PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks

Yoohwan Kim, *Member*, *IEEE*, Wing Cheong Lau, *Senior Member*, *IEEE*, Mooi Choo Chuah, *Senior Member*, *IEEE*, and H. Jonathan Chao, *Fellow*, *IEEE*

Abstract—Distributed Denial-of-Service (DDoS) attacks are a critical threat to the Internet. This paper introduces a DDoS defense scheme that supports automated online attack characterizations and accurate attack packet discarding based on statistical processing. The key idea is to prioritize a packet based on a score which estimates its legitimacy given the attribute values it carries. Once the score of a packet is computed, this scheme performs score-based selective packet discarding where the dropping threshold is dynamically adjusted based on the score distribution of recent incoming packets and the current level of system overload. This paper describes the design and evaluation of automated attack characterizations, selective packet discarding, and an overload control process. Special considerations are made to ensure that the scheme is amenable to high-speed hardware implementation through scorebook generation and pipeline processing. A simulation study indicates that PacketScore is very effective in blocking several different attack types under many different conditions.

Index Terms—Network level security and protection, performance evaluation, traffic analysis, network monitoring, security, simulation.

1 Background

One of the major threats to cyber security is Distributed Denial-of-Service (DDoS) attacks in which victim networks are bombarded with a high volume of attack packets originating from a large number of machines. The aim of such attacks is to overload the victim with a flood of packets and render it incapable of performing normal services for legitimate users. In a typical three-tier DDoS attack, the attacker first compromises relay hosts called agents, which in turn compromise attack machines called zombies that transmit attack packets to the victim. Packets sent from zombie machines may have spoofed source IP addresses to make tracing difficult [25].

DDoS attacks can be launched by unsophisticated casual attackers using widely available DDoS attack tools such as trinoo, TFN2K, Stachedraht, etc. Since an attack in February 2000 [9] that targeted several high profile Web sites, including Yahoo, CNN, eBay, etc., the frequency and magnitude of DDoS attacks has been increasing rapidly, making it a growing concern in our Internet-dependent society. According to a 2003 CSI/FBI Computer Crime and Security Survey [6], 42 percent of respondents of the survey had suffered from DDoS attacks, 28 percent reported financial losses due to DDoS attacks, and the average losses due to DDoS attacks had increased 4.8 times since the year 2002. Recently, the FBI listed a suspect in the Most Wanted list for the charge of launching a DDoS attack against a competitor's Web site [7].

The DDoS problem has attracted much attention from the research community recently. In our observation, there are three major branches of research in DDoS, namely, 1) attack detection, e.g., by monitoring protocol behavior [30], 2) attack traceback, e.g., by packet marking [29], and 3) attack traffic filtering as described below more in detail. The PacketScore scheme discussed in this paper belongs to group 3) attack traffic filtering. Research in attack traffic filtering can be roughly categorized into three areas based on the point of protection:

- **Source-initiated**: Source sites are responsible for guaranteeing that outgoing packets are attack-free. Examples include network ingress filters [8], disabling ICMP, or removing unused services to prevent computers from becoming attack agents, or filtering unusual traffic from the source [24]. However, the viability of these approaches hinges on voluntary cooperation among a majority of ingress network administrators Internet-wide, making these approaches rather impractical given the scale and uncontrollability of the Internet.
- **Path-based**: In this approach, only the packets following the correct paths are allowed [15]. Any packet with a wrong source IP for a particular router port is considered a spoofed packet and dropped, which eliminates up to 88 percent of the spoofed

- *Y. Kim is with the School of Computer Science, University of Nevada, Las Vegas, 4505 Maryland Parkway, Box 4019, Las Vegas, NV 89154. E-mail: yoohwan@cs.unlv.edu.*
- *W.C. Lau is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong. E-mail: wclau@ie.cuhk.edu.hk.*
- *M.C. Chuah is with the Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015. E-mail: chuah@cse.lehigh.edu.*
- *H.J. Chao is with the Department of Electrical and Computer Engineering, Polytechnic Univerisity, 6 MetroTech Center, Brooklyn, NY 11201. E-mail: chao@poly.edu.*

Manuscript received 15 Apr. 2005; revised 17 Jan. 2006; accepted 15 Feb. 2006; published online 4 May 2006.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0048-0405.

packets [4], [28]. In another approach [11], if the number of traveled hops is wrong for a source IP, the packet is dropped, thereby eliminating up to 90 percent of the spoofed packets. These approaches are considered practical, but they have a somewhat high probability of false negatives, i.e., falsely accepting attack packets. Apparently, when packets use unspoofed addresses, which is an emerging trend, none of these approaches works.

- **Victim-initiated**: The victim can initiate countermeasures to reduce incoming traffic. For example, in the pushback scheme [10], the victim starts reducing excessive incoming traffic and requests the upstream routers to perform rate reduction as well. There are other methods based on an overlay network [14], packet marking [18], [31], TCP flow filtering [17], [33] and statistical processing [19], [20], etc. Although victim-initiated protections are more desirable, some methods require changes in Internet protocols or are too expensive to implement.

The industry is adopting more practical approaches, 1) overprovisioning and 2) distributed access points [1]. However, both methods are not only expensive, but have also proven to be vulnerable in recent attacks. More commonly, ISPs rely on manual detection and blocking of DDoS attacks. Once an attack is reported, an offline fine-grain traffic analysis is performed by a subject-matter expert to identify and characterize the attack packets. New filtering rules on access control lists are then constructed and installed manually on the routers. But, the need for human intervention results in poor response time and fails to protect the victim before severe damages are realized. Furthermore, the expressiveness of existing rule-based filtering is too limited, as it requires an explicit specification of all types of packets to be discarded.

The PacketScore scheme has been proposed recently by the authors of this paper [3], [19]. One of the key concepts in PacketScore is the notion of "Conditional Legitimate Probability" (CLP) based on Bayesian theorem. CLP indicates the likelihood of a packet being legitimate by comparing its attribute values with the values in the baseline profile. Packets are selectively discarded by comparing the CLP of each packet with a dynamic threshold. The concept of using a baseline profile with Bayesian theorem has been used previously in anomaly-based IDS (Intrusion Detection System) applications [22], where the goals are generally attack detection rather than real-time packet filtering. In this research, we extend the basic concept to a practical real-time packet filtering scheme using elaborate processes. In this paper, we describe the PacketScore operations for single-point protection, but the fundamental concept can be extended to a distributed implementation for core-routers.

The rest of this paper is organized as follows: In Section 2, we describe the concept of Conditional Legitimate Probability (CLP). In Section 3, we focus on the profiling of legitimate traffic characteristics. In Section 4, score assignment to packets, selective discarding, and overload control are described. In Section 5, an integrated process combining Sections 2, 3, and 4 is described. In Section 6, we evaluate the performance of the standalone packet filtering scheme. Section 7 addresses some of the important issues related to the PacketScore scheme. The paper concludes in Section 8 with the direction of future investigation.

2 Conditional Legitimate Probability

The most challenging issue in blocking DDoS attacks is to distinguish attack packets from legitimate ones. To resolve this problem, we utilize the concept of *Conditional Legitimate Probability* (CLP) for identifying attack packets probabilistically. CLP is produced by comparing traffic characteristics during the attack with previously measured, legitimate traffic characteristics. The viability of this approach is based on the premise that there are some traffic characteristics that are inherently stable during normal network operations of a target network.

We named this scheme *PacketScore* because CLP can be viewed as a score which estimates the legitimacy of a suspicious packet. We will use the terms *CLP* and *score* interchangeably. By taking a score-based filtering approach, the prioritization of different types of suspicious packets is possible and we can avoid the problems of conventional binary rule-based filtering discussed in Section 1. The ability to prioritize becomes even more important when a full characterization of attack packets is not feasible. By dynamically adjusting the cutoff score according to the available traffic capacity of the victim, our approach allows the victim system to accept more potentially legitimate traffic. In contrast, once a rule-based filtering scheme is configured to discard specific types of packets, it does so regardless of the victim network's available capacity.

To formalize the concept of CLP, we consider all the packets destined for a DDoS attack target. Each packet would carry a set of discrete-value attributes $A, B, C, \ldots$. For example, A might be the protocol type, B might be the packet size, C might be the TTL values, etc. We defined $\{a_1, a_2, a_3, \ldots\}$ as the possible values for attribute A, $\{b_1, b_2, b_3, \ldots\}$ as the possible values for attribute B, and so on. During an attack, there are N_n legitimate packets and N_a attack packets arriving in T seconds, totaling N_m.

$$N_m = N_n + N_a,$$
$$(m \text{ for } m\text{easured}, n \text{ for } n\text{ormal, and } a \text{ for } a\text{ttack}).$$

We define $C_n(A = a_i)$ as the number of legitimate packets with attribute value a_i for attribute A. Therefore,

$$\begin{aligned} N_n &= C_n(A = a_1) + C_n(A = a_2) + \ldots + C_n(A = a_i) + \ldots \\ &= C_n(B = b_1) + C_n(B = b_2) + \ldots + C_n(B = b_i) + \ldots \\ &\ldots . \end{aligned}$$

Likewise,

$$\begin{aligned} N_m &= C_m(A = a_1) + C_m(A = a_2) + \ldots + C_m(A = a_i) + \ldots \\ &= C_m(B = b_1) + C_m(B = b_2) + \ldots + C_m(B = b_i) + \ldots \\ &\ldots \end{aligned}$$

$$\begin{aligned} N_a &= C_a(A = a_1) + C_a(A = a_2) + \ldots + C_a(A = a_i) + \ldots \\ &= C_a(B = b_1) + C_a(B = b_2) + \ldots + C_a(B = b_i) + \ldots \\ &\ldots \end{aligned}$$

P_n, the ratio or the probability of attribute values among the legitimate packets, is defined as follows:

$$P_n(A = a_i) = C_n(A = a_i)/N_n, \ where\ i = 1, 2, \ldots$$
$$\left(\sum_i P_n(A = a_i) = 1\right)$$
$$P_n(B = b_i) = C_n(B = b_i)/N_n, \ where\ i = 1, 2 \ldots$$
$$\left(\sum_i P_n(B = b_i) = 1\right)$$
$$\ldots.$$

