HAWK: Halting Anomalies with Weighted Choking to Rescue Well-Behaved TCP Sessions from Shrew DDoS Attacks[1]

Yu-Kwong Kwok, Rohit Tripathi, Yu Chen, and Kai Hwang

University of Southern California, Los Angeles, CA 90089, USA

Abstract. High availability in network services is crucial for effective large-scale distributed computing. While distributed denial-of-service (DDoS) attacks through massive packet flooding have baffled researchers for years, a new type of even more detrimental attack—shrew attacks (periodic intensive packet bursts with low average rate)—has recently been identified. Shrew attacks can significantly degrade well-behaved TCP sessions, repel potential new connections, and are very difficult to detect, not to mention defend against, due to its low average rate.

We propose a new stateful adaptive queue management technique called HAWK (Halting Anomaly with Weighted choKing) which works by judiciously identifying malicious shrew packet flows using a small flow table and dropping such packets decisively to halt the attack such that well-behaved TCP sessions can re-gain their bandwidth shares. Our NS-2 based extensive performance results indicate that HAWK is highly agile.

1 Introduction

Various kinds of malicious attacks have hindered the development of effective wide-area distributed computing. The most notable type of attack is the so-called Distributed Denial-of-Service (DDoS) attack [7], which works by overwhelming the systems with bogus or defective traffic that undermines the systems' ability to function normally. DDoS attacks aims at consuming resources (CPU cycles, system memory or network bandwidth) by flooding bogus traffic at sites so as to deny services to the actual user and prevent legitimate transactions from completing [1]. The TCP, UDP, and ICMP flooding attacks fall in this category.

Unfortunately, while finding effective solutions to combat DDoS attacks has baffled researchers for years, an even more detrimental type of network-based attack has recently been identified [2]. This special class of attack is referred to as low-rate TCP-targeted DDoS attack or shrew attack [2] that denies bandwidth resources to legitimate TCP flows in a *stealthy* manner. Indeed, unlike traditional DDoS attacks, which are easy to detect by observing that the victim site is totally unable to respond,

[1] Manuscript accepted for presentation at ICCNMC 2005 in August 2005.
This research was supported by an NSF ITR Research Grant under contract number ACI-0325409.
Corresponding Author: Kai Hwang, Email: kaihwang@usc.edu, Tel: 213-740-4470, Fax: 213-740-4418.

a shrew attack is very difficult to detect [2] because the adverse effects on well-behaved network connections are not easily observable. Commercial Web sites would then suffer from stealthy losses of potential new connections (hence, new transactions).

The key idea behind a shrew attack is to exploit TCP's *Retransmission Time-Out* (RTO) mechanism to synchronize *intelligent* (i.e., carefully articulated) low average rate bursts. Thus, a shrew attack can also be referred to as *degradation-of-service* or *pulsing* attack, as opposed to the well-known denial-of-service attack. Unlike a regular zombie that paralyzes a system by flooding it with a steady stream of attack traffic, the pulsing zombie attacks its target with irregular small bursts of attack traffic from multiple sources over an extended period of time (see Figure 1). As such, pulsing zombie attacks are more difficult for routers or counter-DDoS mechanisms to detect and trace. The reason is that unlike flooding DDoS attacks, they are slow and gradual, and thus they do not immediately appear as malicious.

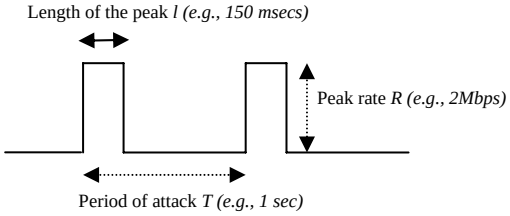


Fig. 1. An illustration of the shrew attack stream with a square waveform.

