Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 66 (2006) 1137–1151
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Collaborative detection and filtering of shrew DDoS attacks using spectral analysis☆

Yu Chen*, Kai Hwang

Internet and Grid Research Laboratory, University of Southern California, Los Angeles, CA 90089, USA

Received 17 December 2005; received in revised form 31 March 2006; accepted 10 April 2006
Available online 12 June 2006

Abstract

This paper presents a new spectral template-matching approach to countering shrew distributed denial-of-service (DDoS) attacks. These attacks are stealthy, periodic, pulsing, and low-rate in attack volume, very different from the flooding type of attacks. They are launched with high narrow spikes in very low frequency, periodically. Thus, shrew attacks may endanger the victim systems for a long time without being detected. In other words, such attacks may reduce the quality of services unnoticeably. Our defense method calls for *collaborative detection and filtering* (CDF) of shrew DDoS attacks. We detect shrew attack flows hidden in legitimate TCP/UDP streams by spectral analysis against pre-stored template of average attack spectral characteristics. This novel scheme is suitable for either software or hardware implementation.

The CDF scheme is implemented with the NS-2 network simulator using real-life Internet background traffic mixed with attack datasets used by established research groups. Our simulated results show high detection accuracy by merging alerts from cooperative routers. Both theoretical modeling and simulation experimental results are reported here. The experiments achieved up to 95% successful detection of network anomalies along with a low 10% false positive alarms. The scheme cuts off malicious flows containing shrew attacks using a newly developed packet-filtering scheme. Our filtering scheme retained 99% of legitimate TCP flows, compared with only 20% TCP flows retained by using the Drop Tail algorithm. The paper also considers DSP, FPGA, and network processor implementation issues and discusses limitations and further research challenges.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Network security; Internet infrastructure; Packet filtering; DDoS attacks; Reduction of quality; Hypothesis test; Digital signal processing (DSP); Distributed computing; Grid systems; Statistical spectral analysis

1. Introduction

Distributed denial-of-service (DDoS) attacks have been identified as a major threat to today's Internet services. These attacks can fully paralyze distributed systems such as ISP core and community networks, collaboration Grids, and P2P systems [5,14,20,24]. A flooding DDoS attack exhausts the network resources such as link bandwidth or CPU power in the victim hosts and thus prevents legitimate users from accessing the victim site.

Recently, a new class of DDoS attacks has been identified on the Internet2 Abilene backbone [9]. These attacks are low-rate, periodic, pulsing type of DDoS attacks. They exploit the transient behavior of a system and gradually reduce the system capacity or service quality. In the literature, this kind of DDoS attacks has been called *shrew DDoS attacks* [15], *pulsing DoS attacks* [16], or *reduction of quality* (RoQ) attacks [12]. For simplicity, we call all of them *shrew attacks* in the sequel.

Shrew attacks exploit, the deficiencies in the *retransmission time-out* (RTO) mechanism of TCP flows. It throttles legitimate TCP flows by periodically sending burst pulses with high peak rate in a low frequency. As such, the TCP flows see congestion on the attacked link every time it recovers from RTO. Indeed, such a shrew attack may reduce the throughput of TCP applications down to almost zero [15]. Given that more than 80% of traffics on the Internet today are using TCP protocol, a majority of existing applications and commercial services are at stake.

☆ This research was supported by NSF Grant 0325409.
* Corresponding author. Fax: +1 213 740 4418.
E-mail addresses: cheny@usc.edu (Y. Chen), kaihwang@usc.edu (K. Hwang).

0743-7315/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2006.04.007

Table 1
Notations and abbreviations used in this paper

Notations	Brief definition	Notations	Definition
T	Attack period (s)	CDF	Collaborative detection and filtering
R	Attack burst rate (K Packets/s)	RoQ	Reduction of quality attacks
L	Attack burst width (s)	DDoS	Distributed denial of Service attacks
R_{d}	Anomaly detection rate	DSP	Digital signal processing
R_{fp}	False positive alarm rate	CTS	Cumulative traffic spectrum
α	Local detection threshold	SFT	Suspicious flow table
β	Global detection threshold	MFT	Malicious flow table
γ	Packet filtering threshold	PSD	Power spectrum density
$\mu(f)$	Template average distribution	ASD	Amplitude spectrum density
$\Phi(f)$	Cumulative traffic spectrum (CTS)	N_{a}	Sample distribution of attack traffic streams
$\Psi(f)$	Cumulative amplitude spectrum (CAS)	N_{o}	Sample distribution of legitimate traffic streams
p	Attack pivot frequency	N_{fa}	Sample traffic flow distribution of shrew attacks
y	CTS spectral pivot point	N_{fo}	Sample distribution of legitimate TCP flows
z	CAS spectral pivot point	θ	Normalized TCP throughput

The low-rate shrew DDoS attacks have high peak rate, but a low average rate to exhibit a stealthy nature. The shrew attacks can damage the victim for a long time without being detected [12]. Countermeasures developed for flooding DDoS attacks are not effective to combat against shrew attacks [12,15,18]. Being masked by background traffic, shrew attacks are very difficult to detect in the time domain, but the situation is not necessarily true in the frequency domain.

We consider a *traffic flow* as a set of packets with the same source and destination IP addresses and port numbers, all traveling in the same direction and applying the same protocol. Each traffic flow can be uniquely identified with a 5-tuple of identifiers. A *traffic stream* is formed with the set of packets arriving at a router at the same physical port during a given time window. Thus a traffic stream may contain multiple TCP or UDP traffic flows.

The shrew attack packets are embedded in TCP or UDP traffic flows. We will consider traffic streams with and without shrew attacks. The trick is to detect the presence of shrew attacks in traffic streams and to filter out attack packets at the refined flow level rather at the stream level.

In this paper, we present a new approach by combining *discrete Fourier transform* (DFT) and a hypothesis test framework to cope with shrew attacks. By calculating the autocorrelation sequence of sampled time series and converting them into frequency-domain spectrum using DFT, we find that the power *spectrum density* (PSD) of a traffic stream containing shrew attacks has much higher energy in low-frequency band than that appeared in the spectrum for legitimate TCP/UDP traffic streams.

Based on this distinction, we develop a distributed *collaborative detection and filtering* (CDF) scheme to detect and segregate the shrew attack flows from legitimate TCP/UDP traffic flows. In addition to software implementation, the scheme can be implemented by network processor or reconfigurable hardware [3]. The DSP hardware pushes spectral analysis down to the lower packet-processing layer.

If the packets are processed by hardware and malicious flows are filtered out timely, the router workload will not increase much at the presence of shrew attacks. Notations, symbols, and abbreviations used in this paper are summarized in Table 1. Only brief definitions are given here, details are given in subsequent sections.

The rest of the paper is organized as follows: Section 2 reviews several directly related research works. Section 3 presents the theoretical foundation of collaborative anomaly detection. Section 4 discusses spectral analysis techniques for collaborative detection. Section 5 presents an adaptive filtering algorithm for cutting off malicious shrew attack flows.

Our simulation setup and performance results are reported in Section 6. Section 7 discusses scalability and implementation issues. We will assess the limitations of using the CDF work to support large-scale network security. Finally, we summarize the research contributions and make a few suggestions for further research.

2. Related previous work

Over the years, a plethora of research has been reported in the area of DDoS defense and traffic control [18,21,24]. While most previous work analyzed Internet traffic patterns in the time domain, researchers have explored the usage of digital signal analysis technology in traffic analysis for network security enforcement [4,7,13,22].

Due to different protocols applied, the periodicity of traffic could be used as a signature for traffic monitoring or for attack detection. The lack of periodicity could indicate that flooding DoS attacks are raging on [7]. In contrast, the PSD of multi-sourced flooding DDoS attacks are distributed in lower frequency band [13]. This could be used as a signature for traffic monitoring and attack detection.

Kuzmanovic and Knightly [15] pioneered the characterization of TCP targeted shrew attacks. They studied the rationale of the shrew attacks and identified the critical parameters that affect the TCP flows. They indicated the limitation of exist-

ing DDoS defense mechanisms against shrew attacks. Chertov et al. [8] investigated the effectiveness of shrew attacks through simulation using NS-2 simulator and emulation experiments on DETER testbed [10].

Taking both theoretical analysis and intensive simulation, Luo et al. [17] explored the impact of shrew attacks against TCP performance under different queue management schemes. Results of both [8,17] validate that variants of shrew attacks can still be effective, if the attack frequency is not precisely tuned to the retransmission timeout.

Sun et al. [25] suggested detecting shrew attacks by matching pattern with prestored attack signature. They use a *deficit round robin* (DRR) algorithm to allocate bandwidth and protect legitimate flows. However, their method cannot distinguish malicious from legitimate flows. Legitimate flows thus suffer in the rate-limit packet filtering process. Previously, we have developed a packet filtering scheme [6] to cut off malicious shrew attack flows.

Luo and Chang [16] have proposed to detect low-rate TCP-targeted DoS attacks using a wavelet approach. They observed anomalies in fluctuation of incoming traffic rate and the declining of outgoing TCP ACKs incurred by pulsing attacks. Previous work has revealed most time-domain characteristics of shrew DDoS attacks [15,16,25], but no model available to describe their properties in the frequency domain. This paper intends to close up this technical gap.

3. Characteristics of shrew DDoS attacks

Based on differences found between traffic spectrum of normal TCP/UDP streams and that of shrew attack streams, we propose to use the cumulative energy distribution function to detect the shrew attacks. Through NS-2 simulation experiments over 8000 randomly generated sample traffic streams, we generate the attack template to implement the new CDF scheme for defense against shrew DDoS attacks.

3.1. The CDF architecture and processing stages

We studied the shrew attack characteristics through intensive simulation using the NS-2 simulator [19], a widely recognized packet-level discrete event simulator. Fig. 1(a) shows the typical network environment to deploy in which our CDF scheme is implemented. Our NS-2 simulations were carried out with many topologies generated by the GT- ITM toolkit from Georgia Tech. [11]. We apply each topology for at least 1000 experiments with shrew attack datasets similar to those used by Chertov et al. [8], Kuzmanovic and Knightly [15], and Sun et al. [25].

A shrew attack stream is modeled by three major parameters including *period of attack T*, *width of burst L*, and the *burst rate R* [15]. Specifically, we generate shrew attack flows with a period T between 0.5 and 3.0 s, the burst period L is in the range (30–90 ms). For single-source attacks, the burse rate R varies in 0.5–2 MB/s. In distributed attacks from multiple sources, R varies in 0.1–2 MB/s. The background traffic without shrew attacks are generated from our analysis of Abilene-I trace dataset released by the PMA Project [1]. This dataset is the first public OC48c backbone trace. We consider routers belonging to the same *autonomous system* (AS) shown by the Internet subnet at the center of Fig. 1(a).

