Future Generation Computer Systems 38 (2014) 36–46

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Detection and defense of application-layer DDoS attacks in backbone web traffic

Wei Zhou[a,*], Weijia Jia[b], Sheng Wen[c], Yang Xiang[c], Wanlei Zhou[c]

[a] *School of Information Science and Engineering, Central South University, Changsha, 410083, PR China*
[b] *Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong*
[c] *School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia*

HIGHLIGHTS

- We propose a novel real-time method to detect application-layer DDoS attacks.
- We integrate the detection algorithm into a DDoS attack defense architecture.
- Our proposed architecture can be deployed in backbone networks.
- We carry out the experiments using real-world backbone traffic from Sina.

ARTICLE INFO

Article history:
Received 19 January 2012
Received in revised form
20 January 2013
Accepted 2 August 2013
Available online 11 August 2013

Keywords:
Model mining
DDoS attack
Backbone

ABSTRACT

Web servers are usually located in a well-organized data center where these servers connect with the outside Internet directly through backbones. Meanwhile, the application-layer distributed denials of service (AL-DDoS) attacks are critical threats to the Internet, particularly to those business web servers. Currently, there are some methods designed to handle the AL-DDoS attacks, but most of them cannot be used in heavy backbones. In this paper, we propose a new method to detect AL-DDoS attacks. Our work distinguishes itself from previous methods by considering AL-DDoS attack detection in heavy backbone traffic. Besides, the detection of AL-DDoS attacks is easily misled by flash crowd traffic. In order to overcome this problem, our proposed method constructs a Real-time Frequency Vector (RFV) and real-timely characterizes the traffic as a set of models. By examining the entropy of AL-DDoS attacks and flash crowds, these models can be used to recognize the real AL-DDoS attacks. We integrate the above detection principles into a modularized defense architecture, which consists of a head-end sensor, a detection module and a traffic filter. With a swift AL-DDoS detection speed, the filter is capable of letting the legitimate requests through but the attack traffic is stopped. In the experiment, we adopt certain episodes of real traffic from Sina and Taobao to evaluate our AL-DDoS detection method and architecture. Compared with previous methods, the results show that our approach is very effective in defending AL-DDoS attacks at backbones.

1. Introduction

DDoS attack is a continuous and critical threat to the Internet. AL-DDoS attack [1], as a major form of current DDoS attacks, utilizes legitimate HTTP requests to overwhelm victim resources. Because an AL-DDoS attack often mimics or occurs in the flash crowd events of popular websites, it becomes more difficult to be addressed. According to the security report from Arbor Networks [2], there has been a sharp escalation in DDoS attacks launched against popular Internet services in recent years.

1.1. Motivation

AL-DDoS attack detection and malicious traffic filtering techniques have long been important but difficult problems to be addressed. Popular web servers are usually the preferable targets for attackers to launch AL-DDoS attacks. In order to protect web servers, researchers have proposed lots of methods to detect AL-DDoS attacks. However, most of them have not met the requirements of detection in the heavy traffic environment. For example, Yi Xie et al. adopted a hidden semi-Markov process to present the behavior of Internet users [1,3]. The hidden semi-Markov approach is a complex algorithm. When users visit a website, it traces and records the whole visiting history of each user. According to our observation on two popular websites (Sina: www.sina.com

* Corresponding author. Tel.: +61 0452224989.
E-mail address: weizhou09@gmail.com (W. Zhou).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.08.002

and Taobao: www.taobao.com), the number of source IP addresses may reach a peak of 10^4 requests per second. It is noticeable that the hidden semi-Markov method is unlikely to perform effectively in backbone traffic. Another typical approach against AL-DDoS attacks is to use CAPTCHA [4,5]. This method requires users to recognize strings in a fuzzy picture and submit a response to a web server for authentication. However, users sometimes consider this operation as a negative experience to surf the Internet. Paper [6] introduced wavelets to identify anomalies in network traffic. But wavelet analysis is generally a post-mortem analysis and cannot be used for online processing. Paper [7] proposed a mechanism to voluntarily increase the bandwidth utilization of legitimate users. However, this approach cannot reduce the network congestion and the load of web servers. Paper [8] proposed a countermeasure that consisted of a suspicion assignment process and a DDoS-resilient scheduler. The suspicion process assigns a continuous 'valued vs. binary' measure onto each client session. It further utilizes these values to determine if and when to schedule the requests of a session. However, this approach is still too time-consuming to detect AL-DDoS attacks in large volume traffic.