As indicated by Kuzmanovic and Knightly [2], it is very difficult to detect such a shrew attack. The main challenge lies in separating a bogus traffic flow from a "flash crowd" [5] without maintaining complicated and expensive per flow state information at the routers. In this paper, we meet this challenge by proposing a novel effective detection technique, called HAWK[2] (*Halting Anomaly with Weighted choKing*), which is an active queue management method based on only *partial state*. Specifically, HAWK works by judiciously monitoring the packet flows with the help of a small flow table. Traffic statistics are accumulated in the flow table using a technique similar to the CHOKe algorithm [3].

A packet flow will be marked as malicious if its traffic statistics in the flow table indicate that the flow is far too bursty over an extended period of time (e.g., 5 secs) with very high rate bursts appearing in short time-spans (e.g., 100 msecs). Once a flow is identified to be malicious, HAWK will drop its packets decisively in order to help the well-behaved sessions to re-gain their entitled bandwidth shares. Furthermore, the HAWK algorithm is able to defend the targeted victim against both single source and distributed shrew attacks while maintaining low overhead in terms of processing and memory resources. Our NS-2 based simulation results indicate that the HAWK algorithm is highly effective.

The rest of the paper is organized as follows. In the next section, we first describe the key characteristics of service-degrading network attacks, and then we introduce

[2] Hawks are natural enemies of shrews.

our HAWK algorithm for traffic burst characterization and corresponding flow classification. Simulation setting and experimental environment details are given in Section 3. In the same section, we present the NS-2 experimental results and provide our interpretations. Finally, we give some concluding remarks in Section 4.

2 The Proposed HAWK Algorithm

We propose to use our HAWK algorithm at the bottleneck link router for characterizing bursts of the attack stream and classifying them into legitimate or illegitimate sources. HAWK maintains a very small state information data structure—the flow table—in order to track down the shrew flows. The flow table only keeps the traffic statistics of potentially malicious flows and confirmed malicious flows, and thus, normally, occupies very little storage space. The maintenance of the flow table is further elaborated below and the storage space requirements are discussed in detail in Section 3.

The router maintains a single queue for all the incoming flows and the average queue size computation is done using exponential moving average as in RED [6] and CHOKe [3]. But unlike from these previous approaches, our decision-making algorithm involves flow table comparisons and statistical computations that are used to characterize and classify flows into different threat level categories. Whenever a new packet arrives at the queue, if the average queue size is less than the minimum threshold of the buffer (Min_{Th}) the packet is admitted into the queue.

Furthermore, HAWK checks each incoming packet against the existing entries in the flow table, and if there is a match the corresponding flow table statistics are updated. In the "random matching" process, the following checking actions are carried out. If the average queue size is greater than the maximum threshold (Max_{Th}) the incoming packet is dropped after checking and updating the flow table statistics. For the intermediate case when the average queue size value is between the minimum (Min_{Th}) and maximum (Max_{Th}) thresholds, we use a mechanism similar to CHOKe [3] by admitting the packet with a probability *p* which depends on the average queue size. For instance, if the queue size is over the maximum threshold (Max_{Th}), the packet is dropped with a probability 1. Similarly, if the queue size is below the minimum threshold (Min_{Th}), the packet is dropped with a probability 0. In the intermediate case, additional checking of the flow table and statistics computations are performed for flow classifications.

In their modeling and simulations, Kuzmanovic *et al.* show the relationship between the throughput of a TCP flow and the denial-of-service inter-burst period. Our NS-2 simulations modeled single flow TCP and single flow DDoS stream interaction and the modeled flow [2]. The inter-burst periods of one second and lower are most fatal to the TCP throughput. The destructive impact reduces as the inter-burst period is increased.

Furthermore, it is found that for the most severe impact without being identified by existing routing architectures, these shrew bursts should occur in a small time window of 100-250 milliseconds. As such, if we take into account the periodicity of bursts with respect to two separate time windows, one having a smaller time scale of 100-

250 milliseconds and the other having a larger time scale of 1-5 seconds, we can classify the attacks into different threat levels.

