

# Detecting Distributed Denial of Service (DDoS) Attacks Through Inductive Learning\*

Sanguk Noh<sup>1</sup>, Cheolho Lee<sup>2</sup>, Kyunghee Choi<sup>2</sup>, Gihyun Jung<sup>3</sup>

<sup>1</sup>School of Computer Science and information Engineering,  
The Catholic University of Korea, Bucheon, Korea  
sunoh@catholic.ac.kr

<sup>2</sup>Graduate School of Information and Communication,  
Ajou University, Suwon, Korea  
cheolholee@cesys.ajou.ac.kr, khchoi@madang.ajou.ac.kr

<sup>3</sup>Division of Electronics Engineering,  
Ajou University, Suwon, Korea  
khchung@madang.ajou.ac.kr

**Abstract.** As the complexity of Internet is scaled up, it is likely for the Internet resources to be exposed to Distributed Denial of Service (DDoS) flooding attacks on TCP-based Web servers. There has been a lot of related work which focuses on analyzing the pattern of the DDoS attacks to protect users from them. However, none of these studies takes all the flags within TCP header into account, nor do they analyze relationship between the flags and the TCP packets. To analyze the features of the DDoS attacks, therefore, this paper presents a network traffic analysis mechanism which computes the ratio of the number of TCP flags to the total number of TCP packets. Based upon the calculation of TCP flag rates, we compile a pair of the TCP flag rates and the presence (or absence) of the DDoS attack into state-action rules using machine learning algorithms. We endow alarming agents with a tapestry of the compiled rules. The agents can then detect network flooding attacks against a Web server. We validate our framework with experimental results in a simulated TCP-based network setting. The experimental results show a distinctive and predictive pattern of the DDoS attacks, and our alarming agents can successfully detect various DDoS attacks.

## 1 Introduction

As the complexity of Internet is scaled up, it is likely for the Internet resources to be exposed to Distributed Denial of Service (DDoS) attacks [2], [6], [10]. There has been a lot of related work [3], [7], [15] which focus on analyzing the pattern of the DDoS attacks to protect users from them. Most of related work keeps track of source Internet Protocol (IP) addresses and checks the distribution of the IP addresses, whether or not

---

\* This work has been supported by the Korea Research Foundation under grant KRF-2002-041-D00465, by the KISTEP under National Research Laboratory program, and by the Catholic University of Korea research fund granted in the program year of 2003.

the DDoS attacks occur. If the randomness of the source IP addresses is getting higher than usual one, their approaches set alarms upon the detection of the DDoS attacks. However, these approaches are useless when attackers reduce the level of randomness of the source IP addresses or when the attackers use the actual IP address instead of the spoofed IP address. To be more generally applicable in realistic settings, therefore, this paper presents a network traffic analysis mechanism of the DDoS attacks using all of the flags, i.e., SYN, FIN, RST, ACK, etc., within Transmission Control Protocol (TCP) header and taking into account relationship between the flags and the network packets. Based upon our analysis mechanism of the DDoS attacks, further, our paper addresses the question of how to detect the DDoS attacks on Web Servers.

To understand the features of DDoS attacks, we introduce the analysis mechanism of the DDoS attacks in two settings: the normal Web server without any attack and the Web server with the DDoS attacks. In these settings, we measure TCP flag rates, which are expressed in terms of the ratio of the number of TCP flags to the total number of TCP packets. For example, the number of SYNs drastically increases in case of SYN flooding attacks which is the most common DDoS attacks. In consequence, the increasing number of SYNs indicates the possibility of the DDoS attacks. Our analysis mechanism calculates the TCP flag rates and provides the basis of the DDoS attacks detection in a TCP-based network environment.