Similarly, $P_a(A = a_i)$ and $P_m(A = a_i)$ are defined as follows:

$$P_a(A = a_i) = C_a(A = a_i)/N_a,$$
$$P_a(B = b_i) = C_a(B = b_i)/N_a, \quad \ldots$$
$$P_m(A = a_i) = C_m(A = a_i)/N_m,$$
$$P_m(B = b_i) = C_m(B = b_i)/N_m, \quad \ldots.$$

Joint probability of multiple attributes is defined as:

$$P_n(A = a_i, B = b_j, \ldots) = C_n(A = a_i, B = b_j, \ldots)/N_n$$
$$P_a(A = a_i, B = b_j, \ldots) = C_a(A = a_i, B = b_j, \ldots)/N_a$$
$$P_m(A = a_i, B = b_j, \ldots) = C_m(A = a_i, B = b_j, \ldots)/N_m.$$

The Conditional Legitimate Probability (*CLP*) is defined as the probability of a packet being legitimate given its attributes:

$$\boldsymbol{CLP(packet\ p) = P(packet\ p\ is\ legitimate \mid p's\ attribute\ A = a_p, attribute\ B = b_p \ldots).}$$

According to Bayes' theorem, the conditional probability of an event E to occur given an event F is defined as:

$$P(E|F) = \frac{P(E \cap F)}{P(F)}.$$

Therefore, CLP can be rewritten as follows:

$$\begin{aligned} CLP(p) &= \frac{P((p = legitimate) \cap (A = a_p, B = b_p, \ldots))}{P(A = a_p, B = b_p, \ldots)} \\ &= \frac{N_n \times P_n(A = a_p, B = b_p, \ldots)/N_m}{N_m \times P_m(A = a_p, B = b_p, \ldots)/N_m} \\ &= \frac{N_n \times P_n(A = a_p, B = b_p, \ldots)}{N_m \times P_m(A = a_p, B = b_p, \ldots)}. \end{aligned} \tag{1}$$

If the attributes are independent,

$$P(A = a_p, B = b_p, \ldots) = P(A = a_p) \times P(B = b_p) \times \ldots,$$

then (1) can be further rewritten as:

$$CLP(p) = \frac{N_n \times P_n(A = a_p) \times P_n(B = b_p) \times \ldots}{N_m \times P_m(A = a_p) \times P_m(B = b_p) \times \ldots}. \tag{2}$$

One of the terms in (2), e.g., $P_n(A = a_p)/P_m(A = a_p)$ is called a partial score. If the percentage of an attribute value a_p is the same in both attack traffic and legitimate traffic, then $P_m(A = a_p) = P_n(A = a_p)$ and the partial score is 1. However, if a_p is found more frequent in legitimate traffic than in attack traffic, then $P_m(A = a_p) < P_n(A = a_p)$, resulting in a partial score greater than 1. On the other hand, if a_p appears more frequently in attack traffic than in the legitimate traffic, the partial score becomes less than 1. It is known that some IP header fields are not evenly distributed over all possible values; rather a unique distribution pattern exists for a site [12], [21], [23]. If attackers do not know the attribute value distribution in legitimate traffic, they are likely to generate random or wrongly guessed attribute values, which makes most of the attack packets to have smaller scores than legitimate packets' scores.

In converting (1) into (2), it is assumed that the attributes are independent. However, some packet attributes are not independent. For example, being a TCP packet implies an 80-90 percent chance of having port 80, rather than a 1/65,536 chance. While we leave the investigation on the independence assumption as a future work, it seems to work in practice because the CLP (p) in (2) is still a good metric for packet prioritization. A large portion of DDoS attack packets get lower CLPs because P_m becomes larger than P_n for the dominant attribute values in the attack. As long as we can assign lower scores to the majority of attack packets, the assumption of independence is not essential to PacketScore operation.

3 Estimating Legitimate Traffic Distribution

Equation (2) shows that we can calculate the probability of a packet's legitimacy by observing the probabilities of the attribute values in legitimate traffic (P_n) and in total traffic (P_m). However, it is practically impossible to know how many packets are legitimate during the attack period, let alone the number of legitimate packets bearing a particular attribute value. For that reason, we utilize an estimate P'_n in place of true P_n. The estimate P'_n is called a *nominal profile* and is collected in advance during normal operations.

A nominal traffic profile consists of single and joint distributions of various packet attributes that are considered unique for a site. Candidate packet attributes from IP headers are:

1. packet size,
2. Time-to-Live (TTL) values,
3. protocol-type values, and
4. source IP prefixes.

Those from TCP headers are:

5. TCP flag patterns and
6. *server* port numbers, i.e., the smaller of the source port number and the destination port number.

Server port number is more stable than sort/destination port numbers because most of the well-known port numbers are small numbers (e.g., below 1,024) and a large portion of Internet traffic uses the well-known port numbers. To increase the number of attributes, we can employ joint distributions of the fraction of packets having various combinations, such as:

7. <packet-size and protocol-type>,
8. <server port number and protocol-type>, and
9. <source IP prefix, TCP flags and packet size>, etc.

Joint distributions often better represent the uniqueness of the traffic distribution for a site, and are harder to guess for the attackers. As many different combinations of single attributes as needed may be used while the storage space permits.

TABLE 1
An Example of a Nominal Profile

TTL Value	Period 1	Period 2	Period 3	Period 4	**Profile**
1	0.5%	0.8%	**1.1%**	0.3%	**1.1%**
2	0.7%	0.5%	0.6%	**0.8%**	**0.8%**
3	3.0%	**3.5%**	2.4%	2.9%	**3.5%**
....	...	...	...	...	**...**
255	**1.3%**	1.2%	0.9 %	1.2%	**1.3%**

TABLE 2
Profile Storage Requirements for Different Iceberg Selection Methods

Threshold Type	**Storage Requirements (Kbytes)**	**Relative Storage Requirements (Kbytes)**
Static threshold	13.6	1.0
Adaptive 90% threshold	76.0	5.6
Adaptive 95% threshold	127.8	9.4
Adaptive 99% threshold	288.3	21.2

During the nominal profiling period, the number of packets with each attribute value is counted and the corresponding ratio is calculated. However, if the profile is created only once during the profiling period, temporally localized traffic characteristics may be misrepresented. To avoid it, the profiling period is broken into subperiods, then the ratios are measured for each subperiod, and one value representing all the subperiods is selected. The principle of PacketScore is to punish the traffic whose attribute value ratio is higher than in profile. Therefore, to accommodate an occasional surge of particular attribute values in legitimate traffic, the highest ratio among the periodic ratios is selected. This strategy has little impact on blocking attack traffic while giving the legitimate traffic a safety margin. Table 1 illustrates this process with an example of TTL values. The boldface values are the highest ratios observed among the periodic values, which are then stored in the profile. If the variance among the periodic ratios is too great to be reliable, it is possible to include only those attribute values with low variance to have a more stable profile.

3.1 Profile Structure

Due to the number of attributes to be incorporated in the profile and the large number of possible attribute values of each attribute, especially for the joint attributes, an efficient data structure is required to implement the profile. For example, the attribute values for TTL are $0, 1, 2, \ldots 255$, thus there are 256 possible attribute values. Each attribute value has a ratio in the profile as illustrated in Table 1 (e.g., 1.1 percent for TTL value 1). To reduce the storage space, we use *iceberg-style* profiles [2], in which only the most frequently occurring attribute values are stored along with their ratio.

Two approaches are possible for selecting the icebergs, i.e., by static threshold and by adaptive threshold. In the static threshold approach, the profile only includes those attribute values which appear more frequently than a preset threshold ratio, say *x percent*. For the attribute values which are absent from the iceberg-style profiles, we use the upper bound (*x percent*) as their ratios. For example with Table 1, if the preset threshold is 1 percent, TTL value 2 is removed from the profile, and TTL value 2 is considered to have 1 percent share in the traffic during the scoring process later. In the adaptive threshold approach, the most frequently appearing attribute values that constitute a preset coverage of the traffic, e.g., *95 percent*, are selected. The corresponding cutoff threshold *y percent* for the given coverage serves as the adaptive threshold, which is also used as the default ratio for the absent items. With such iceberg-style profiles, the nominal profile can be kept to a manageable size. Joint attributes experience an additional problem of combinatorial explosion, so buckets of preset ranges are used instead of the tuples of raw attribute values.

Typical storage requirements for storing six single attributes and two joint attributes are shown in Table 2 for different threshold methods. For the static threshold, we used 0.01, 0.001, and 0.0001, respectively, for single attribute, two-dimensional and three-dimensional joint attributes. Although the static threshold method produces the most space-efficient profiles, adaptive thresholds are considered more robust against wide variations in traffic trace.

The iceberg-based profile is similar to [5] where its unidimensional cluster and multidimensional clusters are comparable to our single attributes and joint attributes, respectively. However, the approaches and purposes are different. While [5] provides valuable information for the network administrator by identifying the dominant traffic type in any combination of attributes within the current traffic, our goal is to take a comprehensive snapshot of the traffic for future reference. The iceberg approach is merely used to reduce the storage requirement rather than traffic aggregation. Since the profiling in PacketScore is based on packet counting and does not require aggregation, it can be done very rapidly, i.e., in a matter of seconds as opposed to minutes [5] for similar size trace data.

3.2 Traffic Profile Stability

PacketScore depends on the stability of the traffic profile for estimating P_n. It has been known that for a given subnet, there is a distinct traffic pattern in terms of packet attribute value distribution for a given time and/or given day [12], [21], [23]. In general, the nominal traffic profile is believed to be a function of time which exhibits periodic, time-of-day, and day-of-the-week variations as well as long-term trend changes.

To further verify traffic profile stability, we conducted an analysis with the packet trace data available from NLANR packet trace archives [26]. All trace data were collected for 90 seconds from 17 sites within the US with the link speed ranging from OC-3 to OC-48. We randomly selected the four sites in Table 3 and total of 49 trace files were downloaded for analysis.

Table 4 shows general statistics for 10 selected traces.