In this paper, we are motivated to design a defense system at the backbone level. This system is able to detect AL-DDoS attacks targeting Internet web servers. Currently, most of these web servers are deployed together in a data center connecting directly to the backbones. Thus, it is critical to implement an effective method to detect AL-DDoS attacks and filter the malicious traffic in backbones before they causes detriments to the web servers. The proposed system has low complexity and can real-timely run in high volume traffic.

1.2. Contribution

The works and primary contributions of this paper are summarized as follows:

- We design an algorithm for mining web traffic models on backbones. The models are used to distinguish the traffic of AL-DDoS attacks from normal heavy traffic and flash crowds. (See Section 3.) We evaluate our approach in both the simulated and real environment. The results show that it is quite effective and accurate. (See Section 5.2.)
- The model mining algorithm is integrated into a modularized defense system. (See Section 4.) The architecture of this defense system has low complexity. The experiments show that the proposed system is quite efficient, and thus suits the backbone environment. (See Section 5.1.)
- We further compare our proposed algorithm and system with previous works. The analysis results show that our method and system exclusively satisfy the requirements of the heavy traffic environment, and thus outperform previous works. (See Section 5.3.)

The rest of this paper is organized as follows. We discuss the related work in Section 2. The details of the proposed algorithm are presented in Section 3. Then, we talk about the defense architecture in Section 4. Experiments and further discussion are presented in Sections 5 and 6 respectively, followed by conclusions in Section 7. Because this paper has lots of abbreviations and variables, for the convenience of readers, we list all of them in Table 1.

2. Related works

In the last decade, researchers have already done lots of work on how to detect various DDoS attacks. Readers can find some related works about how to counter traditional DDoS attacks in papers [9,10]. In this section, we mainly focus on the defense of AL-DDoS attacks.

Table 1
Major notations used in this paper.

Symbol	Explanation
RFV	Real-time frequency vector.
MCP	Model checking period.
MST	Model similarity threshold.
SSP	Self-similarity period.
SSM	Special sequence matrix.
favg	The average frequency for each website URL.
V	The model set.
x_t	The observed number of HTTP packets received.
y_t	The estimated number of HTTP packets received.
d_t	The derivation between the values of y_t and x_t.
sIP	The source IP address.
E_n	The entropy of received traffic.
CP	The collision possibility in bloom filter.
ATDM	The abnormal traffic detection module.
DADM	AL-DDoS attack detection module.
FM	Filter module in the system architecture.

First of all, Y. Xie et al. applied a hidden semi-Markov model (HsMM) to capture the web behavior of users [1,3,11,12]. However, according to our observation to the portal website 'Sina', the number of source IP addresses may reach a peak of 10^4 requests per second in the backbone environment. Thus, HsMM is unlikely to run under such real conditions. G. Oikonomou and J. Mirkovic [13] proposed to assign each webpage with a weight and then build a history graph for the visits of users. This task can be easily realized through the web-crawling techniques [14]. However, when we inspect a famous commercial website 'TaoBao', we find this method become ineffective as there are too many dynamical web pages. Another typical approach is to use 'CAPTCHA' [4,5]. Users are asked to recognize a puzzle figure and the result is then submitted for further authentication. In practice, users may be annoyed and stop to continue this operation. S. Yu et al. adopted the flow correlation algorithm to discriminate suspicious flows [15]. This method requires a packet counting and recording process to be applied on a group of routers. In fact, the routers working on the backbones are too busy to fulfill this task.