We also propose a DDoS attacks detection mechanism using inductive learning algorithms [1], [13] and Bayesian classifier [4]. To identify the DDoS attacks, we endow an alarming agent with a tapestry of reactive rules. The reactive tools [11] are constructed by compiling the results of TCP flag rates and presence (or absence) of flooding attacks into state-action rules. The compilation process exploits the regularities of the DDoS attacks, if any, and enables our alarming agents to detect them. The rules can be obtained from machine learning algorithms which use the results of TCP flag 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 of this paper we discuss related approaches to our analysis and detection mechanism. Section 3 describes the details of our framework, and Section 4 describes a simulated, TCP-based network setting to test our approach. We validate our framework empirically, and discuss the experimental results. In conclusion, we summarize our results and further research issues.

## 2 Related Work

Regarding the analysis and detection of DDoS network flooding attacks, many researchers have investigated the randomness and distribution of source IP addresses. From this perspective, if the randomness of the source IP addresses is getting higher than usual one, they issue alarms based upon the detection of the DDoS attacks.

Gil and Poletto [3] examine flows in one direction vs. flows in the opposite direction over IP packets by using their own data-structure, MULTOPS. Their network

monitoring device using the MULTOPS detects flooding attacks by “the difference between packet rates going to and coming from the attacker.” Their assumption for the detection is based on the disproportional difference between the packet rates, which is caused by randomness of ‘malicious’ packets. Kulkarni et al. [7] trace the source IP addresses and construct Kolmogorov Complexity Metrics [9] for identifying their randomness. The Kolmogorov Complexity Metrics change according to the degree of randomness of spoofed source IP addresses. Actually, the randomness of source IP addresses is very low without any DDoS attack; otherwise, it is very high under DDoS attacks with randomly distributed source IP addresses. However, these approaches are not applicable when attackers reduce the level of randomness of the source IP addresses or when the attackers use the actual IP address instead of the spoofed IP address.

In another approach to detection mechanism, Wang et al. [15] examine the protocol behavior of TCP SYN-FIN (RST) pairs. If there are no DDoS attacks against a TCP-based server, the rate of SYN flag for TCP connection establishment and the rate of FIN flag for TCP connection termination will be the same value, or rarely different in case of retransmission. Otherwise, the rate of SYN, for example, in SYN Flooding attacks, clearly differs from the one of FIN. The metric of SYN-FIN (RST) pairs could be useful to detect SYN Flooding attacks against Web servers. This approach is somewhat similar to our approach in that both of them take into account TCP flags to detect DDoS flooding attacks. However, their method can be applicable only to SYN Flooding attacks. On the other hand, our approach is more general so that our mechanism can be applicable to all types of DDoS attacks, i.e., SYN Flooding attacks, UDP Flooding attacks, ICMP Flooding attacks, and so on. Further, to our best knowledge, applying machine learning algorithms to the flooding detection mechanism is a pretty new approach in this field of research.

### 3 Network Traffic Analysis and DDoS Attacks Detection

We rely on the dynamics of differences between the rates of TCP flag to analyze the features of DDoS attacks. Due to the burstiness of TCP flags, the ratio of the number of a specific TCP flag within TCP header, for example, SYN, FIN, RST, ACK, etc., to the total number of TCP packets, during normal operations on Web servers, clearly differs from the ones under the DDoS attacks.

#### 3.1 Traffic Rate Analysis

We present a network traffic analysis mechanism, Traffic Rate Analysis (TRA). This mechanism calculates two measuring factors: TCP flag rate and protocol rate. The traffic rate analysis uses the traffic flowing into a victim (a host) as inbound, and the traffic flowing from the victim as outbound. A packet collecting agent captures IP packets and classifies them into *TCP*, *UDP*, or *ICMP* packets. In case of the TCP packet, further, the classification procedure separates the packet into TCP header and payload. From the TCP header containing *SYN*, *FIN*, *RST*, *ACK*, *PSH*, and *URG* flags,

the flags are tested to determine whether or not they are set. If any flag of six TCP flags turns on, the agent counts it and sums it up. The packet collecting agents also count the total number of TCP packets during a specific observation period  $t_d$  (sec). Our alarming agents then compute two metrics TCP flag rates and protocol rates. A flag rate is expressed in terms of the ratio of the number of a TCP flag to the total number of TCP packets as follows:

$$\begin{aligned} R_{td}[Fi] &= \frac{\text{total number of a flag } (F) \text{ in a TCP header}}{\text{total number of TCP packets}} \text{ (inbound)} \\ R_{td}[Fo] &= \frac{\text{total number of a flag } (F) \text{ in a TCP header}}{\text{total number of TCP packets}} \text{ (outbound)} \end{aligned} \quad (1)$$

