Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 821315, 6 pages
http://dx.doi.org/10.1155/2013/821315

Research Article

Real-Time Detection of Application-Layer DDoS Attack Using Time Series Analysis

Tongguang Ni, Xiaoqing Gu, Hongyuan Wang, and Yu Li

School of Information Science and Engineering, Changzhou University, Changzhou 213164, China

Correspondence should be addressed to Hongyuan Wang; tiddyddd@163.com

Received 7 June 2013; Accepted 25 August 2013

Academic Editor: Xiaomei Qi

Distributed denial of service (DDoS) attacks are one of the major threats to the current Internet, and application-layer DDoS attacks utilizing legitimate HTTP requests to overwhelm victim resources are more undetectable. Consequently, neither intrusion detection systems (IDS) nor victim server can detect malicious packets. In this paper, a novel approach to detect application-layer DDoS attack is proposed based on entropy of HTTP GET requests per source IP address (HRPI). By approximating the adaptive autoregressive (AAR) model, the HRPI time series is transformed into a multidimensional vector series. Then, a trained support vector machine (SVM) classifier is applied to identify the attacks. The experiments with several databases are performed and results show that this approach can detect application-layer DDoS attacks effectively.

1. Introduction

DDoS attacks have caused severe damage to servers and will cause even greater intimidation to the development of new Internet services. DDoS attacks are categorized into two classes: network-layer DDoS attacks and application-layer DDoS attacks. In network-layer DDoS attacks, attackers send a large number of bogus packets towards the victim server and normally attackers use IP spoofing. The victim server or IDS can easily distinguish legitimate packets from DDoS packets. In contrast, in application-layer DDoS attacks, attackers attack the victim server through a flood of legitimate requests. In this attack model, attackers attack the victim Web servers by HTTP GET requests and pulling large files from the victim server in overwhelming numbers. Also, attackers can run a massive number of queries through the victim's search engine or database query to bring the server down.

To circumvent detection, the attackers increasingly move away from pure bandwidth floods to stealthy DDoS attacks that masquerade as flash crowd. Flash crowd [1, 2] refers to the situation when a very large number of users simultaneously access a website, which may be due to the announcement of a new service or free software download. Because burst traffic and high volume are the common characteristics of application-layer DDoS attacks and flash crowd, it is not easy to distinguish them. Therefore, application layer DDoS attacks may be stealthier and more dangerous for the websites than the general network-layer DDoS attacks.

Most well-known DDoS countermeasure [3] techniques are against network-layer DDoS attacks. Those techniques cannot handle application-layer DDoS attacks. Countering application-layer DDoS attacks becomes a great challenge. Statistical methods is used to detect characteristics of HTTP sessions and employed rate-limiting as the primary defense mechanism in [4]. Constraint random request attacks by the statistical methods are used to defend against the application-layer DDoS attacks in [5]. A CAPTCHA puzzle is used to ensure that the response is generated by a human not by a machine in [6]. A semi-Markov model is proposed to describe the browsing Behaviors of Web surfers in [7], and an improved semi-Markov model is proposed to describe the dynamic behavior process of aggregated traffic in [8]. Recently, trust-based methods [9, 10] were introduced for resisting application-layer DDoS attacks. The common feature of these methods is that a defense system establishes credit records for each user. The credit value given to a sender is designed to be measured based on its history of communication patterns.

In application-layer DDoS attacks, attack sources have been programmed and worked according to their attack

functions, so detection based on its pattern is possible. In this paper, the entropy of HTTP GET requests per source IP address (HRPI) is proposed, which reflects the essential features of application-layer DDoS attacks: the distribution of source IP address and HTTP GET request frequency. To increase the detection accuracy in various conditions, HRPI time series are transformed into a multidimensional vector by estimating the adaptive autoregressive (AAR) model parameters using Kalman filter. Furthermore, a support vector machine (SVM) classifier, which is trained by AAR parameters of HRPI time series, is applied to classify the state of current network traffic and identify the application-layer DDoS attacks.

