

Genetic-Fuzzy Association Rules for Network Intrusion Detection Systems

Ming-Yang Su

Department of Computer Science and
Information Engineering,
Ming Chuan University, Taiwan
*minysu@mail.mcu.edu.tw

Chun-Yuen Lin

Department of Computer and
Communication Engineering,
Ming Chuan University, Taiwan
ololololo.safe@gmail.com

Sheng-Wei Chien, Han-Chung Hsu

Department of Computer Science and
Information Engineering,
Ming Chuan University, Taiwan
lancehtm@hotmail.com,
s110552178@gmail.com

Abstract—A network intrusion detection system (NIDS) based on genetic-fuzzy association rules is presented in the paper, which mines rules in an incremental manner in order to meet the real-time requirement of a NIDS. More precisely, the proposed NIDS adopts the incremental mining of fuzzy association rules from network traffic, in which membership functions of fuzzy variables are optimized by a genetic algorithm. The proposed online system belongs to anomaly detection, not misuse detection. Some denial-of-service (DoS) attacks were experimented in this study to show the performance of the proposed NIDS. The results show that the proposed NIDS can detect DoS attacks in both effectiveness and efficiency.

Keywords—Genetic-fuzzy association rules; incremental mining; membership functions; genetic algorithm; anomaly detection; denial-of-service (DoS) attacks

I. INTRODUCTION

Misuse detection, such as SNORT [1], aims to detect known attacks by characterizing the rules that govern these attacks. Thus, a rules update is most important and is frequently released by IDS vendors. However, the rapid emergence of new vulnerabilities and exploits makes misuse detection difficult to trust day after day. Anomaly detection is designed to capture any deviation from the profiles of normal behavior patterns. It is much more suitable than misuse detection for detecting unknown or novel attacks, but it has the potential to generate too many false positives. Many approaches have been proposed in previous literature concerning the design of anomaly NIDSs, such as neuro-fuzzy [2], support vector machine [3], decision tree [4], Bayesian Neural Networks [5], Naïve Bayes [6], genetic-fuzzy [7, 8], and fuzzy association rules [9, 10, 11, 12, 13, 14].

For a NIDS, effectiveness is usually measured by detection rate, false alarm rate, etc., and efficiency is measured by response time when an attack has occurred. This study focuses on how to create an anomaly NIDS, based on genetic optimized mining of fuzzy association rules, which can detect and react to a flooding attack in real-time. In contrast to traditional static mining for NIDS designs [9, 10, 11, 12, 13, 14], which can only be applied to offline NIDS, this system was designed based on an incremental mining approach, which enables the system to make a decision per time unit. In addition, a genetic algorithm was applied to help select the best membership

functions for the fuzzy variables of mining. Consequently the performance of the proposed response system can be improved.

The remainder of this paper is organized as follows: Section 2 states fuzzy association rules and membership functions of a genetic algorithm; Section 3 introduces the genetically optimized incremental mining algorithm for fuzzy association rules; Section 4 describes the proposed NIDS by genetic-fuzzy association rules; Section 5 shows some experimental results; and Section 6 is the conclusions.

II. FUZZY ASSOCIATION RULES

Agrawal and Srikant proposed the well-known Apriori algorithm [15] in 1994 in which, given two thresholds of $mini_sup$ and $mini_conf$, the algorithm will find all such rules as " $X \Rightarrow Y$ " with $support \geq mini_sup$ and $confidence \geq mini_conf$, where X and Y are subsets of the set of items, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ in the database D has *support* s if the percentage of records in D that contain $X \cup Y$ is s , and has *confidence* c if the ratio of the number of records in D that contain $X \cup Y$ to the number of records in D that contain X is c . Since the Apriori algorithm was designed for mining in databases with binary items, fuzzy association rules mining [16, 17, 18] has been one of the variations which has dealt with quantitative items.

While applying fuzzy association rules to an IDS (intrusion detection system) design, the term item, is replaced by the term feature, and thus, the term itemset is equivalent to the term feature-set, which is a set of features. Let $D = \{r_1, r_2, \dots, r_n\}$ be the database and $I = \{x_1, x_2, \dots, x_m\}$ represent all features appearing in D . The record r_i represents the i th m -tuple in D . Each quantitative feature x_k , $1 \leq k \leq m$, is associated with some fuzzy variables, say $v_1, v_2, \dots, v_t$. Every fuzzy variable is represented by a membership function. For easy representation in the following, MF_{x_i, v_j} , $1 \leq i \leq m$ and $1 \leq j \leq t$, is used to uniquely denote the j th membership function of feature x_i . In the mining of fuzzy association rules [18, 19], a fuzzy itemset consists of two parts: items and fuzzy variables, say $\langle X, U \rangle$, where $X = (x_1, x_2, \dots, x_k) \subset I$ is a collection of items (or features) and $U = (v_1, v_2, \dots, v_k)$ is the collection of corresponding fuzzy variables to X in order. Suppose there are totally n records in the database. Then the support of $\langle X, U \rangle$ is computed as:

