available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach

Ming-Yang Su[a,*], *Gwo-Jong Yu*[b], *Chun-Yuen Lin*[a]

[a]*Department of Computer Science and Information Engineering, Ming Chuan University, Taoyuan Campus, Taiwan*
[b]*Department of Computer Science and Information Engineering, Aletheia University, Taipei, Taiwan*

ARTICLE INFO

Article history:
Received 18 February 2008
Received in revised form
4 December 2008
Accepted 11 December 2008

Keywords:
Network security
Real-time NIDS
Association rules
Fuzzy association rules
Online mining
Incremental mining
Denial-of-service attacks

ABSTRACT

None of the previously proposed Network Intrusion Detection Systems (NIDSs), which are subject to fuzzy association rules, can meet real-time requirements because they all apply static mining approaches. This study proposed a real-time NIDS with incremental mining for fuzzy association rules. By consistently comparing the two rule sets, one mined from online packets and the other mined from training attack-free packets, the proposed system can render a decision every 2 seconds. Thus, compared with traditional static mining approaches, the proposed system can greatly improve efficiency from offline detection to real-time online detection. Since the proposed system derives features from packet headers only, like the previous works based on fuzzy association rules, large-scale attack types are focused. Many DoS attacks were experimented in this study. Experiments were performed to demonstrate the excellent effectiveness and efficiency of the proposed system. The system may not cause false alarms because normal programs supposedly would not generate enough mal-formatted packets, or packets that violate normal network protocols.

1. Introduction

Fuzzy association rules, (Xie, 2005; Gao et al., 2004; Kuok et al., 1997), have received increasing attention in the recent years, and thus have been widely applied to many applications in different fields (Kaya and Alhajj, 2003a,b; El–Semary et al., 2006; Au and Chan, 2003; Pei-Qiliu et al., 2003). Several researchers have adopted fuzzy association rules to design their anomaly-based Network Intrusion Detection Systems (NIDSs) (El–Semary et al., 2006; Bridges and Vaughn, 2000; Florez et al., 2002; Dickerson and Dickerson, 2000; Hossain et al., 2003). They emphasized on the effectiveness of applying fuzzy association rules to design NIDSs (high detection rate, low false alarm rate, etc.), while ignored time efficiency in the NIDSs. In those researches, static mining approaches were adopted to mine out fuzzy association rules from large amounts of records about network traffic to make a decision. It indicates that their NIDSs could only be applied to analyze offline network traffic. Compared with traditional static mining approaches, (El–Semary et al., 2006; Bridges and Vaughn, 2000; Florez et al., 2002; Dickerson and Dickerson, 2000; Hossain et al., 2003), the proposed system of this study can greatly improve efficiency from offline detection to online real-time detection, while retaining the same effectiveness. Similar to their experiments, (El–Semary et al., 2006; Bridges and Vaughn, 2000; Florez et al., 2002; Dickerson and Dickerson, 2000; Hossain et al., 2003), DoS attacks were applied to demonstrate the performance, including the effectiveness and efficiency, of the system.

Some researches have successfully applied neuro-fuzzy (Toosj and Kahani, 2007), genetic-fuzzy (Tsang et al., 2007), and fuzzy logic (Abadeh et al., 2007; Shanmugam and Idris,

* *Corresponding author.*
E-mail addresses: minysu@mail.mcu.edu.tw (M.-Y. Su), yugj@mail.au.edu.tw (G.-J. Yu).

0167-4048/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2008.12.001

2007) to design NIDSs. The main difference between this system and other researches is that, the proposed system makes decisions based on the results of a data-mining algorithm, instead of based on fuzzy inference rules. Thus, those researches (Toosj and Kahani, 2007; Tsang et al., 2007; Abadeh et al., 2007; Shanmugam and Idris, 2007) focused on how to obtain precise fuzzy logic rules by genetic algorithms for direct inference. (Tsang et al., 2007) applied a parallel technique to improve the speed of obtaining these rules. However, the proposed system is an application of a data-mining algorithm, which obtains fuzzy association rules very rapidly. By the deliberate settings of *mini_support* and *mini_confidence* in the mining algorithm, the obtained rules can potentially represent the network traffic as being normal or abnormal. The principle of the proposed system is that, if the current output of the mining algorithm, such as the fuzzy association rules, is significantly different from the rules mined from normal network traffic, then the NIDS may have discovered an attack. The proposed system monitors network traffic and produces one record every two seconds. Once the newest record is gathered, the proposed system uses the incremental mining approach to generate the most recent rules for further comparison to make a decision. That is, all mining processing and similarity computations are completed within two seconds.

While adopting fuzzy association rules to analyze network traffic, especially for a NIDS, the time expense, including online data collection and mining procedures, are vital. Incremental fuzzy-rule mining is suitable to meet real-time demands because it can produce the latest rule set, while a new data record is gathered online. This paper first proposes an online incremental mining algorithm for generating fuzzy association rules, and then presents a real-time intrusion detection system based on the algorithm.

The remainder of this paper is organized as follows. Section 2 introduces the background knowledge. Section 3 briefly describes the incremental mining algorithm. Section 4 presents the proposed real-time NIDS. Section 5 discusses the experimental results. Section 6 is the conclusion.