The rest of the paper is organized as follows. Section 2 discusses the application-layer DDoS attacks and details their properties. Section 3 describes our approach to detect the application-layer DDoS attacks. In Section 4, experiments are presented to validate our detection model. Finally, the conclusion is given in Section 5 and it points out the future work.

2. Application-Layer DDoS Attacks

Application-layer DDoS attacks can be clustered into two types: bandwidth exhausting (HTTP flooding) and resources exhausting [11]. In bandwidth exhausting DDoS attacks, attackers attack the victim server through a flood of legitimate requests. Any zombie machine has to establish a TCP connection with the victim server, which requires a genuine IP address. Attacks mainly focus on the homepage or a hot webpage, and also different web pages. In this case, the sources of the traffic converge to a group of points and high HTTP Get request rate from the attackers.

Besides the flooding attack pattern, application-layer DDoS attacks may focus on exhausting the server resources such as Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth. With increasing computational complexity in Internet applications and larger network bandwidth, server resources may become the bottleneck of these applications. This type of attack is able to use fewer zombies but the attack has an even larger damage to the website. However, the traffic will be similar to the bandwidth exhausting DDoS. As a result, the sources of the traffic converge to a group of points but the targets of the traffic become dispersed in some extent. At the same time, the frequency of HTTP Get request from the attackers is highly large.

On the Web, flash crowd refers to the situation when a very large number of users simultaneously access a popular website, which produces a surge in traffic to the website and might cause the site to be virtually unreachable. DDoS attacks are absolutely different from flash crowd, DDoS attacks are due to an increase in the request rates for a small group of clients while flash crowd is due to an increase in the number of clients. The sources of flash crowd are definitely scattered, conversely, the sources of application-layer DDoS attacks converge to a group of points.

3. Our Approach

3.1. Definition of HRPI. For popular websites, the traffic targeted is a stream of successive HTTP Get requests.

Definition 1. HTTP Get requests in the certain time interval Δt is given in the form of $\langle (x_1, s_1), (x_2, s_2), \ldots, (x_n, s_n) \rangle$. For the (x_i, s_i), x_i is the source IP address and s_i is the number of HTTP Get requests for x_i.

Definition 2. Entropy of HTTP GET requests per source IP address (HRPI) is defined as

$$\mathrm{SRE} = -p(x_i) \sum lbp(x_i), \tag{1}$$

where $p(x_i)$ is the probability of HTTP Get requests belonging to x_i, and $p(x_i) = s_i / \sum_{i=1}^{n} s_i$.

HRPI as a summarization tool is used to quantify the degree of dispersal or concentration of HTTP Get request feature distributions. According to the analysis in Section 2, we deduced the following conclusion (DDoS as 1, normal as 2, flash crowd as 3):

$$\mathrm{HRPI}(3) > \mathrm{HRPI}(2) > \mathrm{HRPI}(1). \tag{2}$$

In most cases, distribution form of source IP address of legitimate users is more uniformly scattered across the Internet; the distribution form of source IP address of attackers is more cumulative in someplaces. In DDoS attacks, several clusters of source IP addresses and lager number of HTTP GET requests are converged, so HRPI value dramatically drops when attacks happen. Conversely, the sources of flash crowd are scattered and there were no such clusters, so it will result in an abnormal increase in HRPI of the network.

3.2. Generation of HRPI Time Series. Adaptive autoregressive AAR (p) model [12] of degree p is defined as

$$y_t = \sum_{k=1}^{p} a_t^k y_{t-k} + e_t, \tag{3}$$

where y_t denotes the observation at instant t and a_t^k denotes the time-varying model parameters. As the traffic collecting device may cause measurement errors, stochastic variable e_t is used to capture this error. The model uses a weighted sum of p previous values to estimate the current observation value. The weights a_t^k ($k = 1, \ldots, p$) are time dependent, and the current value can be predicted as a linear combination of p past values. By using time-varying AAR model, we allow a model of normal behavior to adapt to the changes of the monitored system.