$$\text{Sup}(<X, U>) = \frac{\sum_{i=1}^n \prod_{j=1}^k MF_{x_j \cdot v_j}(r_i[x_j])}{n}, \quad (1)$$

where $r_i[x_j]$ denotes the value of feature x_j of the i th record. For example, suppose four features, *#packet*, *#SYN*, *#ACK*, and *#connection*, are of concern in a NIDS, and each feature has three fuzzy variables, say *low*, *medium* and *high*, then $4 \times 3 = 12$ membership functions are involved, i.e., $MF_{\#packet, low}$ denotes the low function of feature *#packet*, $MF_{\#ACK, high}$ denotes the high function of feature *#ACK*, etc. Suppose there are three records in the database, as shown in Table 1. Then the support of the fuzzy itemset $<(\#packet, \#SYN, \#connection), (low, medium, low)>$ is computed as $(MF_{\#packet, low}(3260) \times MF_{\#SYN, medium}(135) \times MF_{\#connection, low}(27) + MF_{\#packet, low}(2170) \times MF_{\#SYN, medium}(75) \times MF_{\#connection, low}(65) + MF_{\#packet, low}(6123) \times MF_{\#SYN, medium}(213) \times MF_{\#connection, low}(89)) / 3$.

TABLE I. RECORDS IN DATABASE, FOR AN EXAMPLE

<i>#packet</i>	<i>#SYN</i>	<i>#ACK</i>	<i>#connection</i>
3260	135	75	27
2170	75	43	65
6123	213	27	89

A fuzzy association rule has the form of " $<X, U> \Rightarrow <Y, V>$ ", where $<X, U>$ and $<Y, V>$ are two fuzzy itemsets and $X \cap Y = \emptyset$. Let $Z = X \oplus Y$ and $W = U \oplus V$, where $\oplus$ is the concatenation operation. The support of the fuzzy association rule " $<X, U> \Rightarrow <Y, V>$ " is computed as $\text{Sup}(<Z, W>)$, and the confidence of the rule is computed as $\text{Sup}(<Z, W>) / \text{Sup}(<X, U>)$. For example, let us consider the following fuzzy association rule.

$<(\#SYN, \#ACK), (high, low)> \Rightarrow <(\#connection), (medium)>$.

//if *#SYN* is high and *#ACK* is low, then *#connection* is medium

Its support is computed as $\text{Sup}(<(\#SYN, \#ACK, \#connection), (high, low, medium)>)$, and its confidence is computed as $\text{Sup}(<(\#SYN, \#ACK, \#connection), (high, low, medium)>) / \text{Sup}(<(\#SYN, \#ACK), (high, low)>)$.

III. FUZZY ASSOCIATION RULES THROUGH INCREMENTAL MINING WITH GENETIC OPTIMIZATION ON MEMBERSHIP FUNCTIONS

Two important keys need to be addressed for a successful NIDS based on fuzzy association rules. One is applying incremental mining, instead of static mining, to meet real-time

demands. The other is a deliberate design for membership functions. We have proposed an incremental mining algorithm [20] to derive fuzzy association rules from network traffic. Based on the incremental mining algorithm, the paper further extends the algorithm to design an online detection system with genetic optimization in membership functions.

According to the proposed algorithm [20], packet information was collected to form one record every short time unit, and to mine out the newest rule set as the latest record was being gathered. Each current support value of a fuzzy itemset was kept in the memory for a time unit. As the next record was being gathered, the algorithm used the current support value to compute the next one, and then replaced it with the new one, as shown in the example below. Suppose the quantitative values of the four features, *#packet*, *#SYN*, *#ACK* and *#connection*, are measured for each time unit, and in sequence they are $t_1 = (97, 310, 66, 311)$, $t_2 = (215, 208, 75, 210)$, $t_3 = (62, 710, 41, 88)$, ..., $t_i = (230, 86, 31, 720)$, For the fuzzy itemset $<X, U> = <(\#packet, \#SYN, \#connection), (low, medium, low)>$, its support value s at $t_1, t_2, t_3, \dots, t_i, \dots$, is computed individually as:

$$s_{t1} = (MF_{\#packet, low}(97) \times MF_{\#SYN, medium}(310) \times MF_{\#connection, low}(311)) / 1 \rightarrow tmp;$$

$$s_{t2} = (tmp \times 1 + (MF_{\#packet, low}(215) \times MF_{\#SYN, medium}(208) \times MF_{\#connection, low}(210))) / 2 \rightarrow tmp;$$

$$s_{t3} = (tmp \times 2 + (MF_{\#packet, low}(62) \times MF_{\#SYN, medium}(710) \times MF_{\#connection, low}(88))) / 3 \rightarrow tmp;$$

...

$$s_{ti} = (tmp \times (i-1) + (MF_{\#packet, low}(230) \times MF_{\#SYN, medium}(86) \times MF_{\#connection, low}(720))) / i \rightarrow tmp.$$

...

