Monitoring the Application-Layer DDoS Attacks for Popular Websites

Yi Xie and Shun-Zheng Yu, *Member, IEEE*

***Abstract*—Distributed denial of service (DDoS) attack is a continuous critical threat to the Internet. Derived from the low layers, new application-layer-based DDoS attacks utilizing legitimate HTTP requests to overwhelm victim resources are more undetectable. The case may be more serious when such attacks mimic or occur during the flash crowd event of a popular Website. Focusing on the detection for such new DDoS attacks, a scheme based on document popularity is introduced. An Access Matrix is defined to capture the spatial-temporal patterns of a normal flash crowd. Principal component analysis and independent component analysis are applied to abstract the multidimensional Access Matrix. A novel anomaly detector based on hidden semi-Markov model is proposed to describe the dynamics of Access Matrix and to detect the attacks. The entropy of document popularity fitting to the model is used to detect the potential application-layer DDoS attacks. Numerical results based on real Web traffic data are presented to demonstrate the effectiveness of the proposed method.**

***Index Terms*—Application-layer, distributed denial of service (DDoS), popular Website.**

I. Introduction

DISTRIBUTED denial of service (DDoS) attack has caused severe damage to servers and will cause even greater intimidation to the development of new Internet services. Traditionally, DDoS attacks are carried out at the network layer, such as ICMP flooding, SYN flooding, and UDP flooding, which are called Net-DDoS attacks in this paper. The intent of these attacks is to consume the network bandwidth and deny service to legitimate users of the victim systems. Since many studies have noticed this type of attack and have proposed different schemes (e.g., network measure or anomaly detection) to protect the network and equipment from bandwidth attacks, it is not as easy as in the past for attackers to launch the DDoS attacks based on network layer. When the simple Net-DDoS attacks fail, attackers shift their offensive strategies to application-layer attacks and establish a more sophisticated type of DDoS attacks. To circumvent detection, they attack the victim Web servers by HTTP GET requests (e.g., HTTP Flooding) and pulling large image files from the victim server in overwhelming numbers. In another instance, attackers run a massive number of queries through the victim's search engine or database query to bring the server down [1]. We call such attacks application-layer DDoS (App-DDoS) attacks. The MyDoom worm [2] and the CyberSlam [3] are all instances of this type attack.

On the other hand, a new special phenomenon of network traffic called flash crowd [4], [5] has been noticed by researchers during the past several years. On the Web, "flash crowd" refers to the situation when a very large number of users simultaneously accesses a popular Website, which produces a surge in traffic to the Website and might cause the site to be virtually unreachable.

Because burst traffic and high volume are the common characteristics of App-DDoS attacks and flash crowds, it is not easy for current techniques to distinguish them merely by statistical characteristics of traffic. Therefore, App-DDoS attacks may be stealthier and more dangerous for the popular Websites than the general Net-DDoS attacks when they mimic (or hide in) the normal flash crowd. In this paper, we meet this challenge by a novel monitoring scheme.

To the best of our knowledge, few existing papers focus on the detection of App-DDoS attacks during the flash crowd event. This paper introduces a scheme to capture the spatial-temporal patterns of a normal flash crowd event and to implement the App-DDoS attacks detection. Since the traffic characteristics of low layers are not enough to distinguish the App-DDoS attacks from the normal flash crowd event, the objective of this paper is to find an effective method to identify whether the surge in traffic is caused by App-DDoS attackers or by normal Web surfers. Our contributions in this paper are fourfold: 1) we define the Access Matrix (AM) to capture spatial-temporal patterns of normal flash crowd and to monitor App-DDoS attacks during flash crowd event; 2) based on our previous work [6], [7], we use hidden semi-Markov model (HsMM) [8] to describe the dynamics of AM and to achieve a numerical and automatic detection; 3) we apply principal component analysis (PCA) [9] and independent component analysis (ICA) [10], [11] to deal with the multidimensional data for HsMM; and 4) we design the monitoring architecture and validate it by a real flash crowd traffic and three emulated App-DDoS attacks.

The remainder of this paper is organized as follows. In Section II, we put our ideas within the context of prior and ongoing research related to DDoS detection. In Section III, we discuss the App-DDoS attacks and detail their properties. In Section IV, we explain our technique to detect the App-DDoS attacks. In Section V, we conduct experiments using real traffic

Manuscript received August 25, 2006; revised March 31, 2007 and November 08, 2007; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Pacifici. First published June 20, 2008; current version published February 19, 2009. This work was supported in part by the The National High Technology Research and Development Program of China (2007AA01Z449 and by The Research Fund for the Doctoral Program of Higher Education under Grant 20040558043.

The authors are with the Department of Electrical and Communication Engineering, School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510275, China (e-mail: xieyicn@163.com; syu@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TNET.2008.925628

data and three emulated App-DDoS attacks to validate our detection model. In Section VI, we discuss the strengths and limitations of our proposed technique. We conclude our paper in Section VII.

II. Related Work

Our literature survey has noted that researchers attempt to detect DDoS attacks from three different layers: IP layer, TCP layer, and application layer. From all of these perspectives, researchers are investigating various approaches to distinguish normal traffic from the attack one. Here, we survey representative research from each perspective.

Most DDoS-related research has focused on the IP layer. These mechanisms attempt to detect attacks by analyzing specific features, e.g., arrival rate or header information. For example, Cabrera *et al.* [12] used the management information base (MIB) data which include parameters that indicate different packet and routing statistics from routers to achieve the early detection. Yuan *et al.* [13] used the cross-correlation analysis to capture the traffic patterns and then to decide where and when a DDoS attack possibly arises. Mirkovic *et al.* [14] monitored the asymmetry of two-way packet rates and to identify attacks in edge routers. Other statistical approaches for detection of DDoS attacks includes IP addresses [15] and time-to-live (TTL) values [16].

The TCP layer is another main battlefield for detecting DDoS attack. For example, authors [12] mapped ICMP, UDP, and TCP packet statistical abnormalities to specific DDoS attacks based on MIB. Wang *et al.* [17] used the TCP SYN/FIN packets for detecting SYN flooding attacks. In [18], DDoS attacks were discovered by analyzing the TCP packet header against the well-defined rules and conditions and distinguished the difference between normal and abnormal traffic. Noh *et al.* [19] attempted to detect attacks by computing the ratio of TCP flags (including FIN, SYN, RST, PSH, ACK, and URG) to TCP packets received at a Web server.

However, little work has been done on the detection of App-DDoS attacks because there were few such attacks in the past. Ranjan *et al.* [20] used statistical methods to detect characteristics of HTTP sessions and employed rate-limiting as the primary defense mechanism. Yen *et al.* [21] defended the application DDoS attacks with constraint random request attacks by the statistical methods. Other researchers combated the App-DDoS attacks by "puzzle," see, e.g., [3]. Jung *et al.*'s work [5] is most closely related to our own, as they used two properties to distinguish the DoS and normal flash crowd: 1) a DoS event is due to an increase in the request rates for a small group of clients while flash crowds are due to increase in the number of clients and 2) DoS clients originate from new client clusters as compared to flash crowd clients which originate from clusters that had been seen before the flash event.

III. App-DDoS Attacks

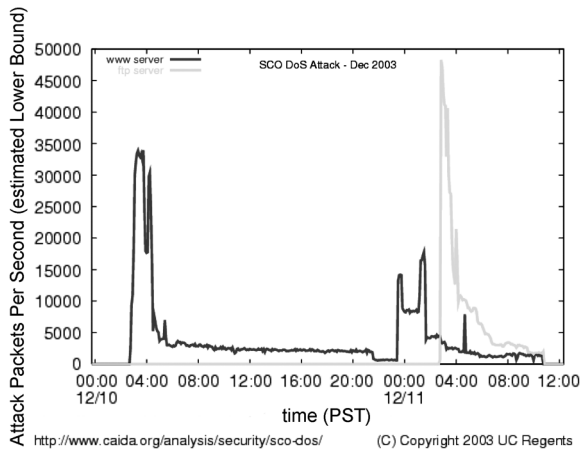
In our opinion, the DDoS attack detection approaches in different scenario can be clustered as: 1) Net-DDoS attacks versus stable background traffic; 2) Net-DDoS attacks versus flash crowd (i.e., burst background traffic); 3) App-DDoS attacks versus stable background traffic; and 4) App-DDoS attacks versus flash crowd. The first two scenarios have been well studied and can be dealt with by most existing DDoS detection schemes (e.g., those presented in Section II) while the other two groups are quite different from the previous ones.

