Adaptive Framework for Network Intrusion Detection by Using Genetic-Based Machine Learning Algorithm

Wafa' S. Al-Sharafat * , Reyadh Sh.Naoum **

*Al Al-Bayt University, Information Technology College, Jordan
** Arab Academy for Financial and Banking Science. IT Collage, Jordan

Summery
Computer networks have expanded significantly in use and in numbers. This expansion makes them target to different attacks. Intrusion Detection System (IDS) is used to identify unknown or new type of attacks or in dynamic environments as mobile networks. As a result, it is necessary to find a ways to implement and operate IDSs. Among different techniques, Genetic-based machine learning algorithm (GBML) which offers a good ability to be adapted to changing environments, robustness to noise and ability to identify unknown attacks. The objective of this paper is to incorporate different techniques into classifier system to detect and classify intrusion from normal network packet. Among several techniques, steady state genetic-based machine leaning algorithm (SSGBML) which will be used to detect intrusions. Steady State Genetic Algorithm (SSGA) and Zeroth Level Classifier system (ZCS) are investigated. SSGA is used as a discovery mechanism for classifiers, while ZCS plays the role of detector by matching incoming environment message with classifiers to determine whether it is normal or intrusion. As a feedback, the environment will make a decision on whether to take action or not. In order to attain the best results, modifying SSGA will enhance our discovery engine. The experiments and evaluations of the proposed method were performed with the KDD 99 intrusion detection dataset.

Key words:
Network intrusion detection, SSGA, Modified SSGBML KDD' 99.

1. Introduction

With the ever-increasing growth of computer networks and emergence of electronic commerce in recent years, computer security has become a priority. Since intrusions take advantage of vulnerabilities in computer systems or use socially engineered penetration techniques, intrusion detection (ID) is often used as another wall of protection. In addition, traditional security techniques as user authentication are not optimal method to protect data from any possible attack. That is due to the vastness of the network activity data and the need to regularly update our intrusion detection systems (IDS) to cope with new unknown attack methods or upgraded computing environments. Thus, ID is becoming one of the main technologies can be used to monitor network traffics and identify network intrusions. There are different taxonomies for IDSs have been suggested [1, 2, 3]. One of these taxonomies depends on the source of audit data that will be used to detect possible intrusions. This was divided into two groups; network intrusion detection (NID) and host-based intrusion detection (HID) [4, 5]. HID identifies the intruders by monitoring host-based traffic, while NID refers to identifying the intruders by monitoring network traffics (packets). This can be accomplished by identifying attacks using their known pattern (signature). Figure 1 shows the computer network with NIDS.

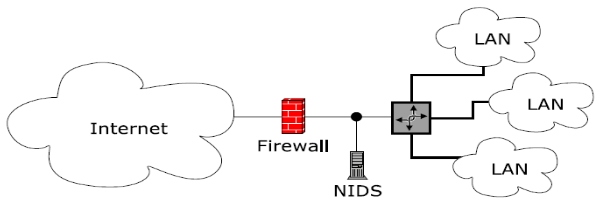


Fig 1: A computer network with network intrusion detection systems

IDS methodologies include statistical models, immune system approaches, protocol verification, file and taint checking, neural networks, whitelisting, expression matching, state transition analysis, machine learning, GAs, and others.

There is a need for adapting technique capable to work effectively in a dynamic environment and identifying known and unknown network attacks. Machine learning with GA as a discovery mechanism can be used to accomplish this task. GA offers the ability to overcome the shortcomings of many existing IDS techniques. GAs possess properties that make them particularly suitable for ID including robustness to noise, self-learning capability, and the ability to build initial rules without the need for a priori knowledge. GAs are intrinsically parallel because they generate multiple offsprings that explore the solution space in multiple directions simultaneously. Parallelism makes GAs well-suited where solution space is extremely large. The adaptability of GA allows the system to be easily retrained to evolve new rules.

Manuscript received April 5, 2009
Manuscript revised April 20, 2009

This paper presents a Modified SSGBML for NID using SSGA. In addition, crossover and mutation probability will be adapted by using Fuzzy Logic. Section 2 provides the dataset commonly used in training and testing the performance of NIDS. Section 3 presents related work for ID. The GBML is presented in section 4, and Section 5 provides several methods for ID evaluations. The experimental results and comparisons with existing approaches presented in Section 6. Finally, conclusion and future research directions are presented in Section 7.