Since the contribution of previous records to the current support of a fuzzy itemset is ephemerally saved in the variable, i.e., *tmp*, the cost of mining time will not be prolonged as the aggregate records increase. The whole algorithm can be found in [20].

Membership functions design is important, and they usually have profound affects on the mined rules. This research adopts a genetic algorithm to select the best membership functions for the features applied in the proposed NIDS system. The membership functions applied in this paper are listed below, in which $\text{abs}(\cdot)$ is the absolute function, $\wedge$ is the power operation, and a, b, c, d, e, f, g , and h are the constants. The output value of a membership function falls in the interval of [0, 1]. Each feature has its own membership functions, i.e., with different constants from feature to feature. The goal of a genetic algorithm is to derive the most appropriate membership functions for every feature. As shown in Figure 1, for a feature, its membership functions before and after genetic evolution, are depicted by solid lines and dashed lines, respectively.

Fuzzy membership functions:

$$\text{Low: } f(x) = 1 / (1 + \text{abs}((x - a) / b) \wedge c)$$

Medium: $f(x) = 1 / (1 + \text{abs}((x - d) / e) ^ f)$

High: $f(x) = 1 / (1 + \exp(-g(x - h)))$

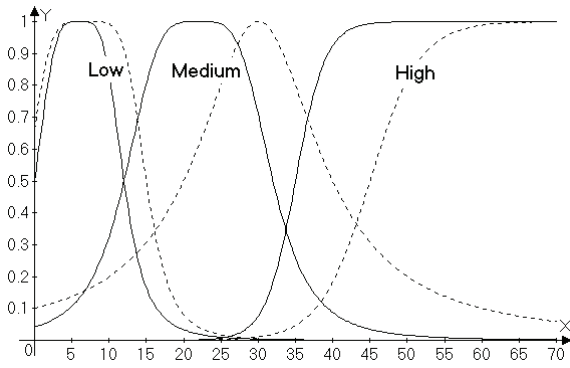


Figure 1 Fuzzy membership functions before and after genetic optimization

Suppose n features are considered in the NIDS design, then one chromosome in the genetic algorithm contains the n features' membership functions, as defined in Figure 2. The first chromosome in the experiment was constructed heuristically. An initial population, containing 30 chromosomes, was derived from the first chromosome by repeated application of the mutation process. In each generation, the fitness of each new chromosome was evaluated according to the performance of the proposed NIDS, using the fuzzy membership functions represented by the chromosome, i.e., the fitness function of the genetic algorithm was to maximize the overall accuracy in a given labeled dataset. A specified percentage of the chromosomes with high fitness were retained for the next generation. Then, parent chromosomes were repeatedly selected from the current generation, and new chromosomes were generated from these parents by crossover and mutation. One generation ended when the number of chromosomes for the next generation had reached the original population, i.e., 30 chromosomes. The evolution process was repeated for 250 generations.

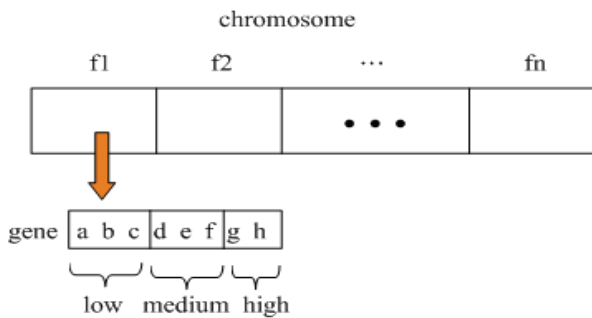


Figure 2 Gene and chromosome structures of evolution

IV. THE PROPOSED NIDS

The proposed system was designed for online detection and response to flooding attacks. A short time unit was defined,

thus, the reaction of the system must be performed once every time unit. Two seconds defined one unit in this research, i.e., one record was generated regardless of the number of captured packets. In the training stage, attack-free network traffic information was collected at the rate of one record every two seconds, and the genetic optimized membership functions were derived in advance from a labeled dataset. The online response system consists of four modules. Module_A collected run-time network traffic information online, at the rate of one record every two seconds, and consistently sent the records to module_B. Module_B applied the genetically optimized incremental mining algorithm to generate the newest fuzzy rule set every two seconds. At the same time, module_C also performed the algorithm every 2 seconds on the attack-free data records, i.e., adding one new attack-free record every 2 seconds to the set of mined data records. The two newest rule sets from module_C and module_B were then sent to module_D for comparison, again, once every two seconds. If their similarity was below the threshold, an anomaly of network traffic could be found. Once an attack had been detected, three actions resulted: an alarm was inserted into the database, module_A was required to gather more detailed information about the attack and, finally, the filtering rule in the firewall could be changed dynamically. In the implementation, a round-robin strategy was applied on the attack-free records in module_C.

