models are used in this manner. In this paper, we have presented an adaptive anomaly-based intrusion detection system.

Fuzzy rule-based modeling is used to create the detection model. In addition, prediction results are delivered to system user for verification. Fuzzy controller module uses verified results in order to tune the detection model. Experimental results show that our proposed architecture could reach a total performance about 15 percent higher than static detection models.

8 Future Works

One of the most important issues about our proposed architecture is the interaction between system-user and intrusion detection system, in order to verify predictions of the system. As means to reduce the number of interactions, system updates in presence of the user could be done in a periodically manner or at specified times that the number of wrong predictions reaches a predefined threshold.

References

[1] D. Anderson, T. Frivold, A. Tamaru and A. Valdes, "Next-generation Intrusion Detection Expert System (NIDES)," Software Users Manual, Beta-Update Release, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, Technical Report SRI-CSL-95-0, May 1994.

[2] D. Barbara´, J. Couto, S. Jajodia and N. Wu, "ADAM: A testbed for exploring the use of data mining in intrusion detection," ACM SIGMOD Record: Special section on Data Mining for Intrusion Detection and Threat Analysis, vol. 30, pp. 15−24, 2001.

[3] T. Bhaskar, N. Kamath and S.D. Moitra, "A hybrid model for network security systems: Integrating intrusion detection system with survivability," International Journal of Network Security, vol. 7, no. 2, pp. 249–260, 2008.

[4] S. M. Bridges and R. B.Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection," Proceedings of the National Information Systems Security Conference, Baltimore, MD, pp. 16-19, 2000.

[5] John E. Dickerson and Julie A. Dickerson, "Fuzzy network profiling for intrusion detection," Proceedings of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, pp. 301–306, Atlanta, USA, July 2000.

[6] E. Eskin, S. J. Stolfo and W. Lee, "Modeling system calls for intrusion detection with dynamic window sizes," Proceedings of the DARPA Information Survivability Conference & Exposition II, Anaheim, CA, pp. 165−175, 2001.

[7] R. E. Haskell, "Neuro-fuzzy classification and regression trees," Proceedings of the Third International Conference on Applications of Fuzzy Systems and Soft Computing, Wiesbaden, Germany, pp. 5-7, October 5-7, 1993.

[8] H. H. Hosmer, "Security is fuzzy!: Applying the fuzzy logic paradigm to the multi-policy paradigm,"

Proceedings of the 1992–1993 Workshop on New Security Paradigms Little Compton, RI, United States, pp. 175-184, 1993.

[9] P. Kabiri and A. A. Ghorbani, "Research on intrusion detection and response: A survey," International Journal of Network Security, vol. 1, no. 2, pp. 84-102, 2005.

[10] S. S. Kandeeban and R. S. Rajesh, "Integrated intrusion detection system using soft computing," International Journal of Network Security, vol. 10, no. 2, pp. 87-92, 2010.

[11] H. Lee, Y. Chung and D. Park, "An adaptive intrusion detection algorithm based on clustering and kernel-method," Proceedings of PAKDD'06, LNAI, vol. 3918, pp. 603-610, 2006.

[12] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and Junxin Zhzng, "Real time data mining-based intrusion detection," Proceedings of the 2nd DARPA Information Survivability Conference and Exposition II, pp. 85-100, June 2001.

[13] Y. Liao, V. R. Vemuri and A. Pasos, "Adaptive anomaly detection with evolving connectionist systems," Journal of Network and Computer Applications, vol. 30, pp. 60–80, 2007.

[14] M. Locasto, K. Wang, A. Keromytis, and S. Stolfo, "Flips: Hybrid adaptive intrusion prevention," Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID), pp. 82-101, Sept. 2005.

[15] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathm, C. Jalali, P. G. Neumann, H. S. Javitz, A. Valdes and T. D. Garvey, "A Real-time Intrusion Detection Expert System (IDES)," Computer Science Laboratory, SRI International, Menlo Park, CA, USA, Final Technical Report, SRI Project 6784, Feb. 1992.