On initially identifying a high burst rate flow over a short time scale, if it is found that the average queue size is larger than the Min_{Th}, we perform the following checking. For each new incoming packet, we randomly pick a packet currently in the queue. If the two packets are found to be from the same flow, then we proceed to update the flow table statistics in order to see if the flow is to be considered as a malicious shrew flow. Otherwise, the packet is admitted without further action.

Once the flow is identified as a high rate stream on a short time scale, we correlate these identified bursty flows over a longer time scale using our HAWK algorithm with the help of the small flow table. Thus, at most of the time, the resource requirement of the flow table is of the order of the number of potential attack sources. A cumulative burst value is maintained along with the packet entry times for each of the identified flows. The cumulative burst value and the associated time provide an insight into the burstiness of the flows.

A large cumulative burst for an extended period of time indicates a potential malicious source. For shorter time scales we use a window size as 200 milliseconds. The rationale behind using this value is that for burst lengths longer than this, in order to maintain the same low average rate the DDoS stream would have to keep its peak rate low, thus decreasing the severity of the attack.

Cumulative Burst gives an insight into the average bursts from a given flow over a series of larger time frames. ***Traffic Burst Rate*** above the threshold values over consecutive one second window is logged. If this trend is found to follow in more than or equal to three blocks within the last five seconds ($\mathbf{C_{thresh}}$), the flow is confirmed as a malicious shrew flow and is blocked. We choose the value of three blocks in five seconds time scale to target the most severe effects of the DDoS streams. Also this provides some leniency to normally bursty flow which may send large but intermittent bursts. But since these natural bursts normally cannot extend symmetrically on a larger time scale of five seconds, we can be sure that our chosen time scale would be unbiased towards these naturally bursty flows. Finally, it should be noted that a time period of five seconds is the *shortest time* to confirm a successful detection. We call this five-second time window as *HAWK Window Size*.

Furthermore, if some legitimate flow shows this behavior, it is good to block such a flow so as to provide fairness to other legitimate flows. Since the pre-filtering at the router input still maintains statistics for these flows, they can be removed from the flow table if they react to routers' congestion indication and do not send large bursts for the next two time windows of one second each. This is again a configurable system parameter.

Periods of more than two seconds are not very severe. Thus, we choose the value of two seconds to balance the tradeoff between an optimal flow table size in presence of normally bursty flows and detecting malicious bursts having higher periods. As such, our algorithm sets a flow as malicious if it detects three or more than three bursts over the threshold within a longer spanning window of five seconds.

Traffic Burst Threshold value is chosen based on the link capacity of the routers' output link. It was identified that any burst lower than one third of the link capacity is not severe enough to produce desired DDoS effect on the legitimate TCP flows. So, in performance study, we set the value of $\boldsymbol{BF_{TH}}$ as one third of the bottleneck link

capacity. The attacker can gather information about the bottleneck link capacity using some of the probing schemes in existence [4].

For distributed shrew attacks, instead of Source Address, we maintain Source Subnet that provides the cumulative traffic coming from the infected subnet to the destination victim. The calculation of packet dropping probability $\boldsymbol{p}$ when the average queue size exceeds the minimum threshold is done as in RED [6] and CHOKe [3], i.e., based on the exponential weighted moving averages of the average queue size. Typical values of suitable RED parameters for a gateway router with limited load would be: $\boldsymbol{Min_{Th}}$ ranging between 30 to 100 packets, $\boldsymbol{Max_{Th}}$ set to $3\boldsymbol{Min_{Th}}$ and $\boldsymbol{w_q}$ = 0.002 [6].

The proposed HAWK algorithm can characterize the low-rate TCP-targeted attack using the flow table driven packet dropping technique, as formalized in the flowchart shown in Figure 2. Upon any packet dropped, the legitimate TCP flows will follow the standard protocol semantics and will cut down their rates in accordance with the end-system congestion avoidance mechanisms of TCP. Thus, the values of $\boldsymbol{C_{burst}}$ and $\boldsymbol{BF_{rate}}$ will always remain much lower than the threshold values for these parameters.