TABLE 3
Trace Data Sites from NLANR

Site	Location	Link Speed	Number of traces downloaded
AIX	NASA Ames to MAE-West	OC12 (655 Mbps)	28
AMP	AMPATH, Miami, Florida	OC12 (655 Mbps)	7
MEM	University of Memphis	OC3 (155 Mbps)	7
PSC	Pittsburgh Supercomputing Center	OC48 (2.5 Gbps)	7

TABLE 4
Statistics of Some Download Traces

File Name	Date	Time (24H)	File Size (MB)	Total # packets (1000)	Avg # packets (1000)	Band-Width (Mbps)	Traffic composition		
							TCP	UDP	ICMP
MEM-1127750202-1.tsh	9/26/05 Mon	9:12	8.4	197	2.381	11.889	0.926	0.06	0.008
MEM-1127791306-1.tsh	9/26/05 Mon	21:10	7.3	171.343	2.066	13.457	0.899	0.086	0.009
MEM-1124810381-1.tsh	8/23/05 Tue	8:28	9.9	231.021	2.787	14.246	0.939	0.053	0.005
MEM-1125415231-1.tsh	8/30/05 Tue	8:35	6.5	151.524	1.827	7.609	0.834	0.147	0.012
AIX-1127750202-1.tsh	9/26/05 Mon	9:31	0.5	12.116	0.149	0.417	0	0	0
AIX-1127835595-1.tsh	9/27/05 Tue	8:42	0.5	12.397	0.158	0.267	0.002	0	0.001
AMP-1127747110-1.tsh	9/26/05 Mon	8:13	31	739.1	9.157	50.311	0.85	0.14	0.003
AMP-1127836180-1.tsh	9/27/05 Tue	8:58	21	496.516	6.14	36.308	0.922	0.075	0.002
PSC-1127747111-1.tsh	9/26/05 Mon	8:38	163	3799.19	47.051	276.604	0.849	0.132	0.018
PSC-1127836180-1.tsh	9/27/05 Tue	9:37	136	3171.69	39.57	233.482	0.864	0.113	0.022

A quick examination of Table 4 revealed that each site had a distinct traffic composition. Especially, we observed that the traffic in AIX is mostly GRE rather than TCP or UDP. For each trace, a profile was created using a 99 percent adaptive coverage method over a series of 10-second windows. We employed a joint attribute composed of three attributes (Protocol type, server port, and packet size) because joint attributes are believed more unique per site than single attributes. For an objective comparison of two profiles, we defined the stability metric S as follows:

$$S = C \times D.$$

C indicates how many items are common to both profiles:

$$C = \frac{(number\ of\ common\ items\ in\ both\ profiles = n)}{(number\ of\ total\ items\ in\ both\ profiles)}.$$

D indicates how closely these common items are related. For example, the two profiles may have 3 percent versus 1.7 percent, or 3.2 percent versus 2.9 percent for a given attribute value. Obviously, the latter shows a stronger resemblance. For an item i that is in both profiles, the comparison ratio is defined as the smaller $(R_{small}(i))$ of two values divided by the larger $(R_{large}(i))$ of two values. D is defined as the average of the comparison ratios. When the comparison ratio is too small, e.g., below 0.01, we may consider it 0.

$$D = \frac{\sum_{i=1}^{n} \frac{R_{small(i)}}{R_{l\arg e}(i)}}{n}.$$

S varies between 0 and 1, from no stability to perfect stability. When two profiles are exactly the same ($C = 1$ and $D = 1$), $S = 1$. When there are no common items (C = 0) or the comparison ratios are zero for all of the common items ($D = 0$), S becomes zero. For a better comparison of the stability at low S values, we define SL as a log version of S:

$$SL = \log_{10} 10C \times \log_{10} 10D, \quad SL = 0, \text{ if } C \leq 0.1 \text{ or } D \leq 0.1.$$

For stability analysis, one reference profile was compared with other profiles and the SL value is calculated. The selected reference profile is from the MEM trace of Tuesday, 27 September 2005, 8:49 a.m. We investigated the following questions:

1. Are the profiles similar for the 10-second windows within a 90-second trace?
2. Are the profiles similar between mornings and evenings at the same site?
3. Are the profiles similar over multiple weeks at the same time of a specific day? (e.g., 8:00 p.m. every Tuesday)
4. Are the profiles different at different sites at the same date and time?

The analysis results are shown in Fig. 1. In Fig. 1a, the profile of the first 10-second window was compared with other 10-second window profiles. It indicates that there is strong correlation among the 10-second windows, thus validating the SL metric. Note that the absolute value of the SL metric is not very useful by itself as its main purpose is for comparison. For example, we can understand that the stability within one trace in Fig. 1a is stronger than the stability for multiple days in Fig. 1c.

Fig. 1b compared seven profiles from 26 September 2005 to 2 October 2005 at approximately 9:00 a.m. (morning) and 8:00 p.m. (evening) each day. It indicates that there is moderate correlation among the daily profiles, although weaker than within the same trace. It should be noted that when the Tuesday profile is compared with itself, the $SL = 1$. It also shows that there is a higher correlation for

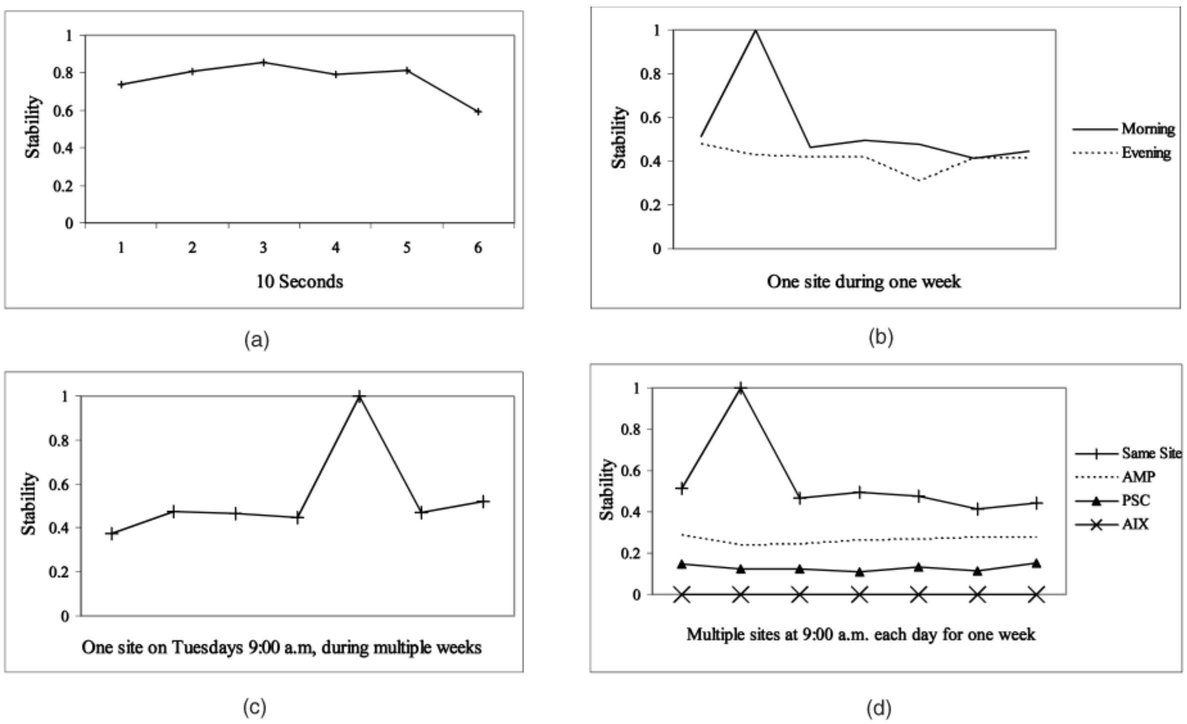


Fig. 1. Traffic profile stability comparisons. (a) Consecutive 10-second windows. (b) Seven consecutive days in a week. (c) Seven consecutive Tuesdays. (d) Four different sites for seven consecutive days in a week.

the same time of day (approximately 9:00 a.m.) than at a different time of day (approximately 8:00 p.m.). Fig. 1c compared the profiles for seven Tuesdays at the same time of day (approximately 9:00AM) from 23 August 2005 to 11 October 2005. Although it spans seven weeks, it still shows a similar correlation to the short-term profiles of 9:00 a.m. as in Fig. 1a. These seven Tuesday morning profiles are slightly closer than the evening profiles in Fig. 1b. However, in Fig. 1d, when compared with other sites, the SL is much lower, showing a much weaker correlation.

In summary, we observe that traffic profiles are most similar among the traffic at the same day at the same time, even over multiple weeks. A traffic profile is still very similar for a different time or day within a site, although stability is slightly lower than the same time of day. On the other hand, there are considerable differences among different sites, so it is necessary to keep separate profiles for each site. By tracking S, we can determine the stability of the profiles for different times or days and can also decide how many profiles are needed. Unless there is a significant difference between profiles, we may use one uniform profile to minimize the maintenance effort. In the above example of Fig. 1b, the morning and evening profiles are quite similar, so it will be unnecessary to keep two profiles. A site administrator may use a guideline such as "Keep a single merged profile if the S values from two profiles are within x percent for every data point." The result in this section provides a good intuition on profile stability, but it is by no means comprehensive. A more exhaustive analysis will be left as a future study.

4 Scoring Packets

4.1 Log-Version CLP

To make it more suitable for real-time processing, we can convert floating-point division/multiplication operations into subtraction/addition operations. By using the logarithmic version of (2) as shown below:

$$\log[CLP(p)] = \left\{ \begin{array}{l} [\log(P_n(A = a_p)) - \log(P_m(A = a_p))] + \\ [\log(P_n(B = b_p)) - \log(P_m(B = b_p))] + \\ [\log(P_n(C = c_p)) - \log(P_m(C = c_p))] + \\ \ldots \end{array} \right\}, \tag{3}$$

we can construct a scorebook for each attribute that maps different values of the attribute to a specific partial score. For instance, the partial score of a packet with attribute A equal to a_p is given by $[\log(P_n(A = a_p)) - \log(P_m(A = a_p))]$. Then, we can sum up the partial scores of different attributes to yield the logarithm of the overall score of the packet. Since N_n/N_m in (2) is constant for all packets within the same observation period, it can be ignored when comparing and prioritizing packets based on their CLP values. The nominal profile P'_n is established in advance and used instead of the true P_n. During the attack period, the same form of traffic profile is collected to make P_m, and those two profiles are combined to create a CLP score table. Table 5 gives an example of such a scorebook for one attribute. When a value is not stored because it is under the iceberg threshold, the threshold value is used instead.