Besides the flooding attack pattern, App-DDoS attacks may focus on exhausting the server resources such as Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth. Research [22] has found, with increasing computational complexity in Internet applications and larger network bandwidth, that server resources may become the bottleneck of those applications. Thus, the App-DDoS attacks may cause more serious problems in the high-speed Internet than in the past.

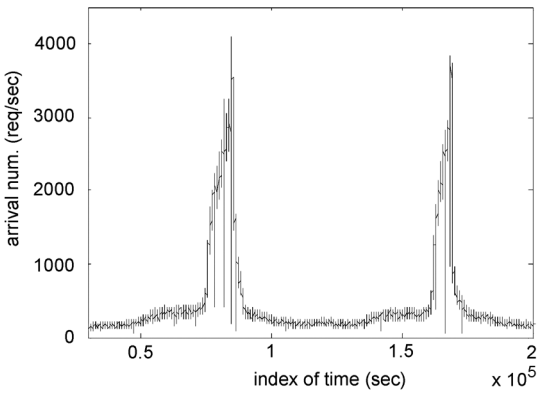
The first characteristic of App-DDoS attacks is that the application-layer requests originating from the compromised hosts are indistinguishable from those generated by legitimate users. Unlike the Net-DDoS attacks, App-DDoS attacks do not necessarily rely on inadequacies in the underlying protocols or operating systems; they can be mounted with legitimate requests from legitimately connected network machines. Usually, App-DDoS attacks utilize the weakness enabled by the standard practice of opening services such as HTTP and HTTPS (TCP port 80 and 443) through most firewalls to launch the attack. Many protocols and applications, both legitimate and illegitimate, can use these openings to tunnel through firewalls by connecting over a standard TCP port 80 (e.g., Code Red virus) or encapsulating in SSL tunnels (HTTPS). Attack requests aimed at these services may pass through the firewall without being identified. Furthermore, attackers may request services to the point where other clients are unable to complete their transactions or are inconvenienced to the point where they give up trying.

This shows, to deal with the third scenario of DDoS attacks, that four issues have to be considered: 1) the Net-DDoS attacks detection methods are unable to collect enough offensive signals for detecting the App-DDoS attacks because they belong to different layers respectively; 2) TCP anomaly detection mechanisms can hardly identify the App-DDoS attacks launched by HTTP requests based on successful TCP connections; 3) in order to establish the TCP connection, attackers have to use the legitimate IP addresses and IP packets, which makes the anomaly detection mechanisms for IP packet become invalid; and 4) the implied premise of most current detection schemes is that the characteristics of DDoS attack traffic differ from normal traffic, which might fail because App-DDoS attacks may mimic the access behaviors of normal users. However, because the background traffic of this scenario is assumed to be stable, some simple App-DDoS attacks (e.g., Flood) still can be monitored by improving existing methods designed for Net-DDoS attacks, e.g., we can apply the HTTP request rate, HTTP session rate, and duration of user's access for detecting.

The second characteristic of App-DDoS attacks is that the attackers aiming at some special popular Websites are increasingly moving away from pure bandwidth flooding to more surreptitious attacks that masquerade as (or hide in) normal flash crowds of the Websites. Since such Websites become more and more for the increasing demands of information broadcast and electronic commerce, network security has to face a new challenge: how to detect and respond to the App-DDoS attacks if



(*a*)



(*b*)

Fig. 1. DDoS and flash crowds. (a) App-DDoS attacks on SCO. (b) Flash crowds from the 1998 World Cup.

they occur during a flash crowd event, i.e., the fourth scenario of our clusters for DDoS attacks.

Besides the issues discussed for the third scenario, the difficulties of dealing with such scenario include: 1) both the flash crowd and App-DDoS attacks are unstable, bursty and huge traffic volume, which is shown in Fig. 1, where (a) [23] shows the burst traffics caused by the typical App-DDoS attacks (Mydoom worm [2]) and (b) shows two flash crowds during the semifinals of the 1998 World Cup [24] and 2) attack nodes may arrange their vicious Web traffic to mimic the normal one by HTTP synthetic tools (e.g., [25]), so the malicious requests differ from the legitimate ones in intent but not in traffic characteristics. Therefore, most current detection mechanisms (e.g., those based on traffic characteristics) become invalid.

The work of Jung *et al.* [5] may not help in this scenario since: 1) it is difficult to associate the amount of resources consumed to a client machine and 2) attack nodes consisting of a large number of geographically widespread machines are increasingly likely to belong to known client clusters. Thus, they cannot be filtered on the IP prefix. Other existing defense methods may be those based on man–machine interaction, e.g., puzzles, passwords, and the CAPTCHAs. However, as Kandula *et al.* in [3] and Ranjan *et al.* in [22] have pointed out, those schemes are not effective for the DDoS attack detection because they may annoy users and introduce additional service delays.

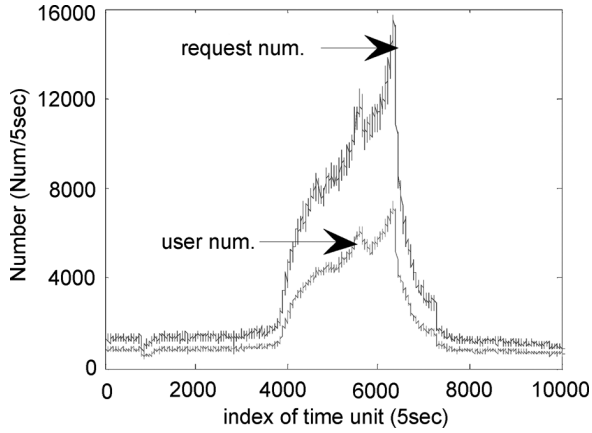


Fig. 2. Flash crowd.

Furthermore, they may deny search engines access to the Web site, and the machine hosting authentication mechanism may be easy to become the new attack targets.

Finally, compared with the consumption of resources such as CPU, memory, and database, App-DDoS attacks may not need to consume a lot of network bandwidth. Therefore, the traditional DDoS detection schemes designed for bandwidth exhausting attacks become ineffective.

IV. Detection Principle

Web user behavior is mainly influenced by the structure of Website (e.g., the Web documents and hyperlink) and the way users access web pages. In this paper, our monitoring scheme considers the App-DDoS attack as anomaly browsing behavior.

We investigate the characteristic of Web access behavior in Figs. 2 and 6. Fig. 2 plots the HTTP request number and the user number per 5 s during the burst Web workload of a semifinal collected from the logs of the 1998 World Cup. From the maximum correlation coefficient 0.9986, between the series of request numbers and that of the user numbers, we can see that the normal flash crowd is mainly caused by the sudden increment of user amount. Fig. 6 plotted in the following experiment section shows that the entropy of the aggregate access behavior against our model does not change much during the flash crowd event, which implies that both the main access behavior profile of normal users and the structure of Website do not have obvious varieties during the flash crowd event and its vicinity area. This conclusion is the same as [5] and is similar to those of other HTTP traces, e.g., Calgary-HTTP, ClarkNet-HTTP, and NASA-HTTP, which can be downloaded freely from [24].

These results are significant to our work. They show that the users' access behavior profile can be used to detect the abnormal varieties of users' browsing process during the flash crowd. Since the document popularity has been widely used to characterize the user behavior and improve the performance of Web server and Internet cache, e.g., [29] and [30], we will extend it to our detection in the rest of this paper.