2. IDS Dataset

The methods in [9,10,12] used the KDD 1999 Dataset [6]. This dataset was derived from the 1998 DARPA Intrusion Detection Evaluation Program held by MIT Lincoln Labs. The dataset was created and simulated in a military network environment in which a typical U.S. Air Force LAN was subjected to simulated attacks. Raw TCP/IP dump data was gathered. The data is approximately 4 GB of compressed TCP dump data which took 7 weeks of network traffic and comprised about 5 million connection records. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted and listed in Table 1. The features exhibited in the dataset can be grouped into 3 categories: basic features of individual TCP connections, content features within a connection, and traffic features computed using a two second time window. Each of the categories and the associated features are shown in [6]. KDD dataset is divided into training and testing record sets. This is too large for our purpose; hence, only concise training dataset of KDD, known as 10% training dataset, was employed here and this sample distributed is shown in Table 2.

The dataset was divided into training and test dataset. Training is used to train the work presented here, while test dataset is used to test it. Test dataset contains additional attacks not described in training dataset. The attacks include the four most common categories of attack:

• Denial of service (DoS) attacks; here, the attacker makes some computing or memory resource which makes the system too busy to handle legitimate requests. These attacks may be initiated by flooding a system with communications, abusing legitimate resources, targeting implementation bugs, or exploiting the system's configuration.

• User to root (U2R) attacks; here, the attacker starts with accessing normal user account and exploits vulnerabilities to gain unauthorized access to the root. The most common U2R attacks cause buffer overflows.

• Remote to user (R2L) attacks; here, the attacker sends packets to a machine, then exploits the machine's vulnerabilities to gain local access as a user. This unauthorized access from a remote machine may include password guessing.

• Probing (PROBE); here, the attacker scans a network to gather information or find known vulnerabilities through actions such as port scanning.

Table 1: KDD Feature

Feature No.	Feature name
1.	duration
2.	protocol type
3.	service
4.	Flag
5.	src_bytes
6.	dst_bytes
7.	land
8.	Wrong_fragment
9.	Urgent
10.	hot
11.	num_failed_logins
12.	Logged in
13.	num_compromised
14.	root_shell
15.	su_attempted
16.	num_root
17.	num_file_creation
18.	num_shells
19.	num_access_files
20.	num_outbound_cmds
21.	is_host_login
22.	is_guest_login
23.	Count
24.	srv_count
25.	Serror_rate
26.	srv_serror_rate
27.	Rerror_rate
28.	srv_rerror_rate
29.	same_srv_rate
30.	diff_srv_rate
31.	srv_diff_host_rate
32.	dst_host_count
33.	dst_host_srv_count
34.	dst_host_same_srv_rate
35.	dst_host_diff_srv_rate
36.	dst_host_same_src_port_rate
37.	dst_host_srv_diff_host_rate
38.	dst_host_serror_rate
39.	dst_host_srv_serror_rate
40.	dst_host_rerror_rate
41.	dst_host_srv_rerror_rate

Table 2: 10% of KDD Dataset

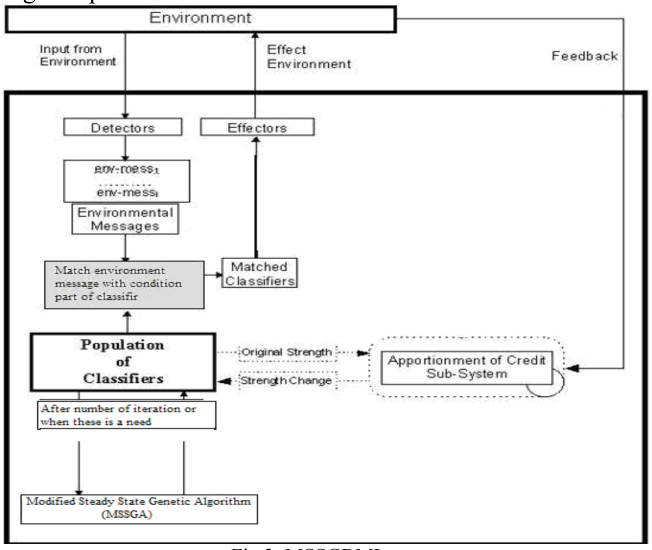
Type	Number of samples	%
Normal	97227	19.69
DoS	391458	79.24
Probe	4107	0.83
R2L	1126	0.23
U2R	52	0.01