2. Association rules and fuzzy association rules

Let $I=\{i_1, i_2, i_3, \ldots, i_m\}$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items, such that $T \subseteq I$. An *association rule* is an implication of form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. Rule $X \Rightarrow Y$ in the transaction set D has *support s* if s% of transactions in D contains $X \cup Y$, and it has *confidence c* if c% of transactions in D that contains X also contains Y. That is, let Sup (X) denote the occurrence frequency of the itemset X in D, $s = \text{Sup}\ (X \cup Y)$ and $c = \text{Sup}\ (X \cup Y)/\text{Sup}\ (X)$. Agrawal and Srikant proposed the well known Apriori algorithm (Agrawal et al., 1993) 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 $s \geq mini\_sup$ and $c \geq mini\_conf$.

Since the Apriori algorithm was designed for mining databases with binary features (item and feature are used interchangeably throughout the paper), fuzzy association rules mining (Xie, 2005; Gao et al., 2004; Kuok et al., 1997) was one of those variations used to deal with quantitative features. Let $T=\{t_1, t_2, \ldots, t_n\}$ be the database, and each transaction t_i represent the ith tuple in T. Moreover, $I=\{i_1, i_2, \ldots, i_m\}$ is used to represent all features appearing in T. Each quantitative feature i_k, $1 \leq k \leq m$, is associated with some fuzzy variables, say $v_1, v_2, \ldots, v_j$. Every fuzzy variable is represented by a membership function. For easy representation in the following, this paper uses $F_{ik \cdot vj}$ to uniquely denote the jth membership function of feature i_k.

Suppose four features, *#packet*, *#SYN*, *#ACK*, and *#connection* are of concern in a NIDS, and each feature has three fuzzy variables, *low*, *medium*, and *high*; then, $4 \times 3 = 12$ membership functions are involved. Thus, $F_{\#packet \cdot low}$ denotes the low function of feature *#packet*, $F_{\#ACK \cdot high}$ denotes the high function of feature *#ACK*, etc. In fuzzy association rule mining, a fuzzy itemset consists of two parts: items and fuzzy variables, denoted as <X, V>, where $X=(x_1, x_2, \ldots, x_i), \subset I$ is a collection of items (or features), and $V=(v_1, v_2, \ldots, v_i)$ is the collection of corresponding fuzzy variables to X, in order. For instance, if X = (*#packet*, *#SYN*, *#ACK*, *#connection*) and V = (*low*, *medium*, *low*, *high*), the fuzzy item set <X, V> represents {*#packet* is *low*, *#SYN* is *medium*, *#ACK* is *low*, *#connection* is *high*}. The support of <X, V> is computed as (Kuok et al., 1997; Kaya and Alhajj, 2003a):

$$\text{Sup}(<X,V>) = \frac{\text{Sum of votes satisfying } <X,V>}{\text{number of records in } T}$$

$$= \frac{\sum_{t_i \in T} \prod_{x_j \in X} F_{x_j \cdot s_j}(t_i[x_j])}{|T|}.$$

Where, $t_i[x_j]$ denotes the value of feature x_j of the ith record, and $|T|$ represents the total number of records.

Instead of ''If X, then Y'' in association rule, a fuzzy association rule now has the form of ''If <X, V>, then <X′, V′>''. Here, <X, V> and <X′, V′> are two itemsets—$X \subset I$, $X' \subset I$ and $X \cap X' = \varnothing$. The first part <X, V> is called an antecedent, and the second part <X′, V′> is called a consequent. Similarly, support s and confidence c of the fuzzy association rule <X, V> ⇒ <X′, V′> are computed as:

$$s = \text{Sup}(<Y,U>) \text{ and}$$

$$c = \text{Sup}(<Y,U>)/\text{Sup}(<X,V>),$$

where <Y, U> is the concatenation itemset of <X, V> and <X′, V′>. For instance, if $X=(x_1, x_2, \ldots, x_i)$, $V=(v_1, v_2, \ldots, v_i)$, $X' = (x'_1, x'_2, \cdots, x'_j)$, and $V' = (v'_1, v'_2, \cdots, v'_j)$, then $Y = (x_1, x_2, \ldots, x_i, x'_1, x'_2, \cdots, x'_j)$ and $U = (v_1, v_2, \ldots, v_i, v'_1, v'_2, \cdots, v'_j)$.

3. Incremental mining algorithm for fuzzy association rules

In order to meet the real-time demands of a NIDS, an incremental mining algorithm is used to derive fuzzy association rules that act as the detection engine within the proposed NIDS. By the algorithm, each current support value of an itemset is briefly retained in the memory. As the next data record is being gathered, the algorithm uses the current support value to compute the next one, and then replaces the current support value with the following value. Suppose the quantitative values of four proposed features—*#packet*, *#SYN*, *#ACK*, and *#connection*—are measured for each time unit, and in sequence are $t_1 = (120, 16, 51, 29)$, $t_2 = (23, 17, 5, 19)$, $t_3 = (189, 76, 41, 67)$, …, $t_i = (72, 34, 9, 36)$, …., For the itemset $<X, V> = \{\#packet \text{ is } low, \#SYN \text{ is } medium, \#connection \text{ is } low\}$, its support value *s* at $t_1, t_2, t_3, \ldots, t_i, \ldots$, is computed individually as:

$$s \text{ at } t_1 = (F_{\#packet.low}(120) \times F_{\#SYN.medium}(16) \times F_{\#connection.low}(29))/1 \rightarrow tmp;$$

$$s \text{ at } t_2 = (tmp \times 1 + (F_{\#packet.low}(23) \times F_{\#SYN.medium}(17) \times F_{\#connection.low}(19)))/2 \rightarrow tmp;$$

$$s \text{ at } t_3 = (tmp \times 2 + (F_{\#packet.low}(189) \times F_{\#SYN.medium}(76) \times F_{\#connection.low}(67)))/3 \rightarrow tmp;$$