There are also some other methods used to defend AL-DDoS attacks. Paper [7,16] proposed to encourage legitimate users voluntarily increasing bandwidth utilization. However, this approach cannot reduce the network congestion and the load of web servers. Jaeyeon et al. [17] characterized AL-DDoS attacks and flash crowds respectively. The results were then used to distinguish abnormal flows from Internet traffic. However, current sophisticated attackers are able to avoid the detection by changing the attack scheme. D. Das et al. [18] proposed an approach which consists of three phases to handle various HTTP flooding scenarios. Paper [8] also introduced a suspicion assignment mechanism and an AL-DDoS resilient scheduler to detect AL-DDoS attacks. According to their experiments, the performance of victims is improved from 40 to 1.5 s. In fact, these methods only release the symptom of being attacked. S. Byers et al. [19] discussed the conceptual AL-DDoS attacks. T. Yatagai et al. [20] introduced the browsing orders and time to detect AL-DDoS attacks. These two methods are too resource-consuming [21]. Khattab et al. [22] leveraged the 'group-testing' theory to detect AL-DDoS attacks. Srivatsa et al. [23] adopted the admission control to limit the number of concurrent clients in online services. However, these methods are challenges to the performance of web servers, which may become the new targets of AL-DDoS attacks. P. Barford et al. [6] introduced wavelets to identify the anomalies in traffic. The wavelets are post-mortem analysis and may not suit online application. G. Yan et al. [24] proposed a non-standard game-theoretic framework that facilitated evaluation of DDoS attacks and defense, but this method is designed for enterprise-scale deployment rather than portal web servers deployed in data centers.

3. Detection prototype

3.1. Real-time processing

In order to real-timely process the traffic, we create a real-time frequency vector ($RFV = \langle favg^1, favg^2, \ldots, favg^m \rangle$), wherein m is the number of resources in the website, such as web pages and images. Each item in RFV denotes the average frequency of one resource being visited. Its value is calculated by Eq. (1) as in

$$\begin{cases} favg_n^k = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n}\left(\dfrac{1}{te_i - te_{i-1}}\right) \\ favg_{n+1}^k = \dfrac{1}{n+1}\left(favg_n^k \times n + \dfrac{1}{te_{n+1} - te_n}\right) \end{cases} \quad (1)$$

wherein we have $1 \le k \le m$. In Eq. (1), the average frequency for each piece of resource $favg_n^k$ is defined as the average reciprocal difference between every pair of successive arrival times te. The parameter n depends on the visitation frequency. The complexity of the algorithm is $O(C)$, where C is a constant. Additionally, the spatial complexity of the algorithm is $2m$ with two parallel vectors recording current $favg_n^k$ and the last arrival time te.

We can dynamically describe the visiting status of web users by using Eq. (1). However, in practice, we still have two problems to be solved:

How to localize items in RFV? The complexity of locating a specific item in a vector is $O(n)$, where n indicates the length of the vector. In backbones, there are usually tens of thousands of 'Get' requests. Therefore, this localization method may not meet the demands of real-time processing. In our implementation, we adopt the Perl language to realize the algorithm. It has strong support for Hash tables and can help us achieve real-time traffic processing. The unique identifier for a resource is its URL. A URL generally has long string length and thus cannot be stored in a Hash table. Accordingly, we use the md5 technique (32 bits) to store the URL as a key in a Hash table.