In this paper, the similarity between the two rule sets was defined as follows. Let S_1 and S_2 be two rule sets. The similarity between them was computed as:

$$\text{sim}(S_1, S_2) = \frac{\text{SCORE1}}{|S_1|} \times \frac{\text{SCORE2}}{|S_2|}, \quad (2)$$

where $|S_1|, |S_2|$ represent the number of rules in the sets, $\text{SCORE1} = \sum_{\forall r \in S_1} \text{score}(r, S_2)$, $\text{SCORE2} = \sum_{\forall r \in S_2} \text{score}(r, S_1)$, and $\text{score}(r, S)$ was defined as:

$$\text{If there is a rule } r' = r \text{ in } S \text{ with support } s' \text{ and confidence } c', \text{ then } \text{score}(r, S) = 1 - \max\left(\frac{|c - c'|}{\max(c, c')}, \frac{|s - s'|}{\max(s, s')}\right), \text{ else } \text{score}(r, S) = -\max(c, s). \quad (3)$$

Two rules, r and r' , are regarded as $r = r'$, if they have the same antecedents and consequents. Finally, let SCORE1 or SCORE2 be 0 if it was a negative value. In our system, for any fuzzy itemset, its support was computed by:

$$\text{Current support} = \text{support due to the latest record} * k + \text{support due to all historical records} * (1 - k), \text{ where } k \text{ is a constant between 0 and 1.} \quad (4)$$

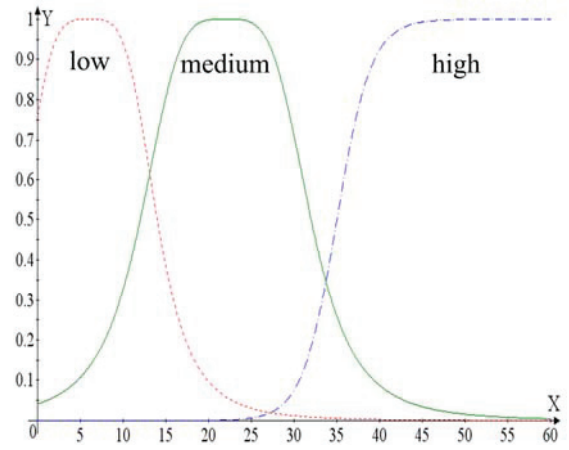
V. SIMULATION RESULTS

In the experiments of this section, a commercial application named IP Traffic [21] was applied to produce background traffic which can generate any amount of TCP/UDP/ICMP packets by hardware limit. Two hosts running IP Traffic played sender and receiver, respectively, and the receiver in the LAN and the sender transmitting packets was deployed through the Internet. During the experiments of this study, the amount of network traffic remained from 0 to 80 Mbps through random connection and random flow size. One laptop launched flooding attacks against the victim located in the LAN through the Internet. Our system, deployed in the LAN, was coded by Microsoft Visual C++ - MFC and run on a laptop with Windows XP. A total of 1000 attack-free traffic records were derived from IP Traffic in advance and stored in the database, one at a time for every two seconds. As these records in the database were exhausted, they were cyclically reused.

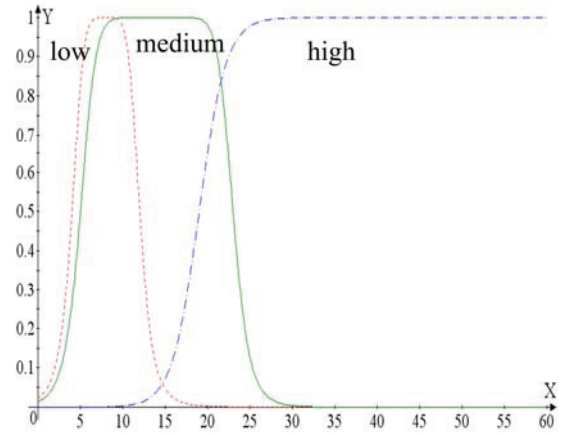
TABLE II. 15-FEATURE LIST

IP:D.IP slots hit	TCP:checksum_error count
IP:MF_Flag count	IP:checksum_error count
TCP:S.IP+URG_Flag+URG data count	ARP:Size error count
IGMP:checksum_error count	TCP:URG_Flag+URG_data count
ARP:S.IP+ARP count	UDP:Same_length_interval count
IGMP:Length>1000 count	ICMP:Length>1000 count
TCP:Sequence_number==0 count	IP:(Total length>1400 <40)&&TTL == 64 count
ICMP: Type error count	