3. Related Work

This section briefly summarizes some of the techniques for ID. The early effort of using GAs for ID can be dated back to 1995, when Crosbie et. al. [7] applied the multiple agent technology and Genetic programming (GP) to detect network attempts. Each agent monitors one parameter of the network packet and GP is used to find the set of agents that collectively determine anomalous network behaviors. This method has the advantage of using many small autonomous agents, but the communication among them is still a problem. Also the training process can be time-consuming if the agents are not appropriately initialized.
In [8] researchers develop a method that integrates fuzzy data mining techniques and genetic algorithms to detect both network misuses and anomalies. In most of the existing GA based IDSs, the quantitative features of network audit data are either ignored or treated. Such features are often involved in intrusion detection. This is because of the large cardinalities of quantitative features. The researchers proposed a method to include quantitative features by introducing fuzzy numerical functions. Their preliminary experiments show that the inclusion of quantitative features and the fuzzy functions significantly improved the accuracy of the generated rules. In this approach, a GA was used to find the optimal parameters of the fuzzy function as well as to select the most relevant network features. Different computing paradigm has been used in [9] where the proposed paradigm was neuro-fuzzy network, fuzzy inferences, and GA to detect intrusion activities in networks. This method firstly used a set of parallel nero-fuzzy classifiers (five layers 4- for type of attack, and one for normal). Then, fuzzy inference used the output from classifiers to take a decision whether the current action is normal or not. The role of GA was used to optimize the classifier engine to give the right decision. This Method also used the same data KDD CUP 99 [6] for training and for testing the system. As a result, this technique will be effectively used to detect intrusion. To enhance their proposed work, feature reduction must be performed instead of using all 41 features. In [10, 11] researchers tried to build an application to enhance the knowledge domain to detect vast range of intrusion by using machine learning (ML) techniques to create rules of Expert System (ES) that can learn from dynamic environment to acquire expert knowledge to be adapted with new attacker behaviors. Knowledge is represented as a set of if-then rules.

4. Intrusion Detection using Steady State Genetic-based Machine Learning Algorithm

ML is the study of computational methods for improving the performance of acquisition of knowledge from experience. Expert performance requires much domain specific knowledge and tries to build ES that can be used in different domains such as industry. Thus, ML will reduce human time-consuming. In addition, ML will increase the level of automation to improve the accuracy and the efficiency of detection systems by discovering and exploiting regularities through training data [12]. SSGBML takes into account the ability to learn from environment to be not restricted to static inputs to learn from. SSGBML merges ES and SSGA that enables learning from incomplete information. In addition, in the early stages for developing our algorithm SSGBML, using Simple Genetic Algorithm (SGA), shows better detection rate rather than using SGA. Different combinations of inputs will be produced to perform rules in the form of {condition} --> action. SSGA will play the role of discovery engine in SSGBML. SSGA is used to give a chance for previous rules from previous generation to participate in detecting intrusions in next generations. In contrast, SGA replaces whole previous generation with new produced generation neglecting the fact that there exist some good rules in previous generation. Therefore, [13] has indicated that SSGA achieved better and faster solution than SGA. This leads us to use SSGA to generate rules from previous rules (act as parents), taking into account how the output will be closed to the problem solution. By modifying SSGBML, MSSGBML, performance will be improved compared with other works. Figure2 provides an overview for MSSGBML.