Kalman filter is an adaptive and recursive data processing algorithm that is suited for online estimation [13, 14]. Kalman filter can process traffic matrix as a whole and all traffic can be estimated simultaneously. This implies that we do not have to consider all the previous data again, to compute the optimal estimates; we only need to consider the estimates from the previous time step and the new measurement.

In our case, we estimate the AAR model parameters from the observed alert series $\{y_t\}$. The true parameters cannot be observed directly and in state space terminology they are called the state X. Now, assume that we have an observation model giving the relation between the unobservable state

and the observations, and an evolution model describing the time-varying nature of the state. So the AAR model can be put in vector form as follows:

$$Y_t = H_t X_t + e_t, \tag{4}$$

where H_t denotes an internal matrix and $H_t = (Y_t, \ldots, Y_{t-p})$. X_t denotes the state vector at instant t and $X_t = (a_t^1, \ldots, a_t^p)^T$.

Without prior information, the evolution of the state is often described with a random walk model [15]. A linear equation is constructed as follows to build a prediction model to correlate X_{t+1} and X_t:

$$X_{t+1} = X_t + w_t, \tag{5}$$

where state noise $w_t$and measurement noise e_t are uncorrelated, zero-mean white-noise processes and with covariance matrices σ_w^2 and σ_e^2, respectively.

For representing the Kalman filter equations, $\widehat{X}$ denotes the estimate of X and $P_{t|t-1}$ denotes error covariance matrix for estimation error of the state at instant t using observations accumulated at instant $t-1$. When initial conditions, $\widehat{X}_0 = E[X_0]$ and error covariance matrix $P_0 = E[(\widehat{X}_0 - X_0)(\widehat{X}_0 - X_0)^T]$, the system state $\widehat{X}_{t|t-1}$ can be estimated iteratively by the following equations:

$$\begin{aligned}
\widehat{X}_{t|t-1} &= X_{t-1}, \\
P_{t|t-1} &= P_{t-1} + C_{w_{t-1}}, \\
K_t &= P_{t|t-1} H_t^T \left(H_t P_{t|t-1} H_t^T + C_{e_t}\right)^{-1}, \\
\widehat{X}_t &= \widehat{X}_{t|t-1} + K_t \left(Y_t - H_t \widehat{X}_{t|t-1}\right), \\
P_t &= \left(I - K_t H_t\right) P_{t|t-1}.
\end{aligned} \tag{6}$$

Using Kalman filter in practice requires initial values for state X_0, error covariance P_0, state noise covariance C_w ($C_w = \sigma_w^2 I$), and observation noise covariance C_e ($C_e = \sigma_e^2 = I$). A common approach is to set $X_0 = 0$, $P_0 = I$, and run the algorithm on a short segment from observation data backwards. The values obtained in this way for X and P_0 are then used to initialize these values in the actual processing run. The adaptation speed increases with C_w and the variance of state estimates is inversely proportional to the value of C_w. Therefore, it should be chosen for a desired balance between state estimate variance and filter adaptation speed according to the application. C_w needs to be set dynamically and according in application.

3.3. Kalman Filter Smoothing. There are three classical smoothing algorithms, fixed-point smoother, fixed-interval smoother, and fixed-lag smoother. We use fixed-lag smoother, since it is suitable for online processing when a small, fixed delay of L observations is allowed [16].