Scoring a packet is equivalent to looking up the scorebooks, e.g., the TTL scorebook, the packet size scorebook, the protocol type scorebook, etc. After looking up the multiple scorebooks, we add up the matching CLP entries in a log-version scorebook. This is generally faster than multiplying the matching entries in a regular scorebook. The small speed improvement from converting a multiplication operation into an addition operation is particularly useful because every single packet must be scored in real-time. This speed improvement becomes more beneficial as the number of scorebooks increases. On the other hand,

TABLE 5
An Example of a Partial Scorebook for One Attribute (e.g., TTL)

TTL Value	Nominal Profile (P'_n)	Measured profile (P_m)	CLP (=P'_n / P_m)	Log version CLP (=log P'_n – log P_m)
1	0.01	0.09	0.111 = 0.01 / 0.09	-0.954 = *log* 0.01 – *log* 0.09
2	0.10	0.25	0.4 = 0.10 / 0.25	*-0.398* = *log* 0.10 – *log* 0.25
3	0.03	0.01	3.0 = 0.03 / 0.01	*0.477* = *log* 0.03 – *log* 0.01
…	…	…	…	…
255	0.10	0.20	0.5 = 0.10 / 0.20	*-0.301* = *log* 0.10 – *log* 0.20

generating a log-version scorebook may take longer than a regular scorebook generation. However, the scorebook is generated only once at the end of each period and it is not necessary to observe every packet for scorebook generation; thus, some processing delay can be allowed. Furthermore, scorebook generation can be easily parallelized using two processing lines, which allows complete sampling without missing a packet.

4.2 Decoupling the Profile Update and Scoring

According to (3) the current packet attribute distributions (P_m) have to be updated constantly whenever a packet arrives. To make wire-speed per-packet score computation possible, we decoupled the updating of packet attribute distribution from that of score computation to allow them to be conducted in parallel, but at different time periods. With such decoupling, the score computation is based on a snapshot of *recently* measured histograms. To be more specific, a frozen set of recent profiles at time period T_1 is used to generate a set of *scorebooks* which is used to score the packets arriving at the next time period, T_2. Packets arriving at T_2 also generate a new profile and scorebook to be used for time period T_3. The time-scale of period T_i is longer than the per-packet arrival time-scale. It can be configured to a fixed length or until the detection of a significant change in the measured traffic profile.

This decoupling introduces a small challenge in catching up with attack profile change. In most cases, the traffic characteristics in adjacent periods are very similar, but during a rapidly changing attack, this assumption may be inaccurate. As a result, the scorebook at T_i does not represent the true scorebook at T_{i+1}, and the PacketScore performance degrades. This can be resolved easily by reducing the time-scale of T_i or by using a packet number-based period instead of a time-based one. We will discuss this in detail in Section 6.7.

4.3 Selective Packet Discarding

Once the score is computed for a packet, selective packet discarding, and overload control can be performed using the score as the differentiating metric. Since an exact prioritization would require offline, multiple-pass operations, e.g., sorting and packet buffering, we take the following alternative approach. First, we maintain the cumulative distribution function (CDF) of the scores of all incoming packets in time period T_i. Second, we calculate the cut-off threshold score $\mathbf{Thd}$ as follows which is illustrated in Fig. 2.

- Total current incoming traffic at period $T_i = \psi_\mathrm{i}$,
- Acceptable traffic at period $T_i = \phi_i$,
- The fraction of traffic permitted to pass = $1 - \Phi_\mathrm{i} = \phi_i/\psi_\mathrm{i}$, and
- The $\mathbf{Thd_{i+1}}$ that satisfies CDF $(\mathbf{Thd_{i+1}}) = \Phi_\mathrm{i}$.

Third, we discard the arriving packets in time period T_{i+1} if its score value is below the cut-off threshold $\mathbf{Thd_{i+1}}$. At the same time, the packets arriving at T_{i+1} create a new CDF, and a new $\mathbf{Thd_{i+2}}$ is calculated for T_{i+2}.

5 The Integrated Process

Fig. 3 depicts the integrated operation between CLP computation and the determination of a dynamic discarding threshold for CLP. A load-shedding algorithm, such as the one described in [13], is used to determine the amount (Φ) of suspicious traffic arriving that needs to be discarded in order to keep the utilization of the victim below a target value. Typical inputs to a load-shedding algorithm include current utilization of the victim, maximum (target) utilization allowed for the victim, and the current aggregated arrival rate of suspicious traffic. Once the required packet-discarding percentage (Φ) is determined, the corresponding CLP discarding threshold, *Thd*, is determined from a recent snapshot of the CDF of the CLP values. The snapshot is updated periodically or upon significant changes in the packet score distribution. The adjustment of the CLP discarding threshold is done on a time-scale which is considerably longer than the packet arrival time-scale.

The entire PacketScore process can be best performed in a pipelined approach as discussed in Section 4.2 in which time is divided into fixed intervals, and each operation is performed based on the snapshot of the previous period. Specifically, the following three operations are performed in pipeline when a packet arrives:

1. Incoming packet profiling:
   - Packets are observed to update P_m.

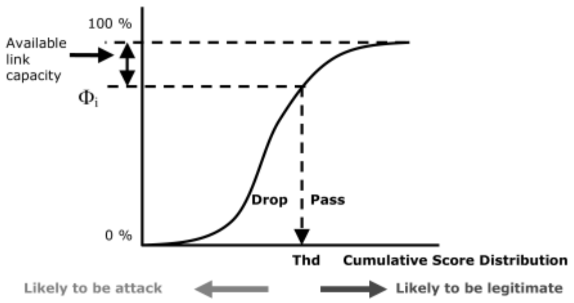


Fig. 2. Selective packet discarding.

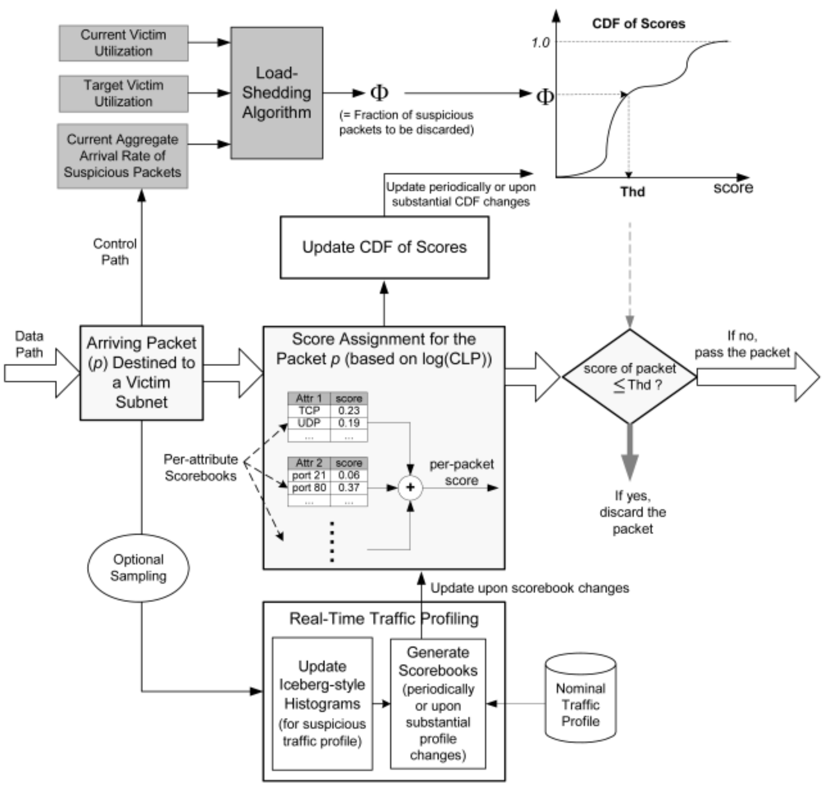

Current Victim Utilization
Target Victim Utilization
Current Aggregate Arrival Rate of Suspicious Packets
Load-Shedding Algorithm
Φ
(= Fraction of suspicious packets to be discarded)
CDF of Scores
1.0
Φ
Thd
score
Update periodically or upon substantial CDF changes
Update CDF of Scores
Control Path
Data Path
Arriving Packet (p) Destined to a Victim Subnet
Score Assignment for the Packet p (based on log(CLP))
Attr 1 | score
TCP | 0.23
UDP | 0.19
Attr 2 | score
port 21 | 0.06
port 80 | 0.37
Per-attribute Scorebooks
per-packet score
score of packet ≤ Thd ?
If no, pass the packet
If yes, discard the packet
Optional Sampling
Update upon scorebook changes
Real-Time Traffic Profiling
Update Iceberg-style Histograms (for suspicious traffic profile)
Generate Scorebooks (periodically or upon substantial profile changes)
Nominal Traffic Profile

Fig. 3. Packet differentiation and overload control.

- At the end of the period, P'_n/P_m is calculated and scorebooks are generated.

2. Scoring:
   - The packets are scored according to the most recent scorebooks.
   - At the end of the period, CDF is generated and the cut-off threshold is calculated.
3. Discarding:
   - The packets are scored according to the most recent scorebooks.
   - The packet is discarded if its score is below the cut-off threshold score.

It is also important to reemphasize that, while CLP-computation is always performed for each incoming packet, selective packet discarding is only performed when the system is operating beyond its safe (target) utilization level. Otherwise, it will set Φ to zero.

5.1 Illustrative Example of a Defense against a SQL Slammer Worm Attack

In an example of how a DDoS attack can be stopped by the PacketScore scheme, Fig. 4 shows the selective discarding of packets generated by the recent SQL Slammer attack (also known as the Sapphire Worm). The attack is comprised of UDP packets with a destination of port number 1434 and a packet size ranging from 371 to 400 bytes. In Fig. 4, each profile contains three iceberg-style histograms, that is, for port number, protocol type, and packet size.