A. Access Matrix

Traditionally, document popularity is defined by the *Request Hit Rate* as $p_{it} = b_{it} / \sum_{i=1}^{N} b_{it}$, where b_{it} is the request number

of the ith document at the tth time unit, and N the number of the Web server's documents. We extend this definition as given in (1), shown at the bottom of the page, where c_{it} is the number of users who request the ith document at the tth time unit, b_{it}/c_{it} is the user's average revisitation to the ith document at the tth time unit, r_{it} is the normalized revisitation, and T is the number of observation time units. Then, we construct a $N \times T$ dimensional Access Matrix (AM), $\mathbf{A}_{N\times T}$, as follows:

$$\mathbf{A}_{N\times T} = [\vec{a_1} \quad \vec{a_2} \quad \cdots \quad \vec{a_T}] = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_N]^{\mathrm{T}} \quad (2)$$

where $\vec{a_t} = (a_{1t}, \ldots, a_{Nt})^{\mathrm{T}}$, $\mathbf{a}_i = (a_{i1}, \ldots, a_{iT})^{\mathrm{T}}$, and $a_{it} = p_{it}$ or r_{it}. We will use $a_{it} = r_{it}$ for the experiment in this paper because it is more suitable to detect the attacks that repeatedly request the same pages such as homepage, "hot" pages, or randomly selected pages from a given set. We note that, in some other cases when the attacks may cause the document popularity away from the Zipf-like distribution, we should let $a_{it} = p_{it}$. We consider a spatial-temporal space constructed by AM in which $\vec{a_t} = (a_{1t}, \ldots, a_{Nt})^{\mathrm{T}}$ presents the spatial distribution of popularity at the tth time unit and $\mathbf{a}_i = (a_{i1}, \ldots, a_{iT})^{\mathrm{T}}$ presents the ith document's popularity varying with time. $\vec{a_t}$ is mainly related to users' interest and Website's structure (e.g., the distribution of contents and hyperlinks between Web pages); $\mathbf{a}_i$ is mainly controlled by users' actions (e.g., click rate and browsing time).

Then, we have the following detection rationale: in considering that most existing methods used on document popularity for modeling user behavior merely focus on the average characteristics (e.g., mean and variance), we use a stochastic process to model the variety of the document popularity, in which a random vector is used to represent the spatial distribution of document popularity and is assumed to be changing with time (i.e., $\vec{a_t}$, for $t = 1, \ldots, T$). The process is controlled by an underlying semi-Markov chain, and a hidden state describes a cluster of document popularities or a type of user behavior. Transition from one hidden state to another implies that user behavior has changed from one type to another. Dwell time of the state can be considered as persistence of the user behavior. Since the model is trained by the normal behavior obtained from a lot of users of the target server, attackers are supposed to be unable to obtain historical access records from the victim Web server or unable to stay in front of the victim to intercept and collect all users' HTTP requests sent to the target and spoof the detector based on the model.

B. Hidden Semi-Markov Model

Based on our previous work [6], [7], we extend the HsMM algorithm to describe the stochastic process on document popularity's spatial distribution varying with time and monitor the App-DDoS attacks occurring during flash crowd event.

HsMM is a hidden Markov model (HMM) with variable state duration. The HsMM is a stochastic finite state machine, specified by $(Q, \{\pi_i\}, \{a_{ij}\}, \{b_i(k)\}, \{p_i(d)\})$, where $Q = \{1, \ldots, M\}$ is a discrete set of hidden states with cardinality M; $q_t \in Q$ denotes the state that the system takes at time t; $\pi_i \equiv \Pr[q_1 = i]$ is the probability distribution for the initial state satisfying $\sum_i \pi_i = 1$; $a_{ij} \equiv \Pr[q_t = j | q_{t-1} = i]$ is the state transition probability from state i to state j satisfying $\sum_j a_{ij} = 1$, for $i, j \in Q$; $\tau_t \in \{1, \ldots, D\}$ denotes the remaining (or residual) time of the current state q_t with D representing the maximum interval between any two consecutive state transitions; $p_i(d) \equiv \Pr[\tau_t = d | q_t = i]$ is the state residual time distribution satisfying $\sum_d p_i(d) = 1$, for $i \in Q$, $d \in \{1, \ldots, D\}$; $b_i(k) \equiv \Pr[\vec{o_t} = \vec{v_k} | q_t = i]$ is the output distribution for given state i, satisfying $\sum_k b_i(k) = 1$, for $i \in Q$, $k \in \{1, \ldots, K\}$; and $\vec{o_t}$ denotes the observed vector at time t, taking values from $\{\vec{v_1}, \ldots, \vec{v_K}\}$.

Then, if the pair process (q_t, τ_t) takes on value (i, d), the semi-Markov chain will remain in the current state i until time $t + d - 1$ and transits to another state at time $t + d$, for $d \in \{1, \ldots, D\}$. The states themselves are not observable. The observables are a series of observations $O = (\vec{o_1}, \ldots, \vec{o_T})$. We adopt the notation $\vec{o_a^b}$ to represents the observation sequence from time a to time b (i.e., $\{\vec{o_t} : a \le t \le b\}$) and assume the "conditional independence" of outputs so that $b_i(\vec{o_a^b}) = \prod_{t=a}^{b} b_i(\vec{o_t})$.

We consider the AM as a multiple-dimensional stochastic process which is controlled by an underlying semi-Markov process. For a given HsMM, the hidden state q_t of the HsMM can be used to represent the spatial distribution of popularity of the N documents at the tth time unit. A transition of the hidden state (i.e., from q_{t-1} to q_t) can be considered as the change of access behaviors from one spatial distribution to another one. Residential duration τ_t of the hidden state q_t can be considered as the time units that the current spatial distribution (or state q_t) will persist.

The parameter estimation of HsMM can be done by the following forward and backward algorithm [8]. The forward and backward variables are defined as follows:

$$\alpha_t(i, d) \equiv \Pr\left[\vec{o_1^t}, q_t = i, \tau_t = d | \lambda\right] \quad (3)$$

$$\beta_t(m, d) \equiv \Pr\left[\vec{o_{t+1}^T} | q_t = i, \tau_t = d, \lambda\right] \quad (4)$$

which can be iteratively calculated by the forward and backward algorithms. Three joint probability functions are defined by

$$\xi_t(i, j) \equiv \Pr\left[\vec{o_1^T}, q_{t-1} = i, q_t = j\right] \quad (5)$$

$$\eta_t(i, d) \equiv \Pr\left[\vec{o_1^T}, q_{t-1} \neq i, q_t = i, \tau_t = d\right] \quad (6)$$

$$\gamma_t(i) \equiv \Pr\left[\vec{o_1^T}, q_t = i\right] \quad (7)$$

$$r_{it} = \frac{\text{average request number per user on the } i^{\text{th}} \text{ document at } t^{\text{th}} \text{ time unit}}{\text{average request amount per user at } t^{\text{th}} \text{ time unit}} = \frac{b_{it}/c_{it}}{\sum_{i=1}^{N}(b_{it}/c_{it})}, \quad i \in [1, N], t \in [1, T] \quad (1)$$

which can be readily determined by the forward and backward variables. Then, the model parameters can be estimated by the following formulas:

$$\hat{\pi}_i = \gamma_1(i) \Big/ \sum_{i=1}^{N} \gamma_1(i) \tag{8}$$

$$\hat{a}_{ij} = \sum_{t=1}^{T} \xi_t(i,j) \Big/ \sum_{t=1}^{T} \sum_{n=1}^{N} \xi_t(i,j) \tag{9}$$

$$\hat{b}_i(\vec{r}) = \sum_{t\vec{o}_t=\vec{r}}^{T} \gamma_t(i) \Big/ \sum_{t=1}^{T} \gamma_t(i) \tag{10}$$

$$\hat{p}_i(d) = \sum_{t=2}^{T} \eta_t(i,d) \Big/ \sum_{t=2}^{T} \sum_{d=1}^{D} \eta_t(i,d). \tag{11}$$

We define the entropy (En) of observations fitting to the HsMM and the average logarithmic entropy (ALE) per observation as follows:

$$En = \Pr\left[\vec{o}_1^T|\lambda\right] = \sum_{i,d} \Pr\left[\vec{o}_1^T, (q_T, \tau_T) = (i,d)|\lambda\right] \tag{12}$$

$$ALE = \ln\left(\Pr\left[\vec{o}_1^T|\lambda\right]\right)/T. \tag{13}$$

The details of the above algorithm can be found in [8]. However, existing algorithms of HsMM will be very complex when the observation is a high-dimension vector with dependent elements in the spatial-temporal matrix of AM. Hence, we use PCA to reduce the dimension of AM and apply the independent component analysis to obtain independent elements.