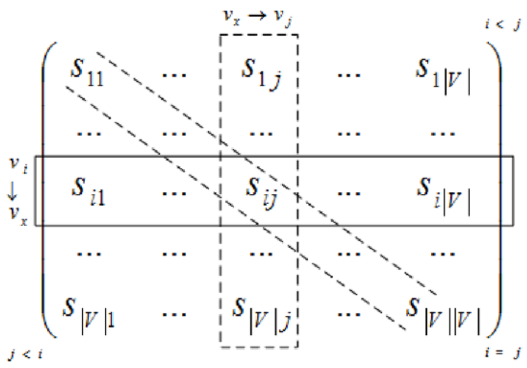
How to deal with the length of RFV? In a large-scale website, such as Sina, the number of resources is massive. We have done statistics on the traffic of Sina. Because a large number of resource will be visited, the length of RFV is quite large. Thus, the real-time processing is almost impossible. In fact, previous research analyzed the real-world traffic traces and concluded that less than 10% of the most popular web pages accounted for more than 90% of requests [17]. In order to mine models from the visitation history of users, the algorithm does not require the accuracy of unit digit level. Therefore, we are able to dramatically decrease the length of RFV by neglecting the records which have small values of average frequency. For example, for a website with 20 000 resources, we can abridge the length of RFV to be less than 2000.

3.2. Traffic models

To derive traffic models from web traffic, we build a model set $V = \{v_1, \ldots, v_p\}$ including the traffic models (*tModel*) and the models between models (*bModel*).

How to get tModels? There is a consensus that the Internet traffic follows the self-similarity distribution [25]. Through our observation on the Sina traffic, we found that it has a period of one day. In this paper, we set the Model Checking Period (MCP) to be one hour and choose the RFV as a *tModel* for the traffic of one hour.

The initial state of V is $\varnothing$. After we add the first RFV into the model set, every successive RFV in a MCP should be checked to determine whether the current RFV is a new *tModel*. We calculate the Pearson correlation coefficient between the current RFV (v) and



$v_x \to v_j$
$i < j$
$v_i \downarrow v_x$
$j < i$
$i = j$
s_{11} … s_{1j} … $s_{1|V|}$
s_{i1} … s_{ij} … $s_{i|V|}$
$s_{|V|1}$ … $s_{|V|j}$ … $s_{|V||V|}$

Fig. 1. The structure of SSM.

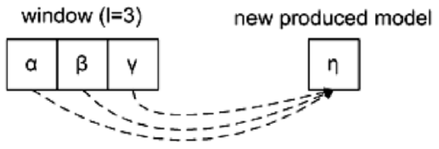


Fig. 2. The structure of window $w(l)$.

tModels ($\{v_i \mid 1 \le i \le p\}$) in the model set, as in

$$r = |\cos\theta| = \left|\frac{v \times v_i}{\|v\| \times \|v_i\|}\right| \quad (2)$$

wherein the parameter θ is the included angle. The value of the correlation coefficient r is in the range between zero and one. When r is close to one, the two vectors are more correlated and they may belong to the same model. Otherwise, when r is close to zero, the current RFV can be deemed as a new model in the set. We set the Model Similarity Threshold (MST) to make the decision.

For the web traffic which follows the self-similarity distribution, V tends to become stable after models mining for a period of time. This indicates the duration has passed the Self-Similarity Period (SSP). For example, the SSP of Sina is set to be 24 h and the MCP to be one hour. Then, V becomes stable when $p \le 24$. We can also adjust the MCP and use a smaller duration on the models. The RFV is the approximation of a real-time model in the web visitation traffic. However, the network traffic has noticeable jitter and it is almost infeasible to present the model. Actually, we can change the MST to adjust the results of the model matching processes.

How to get bModels? We can construct real-time models by the RFV. However, these *tModels* are not sufficient for the detection. In fact, the visitation of users is an important temporal property in the web security. Different *tModels* exist for different temporal patterns. Therefore, we construct a Special Sequence Matrix (SSM) in Fig. 1, and based on it, we introduce another type of models: *bModels*.

bModels are produced dynamically and cause the diameter of the matrix growing dynamically as well. Thus, the SSM is a dynamic spanning matrix. In Fig. 1, the event of v_i emerging before v_j is denoted by $v_i \to v_j$, and we have the following properties: (1) items in the diagonal line of the SSM denotes the number of instances for each *tModel*; (2) each item in the up-triangle of the SSM indicates the number of instances $v_x \to v_j$ after v_j being produced ($x \in (1, i)$); and (3) each item in the down-triangle of the SSM indicates the number of instances $v_i \to v_x$ after v_i being produced ($x \in (1, i)$).