The routers in the same AS collaborate with each other, cooperatively, to defend against the shrew attacks. Both legitimate traffic streams and attack traffic streams are generated from the edge networks. The background traffic consists of large number of TCP and UDP flows, some of them are attack free and some are embedded with attack packets. The victim is attached to the router market as R0. Both legitimate users and attackers are scattered at edge networks.

The CDF scheme is built with a *training process* and a *testing process* in a cascade as illustrated in Fig. 1 (b). The training process consists of a *template generator* and a *pivoting frequency* estimator (Algorithm 1). We have collected 8000 sample traffic streams, half of which are legitimate without attacks (S_o) and the other half with shrew attacks (S_a). The template generator generates *attack template* threshold parameters (α, β, γ) and a *Gaussian template* distribution over 4000 streams in the attack set S_a.

This template distribution is characterized by *template average distribution* denoted by $\mu(f)$ in Table 1. The template generation process will be detailed in Section 3.3. The template spectrum of each stream in sample sets S_a and S_o is characterized by two Gaussian distributions $N_a(\mu_a, \sigma_a^2)$ and $N_o(\mu_o, \sigma_o^2)$, where the subscripts, $_a$ and $_o$ correspond to training streams with and without shrew DDoS attacks. Similarly, the Gaussian distributions $N_{fa}(\mu_{fa}, \sigma_{fa}^2)$ and $N_{fo}(\mu_{fo}, \sigma_{fo}^2)$ are sampling over two training sets at the flow level.

The testing phase is built with three algorithms for determining *pivoting frequency*, *detection of malicious streams*, and *filtering of attack flows* amid legitimate flows. In the testing phase, 4000 incoming traffic streams $X(i)$, were tested to validate the effectiveness of the CDF scheme.

In a way, the complete scheme is based on *template matching*. These traffic patterns are statistically generated to cover a mixture of normal TCP and UDP flows with some shrew attack streams. To detect whether shrew attack streams are embedded in an incoming traffic stream $X(i)$, the energy spectrum $\Phi(f)$ is generated by the DFT engine.

The energy $\Phi(p)$ at the pivotal frequency, where the gap between $\Phi(f)$ and $\mu(f)$ is the maximum, is computed by the *pivoting module* (Algorithm 1). The mean value of the traffic spectrum $\Phi(f)$ is compared with the template average $\mu(f)$ in the detection module (Algorithm 2).

The template distributions $N_a(\mu_a, \sigma_a^2)$ and $N_o(\mu_o, \sigma_o^2)$ are used for anomaly detection. After the attack stream is detected, attack alerts will be sent to the *filtering module* (Algorithm 3), which segregates the shrew attack flows from normal TCP flows. The filtering process uses an *amplitude spectrum* $\Psi(f)$ against two flow-level distributions ($N_{fa}(\mu_{fa}, \sigma_{fa}^2)$ and $N_{fo}(\mu_{fo}, \sigma_{fo}^2)$).

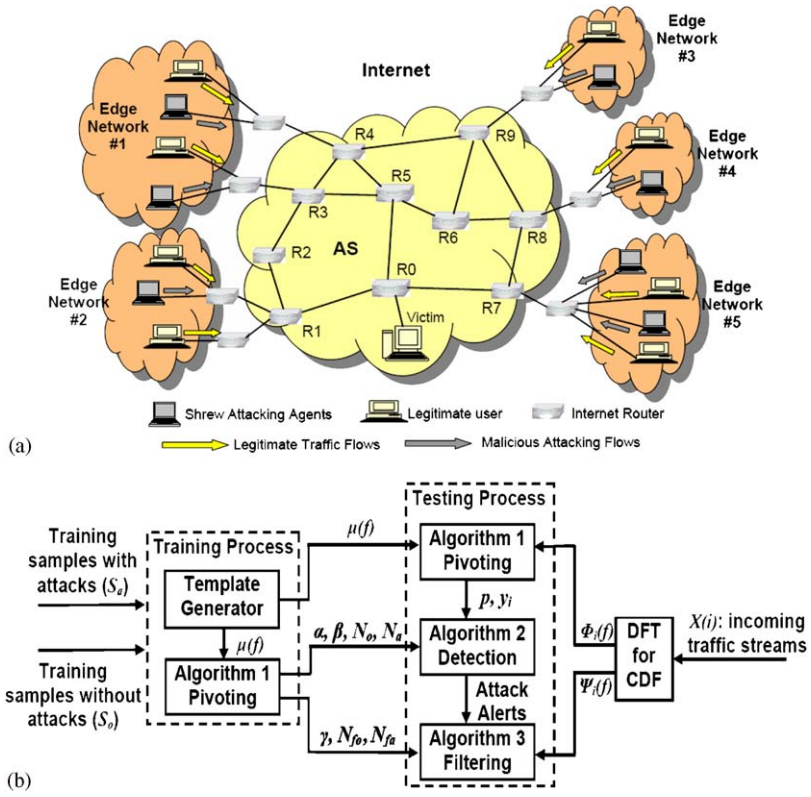

Fig. 1. The collaborative detection and filtering (CDF) architecture for defense against shrew DDoS attacks over multiple routers in the same autonomous system: (a) the CDF implementation network environment; (b) traffic processing stages for CDF scheme.

3.2. Shrew DDoS attack characteristics

Fig. 2 compares the time series of legitimate traffic streams (a), periodic pulsing shrew attack stream (b), and the mixture of legitimate and attack streams (c). Fig. 2(b) presents a shrew attack stream modeled by T, L, and R. When a legitimate TCP stream and a shrew attack stream are both heading for the same destination, we observe: (1) The shrew attack peak rate remains constant while the TCP flow may increase linearly; (2) the shrew stream arrives at the destination periodically, while the TCP flow arrives continuously.

It is difficult to detect periodic pulses using traffic volume analysis method in the time domain. This is because the average bandwidth consumption differs very little between attack-free and attack streams. For higher throughput, the TCP protocol uses a predefined value of RTO with a fixed incrementing pattern [23].

The shrew attacks take advantage of this RTO recovery mechanism by adjusting its attack period and occupying the link bandwidth. The legitimate TCP flows always see a heavily burdened link, when they try to send packets. Thus legitimate TCP flows must undergo a RTO recovery and their sending rate is thus reduced to as low as zero.

The period T is the time interval between two consecutive attack pulses. The burst width L indicates the time period during which attackers send packets in high rate. The burst height exhibits the peak rate by which attacking flow is sent. The period T is calculated by the estimated TCP RTO timer implementation from trusted sources. During the burst with a peak rate R, the shrew pulses create a burst and severe congestion on the links to the victim. The legitimate TCP flows must decrease their sending rate as governed by congestion control mechanism, accordingly.

Figs. 2(a) and (c) compares the sampled time series of packet arrivals of two scenarios: Six TCP flows without any shrew attack stream embedded (Fig. 2(a)), and the same six TCP flows with one shrew attack streams (Fig. 2(b)), which use the attack period $T = 1$ s, burst width $L = 70$ ms, and the burst rate $R = 120$ K packets/s as shown in Fig. 2(b).

3.3. Autocorrelation spectral analysis

We take the number of packet arrivals at the router as the discrete signal series and sample it with a period of 1 ms. We use the highest frequency of 500 Hz. Our sampling effectively plays the role of low-pass filter that gets ride of high frequency noises. Another observation is that the sample also includes packets from legitimate flows. These packets are not necessarily all targeting at the same victim.

The packet arrivals in each training stream or in each incoming traffic stream are modeled by a random process: $\{x(t), t = n\Delta, n \in N\}$, where Δ is a constant time interval, which we assume 1 ms. N is a set of positive integers, and at each time point t, $x(t)$ is a random variable, representing the total number of packets arrived at a router in $(t - \Delta, t]$.

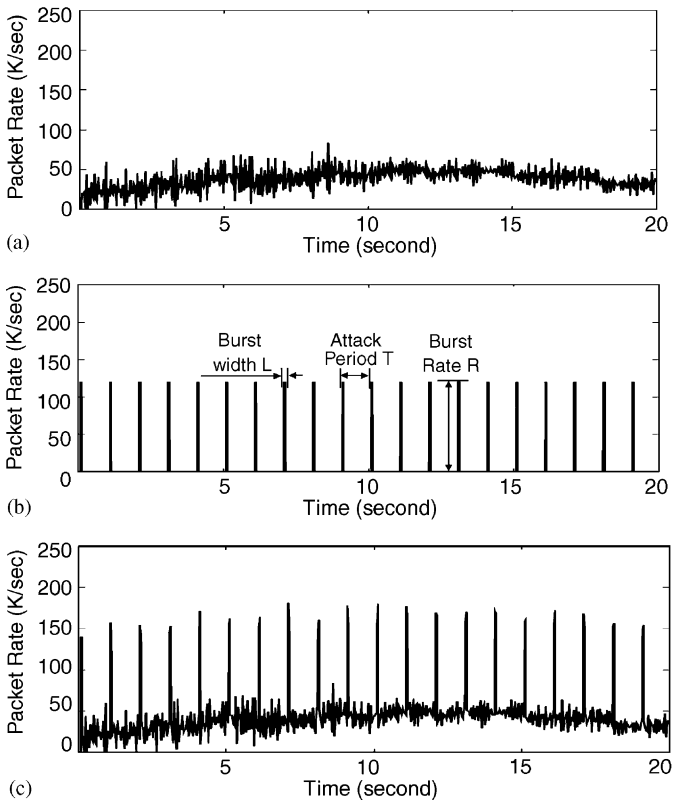

Fig. 2. Three traffic streams under three scenarios—the shrew DDoS attack stream in part (b) hides in six legitimate traffic flows in part (a) to form the traffic stream in part (c): (a) a sample traffic stream consisting of six legitimate TCP and UDP flows without any attack packet; (b) a shrew DDoS flow characterised by high burst rate (R), long attack period (T), and short burst width (L); (c) six normal TCP/UDP flows in part (a) mingled with a single shrew DDoS attack stream from part (b).