All of the 15 features applied in this research are listed in Table 2, which were all derived from IP, TCP, UDP, ICMP, ARP, and IGMP headers. They were adopted to collect packet information and generate one record every two seconds. Some of the features were single condition, e.g., *D.IP slots hit*, and some were compound, e.g., *S.IP + ARP count*. The example, *D.IP slots hit*, for instance denoted how many slots of IP addresses were hit by network packets, according to their destination IP addresses. This study mapped a 32-bit IP address to one of 256 slots by its twice-folded IP address, i.e., mapping IP:x1.x2.x3.x4 to slot $(x1 \oplus x2) \oplus (x3 \oplus x4)$. The first three features in Table 2 before and after genetic optimization on membership functions are depicted in Figure3, Figure 4, and Figure 5, respectively. The before functions that were adopted in [20] are obtained by manually heuristic selections.

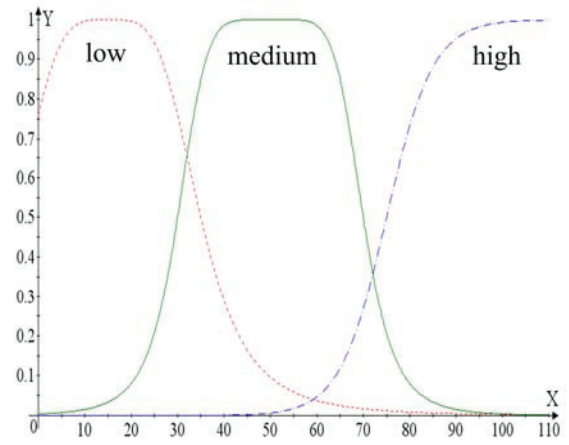


(a) Before

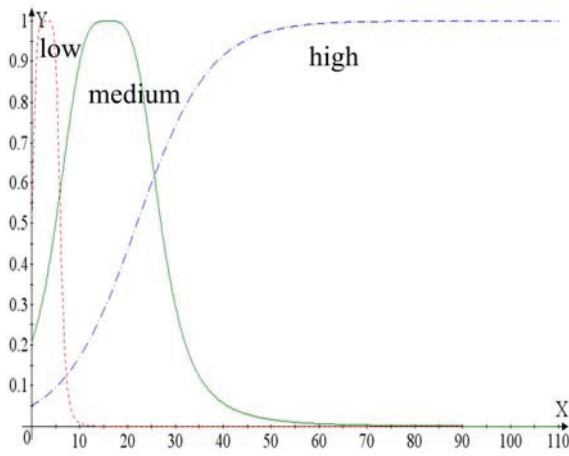


(b) After

Figure 3 Fuzzy membership functions of the feature "IP:D.IP slots hit" before and after genetic optimization

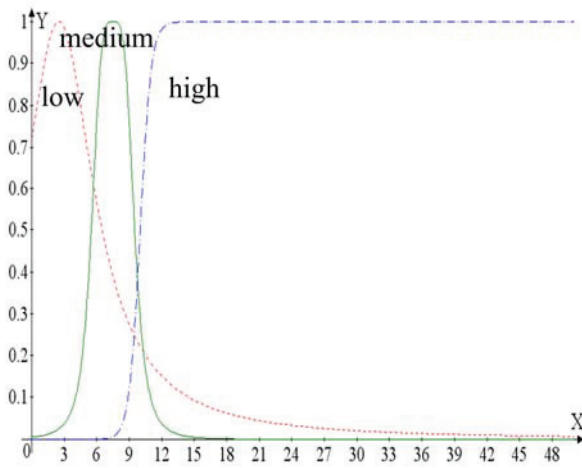


(a) Before

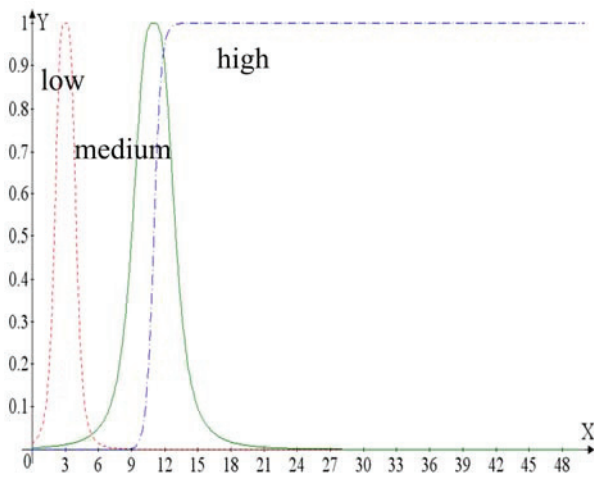


(b) After

Figure 4 Fuzzy membership functions of the feature " IP:MF_Flag count" before and after genetic optimization