During the attack, there is a surge of UDP packets with a destination port number 1434. As the fraction of packets having this destination exceeds the preset iceberg threshold (3 percent in this example) during the attack, port 1434 is recorded in the attack profile. On the other hand, this port number does not appear in the nominal profile because 1434 is not a frequently appearing port number. In the scorebook for the destination port number attribute, the partial score for destination port number 1434 is given by $[\log(0.03) - \log(0.4)] = -1.12$, where the iceberg threshold, 3 percent, i.e., 0.03, is used as a default ratio for noniceberg items. Similarly, the partial scores of a worm packet for the protocol-type and packet-size attributes are given by $[\log(0.1) - \log(0.5)] = -0.7$, and $[\log(0.05) - \log(0.4)] = -0.9$, respectively. Therefore, the score of a worm packet is given by

$$-(1.12 + 0.7 + 0.9) = -2.72.$$

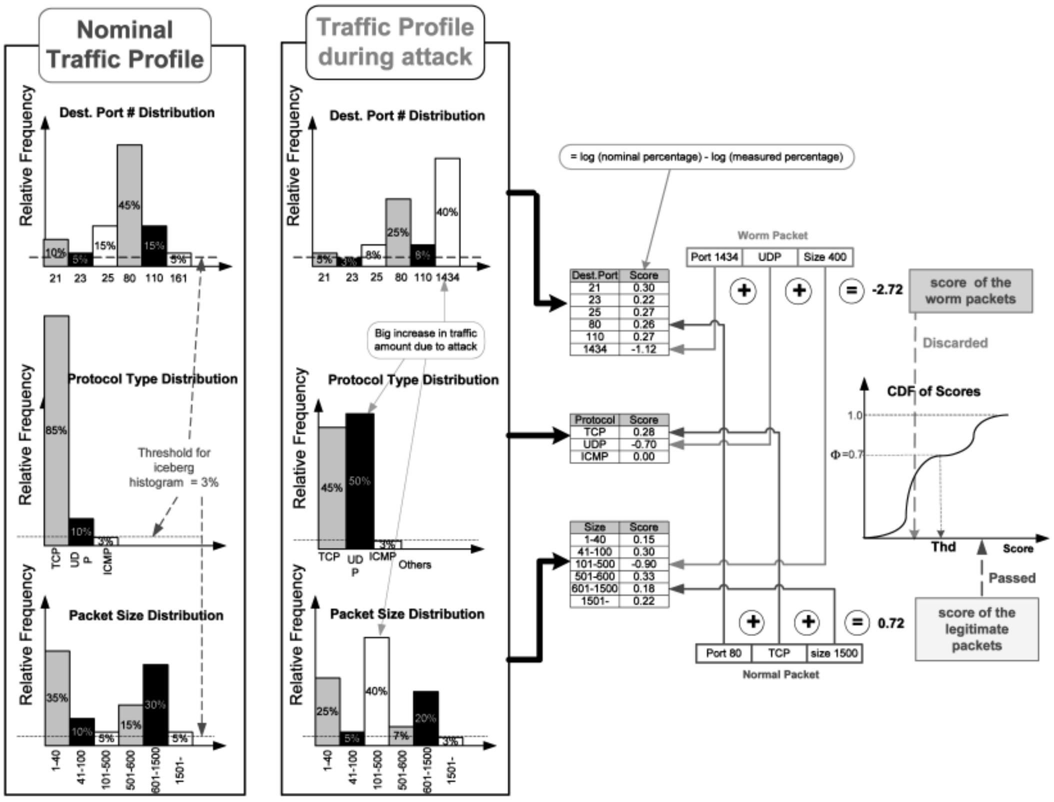
In comparison, the score of a legitimate 1,500-byte TCP packet carrying HTTP traffic destined for port 80 is given by

$$\{[\log(0.45) - \log(0.25)] + [\log(0.85) - \log(0.45)] + [\log(0.3) - \log(0.2)]\} = (0.26 + 0.28 + 0.18) = +0.72.$$

Assuming the cut-off threshold (**Thd**) is +0.5, all the worm packets will be discarded because the score of the worm packets is smaller than **Thd**. The legitimate 1,500-byte TCP packets carrying HTTP traffic, however, are allowed to pass through as their score is higher than **Thd**.

6 Performance Evaluation

We evaluated the performance of the PacketScore scheme via simulation. The simulation programs were written in C++. The profiler program that generated the nominal


Nominal Traffic Profile
Traffic Profile during attack
Dest. Port # Distribution
Protocol Type Distribution
Packet Size Distribution
Relative Frequency
Threshold for iceberg histogram = 3%
Big increase in traffic amount due to attack
= log (nominal percentage) - log (measured percentage)
Worm Packet
Port 1434 | UDP | Size 400
-2.72
score of the worm packets
Discarded
CDF of Scores
Thd
Score
Passed
score of the legitimate packets
0.72
Port 80 | TCP | size 1500
Normal Packet

Dest.Port	Score
21	0.30
23	0.22
25	0.27
80	0.26
110	0.27
1434	-1.12

Protocol	Score
TCP	0.28
UDP	-0.70
ICMP	0.00

Size	Score
1-40	0.15
41-100	0.30
101-500	-0.90
501-600	0.33
601-1500	0.18
1501-	0.22

Fig. 4. Discarding SQL Slammer Worm attack packets.

profile from the packet trace consumed about 1.5 MB of memory space. For each 10-second window comprising about 50,000 packets, the program executed for approximately 0.5 seconds on a 1.5 GHz Intel Pentium PC. The PacketScore filtering program read the nominal profile and packet traces, generated the attack packets, created scorebooks, and selected the packets to drop. It consumed about 40MB of memory space, and executed for approximately 0.5 seconds for each 1-second window comprising 5,000 legitimate and 150,000 attack packets. This amount of traffic was roughly equivalent to 1-2 Gbps speed. We believe that execution time and memory requirements can be greatly improved by optimization and hardware support.

6.1 Simulation Conditions

6.1.1 Trace Data and Profiles

All trace data was from the NLANR archives as described in Section 3.2. The reference profile was obtained from seven days at the MEM site between Monday, 26 September 2005, and Sunday, 2 October 2005, collected at approximately 9:00 a.m. The average pps rate (packets per second) ranged from 861 pps to 3,618 pps and the bandwidth ranges from 2.7 Mbps to 21.6 Mbps. Except in Section 6.5, the Tuesday, 27 September 2005, 9:00 a.m. packet trace was used for the legitimate packet input.

The profile was created from multiple 10-second windows using a 99 percent coverage adaptive threshold as described in Section 3.1. Due to IP address sanitization, the source and destination addresses were not available. The profiles included two single attributes (TTL, TCP flags) and one joint attribute (protocol type + server port + packet size). The server port was the smaller of source and destination port numbers. For non-TCP packets, TCP flag was not used for scoring.

6.1.2 Different Attack Types

We tested PacketScore for the following types of attacks:

- Generic attack: All attribute values of the attack packets were uniformly randomized over their corresponding allowable ranges.
- TCP-SYN Flood attack: All attack packets had a TCP SYN flag set, the size was fixed to 40, and other values were randomized.
- SQL Slammer Worm attack: Attack packets were UDP packets sent to port 1434 with a packet size between 371 and 400 bytes.
- Nominal attack: All attack packets resembled the most dominant type of legitimate packets observed in practice, i.e., 1,500-byte TCP packets with a server-port of 80 and the TCP-flag set to ACK. TTL values might or might not be randomized in this attack. Both cases were examined for TTL.
- Mixed attack: An equal combination of the above four types of attacks.

Except in Section 6.3, a generic attack was used by default.

6.1.3 Attack Intensity and Target Load (ρ_{target})

Except in Section 6.4, where attack intensity varied, an attack of 10 times the nominal traffic amount was applied. The target load (ρ_{target}) for outgoing traffic was read from

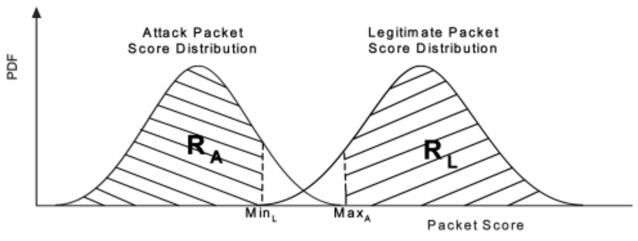

PDF
Attack Packet Score Distribution
Legitimate Packet Score Distribution
RA
RL
MinL
MaxA
Packet Score

Fig. 5. Score differentiation between legitimate and attack packets.

the profile. The profile contained the maximum observed pps in any 10-second window period. This corresponded to 4,874 packets per second.

6.1.4 Time-Based versus Packet Number-Based Windows

The CDF and scorebook must be updated periodically in pipeline processing. The period can be defined as a preset time interval or as a number of packets. The default method was a time-based interval with a 1-second window. We investigated PacketScore's performance by varying the window size from 0.01 second to 10 seconds. In the case of fluctuating traffic volume, a time-based window may not allow for observation of as many packets as necessary to create meaningful profiles. In such a case, packet number-based intervals are more suitable. This is explored in Section 6.6. In all other sections, a time-based 10-second window was used.

6.2 Performance Metrics

To evaluate PacketScore's performance, we first examined the differences in the score distribution for attack and legitimate packets. These differences were quantified using two metrics, namely, $\mathrm{R_A}$ and $\mathrm{R_L}$ as illustrated in Fig. 5.

We defined $\mathrm{Min_L}(\mathrm{Max_A})$ as the lowest (highest) score observed for incoming legitimate (attack) packets and $\mathrm{R_A}(\mathrm{R_L})$ as the fraction of attack (legitimate) packets that had a score below $\mathrm{Min_L}$ (above $\mathrm{Max_A}$). The closer $\mathrm{R_A}$ and $\mathrm{R_L}$ are to 100 percent, the better the score-differentiation power. In practice, score distributions usually have long, thin tails due to very few outlier packets with extreme scores. To avoid the masking effect of such outliers, we took $\mathrm{Min_L}(\mathrm{Max_A})$ to be the first (99th) percentile of the score distribution of legitimate (attack) packets. Since the shape of distribution changes per period, the average of $\mathrm{R_A}$ and $\mathrm{R_L}$ were taken over multiple periods.