Environment
Input from Environment
Effect Environment
Feedback
Detectors
Effectors
env-mess1
env-messi
Environmental Messages
Match environment message with condition part of classifir
Matched Classifiers
Population of Classifiers
Original Strength
Strength Change
Apportionment of Credit Sub-System
After number of iteration or when these is a need
Modified Steady State Genetic Algorithm (MSSGA)
Fig 2: MSSGBML

MSSGBML has a set of components which they are:

4.1 Detector

Detector, Input Unit, takes the input variables that perform network traffic features from network environment. In this unit, the input must be encoded, where in our work, we used real encoding method. Network features have either continuous or discrete values. Furthermore, features are classified as: significant and insignificant features. Significant features have a significant role to detect attacks. Insignificant features don't have a significant role in detecting attack. In this case, these features have been replaced by the least value of number in Java. These features composed the condition part of an environment message. Afterward, determining related features for each type of network attack is exhaustive stage. So, different techniques have been tried to find a relation and a correlation between network features and type of attack that these features can predict. In [14, 15] researchers have tried to find a correlation between features and network attacks. While in [16], researchers have used features for each type of attack according to previous studies that have been accomplished. As a result, we can perform different classes taking the advantages of other results. We used classes contain main features to detect specific type of network attacks. Table 3 shows these classes.

Table 3 : Network Attack classes

Class #	Attack type: Features
Class 1 [17]	DoS: 5,10,24,29,33,34,38,40
	Probe: 2,3,23,34,36,40
	U2R: 3,4,6,14,17,22
	R2L: 3,4,10,23,33,36
Class 2 [18]	DoS: 1,2,3,4,5,6, 12,23,24,31,32,37
	Probe: 1,2,3,4,12,16,25,27,28,29,30,40
	U2R:1,2,3,10,16
	R2L: 1,2,3,4,5,10,22
Class 3 [19]	DoS:1,5,6,23,24,25,26,32,36,38,39
	Probe: 1,2,3,4,5,6,23,24,29,32,33
	U2R: 1,2,3,4,5,6,12,23,24,32,33
	R2L:1,3,5,6,32,33
Class 4 [20]	DoS: 7,8,12,13,23
	Probe: 3,12,27,31,35
	U2R: 14,17,25,38,36
	R2L: 6,11,12,19,22

4.2 Effector

Effector, as output unit, is responsible for firing action of the winning rule to the environment. The result can be either normal, Probe, U2R, R2L or DoS.

4.3 Feedback

Feedback will influence the rule that has been selected to fire its action. That is done by adding positive value (for reward) to the selected rule strength if it gave right prediction for the type of attack. Otherwise it adds a negative value (for penalty).

4.4 Classifier System

The Zeroth Level Classifier System (ZCS) consists of a finite set of classifiers (rules) in the form of {***condition* - *action***}. The rule's condition is a string of characters compound from significant network features with real valued where insignificant features represented don't care (D= least value for number in Java) acts as a wildcard allowing generalization. The action is represented by one character. Initial rules in ZCS are initialized randomly using GA. To initialize fitness for each rule, the fitness functions calculated using equation 1

$$\text{Fitness} = \frac{a}{A} - \frac{b}{B} \tag{1}$$

where ***a*** is the number of correctly identified attacks, ***A*** is the total number of attacks in the training dataset, ***b*** is the number of connections incorrectly classified as attack (false positive), and ***B*** is the total number of normal connections. Resulting fitness value is in the range of [-1, 1]. Then, fitness for rules is calculated as in shown in equation 2.

$$\left(\frac{fi}{\sum_{i=1}^{n} fi}\right) * \left(\frac{DR}{n} - \frac{FPR}{n}\right) \tag{2}$$

where ***n*** is rules number.

Classifier fitness acts as an indicator of the perceived utility of that rule within the system. On receipt of an input message, the rules [N] are scanned and any rules whose condition matches the message is placed in a match set [M]. The selection policy for best [M] to be selected and action set [A] is based on the rules fitness. When an action has been selected, all of the rules in the [M] that advocate this action are placed in action set [A], and the system executes the action as shown in Figure 3. Reinforcement in ZCS consists of redistributing payoff between subsequent action sets. A fixed fraction, β, of the fitness of each member [A] at each time step is placed in a common bucket. A record is kept for the previous action set [A]-1. If this is not empty, then the members of this set receive an equal share of the content of the current bucket, once this has been reduced by a pre-determined discount factor, γ. If a reward is received from the environment, then the β of this value is distributed evenly amongst the members of [A]. Finally, a tax is imposed on the members of [M] that do not belong to [A]. ZCS employs SSGA as a discovery mechanism. Settings the parameter used in ZCS are tuned to gain better results.