…

$$s \text{ at } t_i = (tmp \times (i-1) + (F_{\#packet.low}(72) \times F_{\#SYN.medium}(34) \times F_{\#connection.low}(36)))/i \rightarrow tmp;$$

…

Since the contribution of previous records to the current support of an itemset is ephemerally saved in a variable, i.e., *tmp*, the cost of mining time would not be prolonged as observed records are increased. The time–cost comparison between static mining and incremental mining is illustrated in Fig. 1.

An alphabet character is used to denote a feature's fuzzy variable in the proposed algorithm. Thus, if there are n features, and each has m fuzzy variables, then $n \times m$ consecutive characters will be used. These characters can be regarded as placed in row-major order on an $n \times m$ matrix. For the example of $n = 6$ and $m = 3$, starting with character 'A', the matrix is shown in Fig. 2.

For each character, there are three notations, i.e., ASCII value, position index, and fuzzy variable, and transformations among these notations are very simple. For example, the character 'G' has ASCII value 71, its index is (2, 0), i.e., $[(71-65)/3] = 2$ and $(71-65) \bmod 3 = 0$, and thus it represents the third feature's first fuzzy variable, i.e., $F_{f3.low}$. With such arrangements, each fuzzy itemset can be expressed by

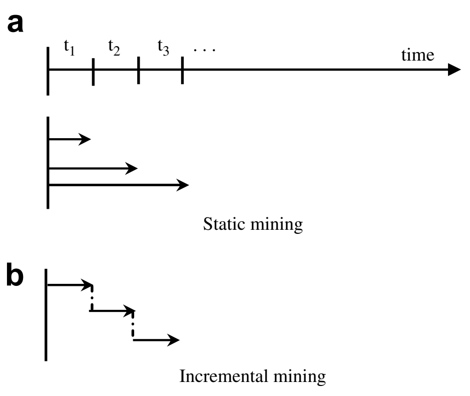

Fig. 1 – Time costs for static mining and incremental mining. (a) Static mining. (b) Incremental mining.

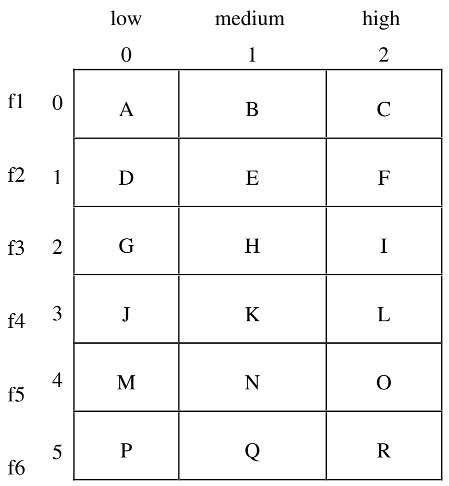

Fig. 2 – Representations of characters: ASCII values, position indexes, and fuzzy variables.

a string. For instance, string "EJR" denotes the fuzzy itemset {f2 is medium, f4 is low, f6 is high}. Since each itemset has to be associated with a variable in order to ephemerally retain its current support value, a node structure is declared as follows. In the algorithm, each fuzzy itemset is represented by a node.

```
struct itemset {
    float SupVal;
    char STRING[maxlen];
    struct itemset *next;
};
```

The variable *SupVal* is used to retain the current support value, and *maxlen* is defined as the longest length of the itemset, which theoretically is less than or equal to the number of features. The proposed algorithm consists of three procedures: *Initialization*, *SupCal*, and *RuleGen*. All nodes are constructed and linked by the *Initialization*. The support value of each node is updated by the *SupCal*; meanwhile, for any node (i.e., fuzzy itemset) with a support value greater than or equal to the *mini_sup*, then the *RuleGen* is called to test all possible rules, print out those rules with confidences greater than or equal to the *mini_conf*. The procedure, *Initialization,* is run only once at the beginning, while the procedures *SupCal* and *RuleGen* are run once for every occasion a new record is gathered.

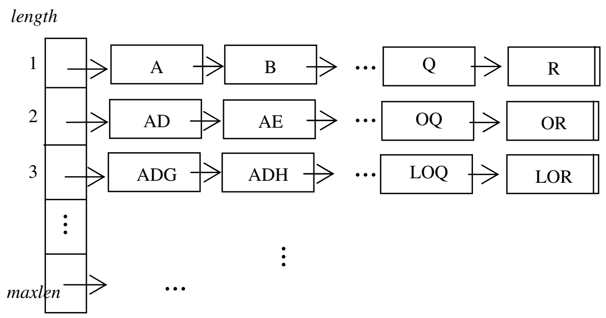

Fig. 3 – All fuzzy itemsets with lengths ≤ *maxlen*.

The matrix is shown in Fig. 2. All fuzzy itemsets will be created and linked in the main memory, as shown in Fig. 3, by the **Initialization** procedure.