Whenever the average queue size is higher than the minimum threshold, a comparison of incoming packet with the already queued packets will result in a success with a high probability if the attack burst was sent during that time frame. The flow table statistics are incremented and the corresponding counter value and the burst parameters for that time frame would progress towards the threshold ultimately, resulting in confirmation of the flow as malicious.

3 NS-2 Simulation Results

In this section, we first describe the simulation set up for evaluating our algorithm in detecting and penalizing attacking hosts and providing fairness to the legitimate users. We use NS-2 to carry out our simulations and we compare the results of our proposed algorithm with those of two well-known active queue management (AQM) algorithms: Drop Tail and CHOKe [3]. As mentioned earlier, response time (i.e., the time duration from the attack launching instant to the attack detected instant) is a very important measure as it determines the duration of damage to a victim site. Thus, we would also examine the response time performance of our algorithm in identifying and blocking the malicious shrew hosts along with the false positives generated using our scheme.

Our simulations consist of a variety of network configurations and traffic patterns simulating both single source as well as multi-source attacks coming from a single LAN and/or distributed LANs. For simulating attacks from different LANs, we use different delay values on the links. The collected statistics are used to plot normalized throughput against attack inter-burst period. The normalized throughput value provides the metric for evaluating the efficiency of our algorithm.

The malicious hosts are modeled as UDP sources sending traffic flows in periodic intervals. The machine GR is the last hop gateway router interfacing to the victim machine connected to the outside network through an AS cloud. We perform statistics collection and computations on the last hop router GR. For a distributed attack

environment the only key parameter that we would like to focus on is the different delays that a flow gets in reaching the victim's end. We achieve this by providing different link delays to each of the malicious hosts.