To estimate the state X_t at instant t with a fixed-lag smoother, we will wait to have observations up to instant $t+L$, where $L > 0$. The state and observation equations have now extended variables. The Kalman filter equations remain the same, and the observation (4) becomes

$$Y_t = [H_t, 0, \ldots, 0] \begin{bmatrix} X_t \\ X_{t-1} \\ \vdots \\ X_{t-L} \end{bmatrix} + e_t. \tag{7}$$

The simplified state (5) can be written as

$$\begin{bmatrix} X_{t+1} \\ X_t \\ X_{t-1} \\ \vdots \\ X_{t-(L-1)} \end{bmatrix} = \begin{bmatrix} X_t \\ X_{t-1} \\ X_{t-2} \\ \vdots \\ X_{t-L} \end{bmatrix} + \begin{bmatrix} w_t \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{8}$$

3.4. SVM Classifier. By sampling the network traffic with time interval Δt, calculating the HRPI of every sample, the HRPI sample series $\{\text{HRPI}_i, i = 1, 2, \ldots, N\}$ is gotten, N is the length of the series. Based on (6)–(8), multidimensional vector $\widehat{X}(i)$ of degree p can be used to describe the state features of network traffic. As a result, detecting DDoS attacks equates to classifying $\widehat{X}(i)$ series virtually.

Support vector machine (SVM) is applied here, which is a well-known data classification technique, to classify AAR parameters vector. SVM method can get the optimal solution whether the sample size tends to be finite or infinite. It can establish a mapping of a nonlinear kernel function, structuring the optimal hyperplane, so problem can be converted into a linearly separable one in the high-dimensional feature space. Besides, it solves the dimension problem and its complexity has nothing to do with the sample's dimension.

Since traffic is only considered as legitimate or attack, it is naturally a binary classification problem. The SVM classifier can be described as

$$\eta = \sum_{i=1}^{M} \alpha_i y_i K(\varphi_i, \varphi) + b, \tag{9}$$

where η is the classification result for the sample, α_i is the Lagrange multiplies, y_i is the category, and $y_i \in \{-1, 1\}$. $K(\varphi_i, \varphi)$ is the kernel function and b is the deviation factor.

The optimal hyperplane that SVM classifier created in the high-dimensional feature space is

$$f(\varphi) = \operatorname{sgn}\left(\sum_{i \in \text{SV}} \alpha_i y_i \left(K(\varphi_r, \varphi_i) + K(\varphi_s, \varphi_i)\right)\right), \tag{10}$$

where

$$b = \frac{1}{2} \sum_{i \in \text{SV}} \alpha_i y_i \left(K(\varphi_r, \varphi_i) + K(\varphi_s, \varphi_i)\right). \tag{11}$$

SV (Support Vector) denotes the support vector and φ_r means positive support vector, φ_s means negative support vector.

The coefficient can be obtained by the following quadratic programming:

$$\begin{aligned} \max \quad & w(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j K\left(x_i, x_j\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_i y_i = 0 \\ & 0 \le \alpha_i \le C \quad (i = 1, 2, \ldots, M), \end{aligned} \tag{12}$$

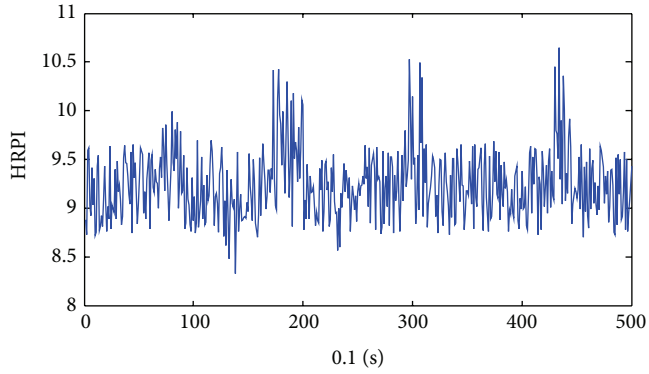
where C is the parameter to price the misclassification. Before the SVM can classify traffics, it should undergo a training process to develop a classification model. We use the LibSVM library [17] to implement SVM.