For each $v_i \to v_j$, we set a window $w(l)$, wherein l is the length of the window. As shown in Fig. 2, for any sequences inside the window, we define a *bModel* $v_i \to v_j$, irrespective of whether there are other models between them. Thus, in Fig. 2, we obtain

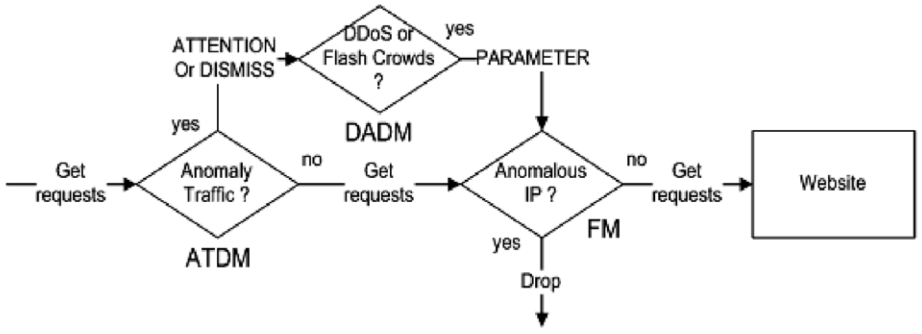


Fig. 3. The system architecture and work flow.

the sequential relations $\alpha \rightarrow \eta, \beta \rightarrow \eta, \gamma \rightarrow \eta$. We also have three instances of *bModels*. For each *bModel*, we record the number of their appearances in the SSM and push each new *bModel* into the window. Therefore, *bModels* can be dynamically obtained.

How to evaluate each tModel and bModel produced in our system? Firstly, by using the SSM, we are able to compute the possibility of each *tModel* by Eq. (3) as in

$$p(v_i) = \frac{\sum_{i=1}^{|V|} s_{ij}}{\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} s_{ij}}. \quad (3)$$

For each *bModel* $v_i \rightarrow v_j$, its support can be calculated by Eq. (4), which denotes the possibility of $v_i \rightarrow v_j$ having the same sequences originating from v_i:

$$p(v_i \rightarrow v_j \mid v_i) = \frac{s_{ij}}{\sum_{j=1}^{|V|} s_{ij}}. \quad (4)$$

Moreover, the confidence of the *bModel* can be calculated by Eq. (5). This denotes the possibility of the sequence $v_i \rightarrow v_j$:

$$p(v_i \rightarrow v_j) = \frac{s_{ij}}{\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} s_{ij}}. \quad (5)$$

3.3. For high detection accuracy and low false positive

The problem of flash crowds is a daunting challenge. This is because the traffic of flash crowds appears similar to a DDoS attack. In order to counter this problem, we implement another RFV for the inbound source IP addresses. Actually, attackers usually deploy a large, distributed botnet to overwhelm the victim. According to previous works [7,13,26,27], most botnets have less than 10,000 zombies. Conversely, a flash crowd is generally originated from tens of thousands of web users requiring the same business-critical content. Therefore, its traffic sources are far more dispersed compared with a real AL-DDoS attack. When the system detects a suspicious occurrence of an AL-DDoS attack, it calculates the entropy of the source IP addresses (sIP) as in Eq. (6). The result depicts the concentration degree of the captured traffic. In fact, the entropy of flash crowds is generally larger than that of a real AL-DDoS attack:

$$En = \sum_{i=1}^{m} P(\mathrm{sIP}_i)\log(P(\mathrm{sIP}_i)) = \sum_{i=1}^{m} favg_n^i \log(favg_n^i). \quad (6)$$