In the following **Initialization** procedure, $maxlen \le n$ denotes the longest fuzzy itemset length of the algorithm concerned. Thus, if *maxlen* is set to 5, for instance, then all itemsets with lengths greater than 5 will be ignored.

```
Procedure Initialization
//Input: features n, degrees m, and the longest itemset length: maxlen ≤ n
//Output: maxlen linked lists in memory; for each list, nodes have the same length and appear in lexicographic order.
//Assumption: Matrix[ ][ ] and length are two global variables.
Matrix[i][j] ← 0, for all i and j; //zero matrix
for (length = 1; length ≤ maxlen; length++) {CreateLinkedList(length, 0);
}
```

```
SubProcedure CreateLinkedList (int len, int start)
if len = 1, then {
 for (i = start; i < n; i++) {
  for ( j = 0; j < m; j++) {
    Matrix[i][j] ← 1;
    Scan Matrix from top to bottom, and left to right, to discover all positions set to 1;
    Dynamically allocate a node using its STRING field to store the corresponding fuzzy itemset;
    Add the node to the tail of the list with length length;
    Matrix[i][j] ← 0;
  }
 }
}//end if
else {//len > 1
 for (i = start; i ≤ n − len; i++) {
  for ( j = 0; j < m; j++) {
    Matrix[i][j] ← 1;
    CreateLinkList(len − 1, i + 1);
    Matrix[i][j] ← 0;
  }
 }
}//end else
```

During the recursive execution, each possible itemset is replaced by a node, and the *STRING* field of each node stores characters, which have corresponding positions in the *Matrix*, marked '1'. Two feasible itemsets, and their corresponding matrix representations are shown in Fig. 4. This paper only describes the *Initialization* procedure. The other two procedures, i.e., *SupCal* and *RuleGen* can be found in (Su et al., 2008a). If *maxlen* is set to 6, *m* (the number degrees) is 3, and *n* (the number of features) is less than 21, then incremental mining can be completed within seconds or milliseconds. Detailed time and memory consumptions can also be found in (Su et al., 2008a).

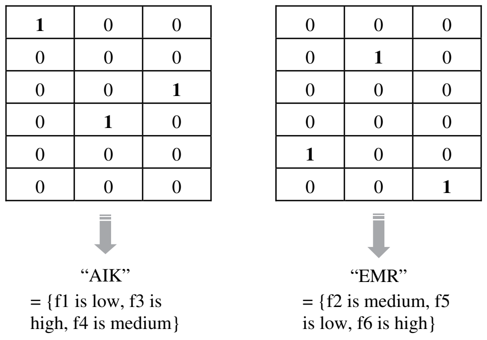
1 0 0
0 0 0
0 0 1
0 1 0
0 0 0
0 0 0
0 0 0
0 1 0
0 0 0
0 0 0
1 0 0
0 0 1
"AIK"
= {f1 is low, f3 is high, f4 is medium}
"EMR"
= {f2 is medium, f5 is low, f6 is high}

Fig. 4 – Two feasible itemsets.

4. The proposed real-time NIDS design

Based on the above incremental mining algorithm, the proposed NIDS architecture is illustrated in Fig 5. In the training stage, network traffic information with attack-free data records were collected and stored in Computer C, with one record for every two seconds. For online testing, Computer A collected network traffic information online, at the rate of one record per two seconds, and consistently sent each record to Computer B. Computer B applied the incremental mining algorithm to generate the newest fuzzy-rule set, every two seconds. At the same time, Computer C performed incremental mining every 2 seconds on the attack-free data records, i.e., adding one record every 2 seconds. The two newest rule sets that come from Computer C and Computer B were consistently sent to Computer D for comparison. If their similarity was below the threshold, an anomaly of network

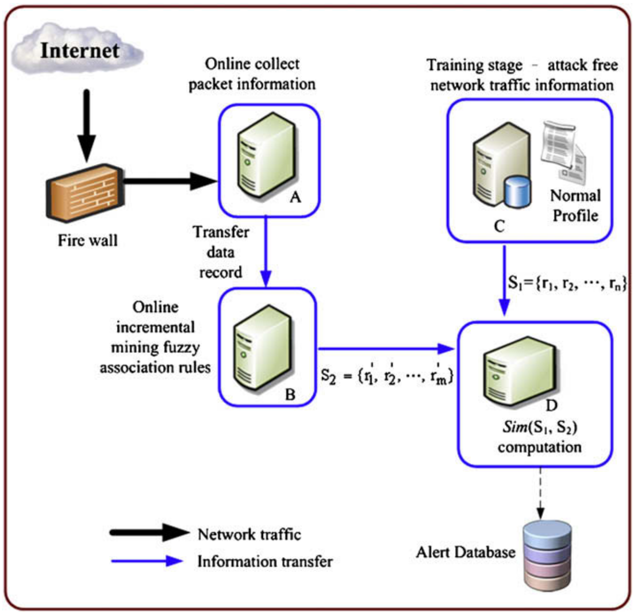
Internet
Online collect packet information
Training stage - attack free network traffic information
A
C
Normal Profile
Fire wall
Transfer data record
$S_1=\{r_1, r_2, \cdots, r_n\}$
Online incremental mining fuzzy association rules
B
$S_2 = \{r_1', r_2', \cdots, r_m'\}$
D
$Sim(S_1, S_2)$ computation
Network traffic
Information transfer
Alert Database

Fig. 5 – The architecture of the proposed NIDS.

