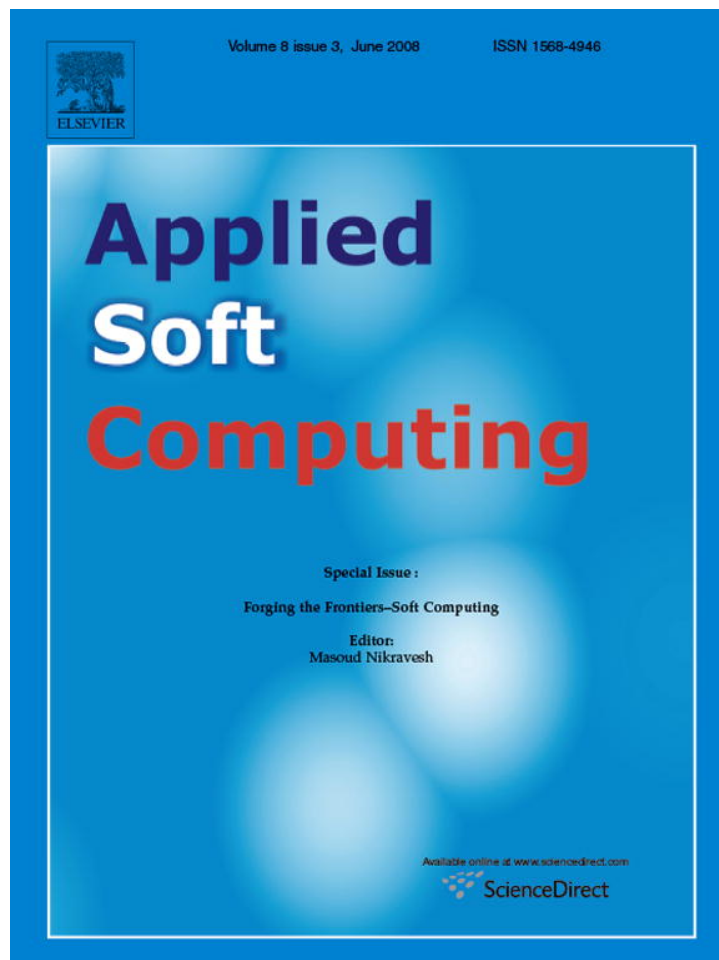




This article was published in an Elsevier journal. The attached copy is furnished to the author for non-commercial research and education use, including for instruction at the author's institution, sharing with colleagues and providing to institution administration.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Compiling network traffic into rules using soft computing methods for the detection of flooding attacks

Sanguk Noh ^{a,*}, Gihyun Jung ^b, Kyunghye Choi ^c, Cheolho Lee ^d

^a School of Computer Science and Information Engineering, The Catholic University of Korea, Bucheon, Republic of Korea

^b Division of Electronics Engineering, Ajou University, Suwon, Republic of Korea

^c Graduate School of Information and Communication, Ajou University, Suwon, Republic of Korea

^d National Security Research Institute, Daejeon, Republic of Korea

Received 21 February 2007; accepted 23 February 2007

Available online 16 October 2007

Abstract

The ability to dynamically collect and analyze network traffic and to accurately report the current network status is critical in the face of large-scale intrusions, and enables networks to continually function despite of traffic fluctuations. The paper presents a network traffic model that represents a specific network pattern and a methodology that compiles the network traffic into a set of rules using soft computing methods. This methodology based upon the network traffic model can be used to detect large-scale flooding attacks, for example, a distributed denial-of-service (DDoS) attack. We report experimental results that demonstrate the distinctive and predictive patterns of flooding attacks in simulated network settings, and show the potential of soft computing methods for the successful detection of large-scale flooding attacks.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Network traffic modeling; Soft computing; Compiled rules; Intrusion detection; Flooding attacks

1. Introduction

As the complexity of Internet is scaled up, it is more likely for the Internet resources to be exposed to various flooding attacks like distributed denial-of-service (DDoS) attacks. It was reported that the DDoS attacks against Web servers such as Yahoo, e-Bay, and E-Trade were extremely costly in terms of economical expense, vendor image loss, credibility, and system stability [5].

In this paper, we focus on characterizing network traffic of Web server, which is one of the most popular services of the Internet and thus has become one of the most important Internet resources being protected. To understand the features of flooding attacks to Web server, the paper introduces an analysis mechanism of network traffic. The analysis mechanism uses *transmission control protocol (TCP) flag rates*, which are expressed in terms of the ratios of the number of packets with specific TCP flags to the total number of TCP packets on

network to be monitored. All six TCP flags (SYN, FIN, RST, ACK, PSH, and URG) of packets appeared on network to be monitored are rated and compared with those on normal network. Since incoming and outgoing network traffic presents completely different characteristics, the flag rates are measured in both incoming and outgoing traffic.

An example to show how the TCP flag rates effectively characterize DDoS attacks can be found in SYN flooding attack. During the attack, the number of SYNs drastically increases and thus SYN flag rate also becomes much larger than the normal one. What it means is that the increase in SYN flag rate indicates the possibility of a DDoS attack. In addition to the flag rates, we also measure *protocol rates*, which are the ratios of the number of packets belonging to specific protocols to the total number of packets on network. The transmission control protocol (TCP), user datagram protocol (UDP), and Internet control message protocol (ICMP) rates for both incoming and outgoing traffic are compared with the normal ones. Under a specific type of the flooding attack, some of protocol rates and flag rates significantly change, compared to normal (without the attack) ones.

We also propose a flooding attack detection mechanism using soft computing methods [12,31], i.e., inductive learning

* Corresponding author. Tel.: +82 2 2164 4579; fax: +82 2 2164 4777.

E-mail addresses: sunoh@catholic.ac.kr (S. Noh), khchung@ajou.ac.kr (G. Jung), khchoi@ajou.ac.kr (K. Choi), chlee@etri.re.kr (C. Lee).

algorithms [2,26] and the probabilistic reasoning of naive Bayesian classifier [7]. To identify the flooding attacks, we endow an alarming agent with a tapestry of reactive rules [23,24]. The reactive rules are constructed by compiling the results of TCP flag rates or protocol rates, and presence (or absence) of flooding attacks into state–action rules. The compilation process exploits the regularities of flooding attacks, if any, and enables our alarming agents to detect them. The compiled rules can be obtained from soft computing algorithms which use the results of TCP flag rates and protocol rates performed offline as their inputs. Further, it is desirable that each of the compilation methods be assigned a measure of performance that compares it to the benchmark. The various compilations available constitute a spectrum of approaches to making detections under various attacks on Web sites.

In the following section, we will discuss related work and, in Section 3, address the question of how to analyze network traffic characteristics and then define two network traffic rates. Section 4 describes a simulated network environment and shows clear factors that indicate the symptoms of various flooding attacks. Section 5 proposes a flooding attack detection mechanism based upon the network traffic model, validates our framework empirically, and presents the experimental results. In conclusion, we summarize our results and discuss further research issues.

2. Related work

Since flooding attacks could be made on any host over the Internet [9,19,21], a number of approaches preventing the attacks or minimizing damages on the hosts have been rigorously proposed in the fields of Internet security. Most research [6,10,14,29] has focused on analyzing the pattern of the flooding attacks. One of them [14] uses the randomness and distribution of source IP addresses. In [6,14], if the randomness of source IP addresses is getting higher than usual, it indicates that a DDoS attack occurs. Gil and Poletto [6] examined IP packet flows of ingress and egress directions, based upon their own data-structure, MULTOPS. Their network monitoring device using the MULTOPS detects the flooding attacks by “the difference between packet rates going out of a victim and coming from the attacker.” Their assumption for the detection is based on the disproportional difference between the packet rates, which is introduced by the randomness of ‘malicious’ packets. Kulkarni et al. [14] traced the source IP addresses and constructed Kolmogorov Complexity Metrics for identifying the randomness of source IP addresses. The Kolmogorov Complexity Metrics [17] changes depending on the degree of randomness of spoofed source IP addresses. However, these approaches are not effective when attackers reduce the level of randomness of source IP addresses or when they use actual IP addresses instead of spoofed ones.