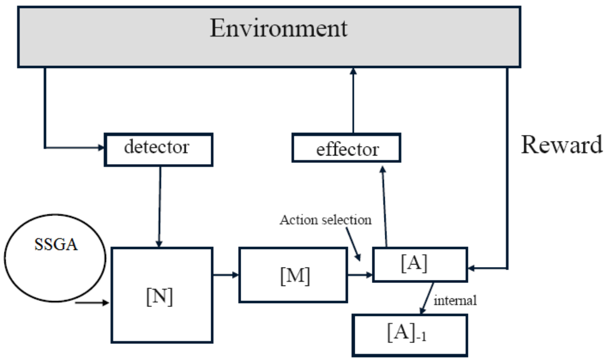


Fig 3: Zeroth Level classifier system (ZCS)

In ZCS, rules clustered according to the type of action that can be taken into five categories: Normal, DOS, Probe, U2R, and R2L. Incoming message spreads to these categories. As a result, best match set will be used to take action. Furthermore, SSGA use this feature to crossover and mutate each category separately as shown in Figure 4.

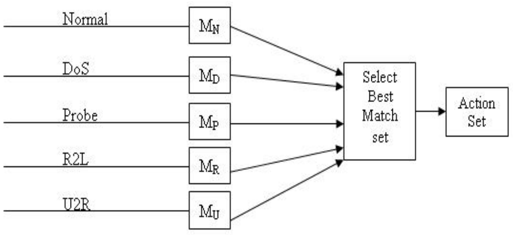


Fig 4: rules categorization

4.5 Steady State Genetic Algorithm

SSGA, as discovery mechanism, will be used as a classifier producer to produce new classifiers from existing ones. SSGA includes a set of operation performed to produce new good classifiers and enhance the performance capabilities for detecting network intrusions. These operations are: selection, crossover, mutation, and replacement of new rules with old ones. According to that, there is a set of decisions must be taken into account to begin using SSGA. First, the type of GA to be used has to be determined. There were a set of studies that can be used to determine which type of GA will be used. The causes for selecting SSGA mentioned early. The results in [13] concerning SSGA will be suitable to be used in solving problems instead of SGA. Based on set of conducted experiments on SSGA and SGA, SSGA gives better and faster results than SGA. By using SSGA, current best solutions are automatically maintained in the population and only the poorest individuals are being replaced. In contrast, in SGA, every individual is replaced in every generation. Thus, there is a much greater pressure to produce individuals that are not degraded by crossover and mutation. Second, selection method for selecting rules as Roulette Wheel, Ranking and other methods mentioned in [21]. Third, parameterized crossover and mutation (Pc, Pm respectively) will be used to determine the possibility if we can accomplish the crossover and mutation on rules. In the next stage for enhancing proposed algorithm, Pc will be adaptive according to the performance of SSGBML in detecting intrusions by applying fuzzy logic. For crossover, the position will be selected randomly and can be either single, or multiple positions. As for Pc, Pm will be used to determine if we can mutate rules component or not. Even more, Pc was adapted to be aware from undesired behavior such as premature convergence. Thus, Pc will be enhanced by using fuzzy logic. Adaptation can be accomplished by using counter for each rule to be an indicator on the rules age. Rule counter value increased automatically with each generation. To formalize the situation, the average and standard deviation will be calculated for rules age to determine if the rule Young, Mid-age, or Old. By applying fuzzy logic on parent's rules age, Pc can be Low, Medium or High. Also, Pm may take the advantage of rules age by applying fuzzy logic on rules age to determine Pm that can be taken into account in improvement stage for our algorithm. So Pm can be Low, Medium or High. Table 4 summarizes the idea of adaptive Pc.