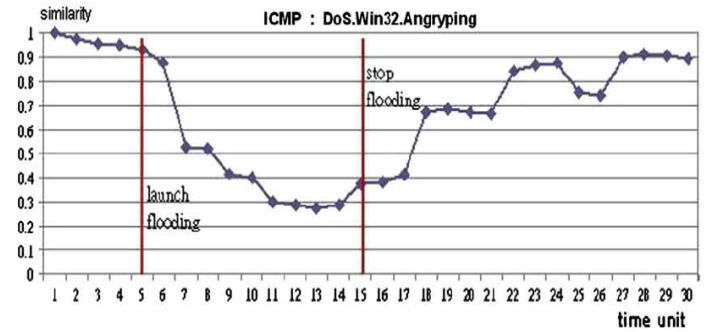
(a) Before



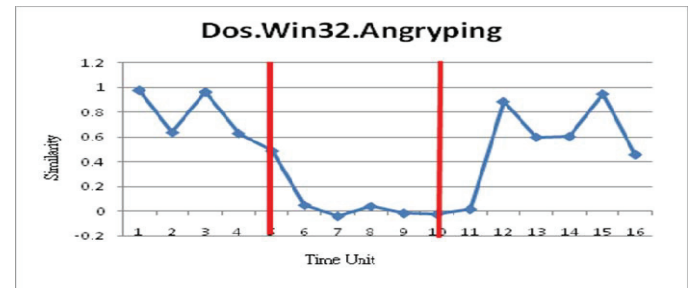
(b) After

Figure 5 Fuzzy membership functions of the feature " TCP:S.IP+URG_Flag+URG data count " before and after genetic optimization

As all 15 features' membership functions after genetic optimization were used to replace that in the original NIDS system [20], the similarity degradation at DoS attacks are more apparent than ever had in [20], as shown in Figures 6-8 for Angryryp Attack, ChinBomb Attack, and DeepThroat Attack, respectively. This represents that the modified NIDS, proposed in this paper, can be more precisely than its predecessor, because false positives can be reduced. The two red vertical lines in each figure denote the attack starting and ending time.

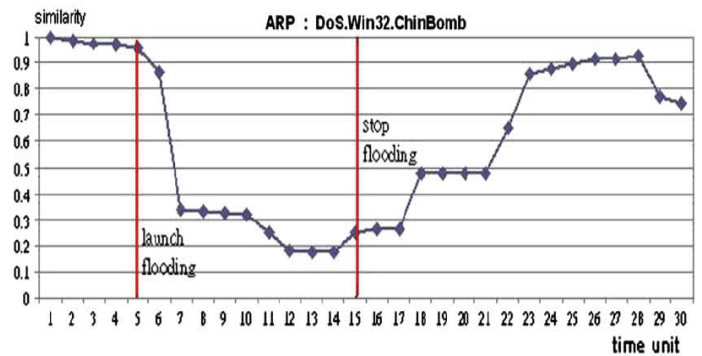


(a) without membership functions genetic optimization

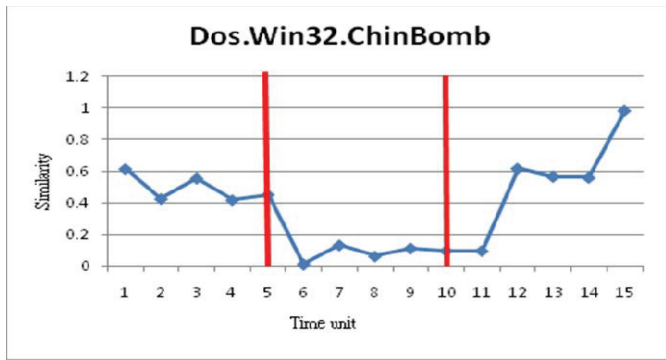


(b) with membership functions genetic optimization

Figure 6 NIDS performance for Angryryp Attack with and without fuzzy membership functions being genetic optimized