Wang et al. [29] proposed another approach to a DDoS attack detection mechanism, which examined the protocol behavior of TCP SYN–FIN (RST) pairs. If there are no DDoS attacks against a TCP-based server, the rate of SYNs for TCP connection establishment is similar to the rate of FINs for TCP

connection termination. On the other hand, the rate of SYNs in the SYN flooding attack clearly differs from that of FINs. Thus, the metrics of SYN–FIN (RST) pairs could be useful to detect the SYN flooding attack against TCP-based servers. This approach is somewhat similar to our approach in the sense that both of them take into account TCP flags to detect flooding attacks. While their method can be applicable only to SYN flooding attacks, however, the approach proposed in this paper is more general and sophisticated so that the proposed approach can be applicable to detecting various flooding attacks, i.e., including SYN flooding attacks, UDP, ICMP, mixed flooding attacks, and so on.

In [1,22,28], they proposed preventive ways to restrict flooding attacks through disconnecting systems involved in the attacks from network or searching-and-removing the programs being utilized for the attacks. The Class Based Queuing (CBQ) technique proposed in [11] divides an output queue into several sub-queues and assigns a low bandwidth to a queue that handles suspicious traffic, meanwhile safe traffic flows in queues with larger bandwidths. Restricting the bandwidth reduces unsafe traffic passed to system (intended to protect) and enables the system to efficiently fight against attack. The on–off feedback control strategy proposed in [18,30] suggests a coordinated defense scheme against DDoS attacks, based on backward-propagation. When a host finds itself becoming a hot spot, it asks the neighboring nodes and routers to reduce influx of attacking packets. The technique may be useful to restrict incoming packets with suspicious symptom but the authors did not suggest any possible way to differentiate the packets utilized for flooding attacks and other safe packets. Intuitively, it is not an easy task at all to distinguish safe packets from unsafe ones.

To distinguish malicious packets from safe packets, we extract features from various network traffic in the presence/absence of a flooding attack using soft computing methods. Applying soft computing methods for rule extraction has been used to detect specific phenomenon in many domains [4,13,20] but, to our best knowledge, it might be first attempt for the detection of flooding attack. In our framework, the soft computing methods compile a pair of network traffic and flooding attack into a set of reactive rules. As we mentioned above, most approaches model the collateral effects of flooding attacks while our approach focuses on the flooding attack itself. In other words, since our framework models the pattern of network traffic on the hosts attacked, we could apply our framework to general flooding attack situations, regardless of the various behavior patterns of unknown flooding attacks.

3. Traffic rate analysis

We rely on the dynamics of network traffic rates, i.e., TCP flag rates and protocol rates, to analyze the characteristics of Web traffic under flooding attacks [16]. That is, we believe that the number of TCP packets with specific flags drastically changes or the number of IP packets belonging to a specific protocol (TCP, UDP and ICMP) sharply increases or decreases, when the flooding attacks occur. This is because the flooding

attacks use a huge amount of packets belonging to a specific protocol with particular flags. For example, when a SYN flooding attack is made, the number of TCP packets with a SYN flag for inbound traffic tremendously increases. In consequence, the TCP protocol rate and the SYN flag rate of inbound traffic go up to nearly one in a very short period.

For a way to efficiently analyze the features of flooding attack traffic, we use a network traffic analysis technique, traffic rate analysis (TRA). The TRA uses the traffic flowing into a victim (a Web server) as the input, and the traffic flowing out of the victim as the output. From its input and output, the TRA extracts two characteristic factors: TCP flag rates and protocol rates. The two rates expressed as the ratios of specific packets to the total number of packets in network are used to characterize the symptoms of the flooding attack and become the training data of the soft computing methods for the detection of the flooding attack (we give a detail of detection mechanism in Section 5). Since the amount of packets a host can handle varies with its hardware and software, we believe that it is reasonable to rely on their rates instead of the absolute amount of packets. That is, we eliminate the ambiguity introduced by counting the number of specific protocol packets with particular flags through normalizing the numbers with the total number of packets flowing on the network.¹

In the TRA, all packets flowing into and out of a Web server are captured and classified. The first classification process determines whether or not a packet belongs to TCP, UDP or ICMP, referencing the protocol field in its header. In case of TCP, further, the second classification procedure is applied and, the packet is separated into TCP header and payload. From the TCP header, which contains SYN, FIN, RST, ACK, PSH, URG flags and other fields, the flags are tested to determine whether or not they are set. If any flag of six TCP flags turns on, the flag is counted. The classification and counting procedure is performed for all captured packets. The numbers of TCP, UDP and ICMP packets are also counted along with the total number of packets in network during a specific pre-determined observation period t_d (s). Then two rates, TCP flag rates and protocol rates, are computed as defined in Eq. (1).

$$\begin{aligned} R_{t_d}[K_i] &= \frac{\text{total number of a flag } (K) \text{ in a TCP header}}{\text{total number of TCP packets}} \quad (\text{inbound}) \\ R_{t_d}[K_o] &= \frac{\text{total number of a flag } (K) \text{ in a TCP header}}{\text{total number of TCP packets}} \quad (\text{outbound}) \end{aligned} \quad (1)$$

Here, t_d stands for observation period. K stands for one of six flags: SYN, FIN, RST, ACK, PSH, and URG. For simplicity, these flags are denoted as S , F , R , A , P , and U for either inbound (i) or outbound traffic (o), respectively. For example, $R_2[A_i]$ represents the ratio of ACK packets to the total number of TCP packets captured for 2 s as an observation period.

The protocol rates are also defined similarly by the ratio of the numbers of TCP, UDP, or ICMP packets to the total number

of IP packets as follows:

$$\begin{aligned} R_{t_d}[[\text{TCP}|\text{UDP}|\text{ICMP}]_i] &= \frac{\text{total number of } [\text{TCP}|\text{UDP}|\text{ICMP}] \text{ packets}}{\text{total number of IP packets}} \quad (\text{inbound}) \\ R_{t_d}[[\text{TCP}|\text{UDP}|\text{ICMP}]_o] &= \frac{\text{total number of } [\text{TCP}|\text{UDP}|\text{ICMP}] \text{ packets}}{\text{total number of IP packets}} \quad (\text{outbound}) \end{aligned} \quad (2)$$

$R_{t_d}[[\text{TCP}|\text{UDP}|\text{ICMP}]_x]$ has the similar meaning as $R_{t_d}[K_x]$ does except $R_{t_d}[[\text{TCP}|\text{UDP}|\text{ICMP}]_x]$ represents protocol rate (one of TCP, UDP or ICMP), instead of flag rate. For example, $R_1[\text{TCP}_o]$ stands for the TCP rate of outbound traffic when the observation period is set to 1 s. As we mentioned before, the use of flag and protocol rates eliminates the possibility of any misunderstanding introduced using absolute numbers of specific packets. In the following section, the flag and protocol rates represent the features of flooding attacks and provide the basic model for the detection of flooding attacks.