[16] M. V. Mahoney, P. K. Chan, "Learning non-stationary models of normal network traffic for detecting novel attacks," Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp. 376−385, 2002.

[17] A. Mitrokotsa, N. Komninos and C. Douligeris, "Protection of an intrusion detection engine with watermarking in ad hoc networks," International Journal of Network Security, vol. 10, no. 2, pp. 93–106, Mar. 2010.

[18] N. Ngamwitthayanon, N. Wattanapongsakorn and D. W. Coit, "Investigation of fuzzy adaptive resonance theory in network anomaly intrusion detection," Proceedings of the 6th International Symposium on Neural Networks: Advances in Neural Networks, pp. 208-217, 2009.

[19] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," Proceedings of the 20th NIST-NCSC National Information Systems Security Conference, Baltimore, MD, USA, pp. 353 − 365, 1997.

[20] A. Rasoulifard, A. Ghaemi, and M. Kahani, "Incremental hybrid intrusion detection using ensemble of weak classifiers," Advances in Computer Science and Engineering: 13th International CSI, Iran, pp. 577-584, March 2008.

[21] M. Saniee, J. Habibi, Z. Barzegar and M. Sergi, "A parallel genetic local search algorithm for intrusion detection in computer networks," Elsevier Journal, Engineering Applications of Artificial Intelligence, vol. 20, no. 8, pp. 1058-1069, December 2007.

[22] M. Tavalaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," IEEE Symposium: Computational Intelligence for Security and Defense Applications, CISDA'09, pp. 1-6, July 2009.

[23] A. N. Toosi and M. Kahani, "A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers," Computer Communications, vol. 30, pp. 2201-2212, 2007

[24] J. Udhayan and T. Hamsapriya, "Statistical segregation method to minimize the false detections during DDoS attacks," International Journal of Network Security, vol. 13, no. 3, pp. 152-160, 2011.

[25] M. Yang, H. Zhang, J. Fu, and F. Yan, "A framework for adaptive anomaly detection based on support vector data description," Lecture Notes in Computer Science, Network and Parallel Computing, pp. 443-450, 2004.

[26] Z. Yu, J. J. P. Tsai, and T. Weigert, "An adaptive automatically tuning intrusion detection system," ACM Transactions on Autonomous and Adaptive Systems, vol. 3, no. 3, pp. 1-25, August 2008.

[27] J. Zeng and D. Guo, "Agent-based intrusion detection for network-based application," International Journal of Network Security, vol. 8, no. 3, pp. 201-210, May 2009.

[28] http://nsl.cs.unb.ca/NSL-KDD/, accessed Dec. 2009.

Appendix I: Fuzzy rules of fuzzy controller.

Table 10: Prediction is false and system must be punished

If MCR is low and Com is low then update intensity is low
If MCR is low and Com is high then update intensity is middle
If MCR is high and Com is low then update intensity is low
If MCR is high and Com is high then update intensity is high

Table 11: Prediction is false and system must be encouraged

If MCR is low and Com is low then update intensity is low
If MCR is low and Com is high then update intensity is middle
If MCR is high and Com is low then update intensity is low
If MCR is high and Com is high then update intensity is low

Farzaneh Geramiraz received her B.S degree in 2006 in Software Engineering from Amirkabir university of Technology and science 2007 is studying artificial intelligence at Shahid Beheshti university of Tehran.

Amir Saman Memaripour is a last-year B.S. student at Shahid Beheshti University in Computer Engineering. His research interests include network security, especially intrusion detection systems, and distributed systems. Moreover, he has a profound background on web applications' security issues.

Maghsoud Abbaspour received his B.S, M.S and Ph.D degree from University of Tehran in 1992, 1996 and 2003 respectively. He joined Computer Engineering department, Shahid Behesht University in 2005.His interests includes sensor and adhoc networks, multimedia on peer to peer networking and network security areas.