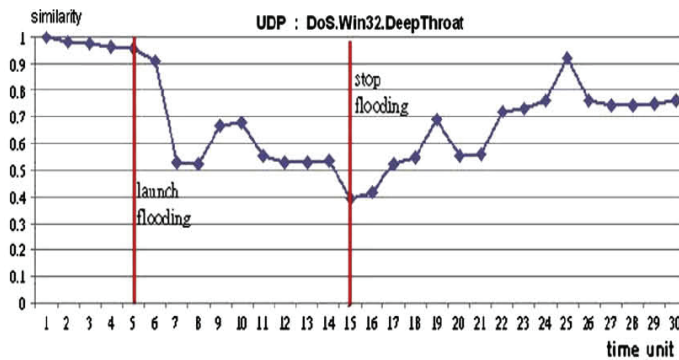


(a) without membership functions genetic optimization

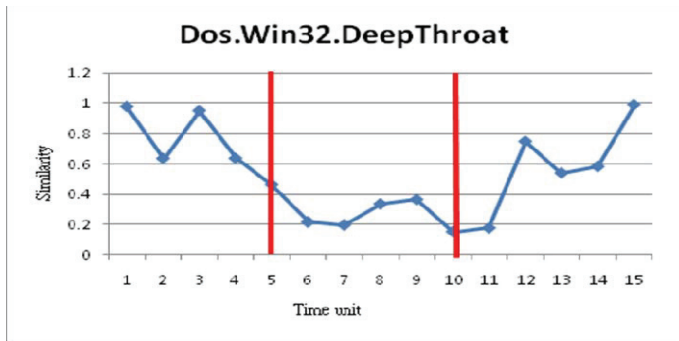


(b) with membership functions genetic optimization

Figure 7 NIDS performance for ChinBomb Attack with and without fuzzy membership functions being genetic optimized



(a) Without membership functions genetic optimization



(b) Without membership functions genetic optimization

Figure 8 NIDS performance for DeepThroat Attack with and without fuzzy membership functions being genetic optimized

Three cases of k in Eq. (4), $k = 0.2$, $k = 0.5$, and $k = 0.8$, were considered separately in the experiments. Detail of the similarity in degradation from the *win32.fakeping* attack is illustrated in Figure 9. A *win32.fakeping* attack was remotely launched at the fifth time unit lasting for 10 time units of flooding; the attack was stopped at the fifteenth time unit. This system deployed in the LAN showed that the similarity began to degrade at the sixth time unit, even more so during the next

ten units, and then started to upgrade at the sixteenth time unit. If the threshold of similarity is set to 0.5, the system, for the case of $k = 0.2$, would generate the first alarm at the seventh time unit (similarity value = 0.483554) and the last alarm at the twentieth time unit (similarity value = 0.478354). Ideally, the NIDS system in the LAN should show an alarm at the sixth time unit because an attack has occurred and to end the alarm at the sixteenth time unit because the attack has disappeared. Since historical data records are taken into consideration in the incremental design, it may be difficult for the similarity to immediately go down below the threshold as an attack occurs, and immediately go up above the threshold as the attack disappears. The smallest similarity value was 0.124654 for the case of $k = 0.2$ in Figure 9, occurred at the fifteenth time unit. It also can be concluded from Figure 9 that during the incremental mining the larger ratio of the latest record, i.e. k value, causes the system to become more susceptible. *Win32.fakeping* ceaselessly pings the victim machine with an abnormal size of payload.

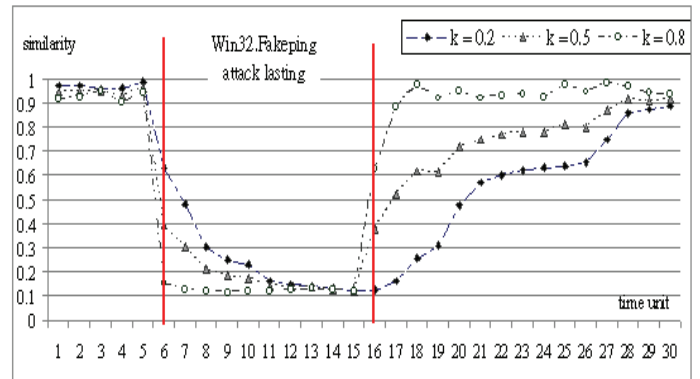


Figure 9 Similarity degradations while being Win32.Fakeping attacked

VI. CONCLUSION REMARKS

In the design, this proposed system has adopted incremental mining of fuzzy association rules, with genetic optimization on the membership functions. Experiments were conducted which demonstrated the effectiveness and efficiency of this intrusion detection system in detecting DoS attacks by an anomaly detection approach.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Council, project no.: NSC 96-2221-E-130-009 and 97-2221-E-130-014.