4. Characterizing the traffic of flooding attacks

Fig. 1 illustrates the simulated setting running on 100 Mbps bandwidth network. The Web server using Apache, Web clients, DDoS attackers, and the network monitoring device including a packet collecting agent and an alarming agent are implemented on LINUX machines. We use the following software packages: SPECweb99 [3,27] running on Web clients generates Web traffic; Tribe Flood Network 2000 (TFN2K) [3,25] running on DDoS attackers simulates flooding attacks with random ports; and libpcap [3,15] used by the network monitoring device captures all packets going into and flowing out of the Web server.

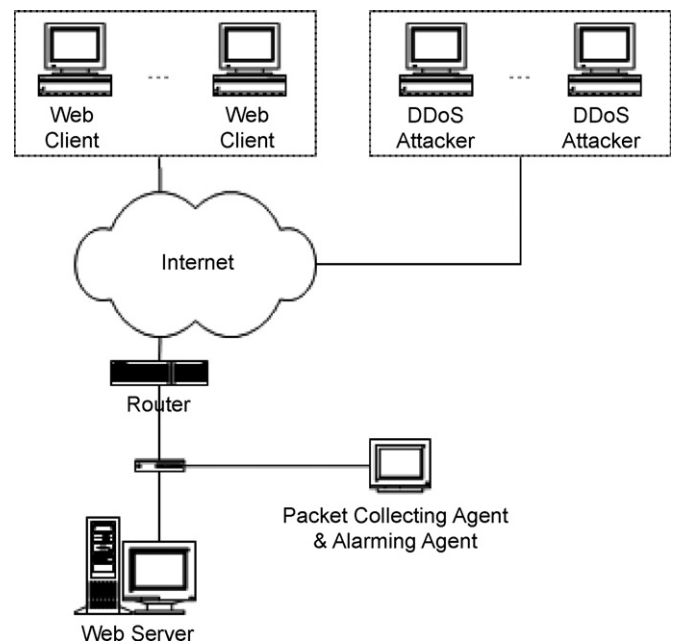


Fig. 1. The simulated network setting.

¹ This is called the normalized traffic rates.

While the Web clients request Web services, the DDoS attackers issue various well-known flooding attacks towards the Web server in diverse traffic pattern. The packet collecting agent, sitting on the network monitoring device, captures IP packets and classifies them into TCP, UDP, or ICMP packets. The agent looks into TCP packets in detail and separates them into headers and payloads. The alarming agent then calculates the two traffic rates, detects DDoS network flooding attacks upon the traffic rate, and then finally issues an alarm in case of a flooding attack.

We measured two traffic rates – flag rates and protocol rates – in two settings: Web traffic without and with flooding attacks. For each network traffic setting, we changed two factors, Simultaneous connections (SC) and Requests per Connection (R/C), to emulate different Web traffic patterns. The SC is the number of HTTP connections at a given time, which approximately mimics the number of users in real network. The R/C represents the number of requests to be issued in a HTTP connection. In our empirical study, we used 5, 10, 50, 100, 150, and 200 for SC and 1, 2, 5, and 10 for R/C. For each value, 10 sets of experiments were performed repeatedly and the traffic rates were averaged to eliminate a possible randomness.

In the following subsections, we present the characteristics of normal Web traffic (without flooding attacks) measured in the simulated network setting. Then the characteristics of normal Web traffic are compared with the ones under various flooding attacks. In the end of this section, some significant changes in the network traffic in the presence of flooding attacks are summarized.

4.1. Normal network traffic rates

Some experimental results of normal Web traffic are depicted in Figs. 2 and 3, when R/C varied from one to five, the observation period t_d (s) was either 1 or 5 s, and SC was set to 200. We varied t_d and SC values very widely but we did not

observe any significant change in the traffic rates with different values of t_d and SC. The simulation results also show that varying the sampling period did not produce any significant difference. Thus, the simulation results with a pair of SC and R/C, and a fixed sampling period presented in the following subsections can be considered as the representative results, and we will discuss the characteristics of these results.

In the protocol rates, as depicted in Figs. 2(b) and 3(b), all of inbound and outbound packets belonged to TCP and thus the TCP protocol rate became exactly one. The experimental results simply show the fact that Web service uses TCP protocol. In the TCP flag rates of inbound traffic, as shown in Figs. 2(a) and 3(a), all flag rates except ACK were less than 0.1. On the other hand, the rate of ACK was close to 1.0. The results show that every normal TCP packet sends an acknowledgement as a notification of receipt, but SYN and FIN packets are sent only once per each connection and other flag packets are seldom transferred. As R/C increased, $R_1[A_i]$ slightly increased or decreased but not significantly when S/C (approximately same as the number of clients attached to the server at a moment) varied. In the outbound flag rates, as shown in Figs. 2(a) and 3(a), SYN and FIN rates were nearly same as those of the inbound traffic. The ACK rate became almost one since the server had to send an acceptance notification to every incoming packet. The Figs. 2(a) and 3(a) show that about 60% of the outbound packets set their PSH flags. Other flag rates were almost less than 0.1 as in the inbound traffic. Two figures with different t_d show that varying t_d did not produce any significance on the traffic rates.

4.2. Traffic rate distributions with various flooding attacks

The flag and protocol rates were measured and analyzed under several well-known flooding attacks: SYN, UDP, and ICMP flooding attacks. For emulating the attacks, the TFN2K [25] was utilized. For SC and R/C, we used 100 and 1,

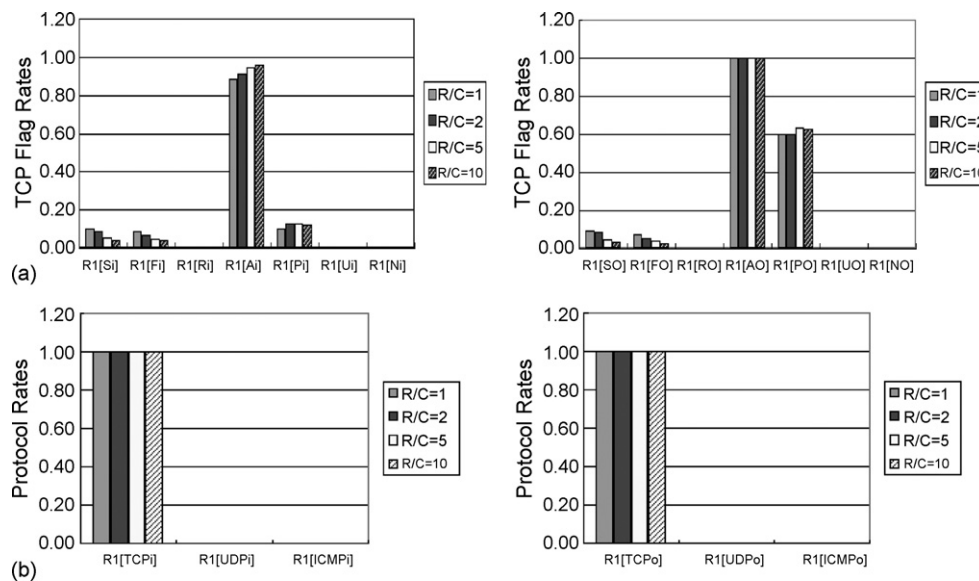


Fig. 2. Network traffic rates of normal Web traffic when SC = 200 and $t_d = 1$ s. (a) TCP flag rates. (b) Protocol rates.