Here,  $t_d$  means the sampling period. In the equation 1,  $K$  stands for one of six flags: *SIN*, *FIN*, *RST*, *ACK*, *PSH*, and *URG* flags, denoted as  $S$ ,  $F$ ,  $R$ ,  $A$ ,  $P$ , and  $U$ , for either inbound ( $i$ ) or outbound ( $o$ ) network traffic. For example,  $R_1[Ai]$  represents the *ACK* flag rate of inbound traffic when the sampling period is one second.

A protocol rate is also defined by the ratio of the number of *TCP*, *UDP*, or *ICMP* packets to the total number of IP packets. Similarly, for example,  $R_2[UDPo]$  stands for the *UDP* protocol rate of outbound network traffic during the sampling period two seconds.

The traffic rate analysis can be applicable even to scaled-up network settings because the mechanism utilizes a rate scheme. This enables us to examine the various traffic patterns and to identify the features of DDoS attacks in various network environments.

### 3.2 Detecting DDoS Attacks Using Machine Learning Algorithms

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 DDoS network flooding attacks upon the traffic rate, and finally issue an alarm in case of DDoS attacks. Let  $S$  be the set of traffic states that the adaptive reasoning agent can discriminate among. Let  $L$  be the set of compilation methods (learning algorithms) that the agent employs.

Given a learning method  $l \in L$ , a compilation procedure of an adaptive reasoning agent implements a function  $\rho: S \rightarrow \{\text{attacks}, \text{no attacks}\}$ , representing whether a flooding attack occurs in the state  $s \in S$ . Thus, various machine learning algorithms compile the models of network traffic into different functions  $\rho$ . We generate the training examples for these learning algorithms from a TCP-based network environment.

## 4 Simulations and Results

We have implemented a simulated network environment using SPECweb99 [14], Tribe Flood Network 2000 (TFN2K) [12], and libpcap [8]. In the simulated, Web-based setting, the SPECweb99 located in Web clients generates web traffic, the TFN2K on DDS attackers simulates DDoS attacks, and the libpcap used by a packet collecting agent captures the stream of network traffic. While the Web clients request of the Web server that they should be serviced, the DDoS attackers make various flooding attacks towards the Web server.

We construct the simulated network environment on LINUX machines, which consist of Web server using Apache, Web clients, DDoS attackers, a network monitoring device including a packet collecting agent and an alarming agent (700 MHz Pentium III, 256 MB memory), and the network bandwidth of 100 Mbps. Figure 1 presents the simulated network setting, and our agents working on the network monitoring device

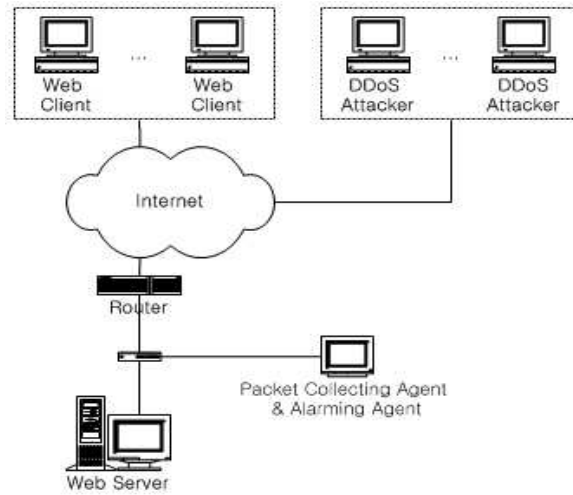


**Fig. 1.** A simulated Network Environment which consists of Web clients, DDoS attackers, Web Server (Victim), and our agents.