This random process was referred to as the *packet process* [7]. We assume a wide sense stationary random process. We define the autocorrelation function of the random signal $x(t)$ in discrete time as follows:

$$R_{xx}(m) = \frac{1}{N-m} \sum_{n=0}^{N-m+1} [x(n)x(n+m)]. \quad (1)$$

The $R_{xx}(m)$ captures the correlation of the packet process and itself at interval m. If there is any periodicity exist, autocorrelation function is capable of enforcing it. The next step is to figure out the periodicity embedded inside the autocorrelation functions. We convert the autocorrelation time series by discrete Fourier transform to generate the *power spectrum density* (PSD) as follows:

$$PSD(f) = DFT(R_{xx}(m), f) = \frac{1}{N} \sum_{n=0}^{N-1} R_{xx}(m) \times e^{-j2\pi fn/N},$$
$$f = 0, 1, 2, \ldots, N-1. \quad (2)$$

3.4. Traffic spectrum and template distribution

A shrew attack stream escapes detection by occupying only a small share of the bandwidth, however, its properties in frequency domain cannot be hidden. After DFT, the PSD captures the periodical pattern of shrew attack stream in the frequency domain.

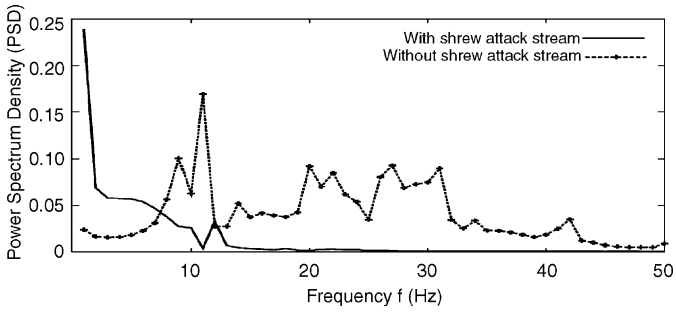

Fig. 3. Comparison of normalized traffic density (PSD) of two traffic streams with and without shrew attacks.

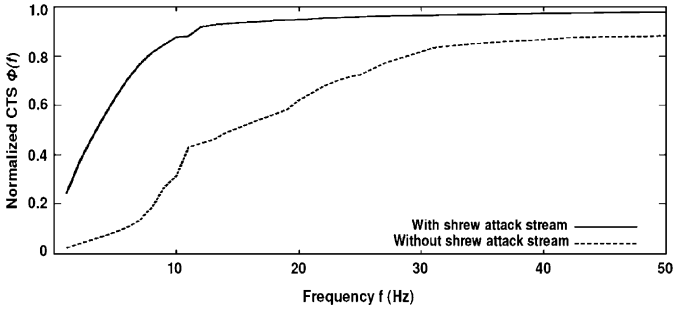

Fig. 4. Comparison of the cumulative energy spectrum of two traffic streams with and without shrew attacks.

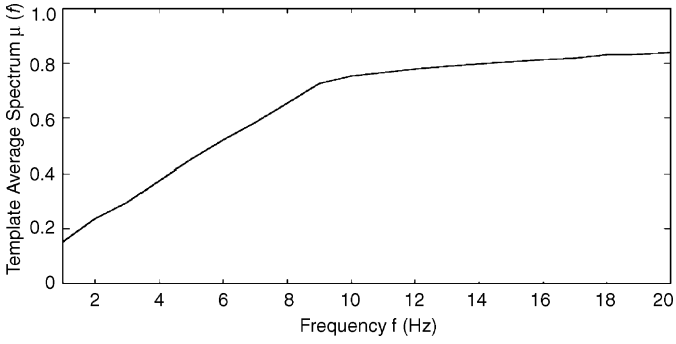

Fig. 5. Average attack template distribution over a low frequency band for all attack streams in training set S_a.

Fig. 3 compares the *traffic density* (or PSD) for two traffic stream patterns corresponding to with and without embedded shrew attacks. It is clear that the embedded shrew attack stream pushes the solid-line PSD curve towards the lower frequency band, while the no-attack stream has a wider frequency range of the traffic density in the dash-line curve in Fig. 3.

After integration of the PSD function over a given frequency range, the resulting energy spectrum is used to determine the ratio of energy accumulated to a given frequency point. This

Table 2
Gaussian distribution of attack template

Freq. f(Hz)	1	3	5	7	9	10	11	13	15	17	19	20
Mean $\mu(f)$	0.15	0.30	0.45	0.58	0.73	0.75	0.77	0.79	0.81	0.82	0.83	0.84
St. dev. $\sigma(f)$	0.11	0.13	0.14	0.14	0.14	0.14	0.13	0.13	0.13	0.10	0.09	0.08

corresponds to the area below the PSD curve. This *cumulative traffic spectrum* (CTS) $\Phi(f)$ is obtained below:

$$\Phi(f) = \sum_{i=1}^{f} PSD(i) \Big/ \sum_{i=1}^{\max} PSD(i). \quad (3)$$

As plotted in Fig. 4, the y-axis is the CTS normalized with respect to the total energy in the whole spectrum [0,500] Hz. For a typical attack stream, more than 90% of the energy is distributed below 20 Hz.

In contrast, for the normal traffic streams without attacks, only 60% the energy is located below 20 Hz. This implies that the energy spectrum offers a sharp distinction to distinguish whether a sampled traffic spectrum contains shrew attacks or not.

Through intensive NS-2 simulation experiments, we obtained 4000 template CTS spectra over the attack dataset S_a. Based on the central limit theorem [2], the template spectrum conforms to a Gaussian distribution $N(\mu, \sigma)$, where μ and σ are the mean and standard deviation.

$$\mu(f) = \sum_{i=1}^{n} \Phi_i(f)/n \;\; \sigma(f) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\Phi_i(f) - \mu(f))^2}, \quad (4)$$

where $n = 4000$ is the size of sample space of S_a, and $\Phi_i(f)$ is the CTS of the ith sample stream.

Fig. 5 plots the template energy distribution for the mean value $\mu(f)$ over the low-frequency range up to 20 Hz. The template spectrum curve sits between the two traffic spectrum curves in Fig. 4. We use this property to distinguish shrew attack stream from regular traffic streams by checking the spectrum gaps with the template average distribution $\mu(f)$.

Table 2 presents the attack template energy distribution on frequency band lower than 20 Hz. At each frequency f, the mean $\mu(f)$ and standard deviation $\sigma(f)$ are listed. These table entries will be used to perform the CDF processes as template references in Algorithm 2. Note, the above spectral analysis can be o applied over both training traffic and incoming traffic streams to be tested.

4. Collaborative anomaly detection

We developed a scheme for *collaborative anomaly detection* using hypothesis test atop the spectral introduced in Section 3.4. Supported by alerts exchange among multiple routers, this scheme detects the shrew attack streams hidden among legitimate TCP/UDP flows. We start with the determination of attack pivot frequency, and then we specify the algorithms developed to perform the collaborative detection processes.

4.1. Pivot frequency in traffic streams

Having established the attack template database, we need to answer a basic question: Which frequency point can result in the highest detection accuracy? This point is called the *pivot frequency*, which varies with each incoming traffic stream. To detect accurately, we must first determine the *pivot frequency* p.

We develop Algorithm 1 to determine the pivot frequency p and a *spectral pivot point* $y = \Phi(p)$ associated with each traffic stream characterized by its CTS spectrum $\Phi(f)$. The pseudo-code specification of Algorithm 1 is given below.

Algorithm 1. Determination of pivot frequency in a traffic stream

Input: $\mu(f)$: Average template CTS spectrum.
$\Phi_i(f)$: The CTS computed from an incoming traffic stream S_i.
Output: p: Pivot frequency for the traffic stream S_i.
$y = \Phi(p)$: Spectral pivot at pivot frequency p
Procedure:
01: Initialize the frequency window (0, 20 Hz)
02: **while** scan through the detection window
03: Calculate gaps at each frequency point $d(f) = \mu(f) - \Phi(f)$
04: **endwhile**
05: Find ρ such that $d(p) = Max\{d(f)|0 \leqslant f \leqslant 20\ Hz\}$ and compute $y = \Phi_i(p)$
06: **return** pivot frequency p and spectral pivot point $y = \Phi_i i(p)$

After sampling a time series of incoming stream, we generate its CTS $\Phi(f)$ through DFT. Then the gap $d(f) = \mu(f) - \Phi(f)$ is calculated using Table 2 entries over all frequencies in a spectral window up to 20 Hz. We obtain the pivot frequency p that must satisfy the following at the maximum gap point:

$$d(p) = Max\{d(f) = \mu(f) - \Phi(f)|\text{for all } f \leqslant 20\,\text{Hz}\}. \quad (5)$$

The *spectral pivot point* $y = \Phi(p)$ will be used in the shrew attack detection. Fig. 6 illustrates how to detect the pivot frequency using the traffic samples shown in Figs. 2(a) and (c). The spectrum of the shrew attack stream is indicated by the top CTS curve. The CTS curve of the attack-free stream lies below the template average spectrum curve at the middle.

For the stream containing shrew attacks, the maximum gap between its CTS curve and the template CTS curve is located at the pivot frequency $p = 6$ Hz. The spectral pivot is thus found at a spectral pivot point denoted as $y_a = \Phi(p) = \Phi(6\,\text{Hz}) = 0.7$. Similarly, for the attack-free stream, the pivot frequency $p = 8$ Hz and $y_o = \Phi(8\,\text{Hz}) = 0.19$.

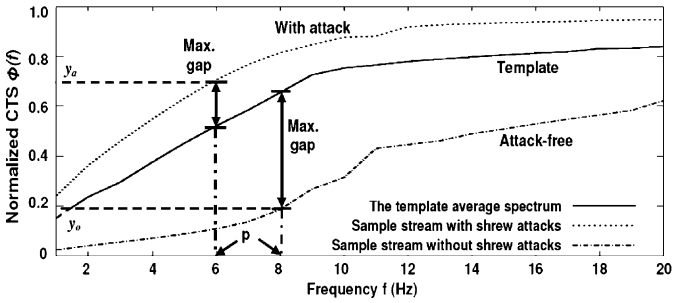

Fig. 6. Determination of the pivot frequency p and the two spectral pivot points y_o for the attack-free stream in Fig. 2(a) and y_a for the attack stream in Fig. 2(c).

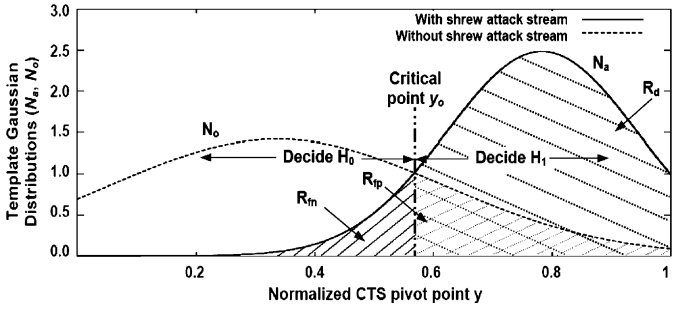

Fig. 7. Template distribution of attack/no-attack streams at the attack pivot frequency.