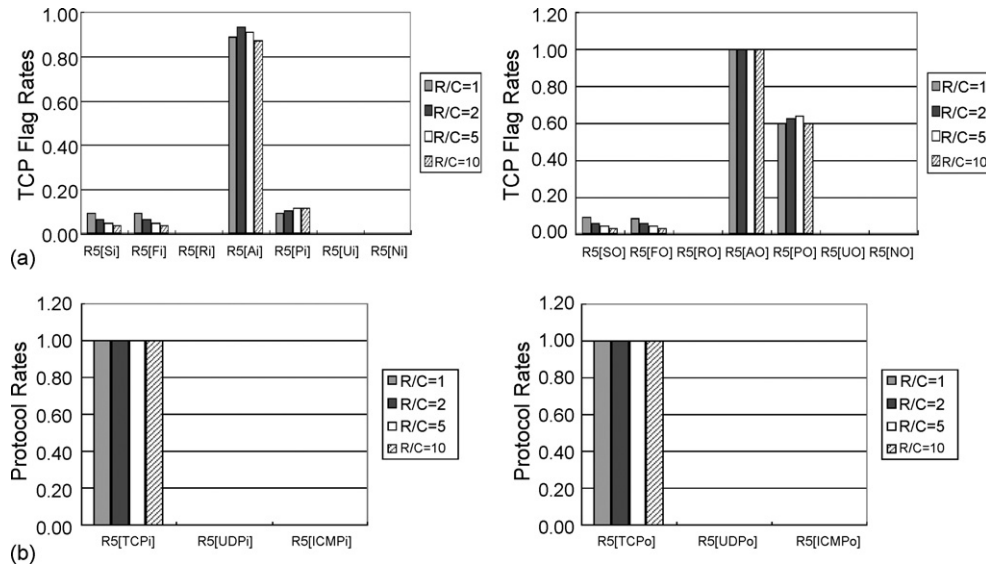


Fig. 3. Network traffic rates of normal Web traffic when $SC = 5$ and $t_d = 5$ s. (a) TCP flag rates. (b) Protocol rates.

respectively, and t_d was set to 1 s. As mentioned before, varying the values of SC , R/C and t_d did not make any difference on the results and we, therefore, illustrate those specific simulation results in the following subsections.

4.2.1. SYN flooding attack

Fig. 4 presents the inbound and outbound TCP flag rates and protocol rates when a SYN flooding attack was made to a Web server. The attack was performed using random ports during the period of 30–70 (s). In Fig. 4, the rates before 30 and after 70 s were normal and the rates during the attack period were influenced by the attack.

In Fig. 4(a), $R_1[A_i]$ went down to about 0.0, due to the SYN's burst during the attack, even though the number of ACK packets did not change significantly. On the other hand, $R_1[S_i]$ and $R_1[U_i]$ in the inbound flag rates drastically changed and went up to almost 1.0. As depicted in Fig. 4(b), the outbound flag rates were not affected nearly by the attack. This reveals that the victim replied to all incoming packets as usual even though the packets were for a DDoS attack. The total number of replied packets might be changed but the rates remained as before. In Fig. 4(c and d), the inbound and outbound protocol rates were just $R_1[TCP_i]$ and $R_1[TCP_o]$, since all utilized packets were TCP packets. Obviously, all other protocol rates were zero.

4.2.2. UDP flooding attack

Fig. 5 shows the inbound and outbound flag rates and protocol rates under a UDP flooding attack. The UDP flooding attack was made between 20 and 60 s. Right after the attack, as shown in Fig. 5(c), $R_1[UDP_i]$ drastically increased while $R_1[TCP_i]$ decreased as much. But the flag rates did not change visibly even though the amount of TCP traffic decreased, since the UDP flooding attack was done with UDP packets. Unlike $R_1[UDP_i]$, the UDP protocol rate – $R_1[UDP_o]$ – in the outbound network traffic did not change significantly and remained in normal data flow, as depicted in Fig. 5(d), because most

incoming UDP packets on random ports did not require any response.

4.2.3. ICMP flooding attack

Fig. 6 shows inbound and outbound TCP flag rates and protocol rates under an ICMP flooding attack, which was made between 20 and 53 s. The inbound traffic rates shown in Fig. 6(a) sporadically went up and down, since a very few TCP packets were allowed to flow into a victim due to the flooding of ICMP, and also only a few number of specific flag packets varied in a particular time. In Fig. 6(b), all flag rates were nearly zero. This result indicates that no TCP flags were flown out of the victim. The rates were simply zero if no packets were observed on monitoring network.

The ICMP protocol rate for inbound traffic, $R_1[ICMP_i]$, increased from nearly zero to one. Like $R_1[UDP_i]$ under UDP flooding attack, the increase of $R_1[ICMP_i]$ under the ICMP flooding attack was natural. One notable thing is that $R_1[ICMP_o]$ sharply went up and down, as depicted in Fig. 6(d), since all of inbound ICMP ping requesting packets asked for a Web server (victim) to reply with ICMP ping acknowledging packets but the victim was not able to continuously send any packet due to the flooding of ICMP packets.

The experimental results in this section imply that, in order to clearly characterize the symptoms of various flooding attacks, the alarming agent needs to calculate all four traffic rates together, which are flag rates and protocol rates for inbound and outbound network traffic, respectively. In Table 1, we summarize the network traffic rates with significant changes.

In Table 1, $\uparrow$ indicates that a specific rate approaches to nearly one and $\downarrow$ means nearly zero. To determine whether or not a SYN flooding attack is made, our alarming agents need to watch the fluctuation of flag rates and, for UDP and ICMP flooding attacks, they need to check protocol rates. For the cases of SYN and UDP flooding attacks, the inbound traffic