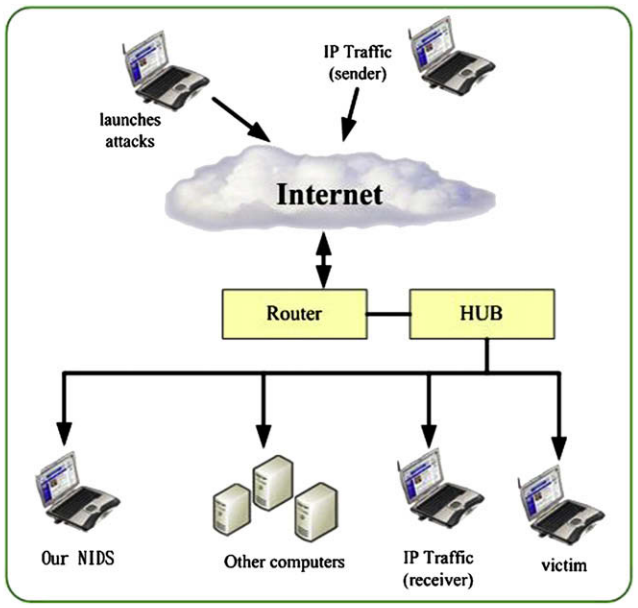

Fig. 6 – The network topology for simulation.

traffic was found. In this paper, the similarity between the two rule sets is defined as follows.

Let S1 and S2 be two rule sets. The similarity between them is computed as:

$$sim(S_1, S_2) = \frac{SCORE\ 1}{|S_1|} \times \frac{SCORE\ 2}{|S_2|},$$

where $|S_1|$, $|S_2|$ represent the number of rules in the sets,

$$SCORE\ 1 = \sum_{\forall r \in S_1} score(r, S_2), \text{ and}$$

$$SCORE\ 2 = \sum_{\forall r \in S_2} score(r, S_1).$$

Here, for a single rule r, with support s, confidence c, and a rule set S, *score*(r, S) is defined as:

If there exists a rule $r_1 = r$ in S with support s_1 and confidence c_1, then

$$score(r, S) \leftarrow 1 - \max\left(\frac{|c - c_1|}{\max(c, c_1)}, \frac{|s - s_1|}{\max(s, s_1)}\right),$$

else $score(r, S) \leftarrow -(c \times s)$.

Two rules, r and r_1, are regarded as equal if they have the same antecedents and consequents. Finally, let SCORE1 or SCORE2 be 0, if it is a negative value.

During the incremental mining, the latest data record is more important o be more than any of the historical data records. Thus, for any itemset in this system, its support is computed by:

$$Currentsuport = supportduetothelatestrecord * 0.2 + supportduetothehistoricalrecords * 0.8.$$

That is, regardless of the pieces records already passed, their total contribution to the current support computation is set to four times that of the latest single record. For example, suppose contiguous n records in time t_1, t_2, ..., t_{n-1}, t_n have their contributions to a specific itemset's support as s_1, s_2, ..., s_{n-1}, s_n, respectively. The current support of the itemset in time t_n is computed as:

$$Currentsupport = 0.2 * s_n + 0.8 * (0.2 * s_{n-1} + 0.8 * (0.2 * s_{n-2} + 0.8 * (\ldots(0.2 * s_3 + 0.8 * (0.2 * s_2 + 0.8 * s_1))\ldots)))$$

Note that the larger ratio of the latest record causes the system to become more susceptible. In the extreme case of the latest record, with a ratio of 1 (the historical records with ratio 0), the system makes decisions absolutely dependent on the current network traffic, and thus, it may generate false positives because network traffic is varied in general. On the other hand, if the ratio of the latest record is close to zero, then the system's reaction to attack will be blunted, unless the flooding attack lasts a long time. According to experiments, the latest record of that ratio, to historical records, is 1 to 4, thus, the system is stable and all experimented DoS attacks can be detected.

Table 1 – Feature list.

No	protocol	Feature
1	TCP	S.IP + SYN count
2	TCP	S.IP + URG_Flag + URG_data count
3	TCP	S.IP + ACK_Flag + ACK count
4	ARP	S.IP + ARP count
5	IP	D.IP slots hit
6	IP	Header length! = 20 count
7	IP	MF_Flag count
8	IP	(Total length > 1400 || < 40) &&TTL = 64 count
9	IP	checksum_error count
10	TCP	ACK_Flag + ACK count
11	TCP	checksum_error count
12	UDP	Same_length_interval count
13	ICMP	Type error count
14	ICMP	checksum_error count
15	IGMP	checksum_error count
16	IGMP	Length > 1000 count

5. Performance analyses

The network topology for the experiment is shown in Fig. 6. A commercial application, named IP Traffic, was applied to produce background traffic, which can generate any amount of TCP/UDP/ICMP/ARP/IGMP packets. Two hosts, running IP Traffic, played sender and receiver, respectively; we deployed the receiver in the LAN, with the sender transmitting packets through the Internet. Using IP Traffic, we can choose protocols and set the contents of packets generated by mathematical laws (Pareto, Uniform, and Exponential) or derived from files, or just generated by a packet generator with configurable contents. Inter-packet delay and packet size can also be selected. During the experiments, the network traffic amount was kept in the range of 0 to 80 Mbps. One laptop launched a DoS attacks against the victim through the Internet. No firewall was applied, and all packets were allowed to pass through the router. The proposed system was coded by