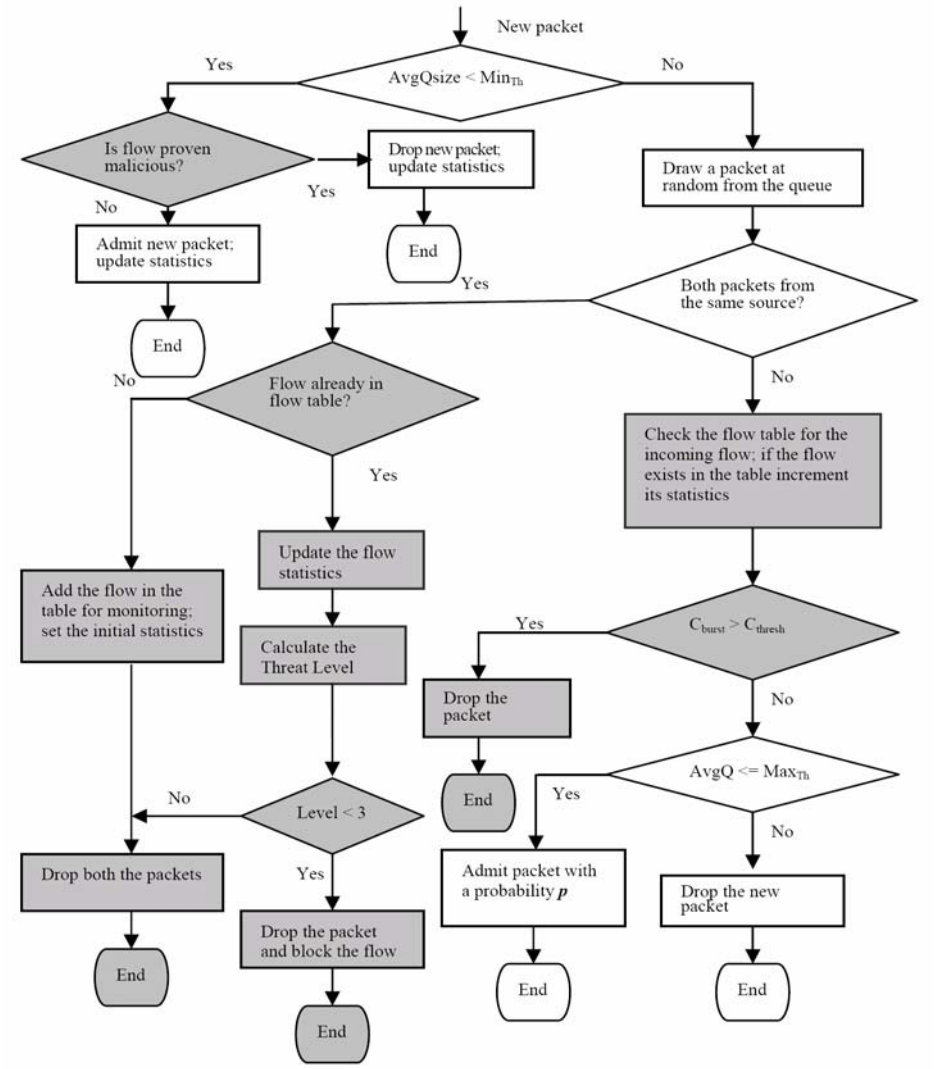


Fig. 2. The proposed HAWK algorithm.

We first use *Normalized Throughput* as the comparison metric, and it is defined as follows:

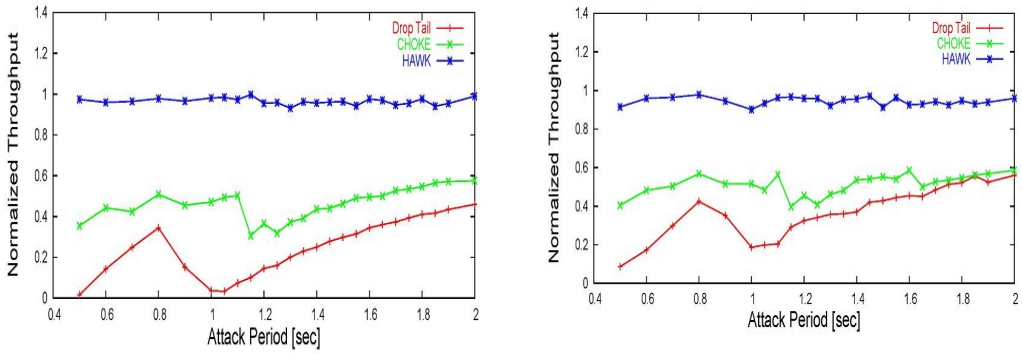
Normalized Throughput =
Average throughput achieved by the TCP flow (or aggregate) with DDoS stream / Throughput achieved without DDoS stream

The value of the normalized throughput gives us an indication of the severity of the damage done by the attack stream. The lower the normalized throughput is, the greater the damage. Unless otherwise specified we use the output link capacity of the last hop router as 2 Mbps, link delay as 120 milliseconds. The shrew attack stream is simulated to generate a square wave having a peak rate of 2 Mbps and a burst length of 150 milliseconds to target TCP flows with average RTT of 150 milliseconds and lower.

Since all the TCP variants are equally vulnerable to the shrew DDoS stream of 50 milliseconds or higher [2], for experimental purpose we use TCP-SACK. Our simulation uses a shorter time scale window of 200 milliseconds and a larger window of five seconds with internal one second blocks. Traffic Burst Threshold value is taken as one third of the bottleneck link capacity.

We first consider the single source scenario. The simulation results of the throughput achieved, under different queuing schemes, by the legitimate TCP flows with different number of shrew DDoS streams are shown in Figure 3. The *x*-axis indicates the period of the burst and the *y*-axis indicates the normalized throughput where value of one indicates the theoretical throughput without the attack stream. It can be clearly seen that under Drop Tail the throughput of the legitimate flow almost reaches zero for period values of one second and lower.

Further increase in the time period of the attack stream increases the throughput of the legitimate flow but it is still far below the actual attainable throughput of one. The results for the CHOKe queuing as shown in Figure 3 indicate a slight improvement in TCP performance but it is clear that CHOKe algorithm cannot achieve the desired goal of providing fair share of the link capacity to the legitimate TCP flows.