We tested our framework in the simulated network environment, as depicted in Figure 1, and measured TCP flag rates. The network traffic models were generated in two settings: the normal Web server without any attack and the Web server with DDoS attacks. For each network traffic setting, we changed two factors Simultaneous Connections (SC) and Requests per Connection (R/C) to get various Web traffic patterns. The SC indicates the number of HTTP connections at a given time, which approximates the number of users. The R/C represents the number of requests to be issued in a HTTP connection. In the experiment, we used 5, 10, 50, 100, 150, and 200 for SC and 1, 2, 5, and 10 for R/C. The sampling time  $t_d$  (sec) to compute TCP flag rates was 1 second.

#### 4.1 Normal Web server without any attack

The experimental results of normal Web traffic are illustrated in Figure 2. Even if SC ranges from 5 to 200, the results of TCP flag rates are almost identical.

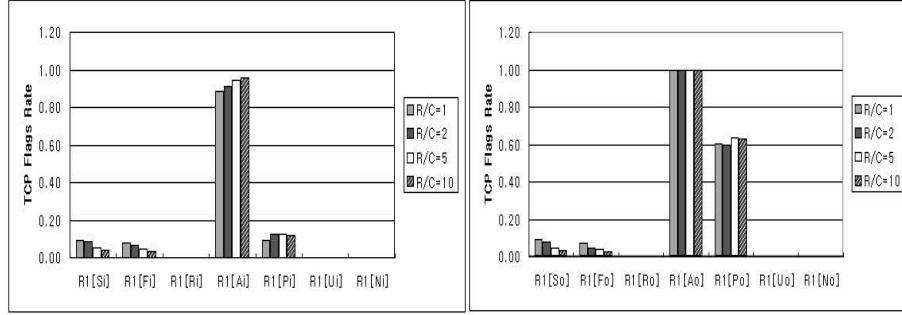


**Fig. 2.** TCP flag rates in the normal Web when SC=200.

We measured TCP flag rates for inbound and outbound network traffic, respectively. In both of inbound and outbound TCP flag rates, the rates of SYN and FIN were less than 0.1. On the other hand, the rate of an ACK flag was close to 1.0. This revealed the fact that most of the TCP packets set an ACK flag bit in their header for the purpose of sending an acknowledgement as a notification of receipt.

#### 4.2 Web server with DDoS attacks

Figure 3 presents the inbound and outbound TCP flag rates when SYN flooding attacks occur.

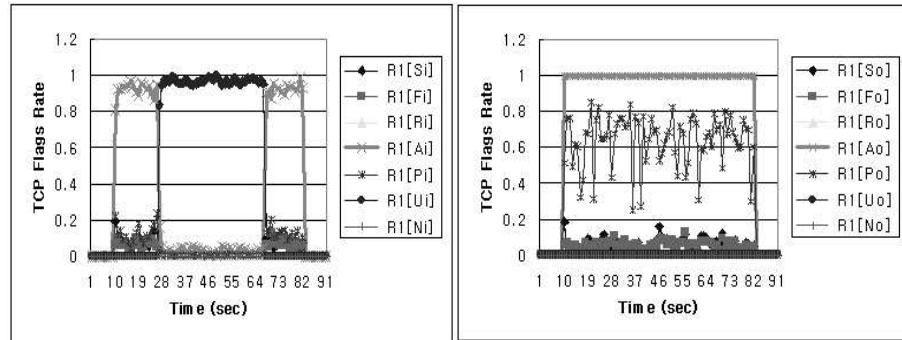


**Fig. 3.** TCP flag rates in the Web with SYN flooding attacks. The TFN2K was used to simulate SYN flooding attacks from 30 to 70 seconds.