C. Principal Component Analysis

PCA, also called the Karhunen–Loeve transform, is one of the most widely used dimensionality reduction techniques for data analysis and compression. It is based on transforming a relatively large number of variables into a smaller number of uncorrelated variables by finding a few orthogonal linear combinations of the original variables with the largest variance. The first principal component of the transformation is the linear combination of the original variables with the largest variance; the second principal component is the linear combination of the original variables with the second largest variance and orthogonal to the first principal component, and so on. In many data sets, the first several principal components contribute most of the variance in the original data set, so that the rest can be disregarded with minimal loss of the variance for dimension reduction of the data [9]. The transformation works as follows.

The average vector of samples is defined as

$$\vec{\mu} = \frac{1}{T} \sum_{t=1}^{T} \vec{a}_t \tag{14}$$

where T is the total number of samples in the data set, $\vec{a}_t = (a_{1t}, \ldots, a_{Nt})^{\mathrm{T}}$ is the tth sample, and a_{it} is the popularity of document i as defined in (2). The deviation from the average is defined as

$$\vec{\phi}_t = \vec{a}_t - \vec{\mu}. \tag{15}$$

The sample covariance matrix of the data set is defined as

$$C = \frac{1}{T} \sum_{t=1}^{T} (\vec{a}_t - \vec{\mu})(\vec{a}_t - \vec{\mu})^{\mathrm{T}} = \frac{1}{T} \sum_{t=1}^{T} \vec{\phi}_t \vec{\phi}_t^{\mathrm{T}} = \frac{1}{T} \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} \tag{16}$$

where $\mathbf{\Phi} = [\vec{\phi}_1\ \vec{\phi}_2\ \cdots\ \vec{\phi}_T]$. To apply PCA to reduce high dimensional data, eigenvalues and corresponding eigenvectors of the sample covariance matrix C are usually computed by the singular value decomposition (SVD) [9]. Suppose $(\lambda_1, u_1), \ldots, (\lambda_N, u_N)$ are N eigenvalue–eigenvector pairs of the sample covariance matrix C. We choose the largest K eigenvectors. Often there will be just a few of large eigenvalues, and therefore K is the inherent dimensionality of the subspace governing the "signal," while the remaining $(N - K)$ dimensions generally contain noise [9]. The dimensionality of the subspace K can be determined by

$$\sum_{i=1}^{K} \lambda_i \Big/ \sum_{i=1}^{N} \lambda_i \geq \alpha \tag{17}$$

where α is given, which represents the contribution ratio of variation in the subspace to the total variation in the original space. We form an $T \times K$ matrix $\mathbf{U}$ whose columns consist of the K eigenvectors. The representation of the data by principal components consists of projecting the data onto the K-dimensional subspace according to the following rules:

$$\vec{d}_t = \mathbf{U}^{\mathrm{T}}(\vec{a}_t - \vec{\mu}) = \mathbf{U}^{\mathrm{T}}\vec{\phi}_t, \quad t = 1, \ldots, T. \tag{18}$$

D. Independent Component Analysis

ICA is a statistical signal processing technique. In contrast to the PCA which is sensitive to high-order relationships, the basic idea of ICA is to represent a set of random variables using basis function, where the components are statistically independent and as non-Gaussian as possible.

The ICA task is briefly described as follows. Given the set of input samples $\mathbf{X} = [\vec{x}_1 \cdots \vec{x}_T]$, where T is the number of samples, and $\vec{x}_t = [x_{1t} \cdots x_{Nt}]^{\mathbf{T}}$ is the N-dimensional observed vector at the t^{th} time unit. The observed vector $\vec{x}_t$ is assumed to be generated by a linear combination of statistically independent and stationary components (sources), i.e.,

$$\vec{x}_t = \mathbf{R}\vec{y}_t, \quad t = 1, \ldots, T \tag{19}$$

where $\vec{y}_t = [y_{1t} \cdots y_{Nt}]^{\mathbf{T}}$ is the N-dimensional statistically independent signal vector at t^{th} time unit and $\mathbf{R}$ is the $N \times N$ mixing matrix. Corresponding to the sample dataset $\mathbf{X}$, the source data set can be denoted as $\mathbf{Y} = [\vec{y}_1 \cdots \vec{y}_T] = [\mathbf{y}_1\ \mathbf{y}_2 \cdots \mathbf{y}_N]^{\mathrm{T}}$. The issue is how to determine the $N \times N$ invertible de-mixing matrix $\mathbf{W} = \mathbf{R}^{-1}$ so as to recover the components of $\vec{y}_t$ by exploiting information hidden in $\vec{x}_t$, i.e., to determine $\mathbf{W}$ such that the components $y_{1t} \cdots y_{Nt}$ of the transformed vector

$$\vec{y}_t = \mathbf{W}\vec{x}_t \tag{20}$$

are mutually independent. We denote $\mathbf{W}$ by its components or vectors as $\mathbf{W} = (\vec{w}_1, \ldots, \vec{w}_N)^{\mathrm{T}}$, where $\vec{w}_i$ is the transpose vector of the i^{th} row of $\mathbf{W}$ with constraint $E\{(\vec{w}_i^{\mathrm{T}}\vec{x}_t)^2\} = 1$.

The estimation of the data model of ICA is usually performed by formulating an objective function and then minimizing or maximizing it, e.g., maximizing some function $F(\mathbf{y}_1, \ldots, y_N)$ that measures the independence. Different algorithms [10] have been proposed to achieve this objective. This paper applies the FastICA [11] which has been widely used for its good performance and fast convergence during estimation of the parameters. Note that the FastICA algorithms require a preliminary whitening (or sphering) of the sample $\mathbf{X}$, which means the observed data $\vec{x}_t$ is linearly transformed into a variable $\vec{z}_t$ by $\vec{z}_t = \mathbf{V}\vec{x}_t$, such that the covariance matrix of $\vec{z}_t$ is unity, i.e., $E\{\vec{z}_t\vec{z}_t^{\mathrm{T}}\} = \mathbf{I}$. This transformation can be accomplished with the above PCA process [10]. In the following paper, we use symbol $\vec{x}_t$ to denote the whitened data, i.e., $E\{\vec{x}_t\vec{x}_t^{\mathrm{T}}\} = \mathbf{I}$.

The FastICA algorithm is based on negentropy. It defines a contrast function J that measures the nonnormality of a zero-mean random variable $\vec{x}_t$. The objective of this algorithm is then to find $\mathbf{W}$ which maximizes the contrast function J, defined by

$$J_G(\vec{w}_i) = \left[E\left\{G\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\} - E\left\{G\left(\underset{w_i}{\vec{\sigma}^2} v\right)\right\}\right]^2 \tag{21}$$

where G is practically any nonquadratic function, v is a standardized Gaussian variable with zero mean and unit variance, and $\underset{w_i}{\vec{\sigma}^2} = E\{(\vec{w}_i^{\mathrm{T}}\vec{x}_t)^2\}$. Thus, $\underset{w_i}{\vec{\sigma}^2} v$ is a Gaussian variable of the same variance as $\vec{w}_i^{\mathrm{T}}\vec{x}_t$. Note that the optimum $\vec{w}_i$ that maximizes $J_G(\vec{w}_i)$ can be determined by

$$\nabla_{\underset{w_i}{\vec{}}} J_G(\vec{w}_i) = 0. \tag{22}$$

Based on the constraints $E\{(\vec{w}_i^{\mathrm{T}}\vec{x}_t)^2\} = 1$ and $\vec{w}_i^{\mathrm{T}}\vec{w}_i = \left\|\vec{w}_i^{\mathrm{T}}\right\|^2 = 1$, we have $\underset{w_i}{\vec{\sigma}^2} = E\{(\vec{w}_i^{\mathrm{T}}\vec{x}_t)^2\} = \vec{w}_i^{\mathrm{T}}\vec{w}_i$. Then, the item $E\{G(\underset{w_i}{\vec{\sigma}^2} v)\}$ of (21) can be replaced by $E\{G(\vec{w}_i^{\mathrm{T}}\vec{w}_i v)\}$, and (22) can be rewritten as

$$\nabla_{\underset{w_i}{\vec{}}} J_G(\vec{w}_i) = E\left\{\vec{x}_t g\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\} - \beta\vec{w}_i = 0 \tag{23}$$

where g is the derivative function of G, β is a constant that can be evaluated by $\beta = E\{\vec{w}_i^{\mathrm{T}}(0)\vec{x}_t g(\vec{w}_i^{\mathrm{T}}(0)\vec{x}_t)\}$, and $\vec{w}_i(0)$ is the value of $\vec{w}_i$ at the optimum. Newton's method is used to solve this equation. Denoting the function on the middle part of (23) by $F(\vec{w}_i)$ and its Jacobian matrix by $JF(\vec{w}_i)$, then

$$JF\left(\vec{w}_i^{\mathrm{T}}\right) = E\left\{\vec{x}_t\vec{x}_t^{\mathrm{T}} g'\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\} - \beta\mathbf{I} \tag{24}$$

where g' is the derivative of g. Since $\vec{x}_t$ satisfies $E\{\vec{x}_t\vec{x}_t^{T}\} = \mathbf{I}$, a reasonable approximation of the first term of (24) seems to be

$$\begin{aligned} E\left\{\vec{x}_t\vec{x}_t^{\mathrm{T}} g'\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\} &\approx E\left\{\vec{x}_t\vec{x}_t^{\mathrm{T}}\right\} E\left\{g'\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\} \\ &= E\left\{g'\left(\vec{w}_i^{\mathrm{T}}\vec{x}_t\right)\right\}\mathbf{I}. \end{aligned} \tag{25}$$

