

# An Online Response System for Anomaly Traffic by Incremental Mining with Genetic Optimization

Ming-Yang Su and Sheng-Cheng Yeh

**Abstract:** A flooding attack, such as DoS or Worm, can be easily created or even downloaded from the Internet, thus, it is one of the main threats to servers on the Internet. This paper presents an online real-time network response system, which can determine whether a LAN is suffering from a flooding attack within a very short time unit. The detection engine of the system is based on the incremental mining of fuzzy association rules from network packets, in which membership functions of fuzzy variables are optimized by a genetic algorithm. The incremental mining approach makes the system suitable for detecting, and thus, responding to an attack in real-time. This system is evaluated by 47 flooding attacks, only one of which is missed, with no false positives occurring. The proposed online system belongs to anomaly detection, not misuse detection. Moreover, a mechanism for dynamic firewall updating is embedded in the proposed system for the function of eliminating suspicious connections when necessary.

**Index Terms:** Anomaly detection, genetic algorithm, firewall, flooding attack, fuzzy association rules, membership functions, online incremental mining.

## I. INTRODUCTION

An intrusion detection system (IDS) can be categorized according to the monitoring strategy it follows and its detection strategy. As for the former, there is a distinction between a host-based intrusion detection system (HIDS) [1] and a network-based intrusion detection system (NIDS) [2], [3]. In general, a HIDS detects transformations in the local integrity of a computer, e.g., the file system, while a NIDS detects intrusions originating from network adapters in the form of protocol packets. A HIDS protects a single host, but a NIDS potentially protects a network. While a monitoring strategy defines the “where,” a detection strategy defines the “how.” Therefore, there exists misuse detection [4], [5] and anomaly detection [3], [6]. Misuse detection, such as SNORT [5], 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. In this paper, the proposed online system is an anomaly NIDS which can detect and respond to flooding attacks in real-time. It is based on a genetically optimized incremental mining algorithm for fuzzy association rules.

Many approaches have been proposed in previous literature concerning the design of anomaly NIDSs, such as neuro-fuzzy [2], support vector machine [6], decision tree [7], Bayesian neural networks [8], Naive Bayes [9], genetic-fuzzy [3], [10], and fuzzy association rules [11]–[16]. However, to the best of our knowledge, all anomaly NIDSs emphasize effectiveness, but neglect efficiency. Usually, effectiveness is measured by detection rate, false alarm rate, etc., and efficiency is measured by response time when an attack has occurred. Much research into anomaly NIDS, such as [2], [3], [6], [7], [10], has evaluated the proposed approach by KDD CUP99 TCPDUMP datasets [17], which meant that the research was designed for offline use and, thus, could not meet the real-time characteristic of a NIDS. This was because the 41 features presented in KDD CUP99 are complicated and varied [2], [18]: The first 9 are intrinsic features which describe the basic features of individual TCP connections and can be obtained from raw TCPDUMP files; features 10 to 22 are content-based features obtained by examining the data portion of a connection and suggested by domain knowledge; features 23 to 31 are traffic-based features computed using a two-second time window (“time-based”); while features 32 to 41 are also traffic-based features, but computed using a window of 100 connections (“host-based”).

This research 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 [11]–[16], 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 II presents background knowledge, including fuzzy association rules and membership functions; Section III introduces the genetically optimized incremental mining algorithm for fuzzy association rules; Section IV describes the online response system proposed by this research; Section V provides experimental results; and Section VI presents our conclusions.

Manuscript received October 15, 2008; approved for publication by Jong Kim, Division III Editor, May 11, 2010.

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

M.-Y. Su is with the Department of Computer Science and Information Engineering, Ming Chuan University, Taoyuan, Taiwan, email: minysu@mail.mcu.edu.tw.

S.-C. Yeh is with the Department of Computer and Communication Engineering, Ming Chuan University, Taoyuan, Taiwan, email: pteryeh@mail.mcu.edu.tw.

## II. BACKGROUND

Fuzzy association rules have been receiving a great deal of attention recently, and have had many applications in different fields [11], [19]–[22]. In fuzzy association rules mining, membership function designs are important, and usually have profound effects on the mined rules. This section will briefly introduce the mining of fuzzy association rules, and the characteristics of membership functions.

Agrawal and Srikant proposed the well-known Apriori algorithm [23] 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 *mini\_sup* and confidence *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 [24]–[26] has been one of the variations which has dealt with quantitative items.