		Parent II		
		Young	Mid-age	Old
Parent I	Young	Low	Medium	Low
	Mid-age	Medium	High	Medium
	Old	Low	Medium	Low

Table 4 : Crossover and Mutation Probability

Fourth, fitness function is one of the SSGBML keys that will be used to give judgment rules. Fitness function in our proposed algorithm takes into account the performance of the rules to detect network intrusion. Fitness calculated as in equation 4.2. Fitness will be used to evaluate strength for all rules within classifier a calculated in equation 3. Beside of rule's fitness, strength can be used as selection mechanism for selecting best rule to fire an action.

$$Strength_i = f_i * age_i \qquad (3)$$

5. Performance Evaluation

One of the main issues involved in solving problems or trying to find optimal solution is how to test these proposed systems. As for NIDS, testing proposed algorithm can provide a good indicator for the proposed algorithm if it can give high performance compared with others or not. If our proposed algorithm has performance less than other algorithms, this encourages us to tune up our algorithm to improve our work. In addition, it is natural to assume that the difficulty of any problem relevant to the scale of these problems. In general evaluating security system is a complex task to be accomplished. The main issue is measuring the performance of IDS effectively. Evaluating IDS can be expressed as how far it can correctly classify intrusions and avoid false detection. The difficulty came from the fact that it has to work properly in unknown situations and deal with new types of attackers in network environment. In previous work, there was a variety of ways used to evaluate IDS. Some of these are False Positive Rate (FPR) and Detection Rate (DR). FPR is defined as "the ratio of incorrectly classified normal examples (false alarms) to the total number of normal examples" [22]. FPR was calculated using Equation 5.1.

$$FPR = \frac{F}{N} \qquad 5.1$$

where F is a number of false alarms and N is the number of total normal records.
In addition, Detection Rate (DR) was defined as "the ratio of correctly classified intrusive examples to the total number of intrusive examples" [22]. The DR is computed using Equation 5.2.

$$DR = \frac{DA}{T} \qquad 5.2$$

where DA is the number of truly detected attacks and T is the number of total attacks.

5. Results

The SSGBML for NID algorithm was tested using the KDD 99 Dataset [6]. Proposed algorithm was trained using 10% of KDD 99 as a training dataset. The training data contains approximately 500,000 connection records. We trained and tested the proposed algorithm using same dataset. After training our algorithm using different classes of features, we extracted the rules and then tested the resulted rules using same dataset. Table 5 provides initial results for the proposed work compared with other work.

Model	DR%
SSGBML	97.45%
ESC-IDS[9]	95.3%
RSS-DSS [23]	94.4%
Fuzzy Inference System[24]	98%
EFRID [25]	95.47%

Table 4: Detection rate (DTR) for the different algorithms performances on the KDD 99 with corrected labels of KDD Cup 99 dataset (n/r stands for not reported)

It can be stated that the proposed algorithm is offered an acceptable level of detection performance compared with others work. Fitness function has a great role in detecting intrusions. For the types of intrusions, DoS, Probe or both had an acceptable detection rate compared with U2R and R2L types.

6. Conclusion

In this paper, a new algorithm was introduced to detect network intrusions and was successfully demonstrated on KDD 99 Dataset, training and testing data. Also, matching difference between environment message and classifier rules became adaptive according to DR values. Discover engine has been improved by using SSGA instead of SGA taking into account the suitable method for selection. Also, when performing some training on SGA, we dramatically reached premature convergence early. So, training phase was stopped and not continued. The proposed work focused on reducing the number of features to be used in classifying and detecting various attacks types. The future work will be continued to use different fitness function since it plays important role in intrusion detection. In addition, we will apply fuzzy logic on Pc, Pm to gain better results.

References

[1] D.Batazrotti, “Testing Network Intrusion Detection Systems", PhD Thesis, Milano University, 2006, Italy.
[2] R.Bace, P. Mell, “Intrusion Detection Systems", NIST special publication on IDS, 16 Augest, 2001.
[3] D.J.Brown , B.Suckow, T.Wary, " A survey Of Intrusion Detection Systems", www.**cse.ucsd.edu**/classes/fa01/cse221/projects/group10.pdf
[4] S.Selvakani, R.S. Rajesh, " Genetic Algorithm for framing rules for intrusion Detection", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.11, November 2007
[5] A.Christie, W. Fithen, J.McHugh, J.Pickel, E. Stoner, "State of the Practice of Intrusion Detection Technologies", TECHNICAL REPORT, Carnegie Mellon University, 2000.
[6] KDD-CUP 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[7] Crosbie M and Spafford E, “Applying genetic Programming to Intrusion Detection”, Proceedings of the AAAI Fall Symposium, 1995.
[8] S.M. Bridges and R.B. Vaugha, " Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection”, Proceedings of 12th Annual Canadian Information Technology Security Symposium, pp.109-122, 2000.
[9] N.Toosi , M.Kahani, " A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers", Computer Communications 30(2007) 2201–2212, 2007