Thus, the Jacobian matrix becomes diagonal and can easily be inverted. β is also approximated by using the current value $\vec{w}_i(k)$ instead of $\vec{w}_i(0)$, and then the following approximate Newton iteration is

$$\vec{w}_i(k+1) = \vec{w}_i(k) - \frac{\left[E\left\{\vec{x}_t g\left(\vec{w}_i^{\mathrm{T}}(k)\vec{x}_t\right)\right\} - \beta\vec{w}_i(k)\right]}{\left[E\left\{g'\left(\vec{w}_i^{\mathrm{T}}(k)\vec{x}_t\right)\right\} - \beta\right]} \tag{26}$$

$$\vec{w}_i^* = \vec{w}_i(k+1)/\left\|\vec{w}_i(k+1)\right\| \tag{27}$$

where $\vec{w}_i^*$ denotes the new value of $\vec{w}_i$.

Using the above algorithm, we can compute all vectors $\vec{w}_i$ for all $i = 1, \ldots, N$ of the de-mixing matrix $\mathbf{W}$.

We use the iterative algorithms proposed in [11] to achieve this objective. The iterative algorithm includes three steps which are implemented after a round iteration of all components.

Step 1) Let $\mathbf{W} = \mathbf{W}/\sqrt{\|\mathbf{W}\mathbf{W}^T\|}$.
Step 2) Let $\mathbf{W} = 1.5\mathbf{W} - 0.5\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{W}$.
Step 3) Repeat Step 2) until convergence.

E. Detection Architecture

The overall procedure of our detection architecture is illustrated in Fig. 3. The scheme is divided into three phases: data preparation, training, and monitoring.

The main purpose of data preparation is to compute the AM by the logs of the Web server.

The training phase includes the three parts, given here.

1) *PCA transition*. The steps are as follows.
   a) Compute the average matrix and difference matrix, respectively.
   b) Compute the eigenvectors and eigenvalues of the covariance matrix.
   c) Sort the eigenvalues and select the first K eigenvectors, where $\alpha = 80\%$ is given in this paper.
   d) Construct the eigenmatrix $\mathbf{U}$ by the first K eigenvectors.
   e) Transform the AM into K-dimensional uncorrelated principal component dataset.
2) *ICA transition*. The steps are as follows.
   a) Use the outputs of the PCA module (i.e., K-dimensional uncorrelated principal component dataset $\mathbf{U}^{\mathrm{T}}\mathbf{X}$) to estimate the unmixing matrix $\mathbf{W}$ by ICA algorithm.
   b) Transform the K-dimensional dataset into independent signals.
3) *HsMM training*
   a) Use the outputs of ICA module as the model training data set to estimate the parameters of HsMM.
   b) Compute the entropy of the training data set and the threshold.

The monitoring phase includes the following steps:

1) Compute the difference matrix between the testing AM and the average matrix obtained in the training phase by the PCA.
2) Using the eigenmatrix $\mathbf{U}$, compute the feature dataset of the testing AM.
3) Using the de-mixing matrix $\mathbf{W}$, compute the independent signals.
4) The independent signals are inputted to the HsMM; entropies of the testing dataset are computed.
5) Output the result based on the threshold of entropy that was determined in the training phase based on the entropy distribution of the training data set.

In the practical implementation, the model is first trained by the stable and low-volume Web workload whose normality can be ensured by most existing anomaly detection systems, and then it is used to monitor the following Web workload for a period of 10 min. When the period is past, the model will be

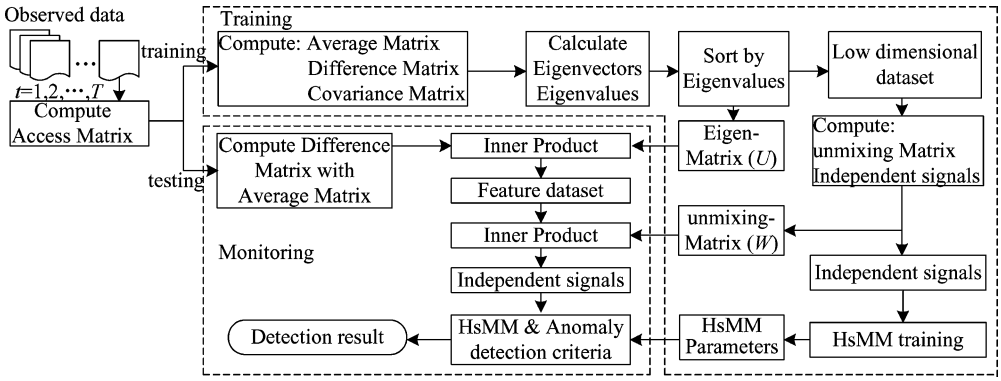


Fig. 3. Monitoring architecture.

updated by the new collected Web workload whose normality is ensured by its entropy fitting to the model. Then, the model is used in anomaly detection for the next cycle.

If some abnormities hiding in the incoming Web traffic are found, the "defense" system will be implemented. For example, we can cluster the Web surfers and evaluate their contributions to the anomalies in the aggregate Web traffic. Then, different priorities are given to the clusters according to their abnormalities and serve them in different priority queues. The most abnormal traffic may be filtered when the network is heavy loaded.

V. Experiments

We implement the algorithm in the NS2 simulator [26]. The network topology is generated by GT-ITM Topology Generator provide by NS2. The simulation includes 1000 client nodes each of which replays one user's trace collected from one of the semifinals of FIFA WorldCup98 [24]. The ratio of randomly selected attack nodes to whole nodes is 10%. Furthermore, we assume the attackers can intercept some of the request segment of normal surfers and replay this segment or "hot" pages to launch the App-DDoS attacks to the victim Web server. Thus, when the attack begins, each potential attack node replays a snippet of another historical flash crowd trace. The interval between two consecutive attack requests is decided by three patterns including constant rate attacks, increasing rate attacks and random pulsing attacks. We use the size of requested document to estimate the victim node's processing time (delay) of each request, i.e., if the requested document is larger, the corresponding processing time will be longer. By this way, we simulate the victim's resource (e.g., CPU) cost by client's requests. Fig. 4(a) shows our simulation scenario. The whole process lasts about 6 h. As shown in Fig. 4(b), the first 2 h data are used to train the model, and the remaining 4 h of data including a flash crowd event are used for test. The emulated App-DDoS attacks are mixed with the trace chose from the period of [3.5 h, 5.5 h]. Fig. 4(c) shows our method on how to collect the observed sequences for detection system. The time unit of this experiment is 5 s. We group 12 consecutive observations into one sequence, the "moving" step is one observation unit and a new sequence is formed using the current observation and the preceding 11 observations. Thus, two consecutive sequences will have 11 overlapped observations. We used 25 consecutive sequences, which last for 36 observation units or 3 min, to detect anomaly accesses. Therefore, in every 5 s when a new observation is obtained, a new sequence is formed by 12 consecutive samples (corresponding to 60 s of traffic). This sequence is then measured by calculation of its entropy for the HsMM. The moving average of entropies over 25 consecutive sequences (corresponding to 36 samples or 180 s of traffic) is used for detection of the attacks.