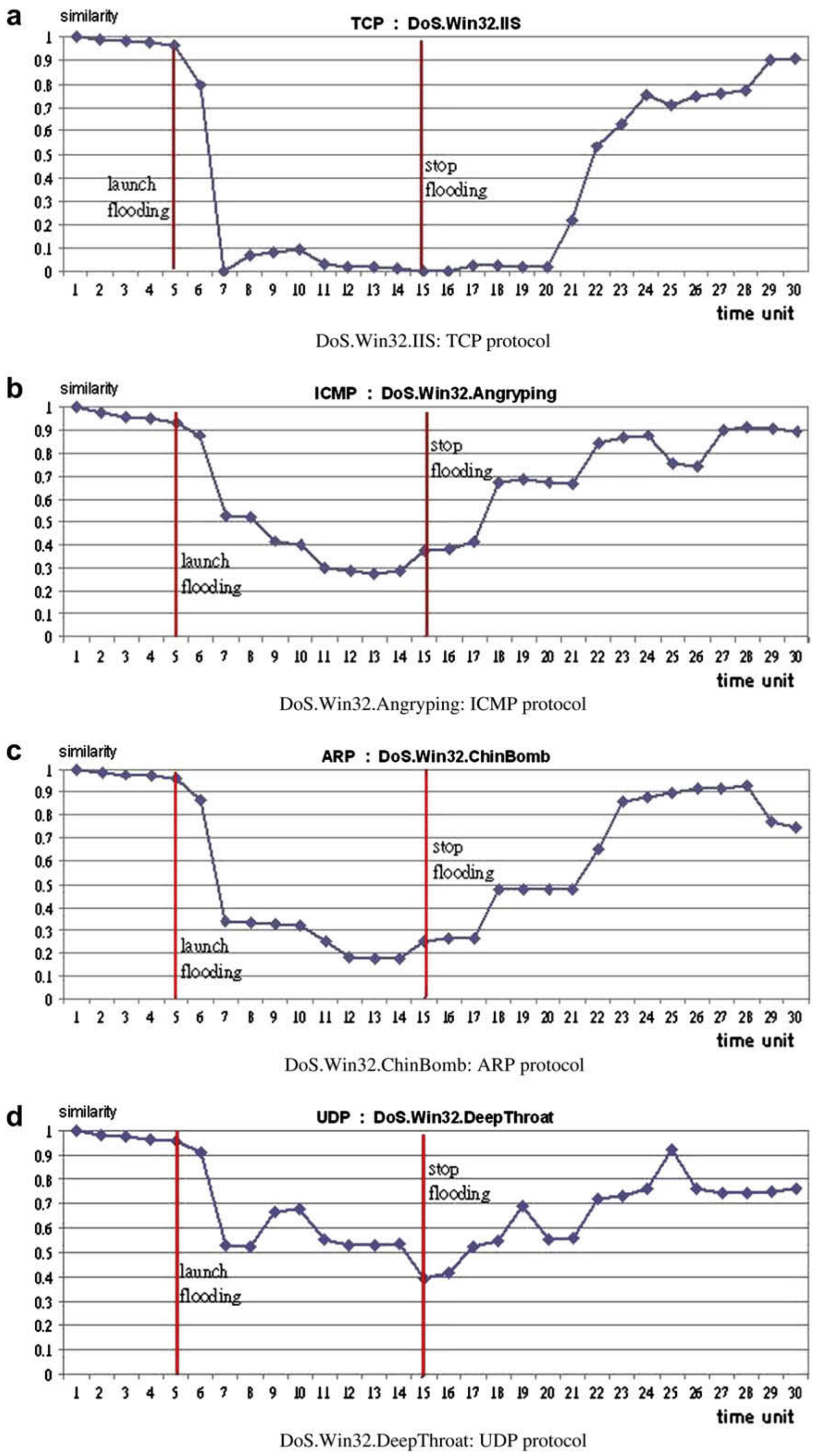

Fig. 7 – Similarity degradation during flooding. (a) DoS.Win32.IIS: TCP protocol. (b) DoS.Win32.Angryping: ICMP protocol. (c) DoS.Win32.ChinBomb: ARP protocol. (d) DoS.Win32.DeepThroat: UDP protocol.

Microsoft Visual C++, and operated on a laptop with a Windows XP operating system.