4. Experiments

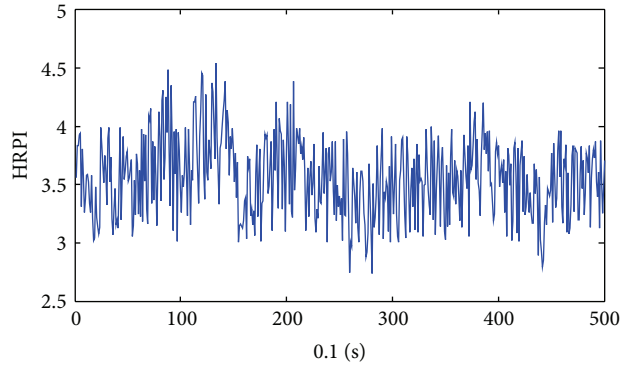
In order to evaluate the performance of our scheme, we divided our study into two groups of experiments: to detect application layer DDoS in normal traffic and in flash crowd.

4.1. Dataset. Normal traffic is the real-life Internet traces collected from the traffic archive of Changzhou university WWW server. The traces contain two weeks worth of all HTTP requests to the web server. We implemented application-layer DDoS attack in a simulator. Simulations are carried out using NS-2 network simulator on Linux platform. For generating attack traffic, there are 50 zombie machines and a web server. Attack rates are 20 HTTP Get requests/s, 30 HTTP Get requests/s, . . . , 60 HTTP Get requests/s, which simulate the attack rates of worm "Mydoom", and every attack lasts 1800s. Flash crowd is collected from the World Cup 98 website [18]. As this is a high arrival rate, we expect our approach to detect this traffic as flash crowd.

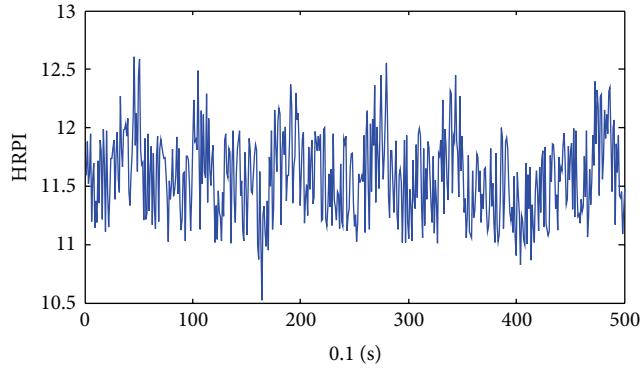
We obtained HRPI time series by multiple sampling and calculation when the sampling interval Δt is 0.1 s. As shown in Figure 1(a), HRPI of normal traffic varies with the time and its mathematical expectation is 9.26. Figure 1(b) shows HRPI of DDoS attack and its mathematical expectation is 3.58, and HRPI of flash crowd is shown in Figure 1(c) with mathematical expectation 11.57. We can see that the HRPI time series are sensitive to DDoS attack and flash crowd, so HRPI can distinguish three types of traffic distinctly.



(a) HRPI of normal traffic



(b) HRPI of DDoS attack



(c) HRPI of flash crowd

Figure 1: HRPI of different web traffics.

4.2. Model Parameters. There are three parameters which may affect the HRPI time series performances. The first one is the parameter p of AAR model. In practice, the model degree is often fixed using some prior knowledge or guidelines. To optimize the goodness of fit verse, model complexity ratio, and also to ease the computational load, we settled $p = 3$ as a degree which allowed the model to capture sufficiently well the normal traffic behavior.

The second one is state noise covariance C_w ($C_w = \sigma_w^2 I$). The adaptation speed of the Kalman filter is determined by the state noise covariance factor σ_w^2. It controls how fast the state adopts the changes in observations and gives a suitable balance in adapting to normal behavior and avoiding incorporating anomalous behavior in to the model. We experimented with different values and chose to use $\sigma_w^2 = 0.0001$.

The third one is the lag of Kalman filter, and we noticed a significant increase in model accuracy when $L = 1$. The increase in accuracy was slower when further increasing L. As larger L means also longer delay in detection, we chose to use $L = 1$.