4.2. Hypothesis test for anomaly detection

We consider two hypothetic events: H_o for an attack-free stream and H_1 for a traffic stream with shrew attacks. Using Algorithm 1, we obtain the *template distribution* $\Phi(p)$ by training 8000 training traffic streams. By central limiting theorem [2], we assume two Gaussian distributions $\Phi(p_o) : N_o(\mu_o, \sigma_o^2) = N(0.33, 0.27^2)$ and $\Phi(p_a) : N_a(\mu_a, \sigma_a^2) = N(0.78, 0.16^2)$ for 4000 attack-free and 4000 streams containing shrew attacks, respectively. These two Gaussian distributions N_o and N_a are plotted in Fig. 7.

The solid-line distribution plotted on the right corresponds to N_a for all attack streams. The dash-line distribution plotted on the left plots N_o, is resulted from all attack-free training streams. In Fig. 7, the detection of a shrew attack stream (event H_1) is under the N_a curve to the right of y_o. The detection of an attack-free traffic stream (event H_o) is under the N_o curve to the left of y_o. We need to choose a local detection threshold α to maximize the *anomaly detection rate* R_d and to minimize the *false positive rate* R_{fp} defined by the following two probability functions:

$$R_d = Prob[H_1|H_1] = \int_{y_o}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_a} \exp\left\{-\frac{(y-\mu_a)^2}{2\sigma_a^2}\right\} dy, \quad (6a)$$

$$R_{fp} = Prob[H_1|H_0] = \int_{y_o}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_o} \exp\left\{-\frac{(y-\mu_o)^2}{2\sigma_o^2}\right\} dy. \quad (6b)$$

Essentially, R_d is the *successful detection probability* that a true alarm is raised, when there is actually a shrew attack. The

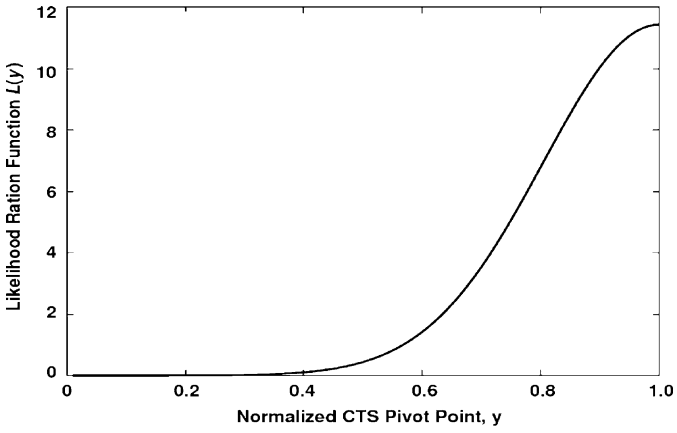

Fig. 8. Variation of the likelihood function with respect to the normalized spectral pivot value.

R_{fp} is the probability of raising a false alarm for the misdetection of an attack-free traffic stream.

In order to tell whether there is any shrew attack stream embedded in a legitimate flow, we set up our hypothesis test rule. Defined below is a *likelihood ratio* $L(y)$ of the pivot point y by the ratio of the two probability density functions, $p_a(y)$ and $p_o(y)$, of the event H_1 and event H_o, respectively,

$$L(y) = \frac{p_a(y)}{p_o(y)} = \frac{\sigma_o}{\sigma_a} \exp\left(\frac{1}{2}\left[\left(\frac{y-\mu_o}{\sigma_o}\right)^2 - \left(\frac{y-\mu_a}{\sigma_a}\right)^2\right]\right), \quad (7)$$

where $p_a(y)$ is the probability density of y in the Gaussian distribution N_a and $p_o(y)$ is the probability density of y in the Gaussian distribution N_o defined as:

$$p_a(y) = \frac{1}{\sqrt{2\pi}\sigma_a} \exp\left\{\frac{(y-\mu_a)^2}{2\sigma_a^2}\right\}, \quad (8a)$$

$$p_o(y) = \frac{1}{\sqrt{2\pi}\sigma_o} \exp\left\{-\frac{(y-\mu_o)^2}{2\sigma_o^2}\right\}. \quad (8b)$$

The variation of the likelihood ratio $L(y)$ at the spectral pivot point is plotted in Fig. 8. The larger is the $L(y)$, the pivot point y is more likely to confirm to the N_a distribution. This implies that the stream is more likely to contain some shrew attack flows.

The *local detection threshold* α is selected to equal certain $L(y)$ corresponding to a well chosen pivotal point y. In another word, given desired detection rate R_d^*, the local detection threshold α is chosen as

$$\alpha = L(y) \text{ for a pivot point } y \text{ such that } R_d \geqslant R_d^*. \quad (9)$$

As shown in Fig. 7, there is an overlapped area under the two Gaussian distributions. To the right of a, we have the R_{fp} in the overlapped area. To the left of α, we have the false negative area corresponding to the misdetection of a real attack as N_o attack. This implies that when $L(y) > \alpha$, the hypothesis H_1 is true. Otherwise, the hypothesis H_0 is true. Based on the template distributions obtained from 8000 training sample streams, we calculated R_d and R_{fp} using Eqs. (6a) and (6b) as plotted in Fig. 9.

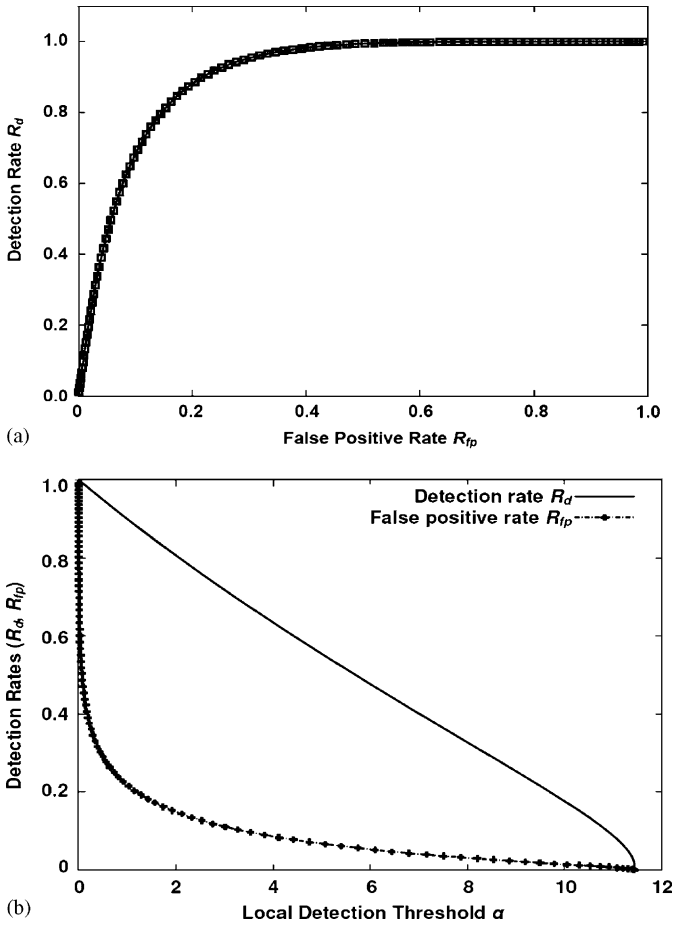

Fig. 9. Plots of the anomaly detection rate R_d and false positive rate R_{fp} averaged over 8000 sample traffic streams in the template database: (a) ROC curve of R_d vs. R_{fp}; (b) effects of variation in local threshold α.

Fig. 9(a) plots the *receiver operating characteristic* (ROC) curve to show the tradeoff between R_d, and R_{fp}. Fig. 9(b) plots the variation of R_d, and R_{fp} as α increase from 0 to 12 in the range of y. For example, to achieve a successful detection rate of 95% with $\alpha = 0.5$, we need to tolerate a false positive rate of 30%.

It is clear that R_d decreases slowly and R_{fp} decreases rapidly with increasing value of α. Consequently, we need one more threshold to keep the detection rate sufficiently high with a small false-positive rate. This has motivated us to develop the collaborative detection scheme.

4.3. Collaborative anomaly detection

With the hypothesis testing, we present in Algorithm 2 a distributed detection scheme for shrew attacks by using multiple collaborative routers. The detection measurements are deployed in upstream routers that are a few hops away from the victim server, because low-rate attacks may throttle legitimate TCP flows destined to the same victim. However, before reaching the target, the shrew attack streams may occupy only a small bandwidth share even at its peak rate, which make it difficult to detect correctly.

Algorithm 2. Collaborative anomaly detection.

Inputs: y_i: The spectral pivot $\Phi(p)$ for an input stream S_i
α: Local detection threshold
β: Global detection threshold
N_a: Sample traffic distribution with shrew attacks
N_o: Sample traffic distribution without shrew attacks

Output: Traffic stream S_i contains either shrew attacks or not

Procedure:
01: Compute $L(y_i)$ using Eq. (6).
02: **Case** 1: $L(y_i) > \alpha$
03: The stream S_i contains shrew attacks
04: Send out alert to collaborative routers
05: Call packet filtering routine (Algorithm 3)
06: **Case** 2: $\beta < L(y_i) < \alpha$
07: Check the alerts from cooperative routers
08: **if** alerts come from cooperative routers, **then**
09: Call packet filtering routine (Algorithm 3)
10: **endif**
11: **Case** 3: $L(y_i) < \beta$
12: The stream S_i contains no shrew attacks

The traffic statistics collected at neighboring routers could be used for a router to verify its detection result. Previously, to reduce false positive alarms, we have to tolerate some higher false negative alarms by choosing a larger local detection threshold α. Algorithm 2 tries to make up this deficiency by selecting a *global cooperative threshold* β that is smaller than α for a guaranteed higher detection rate.

While the likelihood ratio function $L(y)$ is higher than the local threshold α, the router would start the filtering mechanism and multicast alert to its neighbors. Routers whose $L(y)$ is smaller than a but larger than β do not generate alerts. They need to decide whether to start local filtering mechanism by analyzing the resources of alerts received from their neighbors. If $L(y)$ is less than β, there is nothing suspicious. With distributed shrew attacks, alerts from immediate routers are connected to the receiver.