While applying fuzzy association rules to an IDS 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 [26], [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

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

where  $r_i[x_j]$  denotes the value of feature  $x_j$  of the  $i$ th record. For example, suppose four features, #packet, #SYNS, #ACKS, 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.,  $MF_{\#packet, \text{low}}$  denotes the low function of feature #packet,  $MF_{\#ACK, \text{high}}$  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  $\langle (\#packet, \#SYN, \#connection), (\text{low}, \text{medium}, \text{low}) \rangle$  is computed as  $(MF_{\#packet, \text{low}}(3260) \times MF_{\#SYN, \text{medium}}(135) \times MF_{\#connection, \text{low}}(27) + MF_{\#packet, \text{low}}(2170) \times MF_{\#SYN, \text{medium}}(75) \times MF_{\#connection, \text{low}}(65) + MF_{\#packet, \text{low}}(6123) \times MF_{\#SYN, \text{medium}}(213) \times MF_{\#connection, \text{low}}(89)) / 3$ .

A fuzzy association rule has the form of  $\langle X, U \rangle \Rightarrow \langle Y, V \rangle$ ,

Table 1. Records in database as an example.

#packet	#SYN	#ACK	#connection
3260	135	75	27
2170	75	43	65
6123	213	27	89

where  $\langle X, U \rangle$  and  $\langle Y, V \rangle$  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 “ $\langle X, U \rangle \Rightarrow \langle Y, V \rangle$ ” is computed as  $\text{Sup}(\langle Z, W \rangle)$ , and the confidence of the rule is computed as  $\text{Sup}(\langle Z, W \rangle) / \text{Sup}(\langle X, U \rangle)$ . For example, let us consider the following fuzzy association rule.

$\langle (\#SYN, \#ACK), (\text{high}, \text{low}) \rangle \Rightarrow \langle (\#connection), (\text{medium}) \rangle$ .  
 //if #SYN is high and #ACK is low, then #connection is medium  
 Its support is computed as  $\text{Sup}(\langle (\#SYN, \#ACK, \#connection), (\text{high}, \text{low}, \text{medium}) \rangle)$ , and its confidence is computed as  $\text{Sup}(\langle (\#SYN, \#ACK, \#connection), (\text{high}, \text{low}, \text{medium}) \rangle) / \text{Sup}(\langle (\#SYN, \#ACK), (\text{high}, \text{low}) \rangle)$ .

## III. GENETIC OPTIMIZATION OF MEMBERSHIP FUNCTIONS IN INCREMENTAL MINING FOR FUZZY ASSOCIATION RULES

As discussed in Section II, two important keys need to be addressed for a successful NIDS, which are 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 [27] to derive fuzzy association rules from network packets. Based on the incremental mining algorithm, the paper further extends the algorithm to design an online response system with genetic optimization in membership functions.

### A. Incremental Mining for Fuzzy Association Rules

According to the proposed algorithm [27], 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  $t_1 = (97, 310, 66, 311)$ ,  $t_2 = (215, 208, 75, 210)$ ,  $t_3 = (62, 710, 41, 88)$ ,  $\dots$ ,  $t_i = (230, 86, 31, 720)$ ,  $\dots$ . For the fuzzy itemset  $\langle X, U \rangle = \langle (\#packet, \#SYN, \#connection), (\text{low}, \text{medium}, \text{low}) \rangle$ , its support value  $s$  at  $t_1, t_2, t_3, \dots, t_i, \dots$ , is computed individually as

$$s_{t_1} = (MF_{\#packet, \text{low}}(97) \times MF_{\#SYN, \text{medium}}(310) \times MF_{\#connection, \text{low}}(311)) / 1 \rightarrow \text{tmp};$$

Membership functions:

$$\begin{aligned} \text{Low: } f(x) &= 1 / (1 + \text{abs}((x - a) / b)^c) \\ \text{Medium: } f(x) &= 1 / (1 + \text{abs}((x - d) / e)^f) \\ \text{High: } f(x) &= 1 / (1 + \exp(-g(x - h))) \end{aligned}$$