4.3. Experiments and Results

4.3.1. Evaluation Criteria. In this paper, a group of performance metrics in classification problems are used for the evaluation of the results, consisting of FPR, FNR, accuracy,

Table 1: The results of DDoS detection in normal traffic.

	Accuracy	FPR	FNR	Precision	Recall	ROC
P-20	91.52%	8.24%	7.27%	91.62%	92.51%	90.12%
P-30	94.26%	5.05%	3.94%	94.51%	95.28%	95.31%
P-40	96.31%	3.74%	3.08%	96.79%	97.00%	98.79%
P-50	97.24%	2.76%	2.23%	97.30%	97.64%	99.10%
P-60	97.81%	2.05%	1.88%	97.94%	98.02%	99.26%

Table 2: The results of DDoS detection in flash crowd.

	Accuracy	FPR	FNR	Precision	Recall	ROC
T-20	92.33%	6.69%	6.03%	92.66%	93.28%	91.67%
T-30	95.02%	4.10%	3.09%	95.64%	96.37%	96.34%
T-40	96.39%	3.52%	2.58%	96.82%	96.95%	98.99%
T-50	97.68%	2.16%	1.94%	97.22%	98.03%	99.61%
T-60	98.06%	1.47%	1.02%	98.29%	98.49%	99.70%

precision, recall, and ROC. Let TP represent the normal test samples that have been correctly classified and let FP represent the ones that have been wrongly classified. Let TN represent the attacking test samples that have been correctly classified and let FN represent the ones that have been falsely classified. Thus, the False-Positive Rate (FPR) and the False-Negative Rate (FNR) are the proportions of wrongly classified normal test samples and attacking test samples, respectively (FPR = FP/(FP + TN), FNR = FN/(TP + FN)). Accuracy states the overall percentage of correctly classified attacking test samples (accuracy = (TP + TN)/(TP + FP + TN + FN)). Precision as the classifier's safety, states the degree in which messages identified as attacking test samples are indeed malicious (precision = TP/(TP + FP)). Recall as the classifier's effectiveness, states the percentage of attacking test samples that the classifier manages to classify correctly (recall = TP/(TP + FN)). Receiver Operating Characteristic (ROC) as a classifier's balance ability between its FPR and its FNR is a function of varying classification threshold.

4.3.2. Experiment 1: Detect DDoS Attacks in Normal Traffic. We set that the sampling interval Δt is 0.1 s, HRPI time series length N is 100, so the detection time is 10 s. In this experiment, normal traffic contain 600 series, and DDoS attack traffic contain 450 series. Obtained dataset is divided into two parts: training data contains 60% of total data values, testing data contains the rest of the obtained dataset. The kernel function in SVM classifier is radial basis function (RBF) and the robustness of the classifiers is evaluated using 10-fold cross-validation. In order to test the robustness of our method to the disturbance of normal traffic, we do five experiments named as P-20, P-30,. . ., P-60, in which traffic attacks are 20 HTTP Get requests/s,. . ., 60 HTTP Get requests/s mixing normal traffic at the same time.

Table 1 shows the performance results; the detection ratio of our approach increases when the attack traffic volume increases. When normal traffic is much larger than attack traffic, the detection ratio still keeps a high level. This means that our approach can identify the DDoS attack traffic with a high precision, and be sensitive to DDoS attack traffic.

4.3.3. Experiment 2: Detect DDoS Attacks in Flash Crowd. In this experiment, the sampling interval Δt is 0.1 s, and HRPI time series length N is 100, too. We sample flash crowd 500 series, and DDoS attack traffic 350 series. The training and testing method of SVM is the same as experiment 1. We do five experiments named as T-20, T-30, . . . , T-60, by mixing attacking traffic 20 HTTP Get requests/s, . . . , 60 HTTP Get requests/s and flash crowd.