Consider the scenario in Fig. 10(a). Each dot stands for a router in the AS, and the black dots are routers that have detected shrew attack streams embedded in legitimate flows ($L(y) > \alpha$). They multicast the alerts to the cooperative neighbors. The concern of the cooperative range among multiple routers will be studied later through experimental results. To simplify the discussions, we consider in Fig. 10 that each alert is sent to neighbors in two-hop distance. The white dots are routers whose likelihood ratio $L(y)$ is lower than the local threshold α, so they do not generate alert.

Moreover, each node is the root of a *detection tree* that contains all of its neighbors in two-hop distance. The A, B and E is such nodes in Fig. 10(b). These trees keep the multicast group records that each root knows to whom its alert is sent. Distributed attacks are launched from zombies located widely and multiple shrew attack streams approach the victim(s) located in the AS. Once a router detects $L(y) > \alpha$ (black dots in Fig. 10), there are shrew attack streams detected. It would

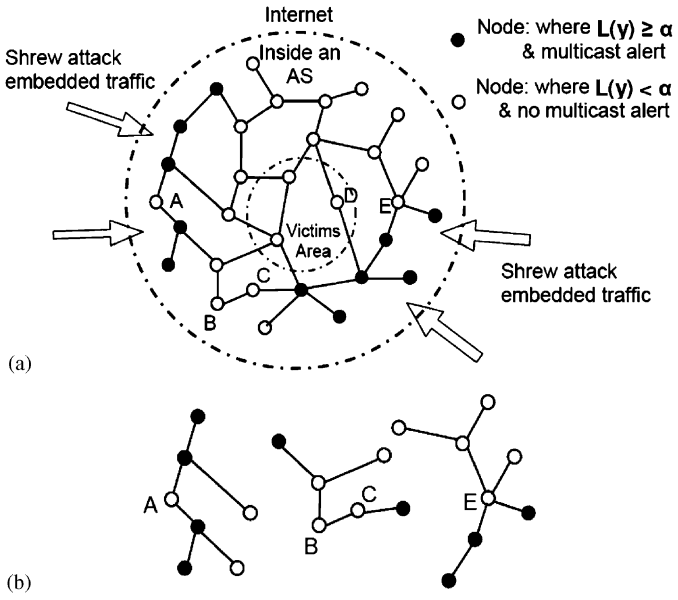

Fig. 10. Collaborative detection scheme and use of detection trees: (a) distributed shrew attacks from two sides of the AS; (b) three detection trees.

multicast an alert along its detection tree and launch countermeasure to identify malicious flows and cut them off timely.

However, shrew attack streams do not appear evenly on all edge routers. Algorithm 2 gives routers a clue to check whether they are in the attacking range. For example, when $L(y) < \alpha$ at node A, alerts from neighbors warn an attack going on. In addition, the decision tree let node A has a vision of where these alerts come from. Node A realizes that it is located at the center of the attacking area.

If $L(y) > \beta$, node A will launch countermeasure against the attacks. In contrast, node B receiving two alerts will not take any further action. All its immediate neighbors have not raised any alert, this leads to a conclusion that there are suspicious flows entering the AS, but node B is not in the attacking range. However, decision trees and received alerts will tell nodes C, D and E that they are at the attacking area. If their local $L(y) > \beta$, then they need to start flow filtering algorithm accordingly.

5. Amplitude filtering of shrew attack flows

Once a router detected there are shrew flows embedded among incoming legitimate traffic streams, it needs take further actions to segregate the malicious attacking flows from legitimate ones and block them. In this section, we present an effective shrew-filtering algorithm based on the flow level analysis of energy distribution over frequency band and hypothesis test.

Essentially, this approach is same as we did in detection mechanism. The major difference lies in the utilization of amplitude spectrum instead of PSD, although they both describe the energy distribution of signal series. The reason is that we found that amplitude spectrum can separate shrew attack flows from TCP flows much clear at flow level.

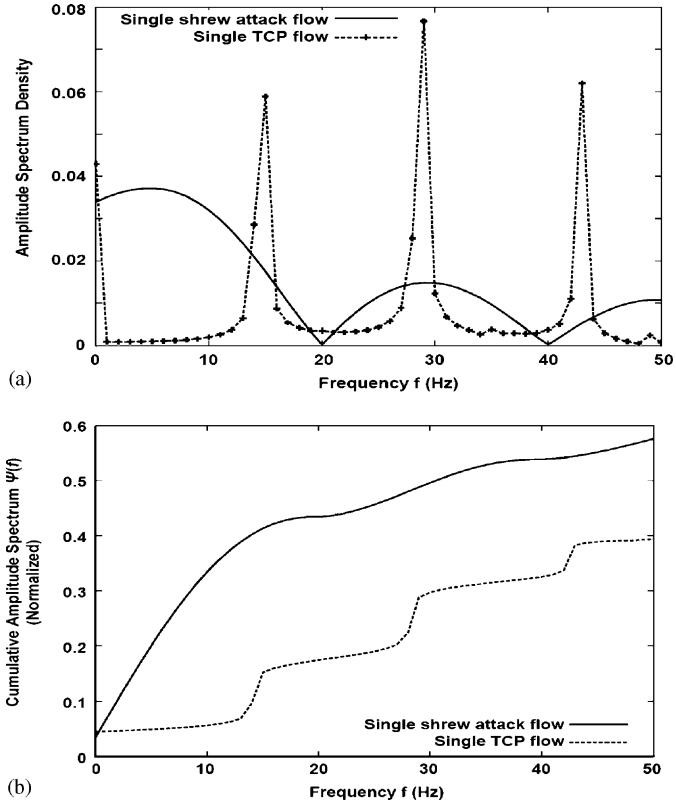

Fig. 11. Amplitude spectral analysis at flow level for a shrew attack flow and for a TCP flow: (a) amplitude spectrum of a shrew attack flow and of a typical TCP flow; (b) CAS of a shrew attack flow and of a typical TCP flow .

5.1. Normalized amplitude spectral analysis

To perform flow level analysis, we treat the number of packet arrivals for each flow as a signal series and sample it every 1 ms. Thus, for each flow time series $x(i)$, we convert it to an *amplitude spectrum density* (ASD) using DFT as follows:

$$ASD(f) = DFT(x(i), f) = \frac{1}{N}\sum_{n=0}^{N-1} x(i) \times e^{-j2\pi fn/N}, \quad f = 0, 1, 2, \ldots, N-1. \tag{10}$$

After integration of the ASD over a given frequency range, the resulting *cumulative amplitude spectrum* (CAS) determines the ratio of energy accumulated to a given frequency point. This is the area below the ASD curve. The normalized CAS $\Psi(f)$ is defined below:

$$\Psi(f) = \sum_{i=1}^{f} ASD(i) \Big/ \sum_{i=1}^{\max} ASD(i)\ . \tag{11}$$

Fig. 11 presents the ASD and the CAS of a typical TCP flow and of a shrew attack flow.

Comparing the normalized CAS $\Psi(f)$ shown in Fig. 11(b), the distance between the two CAS curves is maximum at frequency lower than 20 Hz. Almost 50% of the total energy of a shrew attack flow is located in the major peak ending around 20 Hz in Fig. 11(a).

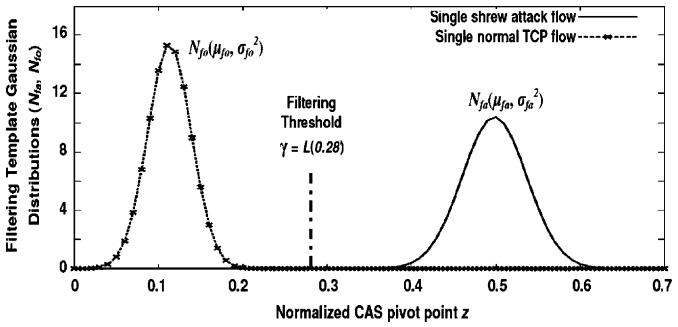

Fig. 12. Normalized cumulative ASD of a shrew attack flow compared with that of a TCP flow at the pivot point.

In contrast, in the same frequency band, it is less than 15% of the total energy located of a normal TCP flow. Similar to the stream level template construction in Section 4, the training process generates the CAS pivot $z = \Psi(p)$ distribution by executing Algorithm 1 to determine the flow-level pivot frequency.

The obtained z assumes a Gaussian distribution according to the central limiting theorem. Fig. 12 shows the Gaussian distribution $N_{\text{fa}}(\mu_{\text{fa}}, \sigma_{\text{fa}}^2) = N(0.50, 0.04^2)$ of a single shrew attack flow and the Gaussian distribution $N_{\text{fo}}(\mu_{\text{fo}}, \sigma_{\text{fo}}^2) = N(0.11, 0.03^2)$ of a normal TCP flow. There is no overlap between the CAS distribution of a normal TCP flow and that of a single shrew attack flow. Actually, the gap between the two mean values is greater than 3.29 σ. A 3σ error level gives us a confidence interval of 99.7% [2].

The same hypothesis detection method is applied at the flow level to distinguish shrew attack flow from the normal TCP flows. Again, we denote H_{o} to represent the hypothesis that the flow is not a shrew attack flow, while H_1 corresponds to a shrew attack flow. To make flow filtering decision, we defined a flow level *likelihood ratio* $L(z)$ at the pivot point z as

$$L(z)=\frac{p_{\text{fa}}(z)}{p_{\text{fo}}(z)}=\frac{\sigma_{\text{fo}}}{\sigma_{\text{fa}}}\exp\left(\frac{1}{2}\left[\left(\frac{z-\mu_{\text{fo}}}{\sigma_{\text{fo}}}\right)^2-\left(\frac{z-\mu_{\text{fa}}}{\sigma_{\text{fa}}}\right)^2\right]\right), \tag{12}$$

where $p_{\text{fa}}(z)$ is the probability density of z in the Gaussian distribution N_{fa}, and $p_{\text{fo}}(z)$ is the probability density of z in Gaussian distribution N_{fo}.

Since there is no overlap between the N_{fa} and N_{fo}, the filtering threshold γ is chosen at $z = 0.28$ where $\gamma = L(z) = 1.9$. Table 3 shows the probability of cutting off a normal TCP flow as a shrew attack flow lower than 0.1% with this γ value. This indicates that the CAS-based flow filtering achieved an accuracy of 99.9%.

With such high filtering accuracy, our CDF scheme can tolerant relatively high false positive rate. When Algorithm 2 raises a false alarm, the router depends on the flow level hypothesis testing to determine whether a flow should be cut off. As indicated by Table 3, the probability that a legitimate TCP flow is filtered off is lower than 0.1%. A high false positive rate R_{fp} actually causes routers to call for unnecessary filtering.