$R_i[Ai]$  went down to about 0.0, due to the SYN's burst during the attack. This indicates that web traffic flow was blocked by enormous amount of SYN packets. On

the other hand,  $R_I[Si]$  and  $R_I[Ui]$  in the inbound flag rates drastically changed and went up to almost 1.0. The outbound TCP flag rates except  $R_I[So]$  were not affected by the attack at all. Since a victim followed the TCP three-way handshaking protocol, it replied to all SYN packets with SYN flags, if the SYN flooding attack was made on open ports. That's the reason why  $R_I[So]$  increased.

### 4.3 Learning Results

To construct compiled rules for our alarming agents, we used three machine learning algorithms: C4.5 [13], CN2 [1], and Bayesian classifier [4]. 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 [1], 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  $R_I[Si]$  and the ACK flag rate for inbound traffic  $R_I[Ai]$ . For the benchmark, we also computed the rate of SYN-FIN pair, which is a core of Wang's SYN flooding detection mechanism [15].

Using the three learning algorithms [5] and the training examples as inputs, we could get the compiled rules as described in Figure 4 and Figure 5.

```
MaxTrain: 719, MinTest: 0
C4.5 Trees:
  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
  (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. 4.** Learning results by TRA.

For the TRA, C4.5 indicated that the SYN flooding attacks occurred if  $R_I[Si]$  was greater than 0.4. The rules obtained by CN2, as shown in Figure 4, 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  $R_1[Si]$  was 0.98 given the class of “attacks.” The learning results for Wang’s work, as shown in Figure 5, were generated over the SYN-FIN pair,  $R_1[Fi]/R_1[Si]$ .

```

MaxTrain: 719, MinTest: 0
C4.5 Trees:
  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
  (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