(a) one shrew flow and one well-behaved TCP flow

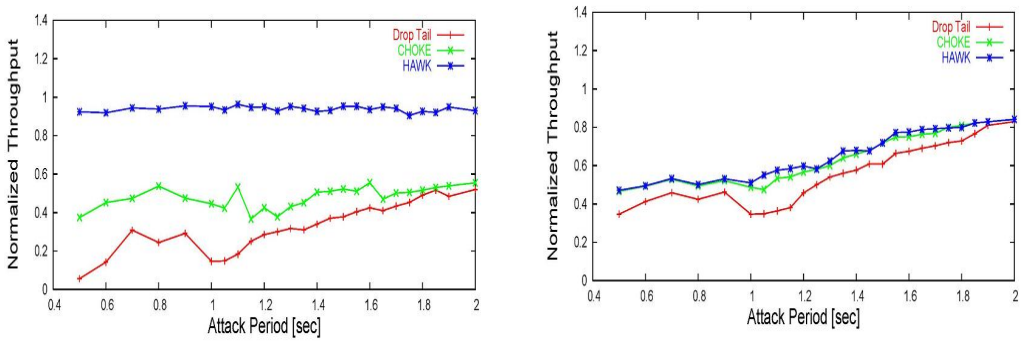
(b) one shrew flow and five well-behaved TCP flows

Fig. 3. Performance comparison among Drop Tail, CHOKe, and HAWK in terms of normalized throughput.

With our HAWK algorithm, we can see that the gain in the TCP throughput is significant throughout the two seconds attack period that we consider in this study. This is due to the fact that for identifying and correlating burst streams we have used three or more blocks of captured bursts within our larger time scale.

Next we consider the multiple-source scenario. The experiment is repeated with five legitimate flows and two DDoS streams so as to find out the impact of attack streams if the attacks are launched from multiple collaborating sites. This kind of scenario is one of the most common cases of distributed denial-of-service attack, where a malicious user compromises a large number of hosts called zombies to launch a coordinated attack with lower peak rate which means that for two DDoS shrews each source sends traffic at half the rate determined in the previous experiment. We would consider two different scenarios here. The link capacity and burst period are kept the same as above in both cases and the effect is seen on five TCP flows.

Firstly, let us consider the case where these zombies are on the same subnet so that all have the same packet delay towards the victim. As shown in the Figure 4(a), the average throughput is almost similar to the previous experiment. Similar to the case of one legitimate flow, the trend shows that the DDoS attack stream has much worse impact for attack periods of one second and lower because of the minimum RTO value of one second for TCP and the best throughput is again given by HAWK. Here a modified adaptive filtering scheme is used where traffic coming from the same subnet is considered together to generate statistics.



(a) attacks on the same subnet

(b) attacks on multiple subnets

Fig. 4. Effect of distributed shrew attacks from the same or different subnets (five well-behaved TCP flows).

Secondly, let us consider the case when these zombies are on different subnets and trying to collaborate for launching a shrew attack on the victim. Being on different subnets, these zombies would have different packet delays towards the victim. This signifies a more realistic scenario if this kind of shrew attack is to be launched from distributed zombies across the globe.

Four zombies are used, each sending at one fourth the peak rate. The link delays from the four zombies till the GR are chosen as 100, 120, 140, 160 milliseconds. As shown in Figure 4(b), the impact of the attack is reduced in this case. This is due to the fact that now the short attack stream from each malicious source reaches at the bottleneck router RV at different times and the router serves legitimate TCP flows more frequently. But the normalized throughput is still less than the ideal value of one.

The result suggests that the different queuing mechanisms CHOKe and HAWK are unable to produce any significant improvement over Drop Tail scheme. This indicates that for lower attack periods, the effect of shrew attack is more prominent. Though it can logically be assumed that with more number of zombies spread out and each sending at a very small fraction of the bottleneck bandwidth, the legitimate TCP flow aggregate would get fair share of the bandwidth.