Table 3
Error levels of filtering at flow level

Error level	Prob. of error (%)	TCP error level	Shrew error level
$\pm\sigma$	68	0.1311 ± 0.026	0.4985 ± 0.038
$\pm 1.65\sigma$	90	0.1311 ± 0.043	0.4985 ± 0.046
$\pm 1.96\sigma$	95	0.1311 ± 0.051	0.4985 ± 0.074
$\pm 3\sigma$	99.7	0.1311 ± 0.078	0.4985 ± 0.114
$\pm 3.29\sigma$	99.9	0.1311 ± 0.086	0.4985 ± 0.125

5.2. Flow filtering atop amplitude hypothesis testing

Based on the above hypothesis test framework, we proposed Algorithm 3 to cut off flows with amplitude spectrum value at the pivot frequency higher than the detection threshold γ. A flow chart is given in Fig. 13 to help our readers follow the shrew filtering process.

Although the source IP addresses are generally spoofed in attack packets, it is safe to use the 5-tuple {*Source IP*, *Source Port*, *Destination IP*, *Destination Port*, *protocol* as flow labels. Our filtering algorithm handles the incoming packets according to records in the *Malicious Flow Table* (MFT), *Suspicious Flow Table* (SFT) and *Legitimate Flow* Table (LFT).

Algorithm 3. Amplitude filtering of shrew attacks at flow level.

Input: $\psi_i(p)$: The amplitude spectrum of an incoming traffic flow F_i identified by the 5-tuple identifier.
γ: The amplitude filtering threshold obtained from the training process

Output: Enable packet dropping in flow F_i

Procedure:

01: **while** flow filtering is called by algorithm 2 on detecting shrew attacks
02: Check flow F_i for membership in various tables
03: **Case** 1: F_i is in LFT
04: Route the flow normally
05: **Case** 2: F_i is in MFT
06: Drop all packets in flow F_i
07: **Case** 3: F_i is in SFT
08: **if** $\Psi_i(p) = \leqslant\gamma$ **Then**
09: Mark F_i as legitimate, move it into LFT
10: **else**
11: F_i contains shrew attacks, move it to MFT
12: Drop all the packets in flow F_i
12: **end If**
13: **Case** 4: F_i is not in any of the three tables
14: Add F_i flow into SFT
15: **endwhile**

If the router has initiated the amplitude filtering algorithm while confirmed the alert, it start checking incoming packets. If a packet label is in the set LFT, this packet is routed normally. If it is in the set MFT, this packet is dropped. If it is in the set SFT, we continue sampling until time out. If there is no matching in any table, this packet belongs to a new flow and it would be added into the SFT, then sampling begins and timer starts.

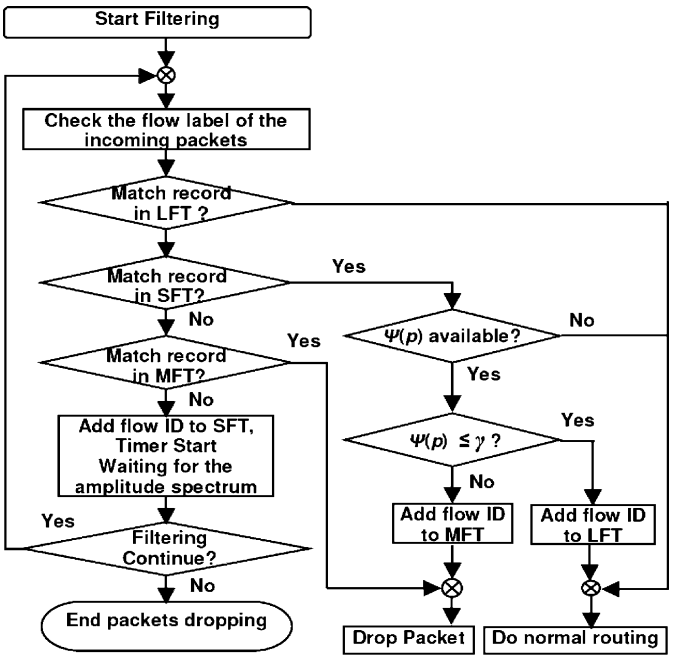

Fig. 13. Control flow in the amplitude-filtering Algorithm 3 (LFT: *Legitimate Flow Table*, SFT: *Suspicious Flow Table*, MFT: *Malicious Flow Table*, $\Psi(p)$: *Cumulative Amplitude Spectrum*).

Once a timer is expired for a flow, we compare its amplitude spectrum likelihood ratio $L(z)$ at pivot point with the threshold γ. If $L(z)$ value is lower than the threshold, we move its record into LFT. All packets in this flow will be routed normally. If $L(z)$ value is higher than the threshold γ, we move it into the MFT and cut this flow. The pseudo-code of amplitude spectral filtering process is specified in Algorithm 3. For flow filtering, we adopt data structures SFT, LFT and MFT tables to track per flow status.

We identify a flow using the 5-tuple {*Source IP*, *Source Port*, *Destination IP*, *Destination Port*, *protocol*}. Routers generally cannot afford to store 13 bytes for each flow for security purpose. To minimize the storage overhead incurred by the 3 large tables needed to implement the algorithm, we store only the output of a hash function with the label as the input.

6. Simulation experiments and performance results

In this section, we evaluate the performance results obtained with NS-2 simulation experiments performed. The simulation assumed a default network parameters in link capacity of 2 MB/s. The RTT of TCP flows are uniformly distributed over 60–240 ms. The shrew attack dataset specified in Section 3.1 are applied here.

While TCP-Tahoe, TCP-Reno and others present similar vulnerability under shrew attacks [15], we adopt the TCP-Reno standards in our experiments. We compare the results of our shrew-filtering algorithm with the well-known *Drop Tail scheme*. We examine the overhead in our CDF algorithms to assess the penalties paid and limitations of our scheme.

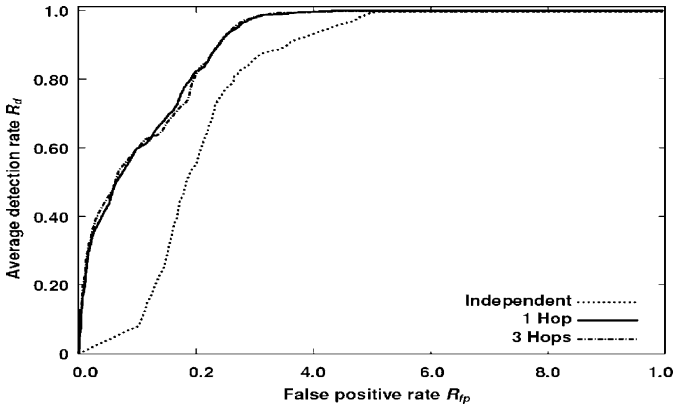

Fig. 14. ROC curve showing the tradeoffs between anomaly detection rate and false positive rate under collaborative detection of shrew attacks, compared with the result of using an isolated router.

6.1. Anomaly detection and false alarm rates

By anomaly, we mean an abnormal network condition caused by shrew attacks. We calculate the actual *anomaly detection rate* R_d as the ratio of detected attack streams over the total number of such traffic streams processed. The *false positive rate* R_{fp} is the ratio of normal traffic flows being wrongly detected as having shrew attacks over total number of legitimate traffic streams.

Besides these two performance metrics, we study the performance of our collaborative detection scheme (Algorithm 2) over several collaborative ranges. A single router works independently is considered no collaboration with its neighbors. Collaborating with 1-hop routers involves 2 to 4 routers at immediate neighbors. The 2-hop neighbors involve 4 to 16 routers within distance 2 from a given router. Finally, we limit to 8 to 40 routers in a 3-hop neighborhood.

The ROC curves shown in Fig. 14 report the performance of Algorithm 2, which is upper bounded by the template ROC performance reported in Fig. 9(a). Three curves are plotted for using 1 router independently and up to 4 and 40 routers, respectively in 1-hop and 3-hop neighborhoods. The lower curve shows the poor detection performance in using a single isolated router. The detection results in using neighboring routers are shown in the top two curves. The neighborhood range has resulted in very little differences. When the false alarm is required to be very low, say below 0.05%, the 3-hop group performs slightly better than the 1-hop group.

This ROC plots clearly shows that almost 98% sucessful detections can be expected, if one can tolerate 30% false alarms. When the false positive rate exceeds 20% the R_d difference in using large collaborative routers diminishes. Of course, more routers used may enhance the detection accuracy.

However, enlarging the collaborative range may trigger also lot more alert messages to propagate among all the routers involved. The message being conveyed here is that using 2–4 router within a 1-hop distance will serve the purpose, sufficiently, based on checking 4000 traffic streams in our experiments.

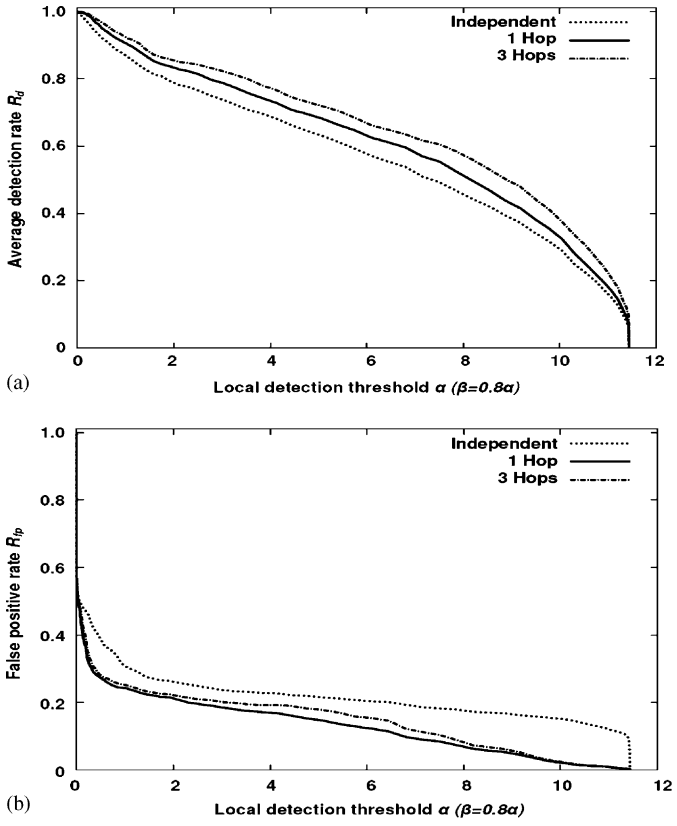

Fig. 15. Shrew attack detection results from testing 4000 traffic streams over 2 to 40 routers under the independent and two collaborative modes: (a) effect of α variation on average detection rate; (b) effect of α variation on false positive rate.