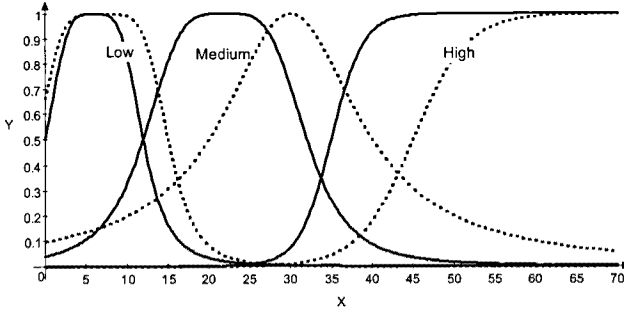


Fig. 1. Membership functions before and after genetic optimization.

$$\begin{aligned} s_{t_2} &= (\text{tmp} \times 1 + (MF_{\# \text{packet}, \text{low}}(215) \times MF_{\# \text{SYN}, \text{medium}}(208) \\ &\quad \times MF_{\# \text{connection}, \text{low}}(210))) / 2 \rightarrow \text{tmp}; \\ s_{t_3} &= (\text{tmp} \times 2 + (MF_{\# \text{packet}, \text{low}}(62) \times MF_{\# \text{SYN}, \text{medium}}(710) \\ &\quad \times MF_{\# \text{connection}, \text{low}}(88))) / 3 \rightarrow \text{tmp}; \\ &\vdots \\ s_{t_i} &= (\text{tmp} \times (i - 1) + (MF_{\# \text{packet}, \text{low}}(230) \\ &\quad \times MF_{\# \text{SYN}, \text{medium}}(86) \times MF_{\# \text{connection}, \text{low}}(720))) / i \rightarrow \text{tmp} \\ &\vdots \end{aligned}$$

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. In algorithm [27], a node structure is declared, and every fuzzy item-set is represented by a node. The node structure contains a field to retain the above tmp value. Since incremental mining requires dynamic updating the support value of each fuzzy itemset, every node is scanned once as the latest record is gathered. The whole algorithm and its performance, including time and memory consumption, can be found in [27].

Theoretically, the total number of nodes (fuzzy itemsets) generated in the algorithm was

$$\sum_{x=1}^{\text{maxlen}} m^x C_x^n \quad (2)$$

where  $n$  is the number of features,  $m$  is the number of degrees of each features,  $\text{maxlen}$  is defined as the longest itemset length which the mining algorithm concerned, and  $C$  was the combinatorial operation. In this paper,  $\text{maxlen}$  is set to 2,  $m$  to 3, and  $n$  to 20. As  $\text{maxlen}$  is set to 2, i.e., only rules derived from large itemsets with length 2 are considered. Thus, the number of rules can be reduced to 200 ~ 350.

#### B. Genetic Optimization of Membership Functions

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

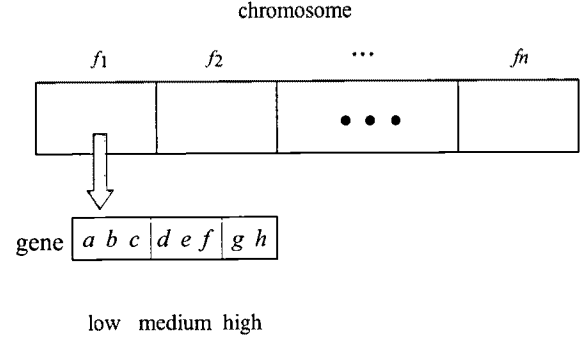


Fig. 2. Gene and chromosome structures of evolution.

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 Fig. 1, for a feature, its membership functions before and after genetic evolution, are depicted by solid lines and dashed lines, respectively.

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 Fig. 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.

#### IV. PROPOSED ONLINE RESPONSE SYSTEM

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. mod-

ule\_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:

$$\text{sim}(S_1, S_2) = \frac{\text{SCORE1} \cdot \text{SCORE2}}{|S_1| \cdot |S_2|} \quad (3)$$

where  $|S_1|$ ,  $|S_2|$  represent the number of rules in the sets,  $\text{SCORE1} = \sum_{r \in S_1} \text{score}(r, S_2)$ , and  $\text{SCORE2} = \sum_{r \in S_2} \text{score}(r, S_1)$ . If there is a rule  $r' = r$  in  $S$  with support  $s'$  and confidence  $c'$ , then  $\text{score}(r, S)$  was defined as:

$$\text{score}(r, S) = 1 - \max\left(\frac{|c - c'|}{\max(c, c')}, \frac{|s - s'|}{\max(s, s')}\right),$$

else

$$\text{score}(r, S) = -\max(c, s). \quad (4)$$

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

$$\begin{aligned} \text{Current support} &= \text{support due to the latest record} \times k \\ &+ \text{support due to all historical records} \\ &\times (1 - k) \end{aligned} \quad (5)$$

where  $k$  is a constant between 0 and 1. For a NIDS design, the importance of the latest data record should be greater than that of any single historical data record. Three cases of  $k$ ,  $k = 0.2$ ,  $k = 0.5$ , and  $k = 0.8$ , were studied in the experiments.