For all the experiments, 16 features were adopted, as listed in Table 1, to collect network traffic information and generate one data record every two seconds. These features were reported as important in detecting DoS attacks (Su et al., 2008b). Each had three degrees: low, medium, and high. Only rules derived from large itemsets, with lengths of

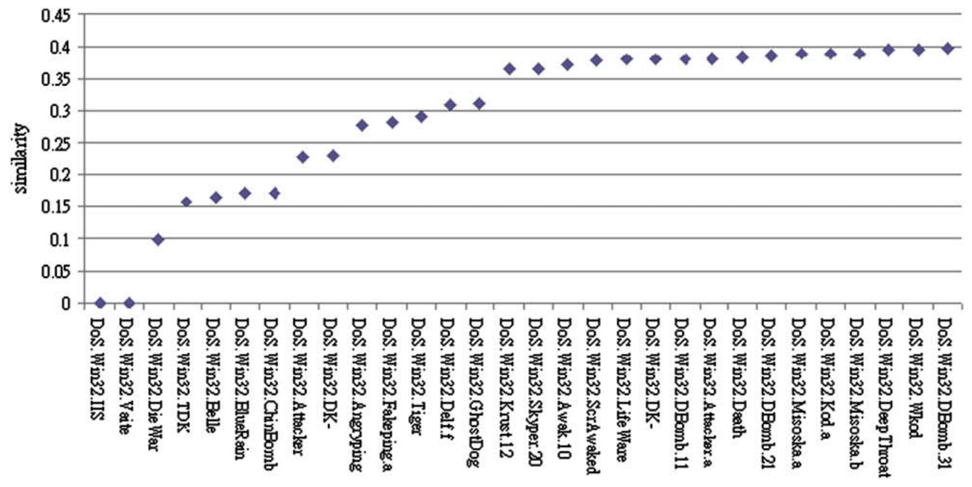

Fig. 8 – The smallest similar values for different DoS attacks.

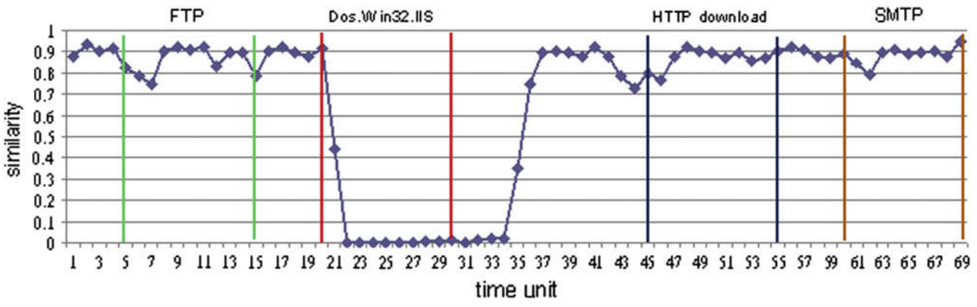

Fig. 9 – No false positives occurred for FTP, HTTP, or SMTP downloads.

2, were considered, i.e., *maxlen* in the **Initialization** procedure is set to 2. Thus, the number of rules could be reduced to 200 ~ 250.

Fig. 7 shows the similarity between the two rule sets, namely one mined from offline attack-free traffic records, and the other mined from online traffic, in degradation during attack occurrence. Four attacks belonging to different protocols were applied and their similarity degradations are illustrated in Fig. 7. The DoS.Win32.IIS attack was launched the at the fifth time unit. After 2 time units, the similarity was down to 0 at the seventh time unit. Lasting for 10 time units (20 seconds) of flooding, the attack was ended at the 15th time unit. The similarity was apparently after the 20th time unit. Historical data records were taken into consideration, and thus, similarity could not immediately return to normal levels. The smallest similarity value is zero, as shown in Fig. 7(a), and 0.2778, 0.1721, and 0.3938 for Fig. 7(b), (c), and (d), respectively.

A total of 30 DoS attacks were tested, and the smallest similarity value for each case is shown in Fig. 8, all are below 0.4. On the other hand, Fig. 9 shows that the proposed system would not generate false positives for FTP, HTTP, or SMTP downloads if the set threshold was 0.5.

Fig. 10 shows the stability of the proposed system. As seen, the similarity between normal online traffic and attack-free traffic (stored in the database) were produced by the application of IP Traffic, by random TCP/IP/UDP/ICMP connections and random total amount of packets. Fig. 10 shows that the

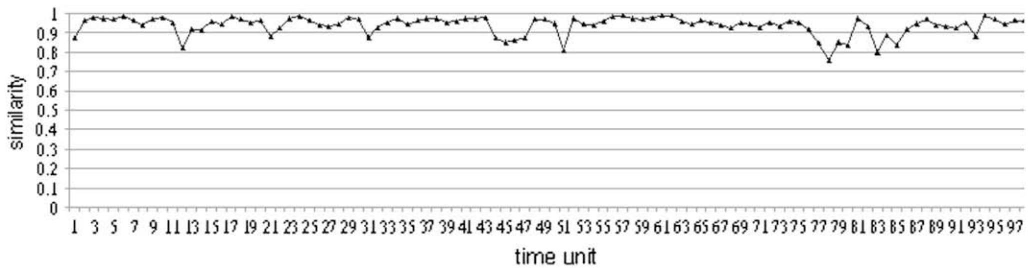

Fig. 10 – Similarities between normal online traffic and attack-free traffic stored in database.

proposed system may not cause false alarms because the similarity is always above 0.75, under the assumption that normal programs would not generate too many malformatted packets or packets that violate normal network protocols. The smallest value in Fig. 10 is 0.759872, and the standard deviation of all similar values is 0.047201.