Table 2 shows the performance results, with the increment of flash crowd, the detection ratio of our approach does not decline rapidly. The FPR and FNR are reduced with the increase of attack rate, and the accuracy, precision, recall, and ROC are ascended with the increase of attack rate.

In the above two groups of experiments, the false negatives come mainly from two aspects: firstly, due to the increase of normal traffic or flash crowd, which makes the HRPI states learn to normal ones, thus making the difference too small for detection. Secondly, the network state shift caused by network random noise results in false negative.

5. Conclusion

Application-layer DDoS attacks detection is a hot and difficult research topic in the field of intrusion detection. Based on the characteristics of DDoS attack, this paper proposes a novel approach to detect DDoS attacks. The work provides two contributions: (1) HRPI is introduced to detect DDoS attacks, and it reflects the essential features of attacks and (2) a detection scheme against DDoS attacks is proposed, and it can achieve high detection efficiency and flexibility.

In our future work, we will make a detailed study of how to set all kinds of parameters in different application scenarios adaptively.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Contact (61070121).

References

[1] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM '11)*, pp. 952–957, April 2011.

[2] G. Oikonomou and J. Mirkovic, "Modeling human behavior for defense against flash-crowd attacks," in *Proceedings of the IEEE International Conference on Communications (ICC '09)*, pp. 1–6, June 2009.

[3] H. Beitollahi and G. Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," *Computer Communications*, vol. 35, pp. 1312–1332, 2012.

[4] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-resilient scheduling to counter application layer attacks under

imperfect detection," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–13, April 2006.

[5] W. Yen and M.-F. Lee, "Defending application DDoS with constraint random request attacks," in *Proceedings of the Asia-Pacific Conference on Communications*, pp. 620–624, Perth, Australia, October 2005.

[6] L. Von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the ACM*, vol. 47, no. 2, pp. 56–60, 2004.

[7] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 54–65, 2009.

[8] Y. Xie, S. Tang, and X. Huang, "Detecting latent attack behavior from aggregated Web traffic," *Computer Communications*, no. 5, pp. 895–907, 2013.

[9] J. Yu, C. Fang, L. Lu et al., "A lightweight mechanism to mitigate application layer DDoS attacks," *Scalable Information Systems*, vol. 18, pp. 175–191, 2009.

[10] P. Du and A. Nakao, "OverCourt: DDoS mitigation through credit-based traffic segregation and path migration," *Computer Communications*, vol. 33, no. 18, pp. 2164–2175, 2010.

[11] H. Beitollahi and G. Deconinck, "Tackling Application-layer DDoS Attacks," *Procedia Computer Science*, vol. 10, pp. 432–441, 2012.

[12] Q.-D. Sun, D.-Y. Zhang, and P. Gao, "Detecting distributed denial of service attacks based on time series analysis," *Chinese Journal of Computers*, vol. 28, no. 5, pp. 767–773, 2005.

[13] R. Yan, Q. Zheng, and H. Li, "Combining adaptive filtering and IF flows to detect DDOS attacks within a router," *KSII Transactions on Internet and Information Systems*, vol. 4, no. 3, pp. 428–451, 2010.

[14] S. Wen, W. Jia, W. Zhou, W. Zhou, and C. Xu, "CALD: Surviving various application-layer DDoS attacks that mimic flash crowd," in *Proceedings of the 4th International Conference on Network and System Security (NSS '10)*, pp. 247–254, Victoria, Australia, September 2010.

[15] S. Haykln, *Adaptive Filter Theory*, Prentice-Hall, Upper saddle River, NJ, USA, 3rd edition, 1996.

[16] J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, and M. Tarvainen, "Processing intrusion detection alert aggregates with time series modeling," *Information Fusion*, vol. 10, no. 4, pp. 312–324, 2009.

[17] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft Research, 1998.

[18] M. Arlitt and T. Jin, "1998 World Cup Web Site Access Logs," 1998, http://ita.ee.lbl.gov/html/contrib/WorldCup.html.