## V. SIMULATION RESULTS AND ANALYSES

Although four attack categories, i.e., DoS, Probe, U2R, and R2L, have been identified in KDD CUP99 datasets [17], all of the related works in the literature based on fuzzy association rules [11]–[16] have applied only one or two flooding attacks for evaluation: El-Semary *et al.* [11] used one attack named *ip-sweep* to evaluate their system; Bridges and Vaughn [12] applied another attack named *mscan* to demonstrate their method's effectiveness; Florez *et al.* [13] applied *mailbomb* to show their performance; Dickerson *et al.* [14] applied two kinds of attacks,

Table 2. Feature list.

TCP: S.IP+SYN count	TCP: S.IP+URG_Flag +URG_data count
TCP: S.IP+ACK_Flag +ACK count	ARP: S.IP+ARP count
IP: D.IP slots hit	IP: Header length!=20 count
IP: MF_Flag count	IP: (Total length>1400  <40) && TTL == 64 count
IP: checksum_error count	TCP: ACK_Flag+ACK count
TCP: checksum_error count	TCP: SYN count
UDP: Same_length_interval count	UDP: Length distribution count
ICMP: Type error count	ICMP: checksum_error count
ICMP: S.IP+ICMP packets count	ICMP: ICMP packets count
IGMP: checksum_error count	IGMP: Length>1000 count

TCP port scans and ICMP (ping) scans in their experiments; Hossain *et al.* [15] used *portscan* to show their effectiveness; while Shanmugam and Idris [16], as [12], applied *mscan*.

The reason for applying only one or two flooding attacks to the evaluations in [11]–[16] may have been due to the following two considerations. First, in order to detect the other two kinds of attacks, U2R and R2L, in KDD CUP99 datasets, a NIDS has to check the packet payload because the malicious actions are due to content-based features, suggested by domain knowledge, in the data portion of a connection. As mentioned in the Introduction, the 41 features presented in KDD CUP99 are complicated and diverse. The above anomaly NIDSs [11]–[16] that were based on fuzzy association rules checked only headers without packet payloads. However, U2R and R2L do not cause any malformation of packets or packet violation to network protocols. Secondly, the amount of packets generated by U2R and R2L in KDD CUP99 may be negligible in huge background traffic. Thus, there may have been no significant difference between rules mined from attacked online network traffic and rules mined from attack-free network traffic.

Attacking tools were downloaded from the VX Heavens website (<http://vx.netlux.org/>) which is maintained by the well-known antivirus lab, Kaspersky. A total of 47 attacking tools of flooding were studied for this research. In the experiments of this section, a commercial application named IP traffic [28] 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++ 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

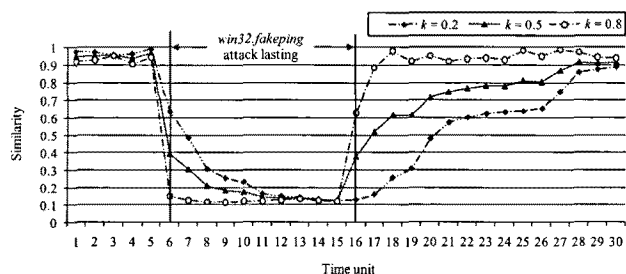


Fig. 3. Similarity degradations while being *win32.fakeping* attacked.

were exhausted, they were cyclically reused.

All of the 20 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 + ACK Flag + ACK number. The former example, D.IP slots hit, 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:  $x_1.x_2.x_3.x_4$  to slot  $(x_1 \oplus x_2)(x_3 \oplus x_4)$ . The latter example denotes the maximal number of packets of source IP addresses belonging to the same slot; (ACK flag = 0 and ACK number empty). Every feature in Table 2 had three degrees: Low, medium, and high.

Three cases of  $k$  in (5),  $k = 0.2$ ,  $k = 0.5$ , and  $k = 0.8$ , were considered separately in the experiments. Detail of the similarity in degradation from the *win32.fakeping* attack is illustrated in Fig. 3. 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 Fig. 3, occurred at the fifteenth time unit. It also can be concluded from Fig. 3 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.