A. Constant Rate Attack

Constant rate attack, the simplest attack technique, is typical among known DDoS attacks. We do not arrange the attack sources to simultaneously launch constant rate App-DDoS attacks and to generate requests at full rate, so that they cannot be easily identified through attack intensity. As shown in Fig. 5(a), we use $(H, t_s, t_e, \Delta t, l)$ to denote the parameters of the constant rate attack. The notation is listed in Fig. 4(d). Three parameters (i.e., $H, \Delta t, l$) are set randomly by each attack node before it launches the attacks.

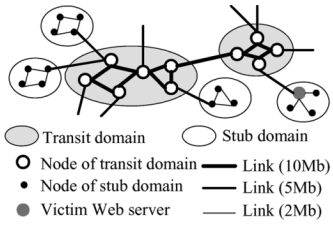
Fig. 6(a) shows the entropy varying with the time, where curve a represents the normal flash crowd's entropy and curve b represents the entropy of flash crowd mixed with constant rate App-DDoS attacks in zone B. Therefore, it is easy to find out that there exist attacks in the period B.

B. Increasing Rate Attack

An abrupt change in traffic volume is an important signal to initiate anomaly detection. The attacker may use the gradually increasing rate, as shown in Fig. 5(b). The state change in the victim network could be so gradual that services degrade slowly over a long period, delaying detection of the attack. We use $(H, t_s, t_e, \Delta t, l_1, l_2, l)$ to denote the parameters of the increasing rate attack. Five variables (i.e., $H, \Delta t, l_1, l_2, l$) are set randomly by each attack node before it launches the attacks. Fig. 6(b) shows the entropy changing with the time, where curve a represents the normal flash crowd's entropy and curve b represents entropy of the traffic that mixes normal flash crowd with increasing rate App-DDoS attacks, where the attacks start gradually in zone B and end gradually in zone D. As it shows, the entropy can be used to discover the attacks in the early beginning.

C. Stochastic Pulsing Attacks

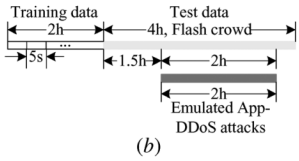
Instead of constantly injecting traffic flows with huge rates into the network, pulsing attacks, which are also called shrew attacks, are much more difficult to be detected and therefore



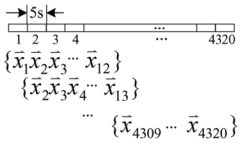
(a)

The simulation topology is generated by GT-ITM Topology Generator provided by NS2. The parameters are given as the following:

1. Three-level hierarchy: transit domain, stub domain and nodes;
2. 2 transit domains; each transit domain has (on average) five nodes; each transit node connects (on average) to five stub domain;
3. each stub domain has (on average) twenty nodes;
4. 10Mps link for domain; 5Mbp for stub domain; 2Mps for nodes;
5. Attack coverage is 10% of the nodes except the 10 transit nodes, i.e., 100 nodes are potential attack nodes;



(b)

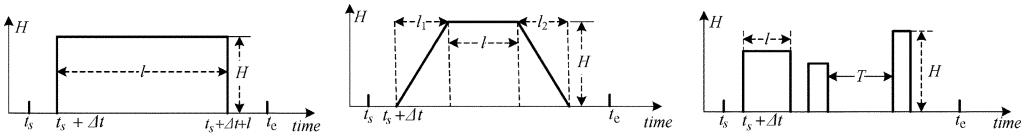


(c)

Symbol	Definition	Setting manner
H	intensity (request rate) of the attack	Stochastic
t_s	presetting start time of attack	Constant
t_e	presetting terminative time of attack	Constant
Δt	Difference between the real start time of attack and t_s	Stochastic
l	Duration of the attack	Stochastic
l_1	Duration of the gradually increasing rate	Stochastic
l_2	Duration of the gradually descending rate	Stochastic
T	Attack period of stochastic pulsing attacks	Stochastic

(d)

Fig. 4. Simulation configuration. (a) Compendious simulation scenarios; (b) Composition of the dataset. (c) Grouping of the dataset. (d) Definition of attack parameters.



(a) (b) (c)

Fig. 5. Different emulated App-DDoS attacks. (a) Constant rate attacks. (b) Increasing rate attacks. (c) Stochastic pulsing attacks.

can damage the victim for a long time without being noticed. Pulsing attackers launch attacks by sending burst pulses periodically. Such attacks have high peak rate while maintaining a low average rate to exhibit "stealthy" behavior.

We emulate a stochastic-type pulsing attack of App-DDoS. As shown in Fig. 5(c), a single source stochastic pulsing attack is modeled as a square waveform HTTP request stream with attack period T, burst length l, and burst rate H. Among the variables, T, l, H, and Δt are preset by each attack node randomly before it is going to fire the next pulsing attack, which makes the different attack nodes have different attack parameters and the same attack node have different attack parameters in different attack periods. Thus, the attacks exhibit a fluctuating rate oscillating between different H and zero, appearing as a stochastic ON/OFF process. Such stochastic pulses are very difficult to be detected by the existing methods that are based on traffic volume analysis, because the average rate of the attacks is not remarkably higher than that of a normal user.

Fig. 6(c) shows the entropy varies with the time, where curve a represents the normal flash crowd's entropy and curve b in zone B represents the entropy of the mixed traffic that contains the normal flash crowd and the stochastic pulsing rate of App-DDoS attacks. As it shows, the attacks can be easily discovered from the normal flash crowd workload by the entropy.

The fact that the entropy series is stable shows that the document popularity is stable. Although the attackers can inject vicious requests into the flash crowd traffic, the original popularity distribution of documents is changed, which causes the entropy series lower than the normal level. Therefore, we can detect the potential App-DDoS attacks by the entropy of document popularity fitting to the model.

D. Performance

In the above scenarios, based on the entropy outputted by the algorithm, we can detect the anomaly caused by the App-DDoS attack. Fig. 7(a) shows the distributions of average entropy. There exist significant differences in entropy distributions between two groups: the normal Web traffic's entropies are larger than -6, but most entropies of the traffic containing attacks are less than -8. The statistical results of the entropy of normal training data and emulated App-DDoS attacks are given as

$$Training\_data: \begin{cases} Average(entropy) = -4.0981 \\ Standard\_Deviation(entropy) = 0.3456 \end{cases}$$

$$Emulated\_Attacks: \begin{cases} Average(entropy) = -8.8275 \\ Standard\_Deviation(entropy) = 1.8377. \end{cases}$$

Fig. 7(b) shows that, if we take -5.3 as threshold value of normal Web traffic's average entropy, the false negative ratio (FNR) is about 1%, and the detection rate is about 90%. Because the number of the remaining principal components will

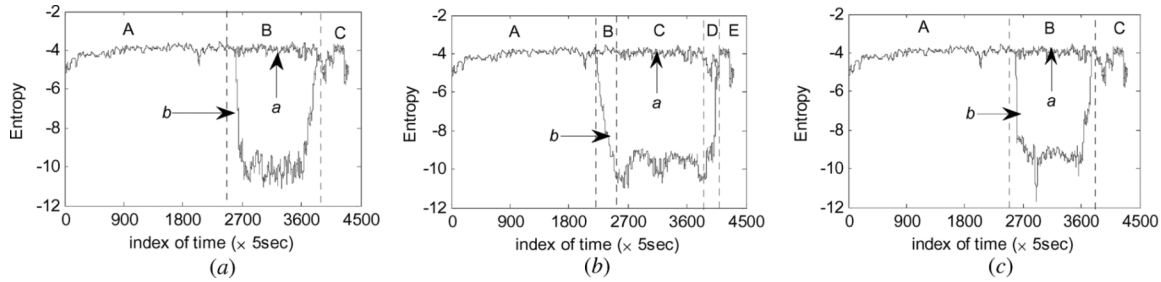


Fig. 6. Entropy versus time of different attacks. (a) Detection for constant rate attacks. (b) Detection for increasing rate attacks. (c) Detection for stochastic pulsing attacks.