REFERENCES

- [1] B. Caswell, J. C. Foster, R. Russell, J. Beale and J. Posluns, Snort 2.0 Intrusion Detection, Syngress Press, 2003.
- [2] A. N. Toosj and M. Kahani, "A New Approach to Intrusion Detection Based on An Evolutionary Soft Computing Model Using Neuro-fuzzy

- Classifiers,” *Computer Communications*, Vol. 30, No. 10, pp. 2201-2212, 2007.
- [3] G. Zhang, J. Yin, Z. Liang, and Y.-G. Cai, “Prior Knowledge SVM-based Intrusion Detection Framework,” in *Proc. of the third Int’l Conf. on Natural Computation*, pp. 489-493, 2007.
 - [4] N. B. Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs Decision Trees in Intrusion Detection Systems,” in *Proc. of the ACM Symp. on Applied Computing*, pp. 420-424, 2004.
 - [5] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian Neural Networks for Internet Traffic Classification,” *IEEE Transactions on Neural Networks*, Vol. 18, No. 1, pp. 223-239, 2007.
 - [6] D. Zuev and A. W. Moore, “Traffic Classification using a Statistical Approach”, in *Proc. of the Sixth Passive and Active Measurement Workshop*, 2005.
 - [7] C.-H. Tsang, S. Kwong, and H. Wang, “Genetic-Fuzzy Rule Mining Approach and Evaluation of Feature Selection Techniques for Anomaly Intrusion Detection,” *Pattern Recognition*, Vol. 40, No. 9, pp. 2373-2391, 2007.
 - [8] M. S. Abadeh, J. Habibi, Z. Barzegar, and M. Sergi, “A Parallel Genetic Local Search Algorithm for Intrusion Detection in Computer Networks,” *Engineering Applications of Artificial Intelligence*, Vol. 20, No. 8, pp. 1058-1069, 2007.
 - [9] A. El-Semary, J. Edmonds, J. Gonzalez-Pino and M. Papa, “Applying Data Mining of Fuzzy Association Rules to Network Intrusion Detection,” in *Proc. of the IEEE Workshop on Information Assurance United States Military Academy*, 2006.
 - [10] S. M. Bridges, R. B. Vaughn, “Intrusion Detection via Fuzzy Data Mining,” in *Proc. of the Canadian Information Technology Security Symposium*, 2000.
 - [11] G. Florez, S. M. Bridges, and R. B. Vaughn, “An Improved Algorithm for Fuzzy Data Mining for Intrusion Detection,” in *Proc. of the IEEE Fuzzy Information*, 2002.
 - [12] J. E. Dickerson and J. A. Dickerson, “Fuzzy Network Profiling for Intrusion Detection,” in *Proc. of the Int’l Conf. on North American Fuzzy Information Processing*, pp. 301-306, 2000.
 - [13] M. Hossain, S. M. Bridges, R. B. Vaughn, Jr., “Adaptive Intrusion Detection with Data Mining,” in *Proc. of the IEEE Conf. on Systems, Man and Cybernetics*, pp. 3097-3103, 2003.
 - [14] B. Shanmugam and N. B. Idris, “Improved Hybrid Intelligent Intrusion Detection System Using AI Technique,” *Neural Network World*, Vol. 17, Issue 4, pp. 351-362, 2007.
 - [15] R. Agrawal, T. Imielinski and A. Swami, “Mining Association Rules Between Sets of Items in Large Databases,” in *Proc. of the ACM SIGMOD*, pp. 207-216, 1993.
 - [16] D. (Walter) Xie, “Fuzzy Association Rules discovered on Effective Reduced Database Algorithm,” in *Proc. of the IEEE Conf. on Fuzzy Systems*, pp. 779-784, 2005.
 - [17] Y. Gao, J. Ma, and L. Ma, “A New Algorithm for Mining Fuzzy Association Rules,” in *Proc. of the Conf. on Machine Learning and Cybernetics*, pp. 1635-1640, 2004.
 - [18] C. Kuok, A. Fu, and M. Wong, “Mining Fuzzy Association Rules in Databases,” in *Proc. of the ACM SIGMOD*, pp. 41-46, 1998.
 - [19] M. Kaya and R. Alhajj, “A Clustering Algorithm with Genetically Optimized Membership Functions for Fuzzy Association Rules Mining,” in *Proc. of the IEEE Conf. on Fuzzy Systems*, pp. 881-886, 2003.
 - [20] M.-Y. Su, G.-J. Yu, and C.-Y. Lin, “A Real-Time Network Intrusion Detection System for Large-Scale Attacks Based on an Incremental Mining Approach,” *Computers & Security*, Vol. 28, No. 5, pp. 301-309, July 2009.
 - [21] IP Traffic, <http://www.omnicor.com/netest.htm>.