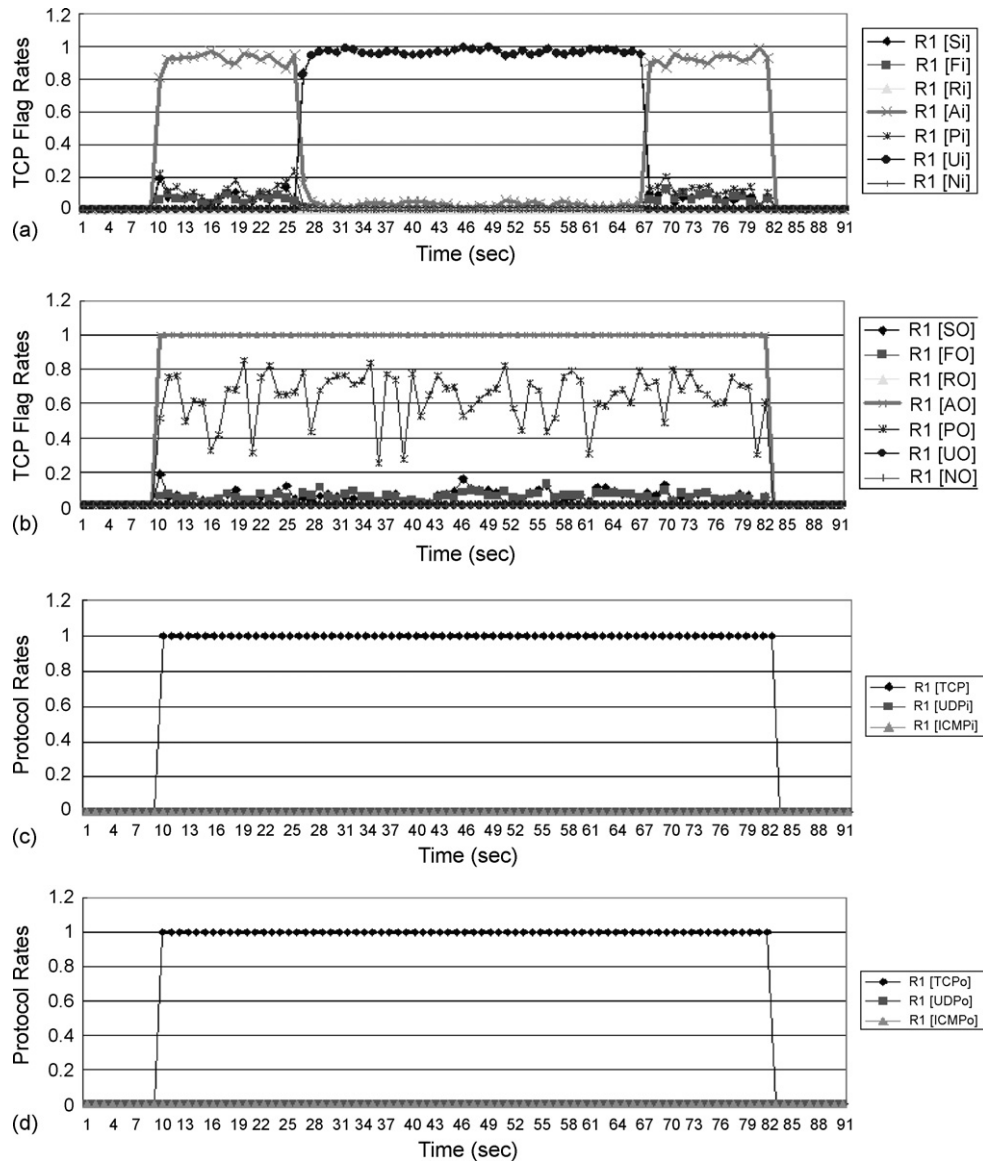


Fig. 4. Network traffic rates under SYN flooding attack. (a) Inbound TCP flag rates. (b) Outbound TCP flag rates. (c) Inbound protocol rates. (d) Outbound protocol rates.

rates are crucial to detect flooding attacks. However, the alarming agents might watch both inbound and outbound traffic rates for the purpose of detecting ICMP flooding attacks. In the next section, we will show how the above traffic rates provide attributes for our flooding attack detection algorithm.

5. Detecting flooding attacks using soft computing methods

We propose a brokering agent architecture, as consisting of a packet collecting agent and an adaptive reasoning agent – an alarming agent – that analyze network traffic, detect network flooding attacks upon the traffic rate, and finally issue an alarm in case of a flooding attack. To inspire adaptability into our alarming agents, we use soft computing methods [12,31], i.e., inductive decision tree algorithms [2,26] and the probabilistic reasoning of Bayesian approach [7], and compile the models of network traffic into a set of rules using them.

5.1. Bayes theorem and decision tree-based induction algorithms

Bayes rule examines whether or not a property observed as evidence belongs to a specific hypothesis (or class), given a set of data distribution. Bayes theorem [7] can be defined as follows:

$$P(h_j|x_i) = \frac{P(x_i|h_j)P(h_j)}{\sum_{j=1}^m P(x_i|h_j)P(h_j)} \quad (3)$$

where

- a set of observable attributes, $X = \{x_1, x_2, \dots, x_n\}$;
- a set of hypotheses in a domain, $H = \{h_1, h_2, \dots, h_m\}$;
- $P(h_j|x_i)$ is the posterior probability of the hypothesis h_j , $h_j \in H$, given that $x_i, x_i \in X$, is an observable event.

In our framework, the set of observable attributes, X , consists of TCP flag rates and protocol rates, and the hypotheses are either

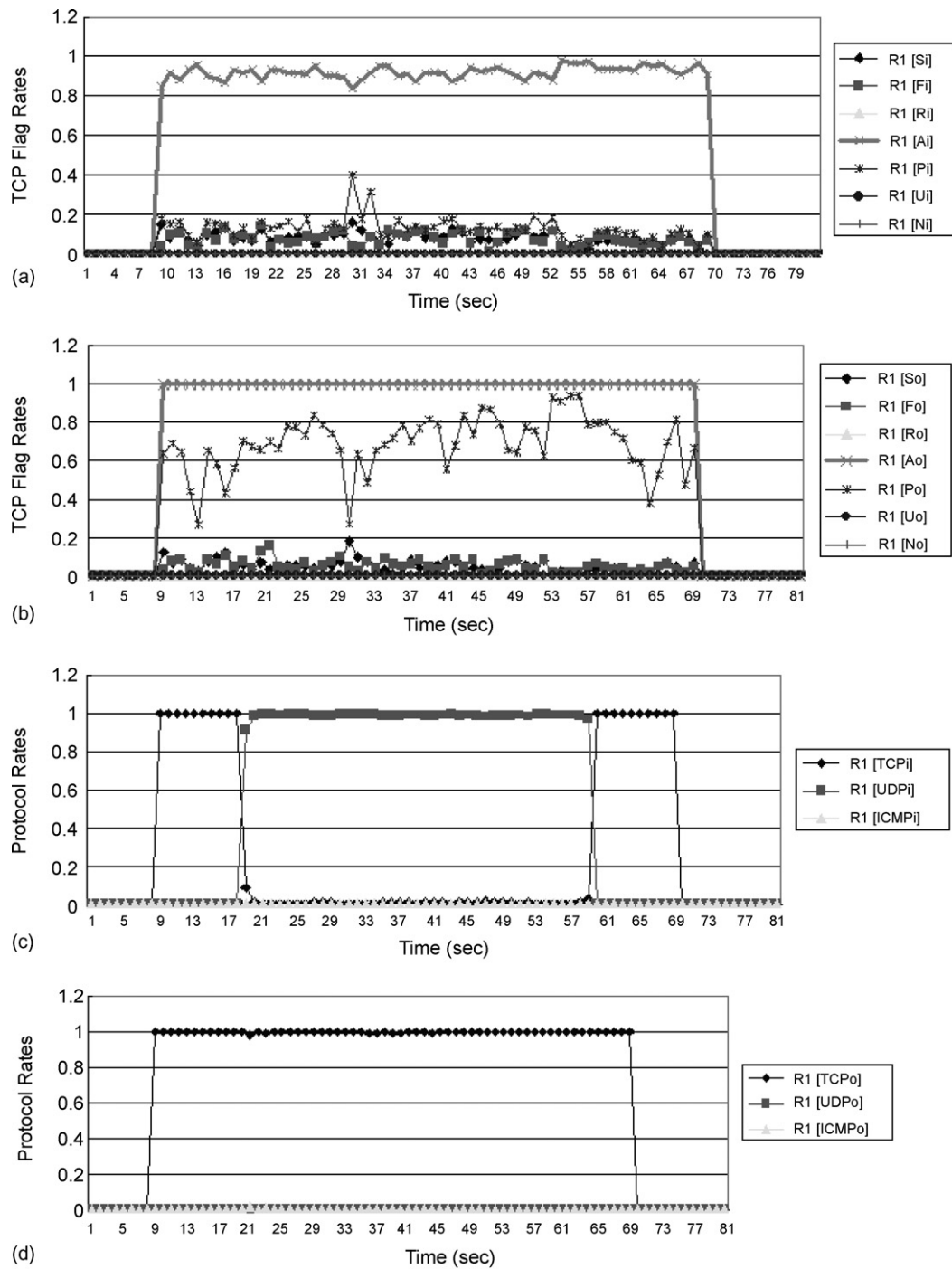


Fig. 5. Network traffic rates under UDP flooding attack. (a) Inbound TCP flag rates. (b) Outbound TCP flag rates. (c) Inbound protocol rates. (d) Outbound protocol rates.