Moreover, attackers often carry out moderate AL-DDoS attacks to avoid detection. As paper [7] stated, the protected service needs enough bandwidth to handle the stream of incoming requests. The web servers can easily satisfy the requirement via providing another access link with better bandwidth. Our system directly connects to the backbone outside the data center, which has significant bandwidth. Therefore, when the moderate AL-DDoS attacks reach the web servers discussed in this paper, it can be expected that the detriments will not be too critical to the performance of the web servers. As suggested by [28], we simply neglect this kind of moderate AL-DDoS attacks.

4. System architecture

We further integrate previous detection algorithms into a modularized AL-DDoS attack defense architecture. It consists of three modules: Abnormal Traffic Detection Module, DDoS Attack Detection Module and Filter Module. The system architecture and the work flows are shown in Fig. 3.

4.1. Abnormal traffic detection module (ATDM)

The ATDM is a real-time analyzer of time series deployed as a head-end sensor. In the real world, regular traffic comes from the legitimate systems. Thus, we only need to observe the abrupt changes in the traffic of HTTP 'Get' requests. Once an anomalous feature is observed, a specific signal of 'ATTENTION' is sent to the DDoS attack detection module for further analysis. The result suggests if this feature indicates a flash crowd or an AL-DDoS attack. There are lots of effective and efficient methods to analyze the heavy traffic in backbones [29,30]. In this paper, we predict the rate of the 'Get' requests and adopt the simple AR model. Besides, in order to decrease the noise in the web traffic, we use the dynamical Kalman filter to calibrate the prediction result. The details of the ATDM are as follows:

We monitor the stream of the HTTP 'Get' requests. Traffic intensity is investigated at constant time intervals, which form a time series denoted by $\{y_1, y_2, \ldots, y_t\}$. In this paper, we measure the traffic intensity by the total number of packages received in a time interval. In fact, the traffic intensity can be predicted by previous observations and any significant changes in it indicate a potential AL-DDoS attack or flash crowd. Therefore, at time t, the significant difference between the observation x_t and the estimated value y_t indicates that current traffic is abnormal. The AR model for predicting the current traffic intensity is shown in

$$y_t = \sum_{k=1}^{p} a_t^k x_{t-k} + e_t \quad (7)$$

wherein y_t is the prediction of x_t. The stationary model parameter is a_t^k and the observation error is e_t. We simply assign a_t^k to be 2^{-k} and the sum of a_t^k to be 1. The model degree p strongly depends on