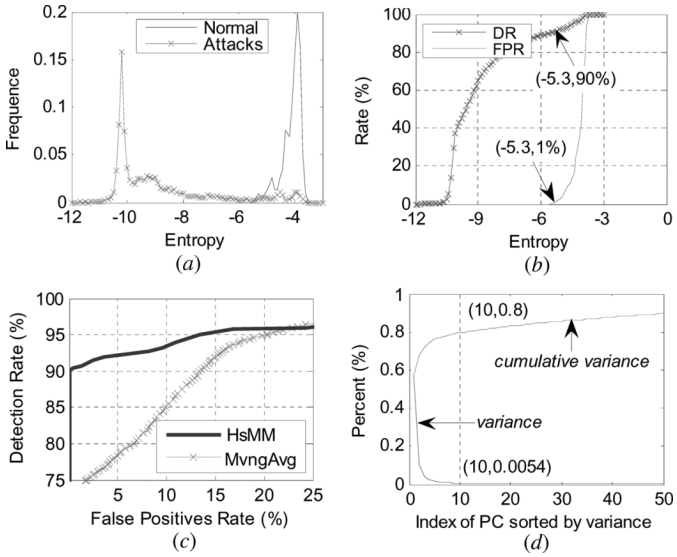


Fig. 7. Performance. (a) Distribution of entropy. (b) Entropy, DR, and FPR. (c) ROC curve. (d) Dontribution ratio *versus* number of PCs.

affect the precision of detection, we use the contribution ratio defined in (17) to decide the first K principal components (PCs). We compared the performance of the proposed scheme with the moving average in implementing anomaly detection. The length of moving average is 60 samples (i.e., 5 min); step of moving average is 12 samples (i.e., 1 min); the cosine distance between the observed vector and the average vector of training data is used as the detection criterion. Fig. 7(c) shows the receiver operating characteristics (ROCs) of our scheme and the moving average method. As it shows, our method is remarkably better than the moving average in the detection rate given the false positive rate. Fig. 7(d) shows the variance versus number of PCs. From this figure, we find the variance is mainly contributed by the first ten PCs whose cumulative ratio is about 80%. This means we can keep the first ten PCs at the cost of losing 20% information.

VI. Discussion

A. Performance of Our Approach

We discuss the performance by the following aspects.

1) Multidimensional Data Processing: Multidimensional detection may become a mainstream method in anomaly detection. However, it is very difficult to deal with the multidimensional observation vector sequence without mass computation or assuming a special distribution for the observed data. Thus, PCA and ICA are used before the HsMM-based detector. Because the elements of each vector obtained through ICA are independent, the joint output probability distribution function of HsMM can be simplified as $b_i(\vec{o_t}) = \prod_{j=1}^{K} b_i(o_{jt})$, where o_{jt} is the j^{th} element of vector $\vec{o_t}$, which enables the detector to implement the multidimensional monitoring with less computation and without special assumption for the distribution of the original data.

The basic goal of PCA is to reduce the dimension of the data. Indeed, it can be proven that the representation given by PCA is an optimal linear dimension reduction technique in the mean-square sense. Such a reduction in dimension has important effect. The computational overhead in the subsequent processing stages is reduced, and the noise that is not contained in the first K components is removed. The main reasons for using the PCA in this paper are: 1) the principle components are typical for the high dimensional data of the problem without sacrificing valuable information and 2) it does not require any special distributional assumption, compared with many statistical methods that often assume a normal distribution or resort to the use of central limit theorem.

2) Advantages of HsMM: HsMM can describe most practical stochastic signals, including non-stationary and the non-Markovian. It has been widely applied in many areas such as mobility tracking in wireless networks, activity recognition in smart environments, and inference for structured video sequences. Many effective algorithms for HsMM parameter estimation have been developed in the literature. In contrast to existing anomaly detection methods developed in biosurveillance [37], the nonstationary and the non-Markovian properties of HsMM can best describe the self-similarity or long-range dependence of network traffic that has been proved by vast observations on the Internet [32], [33].

3) Self-Adaptive Scheme: Based on our experiment (Fig. 6), we found the normal user's access behavior and the Website structure exhibit hours-long stability regardless of whether or not there are flash crowd events occurring during the period, i.e., the popularity of documents is mainly affected by the daily life of the users or information update of the Web pages. Therefore, the model parameters of document popularity change in the period of ten minutes or hours. Hereby, the model parameters can be updated by the self-adaptive implementation [34] in a period of ten minutes, in the way of implementing off-line or asynchronously, which will not affect the online detection. In

our experiments in this paper, the model parameters are updated once per hour.

4) Computational Complexity: In the training phase, it requires $O((MD+M^2)TI)$ computations and $O(MT)$ memory capacity, where M is the number of HsMM's states, D is the maximum interval between any two consecutive state transitions, T is the length of the training samples, and I is the number of iterations. In fact, we can implement off-line training instead of online training in practice. In our experiment, $M = 9$, $D = 60$, and $T = 1440$. With the convergence criteria of 0.01, the iteration times I is about 10. The training can be finished within 10 min by MATLAB.

In the detection phase, the computation comes from PCA, ICA, and HsMM. The computational complexities of the first two modules are $O(NKT)$ and $O(K^2T)$, respectively, where in our experiment $N = 3000$ is the dimension of the original observed vector, $K = 10$ is the number of selected principal components, and $T = 12$ is the length of each test sequence (i.e., one test sequence contains 12 samples and one sample is collected in every 5 s). Because the entropy of the test sequence fitting to the model can be determined by the forward formula given by (12), which requires only $O((MD+M^2)T)$ computations. Thus, the computational complexity for a test sequence is $O(NKT+K^2T+(MD+M^2)T)$, which means about 0.3 M multiplications in this instance. For the computer that is configured with Intel Core 2 1.83-GHz CPU, 2-G RAM and 32 bits operation system, our simulation experiments showed that it requires about 500 ms for performing the anomaly detection in every 5 s of sampling period.

B. Parameter Settings

Before the scheme is applied to detection, some parameters have to be preset, which includes the time unit of observed data, the length of the observed vector sequence, the number of the remaining principal components, and the detection threshold of entropy for anomaly detection in HsMM. We discuss each of them as follows.

The time unit and the length of the observed vector sequence can be set according to the computation ability and memory of the detection system. In this paper, we set the time unit to be 5 s and the length of one observed sequence to be 1 min. Although the small scale of the time unit may bring us high precision, the length of sequence can not be set too short because it may not contain sufficient attack signals for reliable detection. For these considerations, we suggest the time unit is selected in between [5 s, 20 s] and the length of sequence is selected in between [1 min, 5 min].

The number of remaining PCs can be decided by the cumulative variance. We select the largest PCs whose cumulative variance is over 80% in our experiments, which actually resulted in ten PCs in the experiments. The PCs can be selected by a higher cumulative variance, but this may require more computational capacity and memory amount.

In contrast to most current work that decides the detection threshold by subjectivism or empiricism, we use Gaussian distribution to provide a universal and reasonable method for the detection threshold. The Central Limit Theorem (CLT) has told us that given a distribution with a mean (μ) and variance (σ), the sampling distribution approaches a Gaussian distribution. Thus, we can describe the entropy distribution of training data by Gaussian distribution. From the rational of Gaussian distribution, we know $\pm 3\sigma$ error level could give us a confidence interval of 99.7% which is good enough even in high precision detection scenarios. Table I lists the detection threshold setting and their corresponding FPR and DR of our experiments (the mean μ and variance σ have been given in Section V). As indicated in this table, the detection level could be reasonably selected to be $\mu \pm 3.29\sigma$, and this choice ensures us with FPR smaller than 1.5% and DR larger than 90%. This shows that the detection threshold determined by the CLT can achieve pretty high accuracy in detection.

TABLE I
GAUSSIAN DISTRIBUTIONS' DETECTION THRESHOLD

Detection Level	Detection Threshold	FPR	DR
$\mu\pm\sigma$	[-4.4437,-3.7525]	0.2008	0.9545
$\mu\pm1.65\sigma$	[-4.6684,-3.5278]	0.0920	0.9309
$\mu\pm1.96\sigma$	[-4.7755,-3.4207]	0.0753	0.9257
$\mu\pm3\sigma$	[-5.1350,-3.0612]	0.0178	0.9108
$\mu\pm3.29\sigma$	[-5.2352,-2.9610]	0.0129	0.9074

VII. CONCLUSION

Creating defenses for attacks requires monitoring dynamic network activities in order to obtain timely and signification information. While most current effort focuses on detecting Net-DDoS attacks with stable background traffic, we proposed a detection architecture in this paper aiming at monitoring Web traffic in order to reveal dynamic shifts in normal burst traffic, which might signal onset of App-DDoS attacks during the flash crowd event. Our method reveals early attacks merely depending on the document popularity obtained from the server log. The proposed method is based on PCA, ICA, and HsMM. We conducted the experiment with different App-DDoS attack modes (i.e., constant rate attacks, increasing rate attacks and stochastic pulsing attack) during a flash crowd event collected from a real trace. Our simulation results show that the system could capture the shift of Web traffic caused by attacks under the flash crowd and the entropy of the observed data fitting to the HsMM can be used as the measure of abnormality. In our experiments, when the detection threshold of entropy is set -5.3, the DR is 90% and the FPR is 1%. It also demonstrates that the proposed architecture is expected to be practical in monitoring App-DDoS attacks and in triggering more dedicated detection on victim network.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments and suggestions to improve this work. The authors would also like to thank Prof. Z. Lin (Sun Yat-Sen University, China) for his careful reading and valuable suggestions for the improvement of the paper's presentation.