More precisely, Fig. 4 illustrates the changes of number of fuzzy association rules obtained from online network traffic and database attack-free traffic, and the number of equal rules between them. As shown in Fig. 4, there were 342 rules mined







Ming Chan University - NIDS by Incremental Mining						
File(F)	View(V)	Go(G)	Mode	Option	Help(H)	
						
Status	Data - Time	Record	Similarity value	Online Rules	Database Rules	Same Rules
U	2:53:22	--Stop--	--Accuracy System halt--			
	5/23 - 2:53:21	30	0.890282	326	342	326
	5/23 - 2:53:19	29	0.876223	326	342	326
	5/23 - 2:53:17	28	0.859121	326	342	326
	5/23 - 2:53:15	27	0.794299	299	342	299
	5/23 - 2:53:13	26	0.653962	278	342	278
	5/23 - 2:53:11	25	0.636654	278	342	278
	5/23 - 2:53:9	24	0.633332	278	342	278
	5/23 - 2:53:7	23	0.623387	278	342	278
	5/23 - 2:53:5	22	0.601102	278	342	278
	5/23 - 2:53:3	21	0.572566	278	342	278
	5/23 - 2:53:2	20	0.478354	306	342	278
	5/23 - 2:53:0	19	0.307158	308	342	240
	5/23 - 2:52:58	18	0.256313	284	342	216
I	5/23 - 2:52:56	17	0.160502	336	342	210
I	5/23 - 2:52:54	16	0.127377	342	342	210
I	5/23 - 2:52:52	15	0.124654	342	342	210
I	5/23 - 2:52:49	14	0.131876	342	342	210
I	5/23 - 2:52:47	13	0.141161	342	342	210
I	5/23 - 2:52:45	12	0.148196	342	342	210
I	5/23 - 2:52:43	11	0.161958	334	342	210
I	5/23 - 2:52:41	10	0.230539	278	342	210
I	5/23 - 2:52:39	9	0.253450	284	342	216
I	5/23 - 2:52:37	8	0.304854	302	342	234
I	5/23 - 2:52:34	7	0.483554	310	342	278
I	5/23 - 2:52:32	6	0.636107	282	342	282
	5/23 - 2:52:30	5	0.989109	342	342	342
	5/23 - 2:52:28	4	0.962416	342	342	342
	5/23 - 2:52:26	3	0.957629	342	342	342
	5/23 - 2:52:24	2	0.971983	342	342	342
	5/23 - 2:52:22	1	0.973938	342	342	342
U	2:52:20	--System...				

Fig. 4. Execution window of the proposed system for *win32.fakeping*.

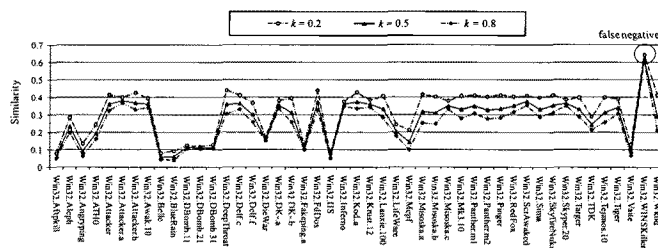


Fig. 5. Smallest similarity values for different flooding attacks.

from traffic records stored in the database. Although the total number of packets generated by IP traffic was set between 0 Mbps and 80 Mbps through random connections, the number of rules obtained was maintained at 342. Also with IP traffic as the background traffic, the win32.fakeping was launched at the fifth time unit. The number of rules mined from online traffic was reduced to 282 at the sixth time unit, and all the 282 rules (set  $S_1$ ) were equal to part of the 342 rule set (set  $S_2$ ) mined from attack-free database records, i.e.,  $S_1$  is a subset of  $S_2$  at this time; thus,  $342 - 282 = 60$  rules in  $S_2$  were not appeared in  $S_1$ . According to the similarity computation formula shown in Section IV, the similarity between  $S_1$  and  $S_2$  was 0.636107 at the sixth time unit. While flooding continued, the system mined 334 rules from online traffic at the eleventh time unit (again, say set  $S_1$ ), in which 210 rules were equal to part of the 342-rule set mined from attack-free database records at the eleventh time unit (set  $S_2$ ): Thus,  $334 - 210 = 124$  rules appeared in  $S_1$ , but not in  $S_2$ ; On the other hand,  $342 - 210 = 132$  rules appeared in  $S_2$ , but not in  $S_1$ . The similarity between  $S_1$  and  $S_2$  at the eleventh time unit was only 0.161958. It was concluded that when attacked by flooding, fuzzy association rules mined from online network traffic could really deflect such an attack if the change of these rules could be deliberately utilized, as with the design of similarity computation in this paper.