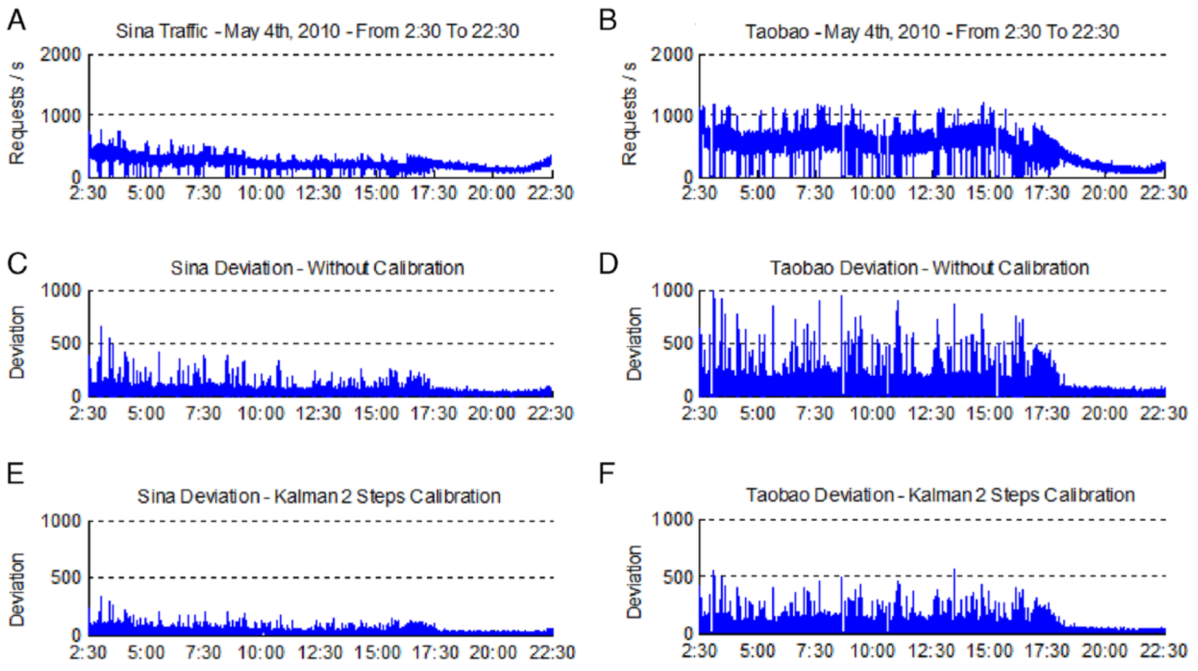


Fig. 4. An example of real web traffic, original derivations and 2-step Kalman calibration.

the properties of the protected websites. We will discuss it in the experiment.

The sensor counts the HTTP ‘Get’ requests every second. Based on previous p records, the AR model calculates the prediction y_t for the next step. The value of y_t needs to be calibrated because of intense fluctuation in the network traffic. We take Fig. 4 as an example. It describes an episode of HTTP ‘Get’ requests from Sina and Taobao on April 10th 2010. According to our investigation, the traffic in Fig. 4(A) and (B) can be considered as normal traffic. We use d_t to denote the deviation between the predicted value y_t and the observation x_t. As shown in Fig. 4(C) and (D) respectively, the deviation d_t has large fluctuation which may significantly affect the detection of abnormal events. In this paper, we introduce the Kalman filter to calibrate the value of y_t. The Kalman filter is an adaptive data processing algorithm which suits the on-line estimation. Fig. 4(E) and (F) show the results of 2-step Kalman calibration. Noticeably, the fluctuation in original deviations has been calibrated and smoothed.

Besides, the Kalman filter is a recursive algorithm which may cause some delay. We adopt the N-step recursion. Therefore, the total complexity of the abnormal traffic detection is $N \times p$. We use observations from x_t to x_{t+N} to estimate the improved prediction. The delay will increase when N becomes larger and the accuracy inversely decreases. Actually, it is necessary to find a trade-off between the prediction accuracy and the delay. We will discuss the parameter N in the experiments.

After the Kalman filter smooths the deviation d_t, the ATDM will report an abnormal event when d_t exceeds a presumed threshold. We calculate the variance of d_t as in

$$\delta_d^2 = \frac{1}{p} \sum_{i=t-p}^{t} \left(d_t - E(d_{t-p}^t)\right)^2 . \tag{8}$$

We consider the threshold of the ATDM as

$$d_t > k\delta_d^2. \tag{9}$$

Although the ATDM is able to detect abnormal traffic in the backbone, we cannot ascertain it as an AL-DDoS. We need to distinguish it from flash crowd events in the module DADM. As is shown in Fig. 3, this function is achieved by the ATDM sending an ‘ATTENTION’ signal to the DADM. This means the DADM does not work all the time in the whole procedure. It only runs when the abnormal traffic arrives. Correspondingly, when the ATDM finds the traffic has changed back to the normal status, it will send a ‘DISMISS’ signal to the DADM and stop continuing the processing.