While $\mathrm{R_A}$ and $\mathrm{R_L}$ can quantify the score differentiation power, the final outcome of selective discarding also depends on the dynamics of the threshold update. We therefore also measured the false positive (i.e., legitimate packets getting falsely discarded), and false negative (i.e., attack packets getting falsely admitted) ratios. To check the effectiveness of the overload control scheme, we compared the actual output utilization ρ_{out} against the target utilization ρ_{target}.

A typical set of score distributions for attack and legitimate packets is shown in Fig. 6a in log-scale. The scores of attack packets and legitimate packets show a clear separation with a slightly overlapping region. Fig. 6b shows the cumulative score distribution in linear-scale, illustrating that the majority of attack packets are concentrated in the low-score region.

Fig. 7 shows a typical score distribution trend over 25 periods under a DDoS attack. The attack packets (high peaks on the upper left side) are concentrated in the lower-score region while legitimate packets (lower right side) have higher scores. The black bar between the two groups represents the cutoff threshold scores for the discard decision, which removes the majority of the attack traffic. PacketScore tracks the score distribution change in each period and adjusts the cutoff threshold accordingly.

6.3 Performance under Different Attack Types

The results are described in Table 6. In most cases, $\mathrm{R_A}$ and $\mathrm{R_L}$ were above 99 percent and false positives were kept low. The result was substantially better than random packet dropping of which the false positive ratio is expected to be 90.7 percent, and better than some of previous methods surveyed in Section 1. Furthermore, ρ_{out} was successfully kept close to its target value in most cases. The false negative ratios were mainly due to the gap between ρ_{target} and $\rho_{\mathrm{legitimate}}$, i.e., the extra capacity left by the legitimate packets that allowed some attack packets to slip through.

It is noteworthy that the false positive probability for the TCP-SYN flood attack was kept at a very low level (0 percent). Although the signature of the TCP-SYN flood packets can easily be derived by any filtering scheme, the

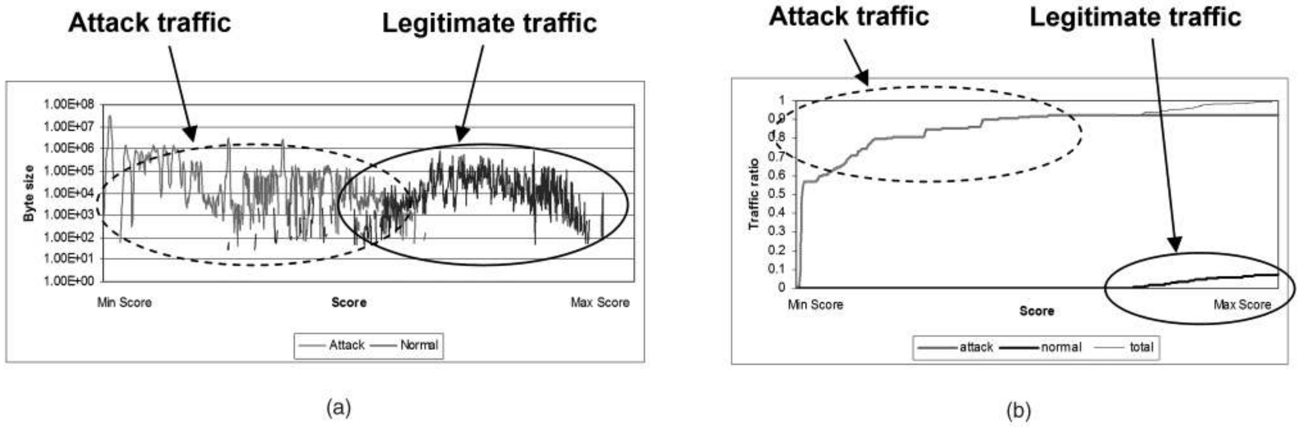


Fig. 6. Score distribution example: (a) Raw score distribution and (b) cumulative score distribution.

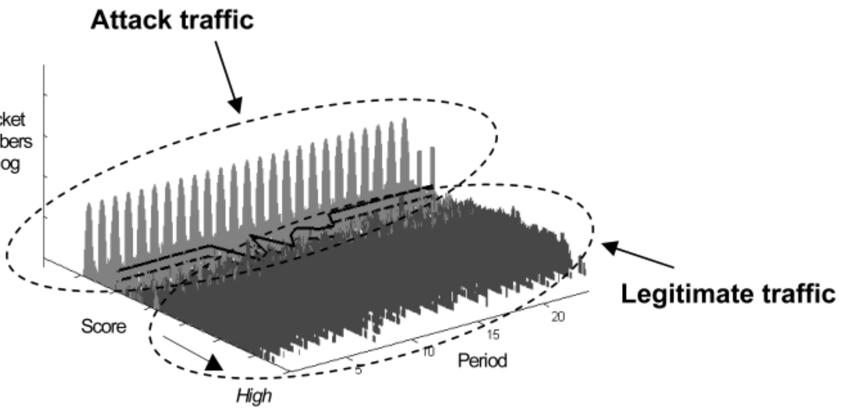


Fig. 7. Score distribution over time.

TABLE 6
Performance against Different Types of Attack (Average $\rho_{legitimate} = 3,618$ pps, $\rho_{target} = 4$)

Attack Type	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
Generic	0.04	4.03	99.9	99.9	0.99
SYN flood	0.0	4.03	100.0	100.0	0.99
SQL Worm	0.0	4.06	100.0	100.0	0.99
Nominal (TTL=100)	0.0	2.68	100.0	100.0	0.85
Nominal (TTL=118)	0.09	3.25	100.0	99.9	0.91
Nominal (TTL=random)	3.30	4.24	93.6	94.7	0.99
Mixed (TTL=100 for nominal)	0.01	4.42	99.8	99.7	1.14

ability of PacketScore to prioritize legitimate TCP-SYN packets over attack packets based on other packet attributes is an essential feature. Without such prioritization, e.g., in the case of stateless rule/signature-based filtering, all TCP-SYN packets would be discarded and, thus, ensure the success of the DDoS attack on the victim.

PacketScore did show some degradation under nominal attack when the TTL values were randomized. Legitimate packets having the same characteristics as attack packets were penalized, but such chances were still quite small, and the false positive ratio was kept to 3.30 percent. The distribution of TTL value in the profile is shown in Fig. 8. When the TTL values were fixed, the performance was better. The TTL value 118 accounted for the largest portion of traffic (29.86 percent) and 100 had a ratio under the adaptive threshold (0.29 percent) for 99 percent coverage.

As PacketScore utilizes more attributes, the performance should become better. While we used only one joint attribute in this experiment, numerous combinations of attributes are possible to increase the number of attributes, and the performance under nominal attack can be further improved. Fig. 9 shows the actual score distribution under mixed attack. It is interesting to see how the scores of each attack type are distributed.

6.4 Performance under Different Attack Intensities

Table 7 shows that the proposed scheme can effectively provide overload control as attack intensifies. Even when the volume of attack packets increased from one time to 20 times the nominal load, the scheme still consistently allowed more than 99.9 percent of legitimate packets to pass through unaffected. The attack packets were admitted only because of the extra capacity ($\rho_{rmtarget} - \rho_{\text{legitimate}}$), as discussed before.

By design, the differentiation power of PacketScore improves as the DDoS attack intensifies. This is because as attack traffic volume increases, the difference between the current traffic profile and the nominal one also increases.

6.5 Nominal Profile Sensitivity

We studied the effect of different nominal profiles using the following:

- Monday profile (25 September 2005, 8:00 p.m.).
- Weekly profile (8:00 p.m. for seven days from 26 September 2005 to 2 October 2005): This is the default profile used previously.

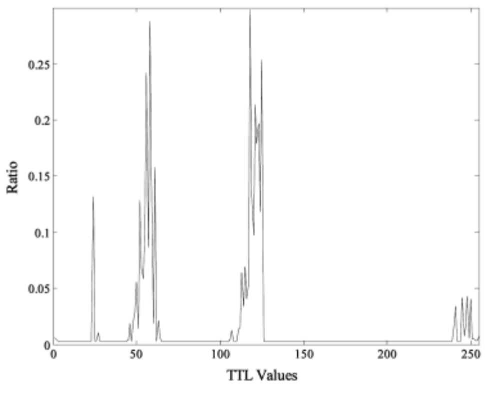


Fig. 8. TTL value ratio distribution in the profile.

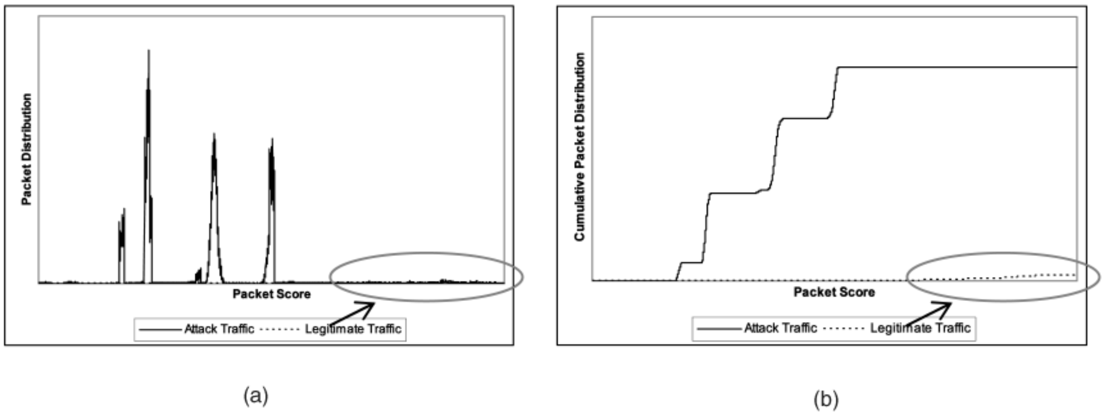


Fig. 9. Score distribution under different types of attack. (a) Packet distribution and (b) cumulative packet distribution.