6.2. Effects of local detection threshold α

Fig. 15 presents the variation of the average detection rate R_{d} and false positive rate R_{fp} with different values of the local threshold α. The α magnitude is related to the selection of the CTS pivot point $y = \Phi(p)$. We compare the performance of independent detection by a single router with collaborative detection using multiple routers. With collaborative ranges in 1, 2 and 3 hops, we find the more routers can results in higher detection rate. For example at $\alpha = 4$, we can achieve the average $R_{\mathrm{d}} = 0.75$ and 0.82, respectively, in using 4 and 40 routers.

As the value of α increase, the average detection rate decreases steadily towards zero. The message is that once should adopt small α, if high detection rate is the major concern. On the other hand, we need to apply larger α to reduce the false positive rate R_{fp} to a reasonably low level as illustrated in Fig. 9(b). For example, to yield $R_{\mathrm{fp}} = 10\%$, corresponding α should be set at 8. The use of independent routers results in an $R_{\mathrm{fp}} = 20\%$, doubling that of using 16–40 routers in 3-hop distance in the AS.

Obviously, there exists a tradeoff between R_{d} and R_{fp} seen by both Figs. 14 and 15. In the next section, we will reveal the effect of the global threshold β, which provides more options to maximize the detection rate and minimize the false alarms. Achieving a 90% detection rate with 25% false alarms by no means imply that these false alarms will block 25% legitimate flows. We use Algorithm 3 to cope with this filtering problem.

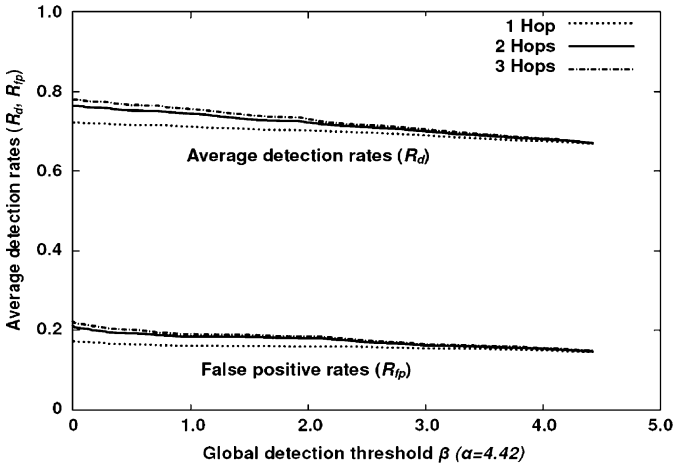

Fig. 16. Effects of global threshold (β) on collaborative detection of shrew DDoS attacks using three cooperative ranges of routers.

6.3. Effects of global detection threshold β

Based on our template database, we realized that we have to choose the global threshold β as such to satisfy the inequalities $0 < \beta < \alpha < 12$. The propose choice of β can increase the average detection rate by another 10% over the use of independent detection. The reason to use two thresholds lies in the effective reduction of false positive rate with increase of the R_{d}. The rule of the thumb is that one should choose β slightly lower than α. This results in good preservation of high detection rate with low false alarms.

Fig. 16 reports the effects of variation of β on the detection rate and false alarm rate, while fixed $\alpha = 4.42$. In case of 3-hop collaboration, a detection rate of 79% is achieved with a false positive rate of 21% by changing the value β. In contrast, a false positive rate of 27% is incurred if the same detection rate of 79% is desired by changing α to 1.92 as shown in Fig. 15.

These results also show that, on the average, both performance metrics are less sensitive to the variations in β or in the collaborative range size. The R_{d} stays between 0.65 and 0.79 and the R_{fp} is restricted between 0.2 and 0.15. As β decreases, the advantage of collaborating with more than 3 hops shows only a small gain. Actually, the global threshold β plays a vital role in fine-tuning, after local threshold α is constrained by keeping the false positive rate low.

6.4. Normalized throughput after packet filtering

Our shrew-filtering Algorithm 3 is triggered by the detection results of Algorithm 2. We compare below the TCP throughput (K packets/s) achieved by Algorithm 3 with that by using the Drop Tail algorithm.

This normalized throughput is a scalar ratio of the average throughput of TCP flows containing shrew DDoS attacks to that of the maximum TCP throughput without any attacks. The met-

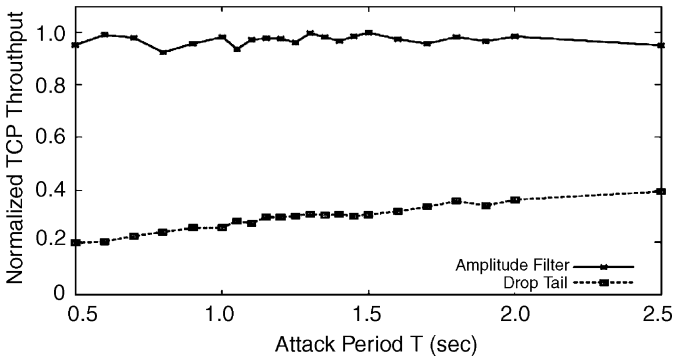

Fig. 17. Normalized TCP throughput of 5 TCP flows after filtering of suspected malicious flows by drop-tail and by our amplitude-filtering algorithm.

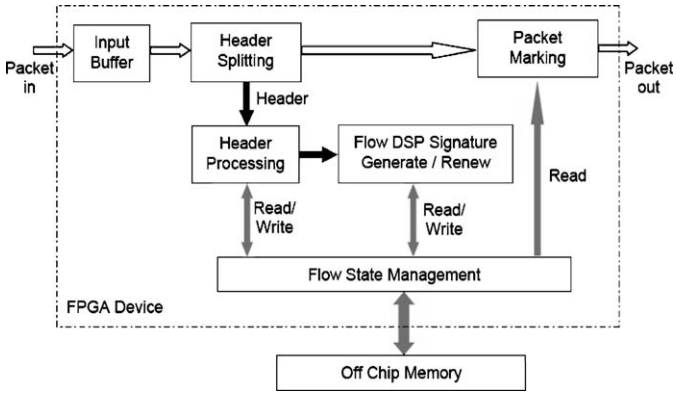

Fig. 18. Block diagram of a shrew attack filter designed for FPGA implementation.

ric θ indicates the damage degree the regular TCP traffic suffer from shrew attacks. The lower is the normalized throughput, the greater is the suffer of legitimate TCP flows.

Fig. 17 compares the throughputs of legitimate TCP flows under attack from multiple distributed shrew streams using the Drop Tail scheme and using our amplitude filtering Algorithm 3. The x-axis is the attack period and the y-axis is the normalized throughput TCP flow retained.

Using the Drop Tail algorithm, the throughput of legitimate TCP flow is reduced to 20% below the attainable throughput. With Algorithm 3, above 95% of the TCP throughput is preserved. Our hypothesis filtering model is effective to cut off malicious shrew attack flows with less than 1% loss of the legitimate TCP flows on the average.

Fig. 17 presents the case where shrew attack streams are distributed in space but synchronized. Four shrew attack streams are from four sources with the same attacking periods and the same burst lengths. However, their peak rate is only $0.25R$, where R is the attack burst rate. This means their average traffic rate is only 25% of that from a single-source attack.

This shows a major advantage of using spectral analysis over bandwidth volume analysis. Even if the shrew attack flows were launched from many zombies to lower the bandwidth utilization, the frequency spectrum assumes the same properties. Although the average detection rate was shown around 82%, this result shows that the TCP flows can retain almost 99.9% of the TCP throughput.

7. Scalability, deployment and limitations

The intensive NS-2 simulation experiments verified that our shrew attack defense mechanism is very promising to cur off low-rate TCP-targeted DDoS attacks. The CDF works by integrating DSP, hypothesis testing, and statistical analysis. It will be more useful to further verify the effectiveness by testing with larger scale enchmark experiments in real-life network environments. We discuss below the scalability, overhead, and accuracy issues and limitations of our defense scheme.

7.1. Scalability and hardware implementation

To deploy a distributed security scheme in core network, the scalability issue is related to the network size, data rate, link capacity, or router number involved. The collaborative shrew attack detection process (Algorithm 2) must count all incoming packets at the network interface. The shrew-filtering process (Algorithm 3) demands storage space and computing power to perform the filtering at flow level.

We have experimented on 4000 traffic streams. Each stream has 8–20 flows. Thus, more than 50,000 flows were process in the simulation experiments. One major obstacle in deploying DDoS defense schemes into the core network routers is the conflict between limited resources in routers and high data rate demanded [21]. It was suggested to implement *intrusion detection system* (IDS) on a FPGA platform, which can process 32,768 complex rules at a data rate as high as 10 Gbps [3].

Our sampling and packets filtering rules are not more complex than those IDS rules. The most time-consuming part is to perform DFT on every traffic stream. This can be solved by using the Xilinx Virtex-4 FPGA, which calculates 512 DSP slices to execute in parallel with a 500 MHz clock. It has been estimated that these hardware implementations may have a speed gain of two orders of magnitude, compared with the software implementations.

Fig. 18 presents a high-level schematic block diagram of the FPGA-based shrew filtering system architecture. The header-processing unit monitors the sampling process and the temporal series are transformed into frequency domain using DSP technique. The packet-marking unit determines the operation against malicious packets as the DSP signature is generated.

Another constraint on scalability lies in the storage demand for sampled spectrum of each traffic flow processed. Per each flow, if we sample for 5 s with a period of 1 ms, then 5000 data points must be processed. If we consider using 8 bits for each spectral line, then 5 kB is needed. This can be easily supported by today's FPGA devices, for example, the Xilinx Virtex-4's SmartRam hierarchy supports up to 10 MB of embedded block RAM.

Table 4
Error levels in filtering with different sampling times

Error level	Traffic types	Sample time periods (λ)				
		1 s	2 s	3 s	4 s	5 s
$\pm 1.96\sigma/95\%$	TCP flow	0.16 ± 0.18	0.14 ± 0.09	0.13 ± 0.07	0.13 ± 0.08	0.11 ± 0.05
	Shrew attack	0.50 ± 0.05	0.47 ± 0.06	0.45 ± 0.07	0.45 ± 0.07	0.50 ± 0.07
$\pm 3\sigma/99.7\%$	TCP flow	0.16 ± 0.27	0.14 ± 0.14	0.13 ± 0.10	0.13 ± 0.12	0.11 ± 0.08
	Shrew attack	0.50 ± 0.08	0.47 ± 0.09	0.45 ± 0.10	0.45 ± 0.11	0.50 ± 0.11
$\pm 3.29\sigma/99.9\%$	TCP flow	0.16 ± 0.30	0.14 ± 0.16	0.13 ± 0.11	0.13 ± 0.13	0.11 ± 0.09
	Shrew attack	0.50 ± 0.08	0.47 ± 0.10	0.45 ± 0.11	0.45 ± 0.12	0.50 ± 0.13

7.2. Overheads and accuracy analysis

The overhead is another critical parameter to evaluate the performance of our amplitude-filtering algorithm. It is defined by the time interval from the moment a shrew attack is detected to the moment the countermeasure responses effectively. This time interval is often varied according to the traffic load on the link. However, the load on the link does not affect the response time of our amplitude-filtering algorithm.