Table 1
Network traffic rates with significant changes

Types of attacks	Inbound		Outbound	
	Flag rates	Protocol rates	Flag rates	Protocol rates
SYN	$R[S_i] \uparrow, R[U_i] \uparrow, R[A_i] \downarrow$			
UDP	$R[UDP_i] \uparrow, R[TCP_i] \downarrow$			
ICMP	$R[ICMP_i] \uparrow, R[TCP_i] \downarrow$		$R[ICMP_o] \uparrow, R[TCP_o] \downarrow$	

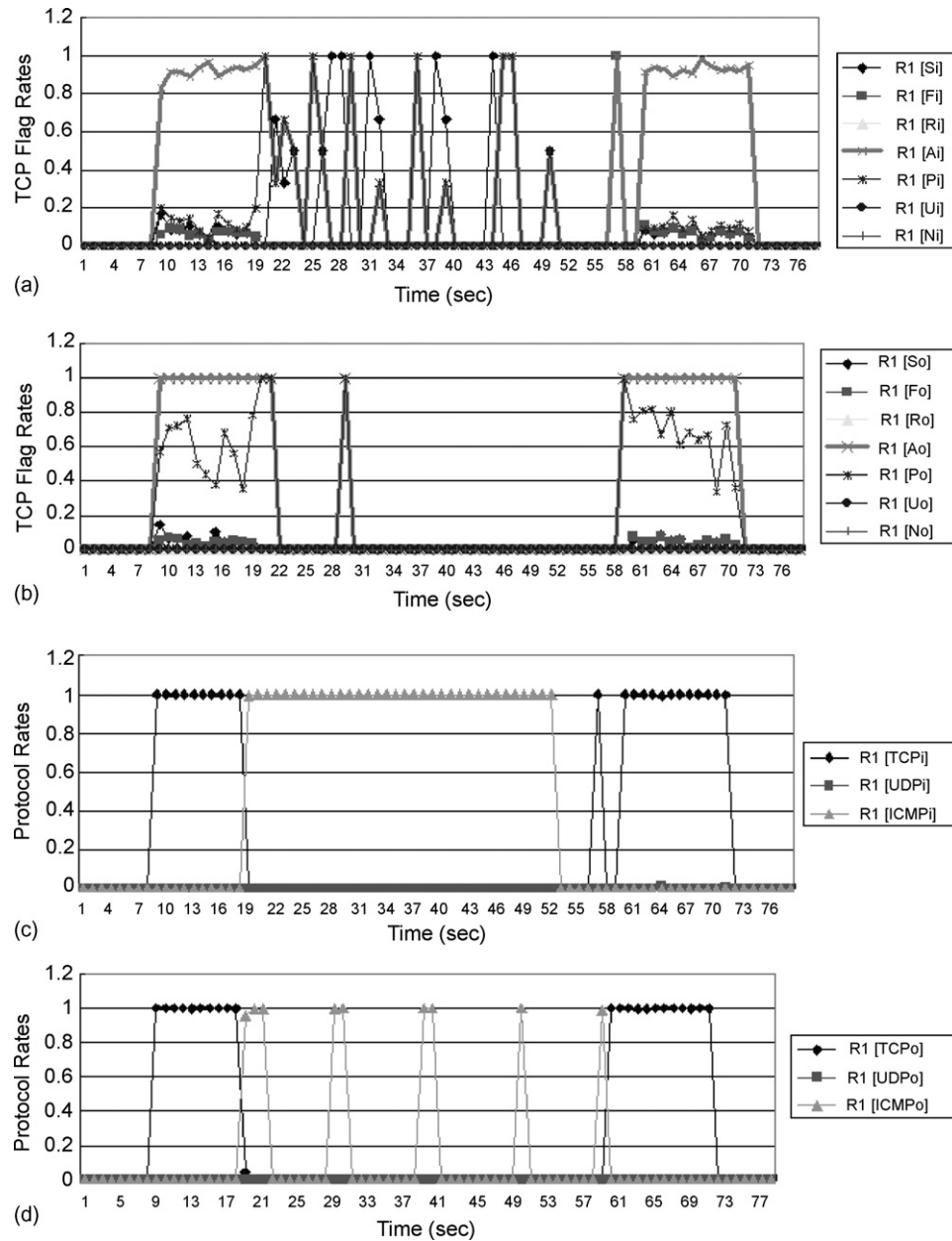


Fig. 6. Network traffic rates under ICMP flooding attack. (a) Inbound TCP flag rates. (b) Outbound TCP flag rates. (c) Inbound protocol rates. (d) Outbound protocol rates.

a flooding attack or no flooding attack. Given the set of data as evidence, Bayes rule allows us to assign probabilities of hypotheses, $P(h_j|x_i)$. Our alarming agents compute $P(h_j|x_i)$ during online, and set an alarm when the probability of a flooding attack given input is greater than that of no flooding attack.

The decision tree approach such as ID3, C4.5 [26] and CN2 [2] is to divide the domain space into classified regions, which are given by a set of classes $C = \{c_1, c_2, \dots, c_m\}$. The basic idea of the decision tree-based induction algorithms finds out a set of ordered attributes, $X = \{x_1, x_2, \dots, x_n\}$, which split the datasets into a correct classification with the highest information gain first. A decision tree has internal nodes labeled with attributes $x_i \in X$, arcs associated with the parent attributes, and leaf nodes corresponding to classes $c_j \in C$.

The decision tree-based induction algorithms, thus, generate a tree representing a model of network traffic in the simulated network setting. Once the tree is built using training data, the optimal rules from the tree can be obtained and are applied to the new network traffic to determine whether or not a flooding attack is made.

5.2. Compilation of network traffic

Let S be the set of traffic states that the adaptive reasoning agent can discriminate among, and let L be the set of compilation methods (soft computing methods) that the agent employs. Given a soft computing methods $l \in L$, a set of compiled decision-making rules of an adaptive reasoning agent

is defined as

$$\rho_l : S \rightarrow \{\text{attacks, no attacks}\} \quad (4)$$

representing whether a flooding attack occurs in the state $s \in S$. Thus, various soft computing methods compile the models of network traffic into different functions ρ_l . We generate the training examples for these algorithms from TCP, UDP and ICMP-based network environments.

5.3. Learning results

To construct compiled rules for our alarming agents, we used three machine learning algorithms: C4.5, CN2, and naive Bayesian classifier. C4.5 represents its output as a decision tree, and the output of CN2 is an ordered set of if–then rules. For the Bayesian classifier, the results are represented as rules specifying the probability of occurrence of each attribute value given a class [2], in our case ‘attacks’ and ‘no attacks.’

In our traffic rate analysis mechanism, under the SYN flooding attack, the attributes of situations that the alarming agents could sense were the SYN flag rate for inbound traffic $R_1[S_i]$ and the ACK flag rate for inbound traffic $R_1[A_i]$. For the benchmark, we also computed the rate of SYN–FIN pair, which is a core of Wang’s SYN flooding detection mechanism [29].

Using the three machine learning algorithms [8] and the training examples as inputs, we could get the compiled rules as described in Figs. 7 and 8.