TABLE 7
Peformance under Different Attack Intensities

Attack Intensity	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
x1	0.0	41.02	99.21	98.99	0.99
x5	0.03	8.31	99.83	99.86	0.99
x10	0.04	4.02	99.87	99.85	0.99
x20	0.07	2.03	99.90	99.85	0.99

TABLE 8
Performance by Different Profiles

	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
Monday profile	29.14	1.67	80.65	70.32	0.99
Weekly profile	0.04	4.03	99.87	99.85	0.99
Monthly profile	3.11	2.10	98.74	82.69	0.99
Different site profile	20.88	5.74	90.86	74.58	0.98

- Monthly profile (8:00 p.m. on Tuesdays for the seven weeks).
- Different site (8:00 a.m. on Tuesday, 27 September 2005, from AMP).

The results are depicted in Table 8. The weekly profile reflected the traffic characteristics best and showed the best performance, followed by the monthly profile. However, the Monday profile and other site profiles were somewhat different from the Tuesday traffic profile and resulted in poor performance.

6.6 Performance under Different Window Scales

Table 9 shows PacketScore performance under different CDF update time scales. The time scale was 1-second. We observed that most of the time scales performed similarly, but performance started to degrade when the time scale became too small because there were not enough packets within the window to build an accurate scorebook.

Packet number-based windows showed a similar performance as Table 10. With an excessively low number of packets (e.g., 500 packets), we observed PacketScore performance degraded. However, with a sufficiently large number of packets ($>$ 1,000 packets), the packet number-based window could be safely applied to a network with fluctuating traffic volume.

6.7 Performance under Changing Attacks

As mentioned in Section 4.2, changing attacks are more challenging due to their time-varying attack packet characteristics. When a change occurs, two measurement periods are required for PacketScore to establish new profiles and scorebooks due to the nature of pipeline processing. During these adjustment periods, the PacketScore scheme can be misled to defend against no-longer-existing attack packets. The effect becomes worse if the attack type changes rapidly compared with the measurement period time scale. The effectiveness of PacketScore can be improved by shortening the CDF update window time.

We observed the effects of changing attack types by alternating the four primary attack types. An attack type was randomly selected and continued for an exponentially distributed period with the average duration as indicated in the results table. Table 11 shows the effects of different window time scales under changing attacks.

As we reduced the measurement window time from 5 seconds to 0.1 second, PacketScore tracked the attack changes very closely, showing as good a performance as in constant attacks. Similar to the false positive ratios, the outgoing traffic amount was better controlled. The PacketScore scheme was effective with very short measurement windows, for example, 0.01 second, as long as the number of packets within a window is meaningful for statistical processing. It is difficult for an attacker to launch a precisely

TABLE 9
Performance under Different Time Scales

Size of time window	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
10 seconds	0.04	3.50	99.88	99.84	1.06
1 second	0.04	4.03	99.87	99.85	0.99
0.1 second	0.16	4.09	99.68	99.80	1.0
0.01 second	2.01	4.38	99.49	99.41	1.0

TABLE 10
Performance under Different Packet Number Windows (Average Legitimate pps = 3,618, Attack pps = 48,740)

Number of packets (legitimate + attack)	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
100,000 (~2 sec)	0.03	3.92	99.90	99.88	0.99
10,000 (~0.2 sec)	0.03	4.00	99.81	99.83	1.0
1000 (~0.02 sec)	0.79	4.12	99.25	99.59	0.99
500 (~0.01 sec)	2.59	4.33	99.53	99.26	0.99

time-coordinated attack at such a resolution due to the nature of the Internet. This allows PacketScore to block nearly all types of changing attacks. A more thorough study on changing attacks has been done in [3].

7 Discussions

7.1 Need for Clean Nominal Profiles

One challenging issue of the PacketScore scheme is the need for a clean baseline profile as in other profile-based systems. This is because DDoS attack traffic is already prevalent on the Internet and a quiet attack-free period may be hard to find. As a result, the constructed nominal profile may be biased by the DDoS traffic and may force PacketScore to accept DDoS traffic that has been reflected in the profile. However, PacketScore is designed to accept specific traffic only up to the maximum ratio that was observed in the past. Therefore, DDoS traffic beyond this ratio will be properly filtered by PacketScore and, thus, cannot succeed in a massive attack. Nevertheless, accepting DDoS traffic is not desirable as it wastes bandwidth. This situation can be improved by constructing a cleaner nominal profile that contains less DDoS traffic.

A cleaner profile can be made one of two ways. First, the packet trace data can be analyzed to identify legitimate flows that show proper two-way communication behavior. The packets from the legitimate flows are used for constructing the profile. Although some traffic flows that do not have continuous packet exchange, such as ICMP, may be left out, PacketScore filtering is already based on an iceberg-style histogram with default value assignment for noniceberg items. The impact on performance of missing some of the packets should be minimal.

Second, we can first use a generic profile to remove those packets that are more likely to be attack packets, and use the remaining packets to create the final nominal profile. The generic profile reflects overall Internet traffic characteristics, e.g., TCP versus UDP ratio, common packet size, common TCP flags, etc. Our preliminary research shows that this two-step profiling is very effective to fight generic attacks. Further study is needed on these methods.

7.2 Attacker's Pretraining

A clever attacker may send a small amount of attack traffic in advance to misconfigure the baseline profile (P'_n) and later bombard victims with similar attack packets. PacketScore can block this type of attack successfully. The score of a packet depends on the *volume* of the specific traffic characteristics during the actual attack, not on the *existence* of the specific traffic characteristics in the nominal profile. During an attack, the attacker must significantly increase the amount of attack traffic to create a DoS condition, making the attack traffic distribution (P_m) substantially different from the baseline profile (P'_n). In other words, the attack traffic attribute ratio in P'_n is relatively low because the amount of pretraining traffic is relatively small compared with legitimate traffic, but the ratio is increased during an attack due to large attack volume compared with legitimate traffic volume. This defeats the attackers' purpose to make P'_n equal to P_m. Equation (2) can be rewritten with a pretrained nominal profile (P'_n).

TABLE 11
Performance under Changing Attack Types

Attack Type	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
Changing (Ave. one attack duration = 1 sec)	5.15	20.50	94.31	97.57	2.65
Changing (Ave. one attack duration = 5 sec)	0.10	21.75	99.70	98.76	2.80

(a)

Attack Type	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
Changing (Ave. one attack duration = 1 sec)	1.65	10.06	99.02	98.55	1.60
Changing (Ave. one attack duration = 5 sec)	0.38	5.53	99.60	99.50	1.14

(b)

Attack Type	% False positive	% False negative	PDF Separation		$\frac{\rho_{out}}{\rho_{target}}$
			% R_A	% R_L	
Changing (Ave. one attack duration = 1 sec)	0.91	6.38	98.78	99.64	1.26
Changing (Ave. one attack duration = 5 sec)	0.32	4.08	99.76	99.91	1.03

(c)

(a) Scorebook update interval = 5 seconds. (b) Scorebook update interval = 1 second. (c) Scorebook update interval = 0.1 second.

$$CLP(p) = \frac{N'_n \times P'_n(A = a_p) \times P'_n(B = b_p) \times \dots}{N_m \times P_m(A = a_p) \times P_m(B = b_p) \times \dots}. \qquad (4)$$

During an attack, P_m for dominant attack attribute values increases; thus, the CLP of an attack packet becomes smaller.

A successful attack can only be launched by simulating the precise attribute value distribution as in the nominal profile (P_n). If the P_m values are the same as the P_n values for all attribute values, the CLP becomes a constant value for all packets; thus, PacketScore loses its attack-distinguishing power. Therefore, securing the nominal profile (P_n) from the attackers is very important.

7.3 DDoS Attacks with Unspoofed Addresses

Not all DDoS attacks spoof the source addresses. PacketScore can successfully defeat this kind of attack because it does not depend on the assumption of spoofed source addresses. Of course, including source addresses or IP prefixes will allow more useful profiles. But, as we have seen in the simulation results, PacketScore utilizes many other IP header fields and can perform well even without examining the source addresses. Attack packets with live addresses still create a change in the operational profile (P_m) and the scorebook, unless they generate the exact same distribution as in the nominal profile, thus allowing PacketScore to distinguish attack packets.

Beyond using live addresses, legitimate-looking attacks are nowadays increasing, i.e., legitimate TCP sessions from zombie machines. As explained in Section 7.2, the traffic from zombie machines will have to increase significantly during an attack, which changes the scorebook and makes PacketScore able to distinguish those attack packets.

7.4 Flash Crowds

Flash crowds are indistinguishable from DDoS attacks to PacketScore. However, PacketScore does not affect flash crowds negatively. As long as the flash crowd capacity does not exceed the acceptable link capacity, the packets are not dropped because PacketScore starts operation only when the incoming traffic exceeds the acceptable capacity. If it does, however, the excessive packets are forced to be dropped anyway with or without PacketScore, so the priority change by PacketScore does not change the amount of dropped packets in a flash crowd.

7.5 Human Intervention

PacketScore does not need human intervention once the profile is established. The only parameter that may need human intervention is ρ_{target}, which is by default set to the maximum observed traffic amount. When there is network capacity available beyond it, the ρ_{target} can be increased manually to accept more potentially legitimate packets.

8 Conclusions and Future Work

We have outlined the process the PacketScore scheme uses to defend against DDoS attacks. The key concept in PacketScore is the Conditional Legitimate Probability (CLP) produced by comparison of legitimate traffic and attack traffic characteristics, which indicates the likelihood of legitimacy of a packet. As a result, packets following a legitimate traffic profile have higher scores, while attack packets have lower scores. This scheme can tackle never-before-seen DDoS attack types by providing a statistics-based adaptive differentiation between attack and legitimate packets to drive selective packet discarding and overload control at high-speed. Thus, PacketScore is capable of blocking virtually all kinds of attacks as long as the attackers do not precisely mimic the sites' traffic characteristics. We have studied the performance and design tradeoffs of the proposed packet scoring scheme in the context of a stand-alone implementation. The newer simulation results in this paper are consistent with our previous research [19]. By exploiting the measurement/scorebook generation process, an attacker may try to mislead PacketScore by changing the attack types and/or intensities. We can easily overcome such an attempt by using a smaller measurement period to track the attack traffic pattern more closely.