4 Conclusions and Future Work

In this paper, we have proposed an adaptive packet-filtering scheme to address the open problem of TCP targeted shrew degradation-of-service attacks. Simulation results demonstrate that our algorithm, called HAWK, outperforms other router assisted queuing mechanisms for combating this special class of network based attacks. Our algorithm is easy to implement and requires a very small storage space that is most of the time only of the order of the number of potential malicious hosts.

Our major on-going work is the implementation of our scheme on the DETER [8] testbed so that we can test the efficacy of the HAWK algorithm in a real environment. Another important research avenue is to extend our scheme to a distributed environment, where multiple routers can interact to identify these attacks even earlier and under wider range of traffic patterns and topologies.

References

1. CERT/CC and FedCIRC, "Advisory CA-2000-01 Denial-of-Service Developments," *Carnegie Mellon Software Eng. Institute*, Jan. 2000.
2. A. Kuzmanovic and E. W. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks— The Shrew vs. the Mice and Elephants," Proceedings of ACM SIGCOMM 2003, Aug. 2003.
3. R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation," INFOCOM 2000, vol. 2, pp. 942–951, Apr. 2000.
4. M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," Proceedings of ACM SIGCOMM '02, Aug. 2002.
5. J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," Proceedings of 11th Word Wide Web Conference.
6. S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397–413, Aug. 1993.
7. S. M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures," Proceedings of the 17th Int'l Conf. Parallel and Distributed Comp. Systems, pp. 536–543, Sept. 2004.
8. DETER and EMIST Projects, "Cyber Defense Technology: Networking and Evaluation," Comm. ACM, pp. 58–61, Mar. 2004. Also from DETER Website: http://www.isi.edu/deter/docs/acmpaper.pdf

Biographical Sketches of Authors:

Yu-Kwong Kwok is an Associate Professor in the Department of Electrical and Electronic Engineering at HKU. Dr. Kwok is currently on leave from HKU and is a Visiting Associate Professor at the University of Southern California. His research interests include Grid computing, mobile computing, wireless communications, network protocols, and distributed computing algorithms. He is a Senior Member of the IEEE. Dr. Kwok is a recipient of the 2003-2004 Outstanding Young Researcher Award given by HKU. He can be reached at ykwok@hku.hk.

Rohit Tripathi received his B.S. degree in Electronics Engineering from Institute of Technology—B.H.U., India in 2000. He received the M.S. degree in Computer Engineering at USC in 2005. He has worked as a Software Engineer at Hughes Software Systems, India, for three years. At Hughes he focused on developing software for IP routing and network based services including Multicasting, VoIP and Network Management. His present research interests are in the area of network security. He can be reached at rohittri@usc.edu.

Yu Chen received his B.S and M.S in Electrical Engineering from Chongqing University, China in 1994 and 1997, respectively. He is presently pursuing the Ph.D. degree in Electrical Engineering at the University of Southern California. His research interest includes Internet security, automated intrusion detection and response systems, and distributed security infrastructure for Grid Computing environment. He can be reached at cheny@usc.edu.

Kai Hwang is a Professor and Director of Internet and Grid Computing Laboratory at the University of Southern California. He received the Ph.D. from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, and distributed computing systems. He has authored or coauthored 7 scientific books and 180 journal/conference papers in these areas. Hwang is the founding Editor-in-Chief of the Journal of Parallel and Distributed Computing. Currently, he is also an Associate Editor of the IEEE Transactions on Parallel and Distributed Systems. He has performed advisory and consulting work for IBM Fishkill, Intel SSD, MIT Lincoln Lab., ETL in Japan, and GMD in Germany. Presently, he leads the NSF-supported ITR GridSec project at USC. The GridSec group develops security-binding techniques for trusted job scheduling in Grid computing, distributed IDS and pushback of DDoS attacks, fuzzy-logic trust models and selfish Grid Computing models, and self-defense software systems for protecting network-centric computing resources. Professor Hwang can be reached at kaihwang@usc.edu or through the URL: http://GridSec.usc.edu/Hwang.html.