Using our model of traffic rate analysis, C4.5 indicated that the SYN flooding attacks occurred if $R_1[S_i]$ was greater than 0.4. The rules obtained by CN2, as shown in Fig. 7, was similar to the ones of C4.5 but the resulting value of the SYN flag rate was 0.48. The Bayesian classifier showed that the average of $R_1[S_i]$ was 0.98 given the class of ‘attacks.’ The learning results

```

MaxTrain: 719, MinTest: 0
C4.5 Trees:
  Decision Tree:

    R[Si] <= 0.4 : WEB (361.0/1.0)
    R[Si] > 0.4 : SYN (359.0)

  Errors = 1/720

CN2 Decision List:
  IF "R[Si]" > 0.48
  THEN class = "SYN" [0 359]
  ELSE
  IF "R[Si]" > 0.02
  THEN class = "WEB" [332 0]
  ELSE
  IF "R[Si]" < 0.02
  THEN class = "WEB" [28 0]
  ELSE
  (DEFAULT) class = "SYN" [0 1]
  Errors = 0/720

Bayes Classifier:
  P(class=WEB) = 0.500000
  P(R[Si]|class=WEB) = G(0.071971,0.051441)
  P(R[Ai]|class=WEB) = G(0.928013,0.051446)
  P(class=SYN) = 0.500000
  P(R[Si]|class=SYN) = G(0.982168,0.067275)
  P(R[Ai]|class=SYN) = G(0.017831,0.067275)
  Errors = 1/720
    
```

Fig. 7. Learning results based upon our model of traffic rate analysis.

```

MaxTrain: 719, MinTest: 0
C4.5 Trees:
  Decision Tree:

    R[Fi]/R[Si] <= 0.038656 : SYN (360.0/2.0)
    R[Fi]/R[Si] > 0.038656 : WEB (360.0/2.0)

  Errors = 4/720

CN2 Decision List:
  IF "R[Fi]/R[Si]" > 0.83
  THEN class = "WEB" [263 0]
  ELSE
  IF "R[Fi]/R[Si]" < 0.04
  THEN class = "SYN" [2 358]
  ELSE
  IF 0.09 < "R[Fi]/R[Si]" < 0.83
  THEN class = "WEB" [91 0]
  ELSE
  IF "R[Fi]/R[Si]" < 0.06
  THEN class = "WEB" [3 0]
  ELSE
  IF "R[Fi]/R[Si]" < 0.45
  THEN class = "SYN" [0 1]
  ELSE
  (DEFAULT) class = "WEB" [1 1]
  Errors = 3/720

Bayes Classifier:
  P(class=WEB) = 0.500000
  P(R[Fi]/R[Si]|class=WEB) = G(0.965380,0.402598)
  P(class=SYN) = 0.500000
  P(R[Fi]/R[Si]|class=SYN) = G(0.003586,0.044161)
  Errors = 8/720
    
```

Fig. 8. Learning results based upon Wang’s work.

for Wang’s work, as shown in Fig. 8, were generated over the SYN–FIN pair, $R_1[F_i]/R_1[S_i]$.

To evaluate the quality of various rule sets generated by different learning algorithms the performance obtained was expressed in terms of the ratio of {total number of alarms – (number of false alarms + number of missed alarms)} to the total number of alarms. The false alarm is defined as the alert turns on when a flooding attack does not occur, and the missed alarm is defined as the alert does not turn on when the attack does occur.

To find a meaningful size of the training set which could guarantee the soundness of the learning hypothesis, we generated several sets of training examples whose size was 48, 96, 144, 192, 240, 480, 720, 960, 1200, and 1440 tuples, respectively. The resulting performances (%) and the sizes of training examples are shown in Fig. 9. In the traffic rate analysis, the best performance was achieved by the rules compiled using Bayesian classifier, as depicted in Fig. 9(a), when the training instances were 720. In the learning curve of the Wang’s work, as depicted in Fig. 9(b), since the performances obtained by C4.5 and Bayes algorithms were almost identical, the rules compiled using C4.5 with 1440 training instances were chosen.

By using the compiled rules, we tested the performances of the two network traffic analysis mechanisms (TRA and Wang’s work) on new sets of network flow patterns. The testing network flows were generated during 100 s. In the testing network environment, the Simultaneous Connections were 7, 15, 40, 70, 130 and 160, the Requests per Connection were 4, 12, 18, and 24, and the DDoS flooding attacks were made using three different time slots, i.e., four 10 s, two 10 s, and one 30 s

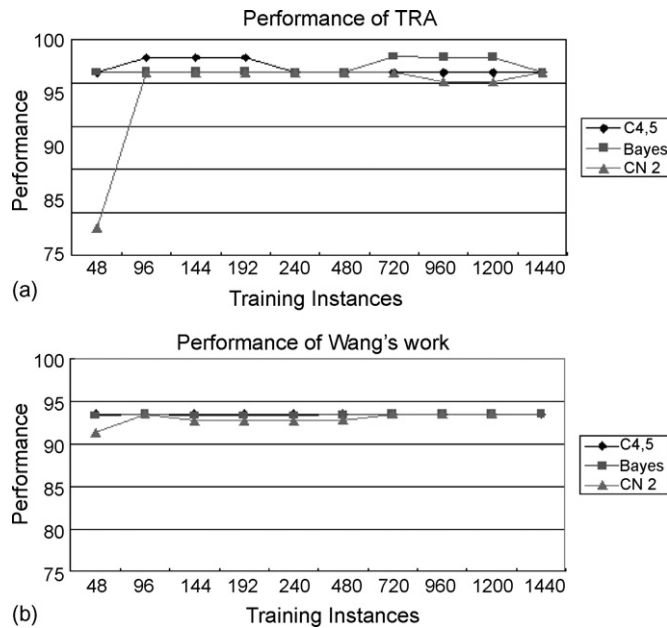


Fig. 9. The SYN flooding attacks detection performances using the resulting rules compiled by C4.5, naive Bayes, and CN2 learning algorithms in (a) TRA and (b) Wang's work, respectively.

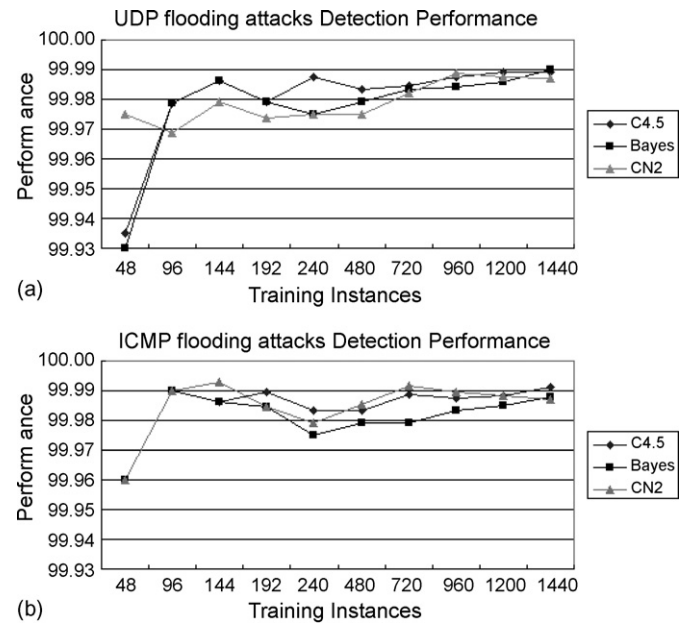


Fig. 10. The (a) UDP and (b) ICMP flooding attacks detection performances using the resulting rules compiled by C4.5, Bayes, and CN2 learning algorithms in our model of traffic rate analysis.

ranging from 30 to 60 s. These combinations, thus, lead to eventually 72 different Web traffics and the flooding attacks.

We analyzed the performance results in Table 2 using the standard analysis of variance (ANOVA) method.

Since the computed value of $f = 178.015076$ in ANOVA exceeds 6.63 ($=f_{0.01,1,142}$), we know that the two mechanisms were not all equally effective at the 0.01 level of significance, i.e., the differences in their performance were not due to chance with probability of 0.99. In the experiment, missed alarms did not happen and all the errors measured were caused by false alarms. Our alarming agent's performance using the TRA mechanism was better than that of Wang's SYN flooding detection mechanism. This result indicates that $R_1[A_i]$ was more crucial than $R_1[F_i]$ in case of the SYN flooding attacks detection.

To measure the performance of the three learning algorithms, we applied the rules compiled using them to new network traffic characterized by TRA mechanism. In the experiment, the performances of our alarming agents using the three learning algorithms were exactly the same. ANOVA revealed that the three machine learning algorithms were all equally effective.

We also tested our alarming agent's performance in UDP and ICMP flooding attacks using the same set of learning algorithms.