We are currently investigating the generalized implementation of PacketScore for core networks. PacketScore is suitable for the operation at the core network at high speed, and we are working on an enhanced scheme for core network operation in a distributed manner. In particular, we plan to investigate the effects of update and feedback delays in a distributed implementation, and implement the scheme in hardware using network processors. Second, PacketScore is designed to work best for a large volume attack and it does not work well with low-volume attacks. We intend to explore and improve PacketScore performance in the presence of such attack types, e.g., bandwidth soaking attacks described in [31] or low-rate attacks [15]. Finally, a thorough investigation on the stability of traffic characteristics shall be performed as mentioned in Section 3.

Acknowledgments

W.C. Lau's work was supported by a CUHK/RGC Direct Grant # 2050368.

References

[1] Akamai Technologies, Inc., http://www.akamai.com, 2006.
[2] B. Babcock et al., "Models and Issues in DataStream Systems," *ACM Symp. Principles of Database Systems,* June 2002.
[3] M.C. Chuah, W. Lau, Y. Kim, and H.J. Chao, "Transient Performance of PacketScore for Blocking DDoS Attack," *Proc. IEEE Int'l Conf. Comm.,* 2004.
[4] Cisco IOS Security Configuration Guide, Release 12.2 "Configuring Unicast Reverse Path Forwarding," pp. SC-431-SC-446, http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fothersf/scfrpf.pdf. 2006, 2006.
[5] C. Estan, S. Savage, and G. Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic," *Proc. 2003 ACM SIGCOMM,* pp 137-148, 2003.
[6] CSI/FBI Survey, http://www.gocsi.com/forms/fbi/csi_fbi_survey.jhtml, 2006.
[7] FBI Fugitive, http://www.fbi.gov/wanted/fugitives/cyber/echouafni_s.htm, 2006.
[8] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2827, 2000.
[9] L. Garber, "Denial-of-Service Attacks Rip the Internet," *Computer,* pp. 12-17, Apr. 2000.
[10] J. Ioannidis and S.M. Bellovin, "Implementing Pushback: Router-Based Defense against DDoS Attacks," *Proc. Network and Distributed System Security Symp.,* Feb. 2002.
[11] C. Jin, H. Wang, and K.G. Shin, "Hop-Count Filtering: An Effective Defense against Spoofed Traffic," *Proc. ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.
[12] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," *Proc. Int'l World Wide Web Conf.,* May 2002.
[13] S. Kasera et al., "Fast and Robust Signaling Overload Control," *Proc. Int'l Conf. Network Protocols,* Nov. 2001.

[14] A.D. Keromytis, V. Misra, and D. Rubenstein, "SOS: An Architecture for Mitigating DDoS Attacks," *IEEE J. Selected Areas in Comm.*, vol. 22, no. 1, pp. 176-188, Jan. 2004.
[15] A. Kuzmanovic and E.W. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)," *Proc. ACM SIGCOMM 2003,* Aug. 2003.
[16] H. Kim and I. Kang, "On the Effectiveness of Martian Address Filtering and Its Extensions," *Proc. IEEE GLOBECOM,* Dec. 2003.
[17] Y. Kim, J.Y. Jo, H.J. Chao, and F. Merat, "High-Speed Router Filter for Blocking TCP Flooding under Distributed Denial-of-service Attack," *Proc. IEEE Int'l Performance, Computing, and Comm. Conf.,* Apr. 2003.
[18] Y. Kim, J.Y. Jo, and F. Merat, "Defeating Distributed Denial-of-Service Attack with Deterministic Bit Marking," *Proc. IEEE GLOBECOM,* Dec. 2003.
[19] Y. Kim, W.C. Lau, M.C. Chuah, and H.J. Chao, "PacketScore: Statistics-Based Overload Control against Distributed Denial-of-Service Attacks," *Proc. IEEE INFOCOM,* Mar. 2004.
[20] Q. Li, E.C. Chang, and M.C. Chan, "On the Effectiveness of DDoS Attacks on Statistical Filtering," *Proc. 2005 IEEE INFOCOM,* 2005.
[21] D. Liu and F. Huebner, "Application Profiling of IP Traffic," *Proc. 27th Ann. IEEE Conf. Local Computer Networks (LCN),* 2002.
[22] M. Mahoney and P.K. Chan, "Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks," *Proc. ACM 2002 SIGKDD,* pp 376-385, 2002.
[23] D. Marchette, "A Statistical Method for Profiling Network Traffic," *Proc. First USENIX Workshop Intrusion Detection and Network Monitoring,* Apr. 1999.
[24] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," *Proc. 10th IEEE Int'l Conf. Network Protocols,* Nov. 2002.
[25] D. Moore, G.M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proc. 10th USENIX Security Symp.,* Aug. 2001.
[26] NLANR PMA Packet Trace Data, http://pma.nlanr.net/Traces, 2006.
[27] K. Park and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," *Proc. IEEE INFOCOM,* pp. 338-347, 2001.
[28] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proc. ACM SIGCOMM,* pp. 15-26, 2001.
[29] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *IEEE/ACM Trans. Networking,* vol. 9, no. 3, June 2001.
[30] H. Wang, D. Zhang, and K.G. Shin, "Change-Point Monitoring for the Detection of DoS Attacks," *IEEE Trans. Dependable and Secure Computing,* vol. 1, no. 4, Oct.-Dec. 2004.
[31] Y. Xu and R. Guérin, "On the Robustness of Router-Based Denial-of-Service (DoS) Defense Systems," *ACM SIGCOMM Computer Comm. Rev.,* vol. 35, no. 3, July 2005.
[32] A. Yaar and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proc. IEEE Symp. Security and Privacy,* 2003.
[33] A. Yaar and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," *Proc. 2004 IEEE Symp. Security and Privacy,* 2004.

Yoohwan Kim received the bachelor's degree in economics from Seoul National University, Korea, in 1989 and the master's and PhD degrees in computer engineering from Case Western Reserve University, Cleveland, Ohio, in 1994 and 2004, respectively. He is an assistant professor of computer science at the University of Nevada, Las Vegas (UNLV). Before joining UNLV in 2004, he worked in software and communication networking industry for several years. He was a member of technical staff at Lucent Technologies, Whippany, New Jersey, developing software for wireless networking equipment between 1997 and 1999. In 2000, he cofounded and managed a New Jersey-based software company that developed technologies for delivering and customizing video advertising over the Internet. His current research interests include network security, Internet traffic analysis, software architecture, and real-time embedded software design. He is a member of the IEEE and the IEEE Computer Society.

Wing Cheong Lau received the BS degree in engineering from The University of Hong Kong and the MS and PhD degrees in electrical and computer engineering from The University of Texas at Austin. He is an associate professor with the Department of Information Engineering at The Chinese University of Hong Kong (CUHK), where he also serves as the director of the Mobile Technologies Center (MobiTeC). From 1995 to 1997, he was with Southwestern Bell Technology Resources (now AT&T Labs), Austin, Texas, responsible for broadband network architectural design and performance analysis. From 1997 to 2004, he was with the Performance Analysis Department, Bell Laboratories, Lucent Technologies, Holmdel, New Jersey, where he served as a performance consultant and system architect. Prior to joining CUHK, he was with Qualcomm, San Diego, actively contributing to the design and standardization of IETF and 3G Mobility Management protocols and architecture. His research interest includes networking protocol design and performance analysis, traffic characterization, system modeling and network security for high-speed wired and wireless networks. He is a senior member of the IEEE and the IEEE Computer Society.

Mooi Choo Chuah received the first class honors bachelor's degree electrical engineering from University of Malaya, Malaysia, and the master's and PhD degree's from the University of California, San Diego. She is the director of Wireless Infrastructure and Network Security Laboratory (WiNS Labs) and an associate professor of Computer Science and Engineering Department at Lehigh University. Prior to joining Lehigh, she spent 12 years at Bell Laboratories, Lucent Technologies, Holmdel, New Jersey, where she conducted researches in future wireless system design, network security, resource, and mobility management design. She has been awarded 35 US patents, one Canadian patent, and has 20 more pending in various areas, e.g., wireless MAC, IP protocol design, mobility management, etc. Her current research interests include Internet and wireless security, protocol, system design for disruption tolerant networks, and ad hoc and sensor networks design. She is a senior member of the IEEE and a member of Sigma Xi society.

H. Jonathan Chao received the BS and MS degrees in electrical engineering from National Chiao Tung University, Taiwan, and the PhD degree in electrical engineering from The Ohio State University. He is department head and a professor of electrical and computer engineering at Polytechnic University, New York, where he joined in January 1992. He has been doing research in the areas of network security, terabit switches/routers, quality of service control, and optical networking/switching. He holds more than 20 patents and has published more than 150 journal and conference papers in the above areas. He has also served as a consultant for various companies, such as Lucent, NEC, and Telcordia. During 2000-2001, he was cofounder and CTO of Coree Networks, New Jersey, where he led a team to implement a multiterabit MPLS (Multi-Protocol Label Switching) switch router with carrier-class reliability. From 1985 to 1992, he was a member of technical staff at Telcordia, where he was involved in transport and switching system architecture designs and ASIC implementations. From 1977 to 1981, he was a senior engineer at Telecommunication Labs of Taiwan performing circuit designs for a digital telephone switching system. Professor Chao is a fellow of the IEEE for his contributions to the architecture and application of VLSI circuits in high-speed packet networks. He received the Telcordia Excellence Award in 1987. He is a corecipient of the 2001 Best Paper Award from the *IEEE Transaction on Circuits and Systems for Video Technology*. He coauthored two networking books, *Broadband Packet Switching Technologies—A Practical Guide to ATM Switches and IP Routers* (New York: Wiley, 2001) and *Quality of Service Control in High-Speed Networks* (New York: Wiley, 2001). He has served as a guest editor for the *IEEE Journal On Selected Areas in Communications* (*JSAC*) and the *IEEE/ACM Transactions on Networking* from 1997-2000.