6. Conclusions

Previous researches (El–Semary et al., 2006; Bridges and Vaughn, 2000; Florez et al., 2002; Dickerson and Dickerson, 2000; Hossain et al., 2003) have shown that fuzzy association rules can be effectively applied to design NIDSs. However, none had achieved detection in real time because they collected the records of network traffic information first, and then analyzed these records by static mining. Based on the proposed incremental mining algorithm, this study designed a real-time intrusion detection system for large-scale attacks, and similar to (El–Semary et al., 2006; Bridges and Vaughn, 2000; Florez et al., 2002; Dickerson and Dickerson, 2000; Hossain et al., 2003), tested the system by DoS attacks. The results showed that this system exhibited excellent performance under DoS attacks. The main contribution of this study was to realize a real-time anomaly-based NIDS by an incremental mining approach that is able to make a decision every two seconds. The proposed system is a non-adaptive NIDS, unless parameters in the system or features in Table 1 are changed by human input.

Acknowledgments

This work was partially supported by the National Science Council under contracts NSC 95-2221-E-130-003 and 96-2221-E-130-009.

References

Abadeh Mohammad S, Habibi Jafar, Barzegar Zeynab, Sergi Muna. A parallel genetic local search algorithm for intrusion detection in computer networks. Engineering Applications of Artificial Intelligence 2007;20:1058–69.

Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. SIGMOD 1993.

Au Wai-Ho, Chan Keith CC. Mining fuzzy association rules in a bank-account database. IEEE Transactions on Fuzzy Systems 2003;11(2):238–48.

Bridges Susan M, Vaughn Rayford B. Intrusion detection via fuzzy data mining. In: The Proceedings of the Canadian Information Technology Security Symposium; 2000.

Dickerson JE, Dickerson JA. Fuzzy network profiling for intrusion detection. Fuzzy Information Processing Society; 2000.

Aly El–Semary, Janica Edmonds, Jes′us Gonz′alez-Pino, Mauricio Papa. Applying data mining of fuzzy association rules to network intrusion detection. In: The Proceedings of the IEEE Workshop on Information Assurance United States Military Academy; 2006.

German Florez, Bridges Susan M, Vaughn Rayford B. An improved algorithm for fuzzy data mining for intrusion detection. In: The Proceedings of the IEEE Fuzzy Information; 2002.

Ya Gao, Jun Ma, Lin Ma. A new algorithm for mining fuzzy association rules. In: The Proceedings of the Conference on Machine Laming and Cybemetics; 2004.

Hossain Mahmood, Bridges Susan M, Vaughn Jr. Rayford B. Adaptive intrusion detection with data mining. In: The Proceedings of the IEEE Conference on Systems, Man and Cybernetics; 2003.

IP Traffic, http://www.omnicor.com/netest.htm.

Mehmet Kaya, Reda Alhajj. A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. In: The Proceedings of the IEEE Conference on Fuzzy Systems; 2003a.

Kaya M, Alhajj R. Facilitating fuzzy association rules mining by using multi-objective genetic algorithms for automated clustering. In: The Proceedings of the IEEE Conference on Data Mining; 2003b.

Kuok C, Fu A, Wong M. Mining fuzzy association rules in databases. In: The Proceedings of the ACM Conference on Information and Knowledge Management; 1997.

Pei-Qiliu, Zeng-Zhi Li, Yin-Liang Zhao. Algorithm of mining fuzzy association rules in network management. In: The Proceedings of the IEEE Conference on Machine Learning and Cybernetics; 2003.

Shanmugam Bharanidharan, Idris Norbik Bashah. Improved hybrid intelligent intrusion detection system using ai technique. Neural Network World 2007;17(4):351–62.

Su Ming-Yang, Yeh Sheng-Cheng, Chang Kai-Chi, Wei Hua-Fu. Using incremental mining to the fuzzy rules for real-time network intrusion detect intrusion detection systems. In: The Proceedings of the IEEE Conference on Advanced Information Networking and Applications – Workshops, March 25–28, 2008a. p. 50–5.

Su Ming-Yang, Chang Kai-Chi, Wei Hua-Fu, Lin Chun-Yuen. Feature weighting and selection for a real-time network intrusion detection system based on GA with KNN. Lecture Note on Computer Science (LNCS) 2008b. To appear.

Toosj Adel N, Kahani Mohsen. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Computer Communications 2007;30:2201–12.

Tsang Chi-Ho, Kwong Sam, Wang Hanli. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognition 2007;40:2373–91.

Xie Dong (Walter). Fuzzy association rules discovered on effective reduced database algorithm. In: The Proceedings of the IEEE Conference on Fuzzy Systems; 2005.

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 Campus, Taiwan. His research interests include network security, intrusion detection/prevention, malicious code detection, sensor networks, and wireless Ad hoc network security.

Gwo-Jong Yu received a B.S. degree in Computer Science from the Chung-Yuan Christian University, Chung-Li, Taiwan in 1989, and the Ph.D. degree in Computer Science from the National Central University, Chung-Li, Taiwan in 2001. His Ph.D. thesis is on digital watermarking for copyright protection and image authentication. Since August 2001, he has been an assistant professor in the department of Computer and

Information Science, Aletheia University, Taiwan. He is a member of IEEE. In August 2006, he became an associate professor in the department of Computer Science and Information Engineering, Aletheia University, Taiwan. His research interests include Bluetooth, Ad hoc networking, wireless sensor networks, and network security.

Chun-Yuen Lin received his B.S. degree from the Department of Computer Science and Information Engineering of Ming Chuan University, Taiwan in 2008. He is now pursuing his M.S. degree at the Ming Chuan University. His current research areas include network security, intrusion detection, and wireless security.