All 47 flooding attacks were tested and their smallest similarity values, as shown in Fig. 5, were all below 0.5, except for *win32.winskiller*. If the threshold was set to 0.5, only *win32.winskiller* caused a false negative. *win32.winskiller* attacked the NetBIOS name service protocol located in the upper UDP. The IP and UDP headers of *win32.winskiller* packets were normal. In our system, all mal-formatted NetBIOS headers belonged to the data payload of UDP and, thus, could be ignored, which was why the smallest similarity could go no further down.

If the LAN was simultaneously attacked by two or more flooding attacks, the system could detect such mixed attacks more rapidly and effectively, because (3) the amount of malicious (or mal-formatted) packets was much more than those under a single attack, and (4) the chance of an anomaly, caused by the features listed in Table 2, was increased. For example, the smallest similarity values of *win32.kod.a* and *win32.kod.a* in the case of  $k = 0.2$  were 0.42649986 and 0.3746681, respectively. While simultaneously mixing these two attacks, the smallest similarity value was reduced to 0.23415686. Finally, to consider the variety of network applications, to avoid false positives, the threshold was set to 0.5 in this research, as determined by experiments.

## VI. CONCLUSION

Many anomaly NIDSs, e.g., [2], [3], [6], [7], [10], in the literature have applied KDD CUP99 datasets [17] in their experiments: All belong to offline detections because many of the 41 features proposed by KDD CUP99 are content- or connection-based [2], [18]. This study focuses on online real-time response to anomaly traffic caused by DoS or Worm flooding attacks. In the design, this proposed system has adopted incremental mining of fuzzy association rules, with genetic optimization on the membership functions. A mechanism for dynamic firewall updating is embedded in the proposed system, so that it can cut off some suspicious connections in real-time. Experiments were conducted which demonstrated the effectiveness and efficiency of this intrusion response system in preventing flooding attacks.