[10] M. Sabhnani, G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context", Proceeding of International Conference on Machine Learning: Models, Technology and Application, Las Vegas, Nevada, USA, June 2003
[11] Ch. Sinclair, L. Pierce, S. Matzner, "An Application of Machine Learning to Network Intrusion Detection", 15th Annual Computer Security Applications Conference Phoenix, Arizona , December 6-10, 1999
[12] A.Osareh, Bita Shadgar, " Intrusion Detection in Computer Networks based on Machine Learning Algorithms", International Journal of Computer Science and Network Security, VOL.8 No.11, November 2008
[13] J.Jones, T.Soule, " Comparing Genetic Robustness in Generational vs. Steady State Evolutionary Algorithms", GECCO'06, Seattle, Washington, USA. July 8–12, 2006, ©Copyright 2006 ACM
[14] I.Guayan , A.Elisseeff, "An Introduction to Variable and Selection" , Journal of Machine Leaning Research 3, March 2003
[15] L.Yu, H.Lin, "Feature Selection for High-Dimensional Data: A Fast Correlation-based Filter Solution", Proceeding of 20th International Conference on Machine Learning (ICML-2003), Washington D.C., August 2003.
[16] T. S. Chou, K. K. Yen, and J. Luo, "Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms", International Journal of Computational Intelligence 4;3 © www.waset.org Summer 2008.
[17] A.Zaind, M. Maarof, S.Shamsnddin, A. Abraham,"Ensamble of One-class Classifier for Ntwork Intrusion Detections". www.**softcomputing.net**/ias08_1.pdf
[18] T.S.Chou, K.K.Yen, J.LNO," Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms", International Journal Of Computational Intelligence, 2008.
[19] S.Mukkamala, A.h.Sung," Identifying Significant Features for Network Forensic Analysis using Artificial Intelligent Techniques", International Journal of Digital Evidence, Vol 1, Issue 4, Winter 2003.
[20] S.Mukkamala, A.h.Sung, A. Abraham, Modeling Intrusion Detection System using Linear Genetic Programmimg Approaches", LNCS 3029, Springer Hiedelberg, pp 633-642, 2004.
[21] M.Mitchell, "An Introduction to Genetic Algorithm", MIT Press, 1996.
[22] L. Kuang, " DNIDS: A Dependable Network Intrusion Detection System Using the CSI-KNN Algorithm", Master thesis , Queen's University , Canada, September 2007.
[23] D.Song, M.I.Heywood, A.N.Ziniric-Heywood, " Training genetic programming on half million patterns: an example from anomaly detection", IEEE Transaction on Evolutionary Computation 9(3) p.225-239 , 2005.
[24] T.P.Fries, " A Fuzzy-Genetic Approach to Network Intrusion Detection", GECCO'2008, Atlanta, Georgia, USA, July 12-16,2008.
[25] J.Gomez, D.Dasgupta, "Evolving Fuzzy Classifier for Intrusion Detection", Proceedings of the IEEE, 2002.

Wafa' Al-Sharafat was born in Jordan. She received the B.S. in Computer Science from Hashemite University, Jordan in 2001 and M.Sc. degree in Computer Information system from Arab Academy for Banking and Financial Science, IT collage, Jordan in 2004. She is currently pursuing her PhD Degree in Computer information system from Arab Academy for Banking and Financial Science, IT collage, Jordan. Presently she is working as an Instructor in Computer Information System department in Al Al-Bayt University, IT Collage. Her main interests in Machine Learning, Genetic Algorithm, E-commerce.

Prof.Dr. Reyadh Sh.Naoum was born in Iraq. He received the B.S. in Mathematics from Basrah University, Iraq in 1969 and M.Sc. degree in Computing from Basrah University, Iraq in 1976 and PhD in computing from England in 1987. Presently he is working as Prof. in Arab Academy for Banking and Financial Science, IT collage, Jordan. His main interests in Neural Computing, Genetic Computing and Numerical Software.