REFERENCES

[1] K. Poulsen, "FBI Busts Alleged DDoS Mafia," 2004. [Online]. Available: http://www.securityfocus.com/news/9411

[2] "Incident Note IN-2004-01 W32/Novarg. A Virus," CERT, 2004. [Online]. Available: http://www.cert.org/incident_notes/ IN-2004-01.html
[3] S. Kandula, D. Katabi, M. Jacob, and A. W. Berger, "Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds," MIT, Tech. Rep. TR-969, 2004 [Online]. Available: http://www.usenix.org/events/nsdi05/tech/ kandula/kandula.pdf
[4] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "Modeling, Analysis and Simulation of Flash Crowds on the Internet," Storage Systems Research Center Jack Baskin School of Engineering University of California, Santa Cruz Santa Cruz, CA, Tech. Rep. UCSC-CRL-03-15, Feb. 28, 2004 [Online]. Available: http://ssrc.cse.ucsc.edu/, 95064
[5] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *Proc. 11th IEEE Int. World Wide Web Conf.*, May 2002, pp. 252–262.
[6] Y. Xie and S. Yu, "A detection approach of user behaviors based on HsMM," in *Proc. 19th Int. Teletraffic Congress (ITC19)*, Beijing, China, Aug. 29–Sep. 2 2005, pp. 451–460.
[7] Y. Xie and S. Yu, "A novel model for detecting application layer DDoS attacks," in *Proc. 1st IEEE Int. Multi-Symp. Comput. Computat. Sci. (IMSCCS\06)*, Hangzhou, China, Jun. 20–24, 2006, vol. 2, pp. 56–63.
[8] S.-Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit duration hidden Markov model," *IEEE Signal Process. Lett.*, vol. 10, no. 1, pp. 11–14, Jan. 2003.
[9] L. I. Smith, A Tutorial on Principal Components Analysis [EB/OL], 2003 [Online]. Available: http://www.snl.salk.edu/~shlens/pub/ notes/pca.pdf
[10] A. Hyvärinen, "Survey on independent component analysis," *Neural Comput. Surveys*, vol. 2, pp. 94–128, 1999.
[11] A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 626–634, Jun. 1999.
[12] J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra, "Proactive detection of distributed denial of service attacks using MIB traffic variables a feasibility study," in *Proc. IEEE/IFIP Int. Symp. Integr. Netw. Manag.*, May 2001, pp. 609–622.
[13] J. Yuan and K. Mills, "Monitoring the macroscopic effect of DDoS flooding attacks," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 4, pp. 324–335, Oct.-Dec. 2005.
[14] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. Int. Conf. Network Protocols*, 2002, pp. 312–321.
[15] T. Peng and K. R. M. C. Leckie, "Protection from distributed denial of service attacks using history-based IP filtering," in *Proc. IEEE Int. Conf. Commun.*, May 2003, vol. 1, pp. 482–486.
[16] B. Xiao, W. Chen, Y. He, and E. H.-M. Sha, "An active detecting method against SYN flooding attack," in *Proc. 11th Int. Conf. Parallel Distrib. Syst.*, Jul. 20–22, 2005, vol. 1, pp. 709–715.
[17] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proc. IEEE INFOCOM*, 2002, vol. 3, pp. 1530–1539.
[18] L. Limwiwatkul and A. Rungsawangr, "Distributed denial of service detection using TCP/IP header and traffic measurement analysis," in *Proc. Int. Symp. Commun. Inf. Technol.*, Sappoo, Japan, Oct. 26–29, 2004, pp. 605–610.
[19] S. Noh, C. Lee, K. Choi, and G. Jung, "Detecting Distributed Denial of Service (DDoS) attacks through inductive learning," *Lecture Notes in Computer Science*, vol. 2690, pp. 286–295, 2003.
[20] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-resilient scheduling to counter application layer attacks under imperfect detection," in *Proc. IEEE INFOCOM*, Apr. 2006 [Online]. Available: http://www-ece.rice.edu/networks/papers/dos-sched.pdf
[21] W. Yen and M.-F. Lee, "Defending application DDoS with constraint random request attacks," in *Proc. Asia-Pacific Conf. Commun.*, Perth, Western Australia, Oct. 3–5, 2005, pp. 620–624.
[22] S. Ranjan, R. Karrer, and Knightly, "Wide area redirection of dynamic content by Internet data centers," in *Proc. 23rd Ann. Joint Conf. IEEE Comput. Commun. Soc.*, Mar. 7–11, 2004, vol. 2, pp. 816–826.
[23] [Online]. Available: http://www.caida.org/analysis/security/sco-dos/
[24] [Online]. Available: http://ita.ee.lbl.gov/html/traces.html
[25] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle, "Stochastic models for generating synthetic HTTP source traffic," in *Proc. IEEE INFOCOM*, 2004, vol. 3, pp. 1546–1557.
[26] NS2 [Online]. Available: http://www.isi.edu/nsnam/ns/
[27] W. Wang, X. Guan, and X. Zhang, "A novel intrusion detection method based on principle component analysis in computer security," in *Proc. Int. Symp. Neural Networks*, Dalian, China, Aug. 19–21, 2004, pp. 657–662, Part II.
[28] Y. Xie and S. Yu, "A dynamic anomaly detection model for web user behavior based on HsMM," in *Proc. 10th Int. Conf. Comput. Supported Cooperative Work in Design (CSCWD 2006)*, Nanjing, China, May 3–5, 2006, vol. 2, pp. 811–816.
[29] S. Burklen, P. J. Marron, S. Fritsch, and K. Rothermel, "User centric walk: An integrated approach for modeling the browsing behavior of users on the Web," in *Proc. 38th Ann. Simulation Symp.*, Apr. 4–6, 2005, pp. 149–159.
[30] C. Roadknight, l. Marshall, and D. Vearer, "File popularity characterisation," *ACM SIGMETRICS Performance Eval. Rev.*, vol. 23, no. 4, pp. 45–50, Mar. 2000.
[31] A. M. G. Cooper, R. Tsui, and M. Wagner, Summary of Biosurveillance-Relevant Technologies. [Online]. Available: http://www.cs.cmu.edu/~awm/biosurv-methods.pdf
[32] S. Z. Yu, Z. Liu, M. Squillante, C. Xia, and L. Zhang, "A hidden semi-Markov model for web workload self-similarity," in *Proc. 21st IEEE Int. Performance, Computing, Commun. Conf.*, Phoenix, AZ, Apr. 3–5, 2002, pp. 65–72.
[33] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the selfsimilar nature of ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

Yi Xie received the B.S. and M.S. degrees from Sun Yat-Sen University, Guangzhou, China, where he is currently working toward the Ph.D. degree in the Department of Electrical and Communication Engineering, School of Information Science and Technology.

From July 1996 to July 2004, he was with the Civil Aviation Administration of China (CAAC). He was a Visiting Scholar with George Mason University from 2007 to 2008. His research interests are in network security and Web-user behavior model algorithm.

Shun-Zheng Yu (M'08) received the B.Sc. degree from Peking University, Peking, China, and the M.Sc. and Ph.D. degrees from Beijing University of Posts and Telecommunications, Beijing, China.

He was a Visiting Scholar with Princeton University and IBM Thomas J. Watson Research Center during 1999 to 2002. He is currently a Professor with the School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, China. Networking, traffic modeling, anomaly detection, and algorithms for hidden semi-Markov model have been of particular interest over recent years.