In Table 4, we observe different error levels of the normal TCP and of the shrew attack streams. The confidence level of detecting and filtering shrew flows is very high (99.9%) while $\lambda \geqslant 2$ s. With $\lambda = 1$ second, we observed an overlap in $\pm 3\sigma$ and $\pm 3.29\sigma$ error ranges, but no overlap at $\pm 1.96\sigma$ error level. This implies that the filtering accuracy is not sensitive to the sampling period.

7.3. Impacts of IP spoofing and table overflows

As long as the attack stream presents similar periodicity, we can detect the shrew DDoS attacks accurately. However, the performance of shrew filtering algorithm may suffer from the dynamics of source IP spoofing.

In a flow-level sampling and filtering procedure, we uniquely identify each flow using the 5-tuple {*Source IP*, *Source Port*, *Destination IP*, *Destination Port*, *protocol*}. This may add some flow labeling overhead, if the attacker spoof the source IP dynamically. If the spoofing source IP set is very large, attacker may cause the MFT to overflow.

The MFT overflow may decrease the performance of shrew filtering process. It could be worse if the attacker use different source IP in sending every pulse. One possible solution to solve this dynamic spoofing problem is to perform also rate limiting at network interfaces, when we detect the existence of shrew attack streams.

Another issue is that short-lived burst traffic flows may be mistakenly marked as low-rate shrew attacks. Although they are marked as malicious flow, they have already passed the filter and thus do not affect the legitimate TCP flow. Meanwhile, short-lived burst traffic may waste space in the MFT.

In addition, long-lived pulsing nature traffic can bring down the throughput of TCP flows sharing links with them. Therefore, when TCP flows are throttled heavily, it is reasonable to block these pulsing streams, even they are not generated purposely to impair the normal TCP flows.

8. Conclusions and further research

Our research contributions are summarized below in five technical aspects. Some of the identified topics deserve continued effort to mature the defense technology:

A. *Stream-level detection and flow-level filtering*: Leveraging spectral analysis, our hypothesis testing model makes the spectral template matching effective in detecting shrew DDoS attacks at traffic streaming level and in cutting off malicious flows at a refined flow level.
B. *Spectral modeling of Internet traffic patterns*: Our CDF model offers a theoretical foundation on defense not only against shrew DDoS attacks but also extensible to cope with flooding type of DDoS attacks. This area demands further research and experiments to prove the idea.
C. *Detection accuracy versus false alarms*: Through extensive NS-2 simulation experiments, we report encouraging results on successful anomaly detection with low false positive alarms. This implies possible tradeoffs between anomaly detection and false alarms.
D. *Appealing to Hardware implementations*: Our CDF scheme appeals to both DSP software and FPGA hardware implementation. The scheme can be also implemented on network processors. One can push frequency-domain monitoring down to a lower level in packet processing hierarchy. The DSP chips, FGPA, and network processors will all reduce packet-processing time on routers.
E. *Protecting legitimate TCP/UDP flows*: The shrew-filtering process (Algorithm 3) drops all packets in malicious flows detected. Our simulation results show almost a complete cutoff of the malicious shrew attack flows with less than 0.1% loss of the legitimate TCP flows. This avoids the loss of legitimate TCP/UDP packets, as often found in using rate-limiting algorithms.

For continued effort, we are porting and testing the CDF defense schemes on the DETER testbed [10], jointed developed by University of California at Berkeley and USC Information Science Institute. More benchmark results will be reported later to further verify the effectiveness of the CDF scheme.

We are also exploring the use of FPGA and network processors to solve the security problems in hardware. We aim to relieve the burdens of end users, core ISP gateways, and edge network routers by speeding up the defense process with DSP technologies. The ultimate goal is to achieve automated intrusion detection and responses in real-time.

Acknowledgments

We would like to acknowledge the support of this work by NSF ITR Grant 0325409. This work extends from our preliminary filtering results reported in [6]. Both theoretical modeling and experimental results are newly reported here. We thank Dr. Yu-Kwong Kwok of the University of Hong Kong for his earlier contributions to frequency-domain filtering techniques for cutting off shrew DDoS attacks.

References

[1] Abilene-I data set, the Passive Measurement and Analysis (PMA) project, ⟨http://pma.nlanr.net/traces/long/ipls1.html⟩.
[2] R. Allen, D. Mills, Signal analysis: Time, Frequency, Scale, and Structure, Wiley, New York, 2004.
[3] M. Attig, J. Lockwood, A framework for rule processing in reconfigurable network systems, in: Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA, 17–20 April, 2005.
[4] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, in: Proceedings of ACM Internet Measurement Workshop, Marseille, France, 6–8 November, 2002.
[5] Y. Chen, K. Hwang, Collaborative change detection of DDoS attacks on community and ISP networks, in: IEEE International Symposium on Collaboration Technologies and Systems (CTS'06), Las Vegas, NV, 15–17 May, 2006.
[6] Y. Chen, K. Hwang, Y.K. Kwok, Filtering of shrew DDoS attacks in frequency domain, in: The First IEEE LCN Workshop on Network Security (WoNS), Sydney, Australia, 15–17 November, 2005.
[7] C.M. Cheng, H.T. Kung, K.S. Tan, Use of spectral analysis in defense against DoS attacks, Proceedings of 2002 IEEE GLOBECOM, Taipei, China.
[8] R. Chertov, S. Fahmy, and N. Shroff, Emulation versus simulation: a case study of TCP-Targeted denial of service attack, in: Proceedings of Second International IEEE CreateNet Conference on Testbeds and Research Infrastructures, March 2006.
[9] M. Delio, New Breed of Attack Zombies Lurk, ⟨http://www.wired.com/news/technology/0,1282,43697,00.html⟩, 31 October, 2005.
[10] DETER and EMIST Network Project, Cyber defense technology networking and evaluation, Comm. ACM, 47(3) (2004).
[11] GT-ITM: Georgia Tech Internet Topology Models, ⟨http://www.cc.gatech.edu/projects/gtitm/⟩, 3 November, 2005.
[12] M. Guirguis, A. Bestavros, I. Matta, Y. Zhang, Reduction of quality (RoQ) attacks on internet and systems, in: IEEE INFOCOM, Miami, FL, March 2005.
[13] A. Hussain, J. Heidemann, C. Papadopoulos, A framework for classifying denial of service attacks, ACM SIGCOMM, Karlsruhe, Germany, August 2003.
[14] K. Hwang, Y.-K. Kwok, S. Song, M. Cai, Y. Chen, Y. Chen, Security binding and worm/DDoS defense for trusted grid computing, Internat. J. Critical Infrastructures 2 (4) (2005).
[15] A. Kuzmanovic, E. Knightly, Low-rate TCP-targeted denial of service attacks—the shrew vs. the mice and elephants, in: Proceedings of 2003 ACM SIGCOMM, Karlsruhe, Germany, 25–29 August, 2003.
[16] X. Luo, R. Chang, On a new class of pulsing denial-of-service attacks and the defense, in: Proceedings of Network and Distributed System Security Symposium (NDSS'05), San Diego, CA., 2–5 February, 2005.
[17] X. Luo, R. Chang, E. Chan, Performance analysis of TCP/AQM under denial-of-service attacks, in: Proceedings of IEEE MASCOTS, Atlanta, GA, September 2005.
[18] R. Mahajan, S. Floyd, D. Wetherall, Controlling high-bandwidth flows at the congested router, in: Proceedings of ACM Ninth International Conference on Network Protocols (ICNP), Riverside, CA, 11–14 November, 2001.
[19] S. McCanne, S. Floyd, NS-2 Network Simulator, ⟨http://www.isi.edu/nsnam/ns/⟩, 1997.
[20] D. Moore, G. Voelker, S. Savage, Inferring Internet denial-of-service activity, in: Proceedings of 10th USENIX Security Symposium, Washington, DC, August 2001.
[21] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, R. Govindan, COSSACK: coordinated suppression of simultaneous attacks, in: Proceedings of DISCEX III, 2003.
[22] C. Partridge, D. Cousins, A. Jackson, R. Krishnan, T. Saxena, W. Strayer, Using signal processing to analyze wireless data traffic, Proceedings of ACM Workshop on Wireless Security, Atlanta, GA, 28–28 September, 2002.
[23] V. Paxson, M. Allman, Computing TCP's retransmission timer, Internet RFC 2988, November 2000.
[24] S. Specht, R. Lee, Distributed denial of service: taxonomies of attacks, tools and countermeasures, in: Proceedings of 2004 PDCS, San Francisco, CA, 15–17 September, 2004.
[25] H. Sun, J. Lui, D. Yau, Defending against low-rate TCP attacks: dynamic detection and protection, in: Proceedings of 2004 IEEE International Conference on Network Protocols (ICNP), Berlin, Germany, 5–8 October, 2004.

Yu Chen received the B.S. degree from Chongqing University, China in 1994. He earned the M.S. degree in Computer Engineering from the University of Southern California (USC) in 2002. He is currently a Ph.D. candidate in Computer Engineering and works at USC Internet and Grid Research Laboratory. His research interest includes Internet infrastructure security, DDoS attack detection and defense, Internet traffic analysis and distributed security infrastructure. He can be reached at cheny@usc.edu.

Kai Hwang is a Professor of Computer Engineering and Director of Internet and Grid Research Laboratory at the University of Southern California (USC). He received the Ph.D. degree from the University of California, Berkeley in 1972. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet security, and Grid and distributed systems.
Dr. Hwang has published over 190 original scientific papers and four popular textbooks. His latest books, *Scalable Parallel Computing* and *Advanced Computer Architecture*, are being adopted worldwide and translated into four languages. He has also edited a number of advanced research books, including the series of *Annual Reviews in Scalable Computing* (World Scientific Publisher. Presently, he leads a NSF-supported GridSec project in developing security binding and distributed defense systems against network worms and DDoS attacks for trusted Grid, P2P, and Internet computing. Contact him via kaihwang@usc.edu or http://GridSec.usc.edu/Hwang.html.