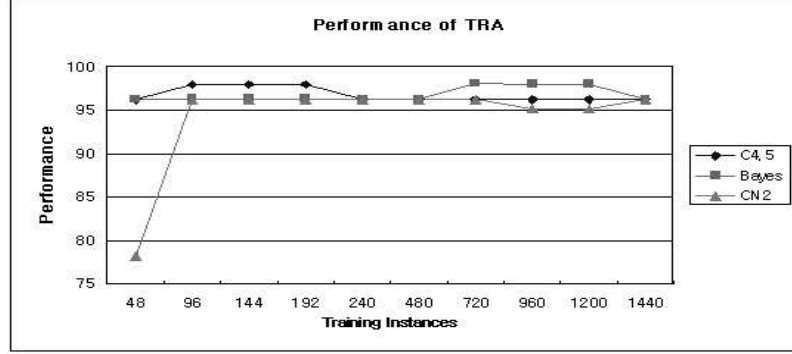
```

**Fig. 5.** Learning results by Wang’s work [15].

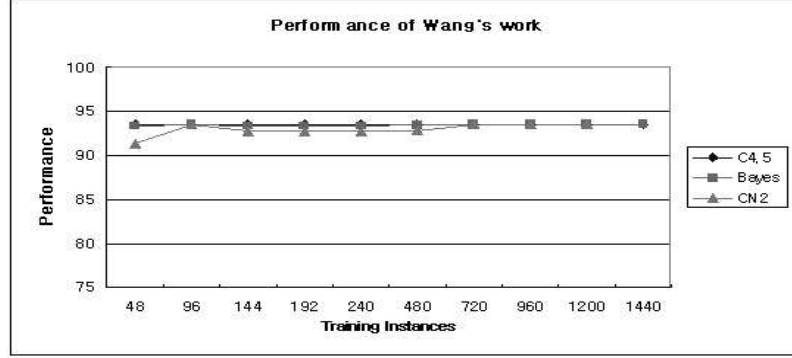
To evaluate the quality of various rule sets generated by different learning algorithms the performance obtained was expressed in terms of the ratio of {total number of alarms – (number of false alarms + number of missed alarms)} to the total number of alarms. The false alarm is defined as the alert turns on when the DDoS attack does not occur, and the missed alarm is defined as the alert does not turn on when the DDoS 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 Figure 6. In the traffic rate analysis, the best performance was achieved by the rules compiled using Bayesian classifier, as depicted in Figure 6 (a), when the training instances were 720. In the learning curve of the Wang’s work, as depicted in Figure 6 (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 seconds. In the testing network environment, the Simultaneous Connections (SC) were 7, 15, 40, 70, 130 and 160, the Requests per Connection (R/C) were 4, 12, 18, and 24, and the DDoS flooding attacks were made using three different time slots, i.e., four 10 seconds, two 10 seconds, and one 30 seconds ranging from 30 to 60 seconds. These combinations, thus, lead to eventually 72 different Web traffics including the DDoS attacks.



(a) Detection performance using the compiled rules in TRA



(b) Detection performance using the compiled rules in Wang's work

**Fig. 6.** The DDoS attacks detection performances using the resulting rules compiled by C4.5, Bayes, and CN2 learning algorithms in TRA and Wang's work.

We analyzed the performance results in Table 1 using the standard analysis of variance (ANOVA) method.

Table 1: Performances in TRA and Wang's work

| Methods     | Performances (%)       |
|-------------|------------------------|
| TRA         | 99.881321 +/- 0.316758 |
| Wang's work | 97.022569 +/- 1.791220 |
| ANOVA       | $f = 178.015076$       |

Since the computed value of  $f = 178.015076$  in ANOVA exceeds  $6.63 (= f_{.01,1,142})$ , 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 didn't 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.

## 5 Conclusions

We investigated the traffic rate analysis (TRA) as a traffic flow analysis mechanism and, using our TRA mechanism, analyzed TCP-based network flows under DDoS attacks. Further, we detected the DDoS network flooding attacks using the state-action rules compiled by machine learning algorithms, 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.

## References

1. Clark, P. and Niblett, T.: The CN2 Induction Algorithm. *Machine Learning Journal* 3(4) (1989) 261-283
2. Garber, L.: Denial-of-Service Attacks Rip the Internet, *IEEE Computer*, vol. 33(4) (2000) 12-17
3. Gil, T.M. and Poletto, M.: MULTOPS: a data-structure for bandwidth attack detection, In *Proceedings of the 10th USENIX Security Symposium* (2001) 23-38
4. Hanson, R., Stutz, J., and Cheeseman, P.: Bayesian Classification Theory. *Technical Report FIA-90-12-7-01*, NASA Ames Research Center, AI Branch (1991)
5. Holder, L.: ML v2.0: Machine Learning Program Evaluator, available on-line: <http://www-cse.uta.edu/~holder/ftp/ml2.0.tar.gz>
6. Houle, J.K., and Weaver, M.G.: Trends in Denial of Service Attack Technology, CERT Coordination Center (2001)
7. Kulkarni, A.B., Bush, S.F., and Evans, S.C.: Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics. TR176, GE Research Center (2001)
8. Lawrence Berkeley National Labs Network Research Group.: libpcap, available on-line: <http://ftp.ee.lbl.gov>
9. Li, M., and Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications, *Springer-Verlag* (1997)
10. Moore, D., Voelker, G.M., and Savage, S.: Inferring Internet Denial-of-Service Activity. In *Proceedings of the 10th USENIX Symposium* (2001) 9-22
11. Noh, S. and Gmytrasiewicz, P. J.: Towards Flexible Multi-Agent Decision-Making Under Time Pressure. In *Proceedings of IJCAI* (1999) 492-498
12. Packet Storm.: *Tribe Flood Network 2000 (TFN2K) DDoS tool*, available on-line: [http://packetstormsecurity.org/distributed/TFN2k\\_Analysis-1.3.txt](http://packetstormsecurity.org/distributed/TFN2k_Analysis-1.3.txt)
13. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers (1993)
14. Standard Performance Evaluation Corporation.: *SPECweb99 Benchmark*, available on-line: <http://www.spec.org/osg/web99>
15. Wang, H., Zhang, D., and Shin, K.G.: Detecting SYN Flooding Attacks. In *Proceedings of IEEE INFOCOM*, vol. 21(1) (2002) 1530-1539