Table 2
Performances in TRA and Wang's work

Methods	Performances (%)
TRA	99.881321 ± 0.316758
Wang's work	97.022569 ± 1.791220
ANOVA	$f = 178.015076$

As depicted in the Fig. 10, our alarming agents also showed fairly good performances in detecting UDP and ICMP flooding attacks. The detection performances of UDP and ICMP flooding attacks were almost 100% and higher than that of SYN flooding attacks. Since there were no UDP and ICMP packets in the normal Web traffic, our alarming agents could not miss their alarms for the UDP and ICMP flooding attacks whenever the attacks were made on the TCP-based Web server.

6. Conclusions

We investigated the traffic rate analysis as a network traffic flow analysis mechanism and, using our TRA mechanism, analyzed TCP-based network flows. The experiments on the TCP-based server showed the distinctive patterns of flag rates and protocol rates in case of the flooding attacks, i.e., SYN, UDP, and ICMP flooding attacks, and in absence of a flooding attack, respectively. Further, we detected the network flooding attacks using a set of rules compiled by various machine learning algorithms (C4.5, CN2, and naive Bayes Classifier), and compared our detection performance to the benchmark. The combination of traffic rate analysis and flooding attacks detection mechanism enables Internet resources to be safe and stable from the ongoing flooding attacks.

In our future research, to determine the reliability of our method, we will continuously test our framework in different network settings, for example, SMTP and FTP servers. We will also extend our work to various kinds of worm propagation in scaled-up network settings. As part of our ongoing work, we are analyzing the features of non-autonomous propagation worms, for example, email attachment worms using SMTP protocol. We hope that our framework will be applicable when the fluctuation of network traffic becomes prohibitive, and be

valuable to detect flooding attacks as well as worms on the Internet.

Acknowledgements

The authors would like to thank the editors and the anonymous reviewers for their helpful comments. The corresponding author, Prof. Noh, has greatly profited from discussing with Prof. Lotfi A. Zadeh and Dr. Masoud Nikravesh, while he has been a visiting professor within The Berkeley Initiative in Soft Computing (BISC) group at University of California, Berkeley.

This work has been supported by the Agency for Defense Development, Korea, under Grant UD060072FD “A Study on the Multi-Spectral Threat Data Integration of ASE,” 2006.

References

- [1] BindView's RAZOR Security Team, Zombie Zapper, available on-line: http://razor.bindview.com/tools/ZombieZapper_form.shtml, 2001.
- [2] P. Clark, T. Niblett, The CN2 induction algorithm, *Mach. Learn. J.* 3 (4) (1989) 261–283.
- [3] D. Dittrich, Distributed denial of service (DDoS) Attacks/tools, available on-line: <http://staff.washington.edu/dittrich/misc/ddos/>, 2006.
- [4] A.E. Elalfi, R. Haque, M.E. Elalami, Extracting rules from trained neural network using GA for managing E-business, *Appl. Soft Comput.* 4 (2004) 65–77.
- [5] L. Garber, Denial-of-service attacks rip the Internet, *IEEE Comput.* 33 (4) (2000) 12–17.
- [6] T.M. Gil, M. Poletto, MULTOPS: a data-structure for bandwidth attack detection, in: *Proceedings of the 10th USENIX Security Symposium*, 2001, pp. 23–38.
- [7] R. Hanson, J. Stutz, P. Cheeseman, Bayesian Classification Theory, Technical Report FIA-90-12-7-01, NASA Ames Research Center, AI Branch, 1991.
- [8] L. Holder, ML v2.0: Machine Learning Program Evaluator, available on-line: <http://www-cse.uta.edu/~holder/ftp/ml2.0.tar.gz>.
- [9] K. Houle, G. Weaver, N. Long, R. Thomas, Trends in denial of service attack technology, CERT Coordination Center White Paper, available on-line: http://www.cert.org/archive/pdf/DoS_trends.pdf, 2001.
- [10] A. Householder, A. Manion, L. Pesante, G.M. Weaver, Managing the Threat of Denial-of-Service Attacks, CERT Coordination Center White Paper, available on-line: http://www.cert.org/archive/pdf/Managing_DoS.pdf, 2001.
- [11] F. Kargl, J. Maier, M. Weber, Protecting web servers from distributed Denial of service attacks, in: *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 514–524.
- [12] F.O. Karray, C.D. Silva, *Soft Computing Intelligent Systems Design: Theory, Tools and Applications*, Addison Wesley, Harlow, England, 2004.
- [13] N. Kasabov, *Adaptation and interaction in dynamical systems: modelling and rule discovery through evolving connectionist systems*, *Appl. Soft Comput.* 6 (2006) 307–322.
- [14] A.B. Kulkarni, S.F. Bush, S.C. Evans, Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics, Technical Report 2001CRD176, GE Research and Development Center, 2001.
- [15] Lawrence Berkeley National Laboratory's Network Research Group, libpcap: the Packet Capture library, available on-line: <http://ftp.ee.lbl.gov/>, 2002.
- [16] C. Lee, S. Noh, K. Choi, G. Jung, Characterizing DDoS attacks with traffic rate analysis, in: *Proceedings of e-Society*, 2003, pp. 81–88.
- [17] M. Li, P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, second ed., Springer Verlag, New York, 1997.
- [18] J.C. Liu, K.G. Shin, C.C. Chang, Prevention of congestion in packet-switched multistage interconnection networks, *IEEE Trans. Parallel Distrib. Syst.* 6 (5) (1995) 535–541.
- [19] D. Moore, G.M. Voelker, S. Savage, Inferring Internet Denial-of-Service activity, in: *Proceedings of the 10th USENIX Symposium*, 2001, pp. 9–22.
- [20] D. Moshou, I. Hostens, G. Papaioannou, H. Ramon, Dynamic muscle fatigue detection using self-organizing maps, *Appl. Soft Comput.* 5 (2005) 391–398.
- [21] P. Mutaf, Defending against a Denial-of-Service attack on TCP, in: *Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection*, 1999.
- [22] The National Infrastructure Protection Center (NIPC), Potential Distributed Denial of Service (DDoS) Attacks, ADVISORY 01-021, 2001.
- [23] S. Noh, P.J. Gmytrasiewicz, Towards flexible multi-agent decision-making under time pressure, in: *Proceedings of IJCAI*, 1999, pp. 492–498.
- [24] S. Noh, C. Lee, K. Choi, G. Jung, Detecting Distributed Denial of Service (DDoS) Attacks Through Inductive Learning, *Lecture Notes in Computer Science*, 2690, Springer Verlag, New York, 2003, pp. 286–295.
- [25] Packet Storm, Tribe Flood Network 2000 (TFN2K) DDoS tool, available on-line: http://packetstormsecurity.org/distributed/TFN2k_Analysis-1.3.txt, 2000.
- [26] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Francisco, 1993.
- [27] Standard Performance Evaluation Corporation, SPECweb99 Benchmark, available on-line: <http://www.spec.org/osg/web99>, 2005.
- [28] TheoryGroup, Remote Intrusion Detector (RID), available on-line: <http://www.theorygroup.com/Software/RID>, 2001.
- [29] H. Wang, D. Zhang, K.G. Shin, Detecting SYN flooding attacks, in: *Proceedings of IEEE INFOCOM '02*, 2002, pp. 1530–1539.
- [30] Y. Xiong, S. Liu, P. Sun, On the defense of the distributed denial of service attacks: an on-off feedback control approach, *IEEE Trans. Syst. Man Cybern.—Part A: Syst. Hum.* 31 (4) (2001) 282–293.
- [31] L.A. Zadeh, Fuzzy logic, neural networks and soft computing, *Commun. ACM* 37 (3) (1994) 77–84.