## REFERENCES

- [1] K. Lu, Z. Chen, Z. Jin, and J. Guo, "An adaptive real-time intrusion detection system using sequences of system call," in *Proc. IEEE Canadian Conf. Electric. Comput. Eng.*, 2003, pp. 789–792.
- [2] A. N. Toosj and M. Kahani, "A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers," *Comput. Commun.*, vol. 30, no. 10, pp. 2201–2212, 2007.
- [3] 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.
- [4] S. Lekkas and L. Mikhailov, "Towards the development of OMNIVOR: An evolving intelligent intrusion detection system," *Appl. and Innovations in Intell. Syst.*, pp. 303–308, 2007.
- [5] B. Caswell, J. C. Foster, R. Russell, J. Beale, and J. Posluns, *Snort 2.0 Intrusion Detection*. Syngress Press, 2003.
- [6] G. Zhang, J. Yin, Z. Liang, and Y. G. Cail, "Prior knowledge SVM-based intrusion detection framework," in *Proc. Int. Conf. Natural Comput.*, 2007, pp. 489–493.
- [7] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs. decision trees in intrusion detection systems," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 420–424.
- [8] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, 2007.
- [9] D. Zuev and A. W. Moore, "Traffic classification using a statistical approach," in *Proc. Passive and Active Meas. Workshop*, 2005.
- [10] M. S. Abadeh, J. Habibi, Z. Barzegar, and M. Sergi, "A parallel genetic local search algorithm for intrusion detection in computer networks," *Eng. Appl. of Artificial Intell.*, vol. 20, no. 8, pp. 1058–1069, 2007.
- [11] A. E.-Semary, J. Edmonds, J. G. A. Pino and M. Papa, "Applying data mining of fuzzy association rules to network intrusion detection," in *Proc. IEEE Workshop Inf. Assurance United States Military Academy*, 2006.
- [12] S. M. Bridges and R. B. Vaughn, "Intrusion detection via fuzzy data mining," in *Proc. Canadian Inf. Technol. Security Symp.*, 2000.
- [13] G. Florez, S. M. Bridges, and R. B. Vaughn, "An improved algorithm for fuzzy data mining for intrusion detection," in *Proc. IEEE Fuzzy Inf.*, 2002.
- [14] J. E. Dickerson and J. A. Dickerson, "Fuzzy network profiling for intrusion detection," in *Proc. Int. Conf. North American Fuzzy Inf. Process.*, 2000, pp. 301–306.
- [15] M. Hossain, S. M. Bridges, and R. B. Vaughn Jr., "Adaptive intrusion detection with data mining," in *Proc. IEEE Conf. Syst., Man and Cybern.*, 2003, pp. 3097–3103.
- [16] B. Shanmugam and N. B. Idris, "Improved hybrid intelligent intrusion detection system using AI technique," *Neural Netw. World*, vol. 17, no. 4, pp. 351–362, 2007.
- [17] *KDD CUP 1999 for intrusion detection evaluation*, ACM Special Interest Group on Knowledge Discovery and Data Mining. [Online]. Available: <http://www.sigkdd.org/kddcup/index.php?section=1999&method=data>
- [18] *The UCI KDD Archive*, UCI Knowledge Discovery in Databases Archive. [Online]. Available: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup>
- [19] M. Kaya and R. Alhajj, "A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining," in *Proc. IEEE Conf. Fuzzy Syst.*, 2003, pp. 881–886.
- [20] W.-H. Au and K. C. C. Chan, "Mining fuzzy association rules in a bank-account database," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 2, pp. 238–248, 2003.
- [21] M. Kaya and R. Alhajj, "Facilitating fuzzy association rules mining by using multi-objective genetic algorithms for automated clustering," in *Proc. IEEE Conf. Data Mining*, 2003, pp. 561–564.
- [22] P.-Qiliu, Z.-Z. Li, and Y.-L. Zhao, "Algorithm of mining fuzzy association rules in network management," in *Proc. IEEE Conf. Mach. Learning and Cybern.*, 2003, pp. 123–127.
- [23] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD*, 1993, pp. 207–216.
- [24] D. W. Xie, "Fuzzy association rules discovered on effective reduced database algorithm," in *Proc. IEEE Conf. Fuzzy Syst.*, pp. 779–784, 2005.
- [25] Y. Gao, J. Ma, and L. Ma, "A new algorithm for mining fuzzy association rules," in *Proc. Conf. Mach. Learning and Cybern.*, 2004, pp. 1635–1640.
- [26] C. Kuok, A. Fu, and M. Wong, "Mining fuzzy association rules in databases," in *Proc. ACM SIGMOD*, 1998, pp. 41–46.
- [27] M.-Y. Su, S.-C. Yeh, and K.-C. Chang, "Using incremental mining approach to analyze network traffic online based on fuzzy rules," *J. Internet Technol.*, vol. 9, no. 1, pp. 77–86, Feb. 2008; A simplified version also appeared in *Proc. IEEE Conf. Adv. Inf. Netw. and Appl.*, 2008, entitled - "Using incremental mining to generate fuzzy rules for real-time network intrusion detection systems."
- [28] *IP Traffic*, Omnicor. [Online]. Available: <http://www.omnicor.com/netest.htm>



**Ming-Yang Su** received his B.S. degree from the Department of Computer Science and Information Engineering of Tunghai University, Taiwan in 1989, and received his M.S. and Ph.D. degrees from the same Department of the National Central University and National Taiwan University in 1991 and 1997, respectively. He is an IEEE Member, and currently an Associate Professor of the Department of Computer Science and Information Engineering at the Ming Chuan University, Taoyuan, Taiwan. His research interests include network security, intrusion detection/prevention, malware detection, wireless ad hoc network, and wireless sensor networks.



**Sheng-Cheng Yeh** was born in Taipei, Taiwan, R.O.C., in 1966. He received the B.S. degree in Electrical Engineering from National Taiwan Institute of Technology (National Taiwan University of Science and Technology), Taipei, Taiwan, in 1991, the M.S. and Ph.D. degrees in Electrical Engineering from National Central University, Taoyuan, Taiwan, in 1993 and 2000, respectively. Since 2003, he has been a Faculty Member of the Department of Computer and Communication Engineering, Ming Chuan University, Taoyuan, Taiwan, where he is currently an Associate Professor. His research interests include the design and analysis of computer networks, wireless communications, network security, and locating technologies.