4.2. AL-DDoS attack detection module (DADM)

Our work is concerned with four types of abnormal traffic: (1) ‘Repeated Request AL-DDoS Attack’: This type of attack mainly focuses on the homepage or a hot webpage, so the targets of the traffic converge to one or two points. We also suppose the attack is launched by a certain quantity of bots, like MyDoom [31] and Code Red [32]. Thus, the sources of the traffic will converge to a group of points. (2) ‘Recursive Request AL-DDoS Attack’: This type of attack performs as a ‘Website Spider’ [33]. Its traffic is scattered onto different web pages. As a result, the sources of the traffic converge to a group of points but the targets of the traffic become dispersed. (3) ‘Repeated Workload AL-DDoS Attack’: This type of attack is able to use less bots but causes even greater damage to the website. However, the traffic will be similar to the case of ‘Repeated Request AL-DDoS’. A group of bots continually send requests of a large image or database searching operations. (4) ‘Flash Crowds’: A flash crowd describes an event of an unexpected visiting surge to a website. It results from the announcement of a new service or free software download. The sources of flash crowds are definitely scattered because they are the HTTP ‘Get’ requests originated from legitimate users. Conversely, the targets of flash crowds converge to one or two points because the visitors are mainly interested in one webpage or resource.

When an ‘ATTENTION’ signal arrives from the head-end sensor, the DADM is activated. The detection method discussed in Section 3 is embedded in this module. It traces each incoming source IP address and the required webpage, and then records the average frequencies in the related RFV. The detailed work flow is shown in Fig. 5. In the heavy traffic, the average frequency can be considered as the possibility of each required webpage. Actually, the processing of the RFV is quite efficient for a backbone environment

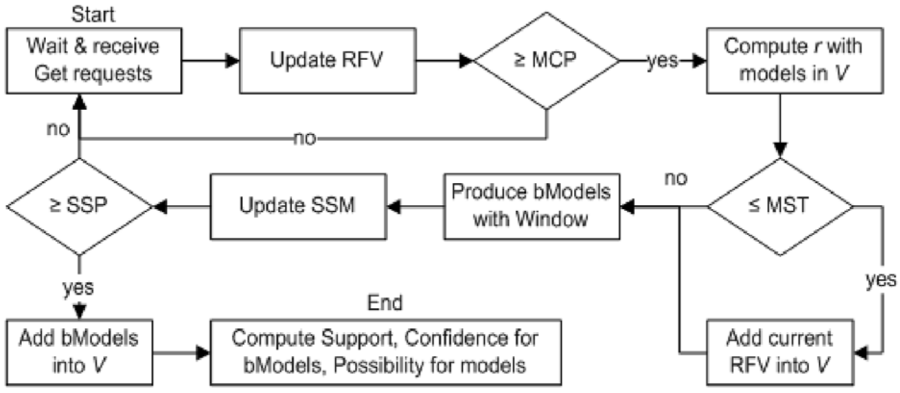

Start
Wait & receive Get requests
Update RFV
≥ MCP
yes
Compute r with models in V
no
no
≤ MST
no
Produce bModels with Window
Update SSM
≥ SSP
yes
yes
Add bModels into V
End
Compute Support, Confidence for bModels, Possibility for models
Add current RFV into V

Fig. 5. The detailed workflow of DADM.

and produces a series of traffic models. For *tModels*, we compute their probabilities by Eq. (3). For *bModels*, we are able to calculate their support and confidence values by Eqs. (4) and (5), and use (confidence-support)/2 instead of the probability. The DADM compares the current model with the models of normal traffic in the model set, if the probability of the current model is smaller than a presumed threshold, this abnormal traffic will be considered as an AL-DDoS attack or a flash crowd.

Moreover, in order to distinguish the AL-DDoS attack and the flash crowd for each abnormal traffic, the values of entropy on each *tModel* or *bModel* are calculated to describe the distribution of the incoming sources and the target URLs. For the convenience of analysis, we name S as the RFV of source IP addresses, T as the URLs of required web pages, numeral one as the 'repeated request AL-DDoS attack', numeral two as the 'recursive request AL-DDoS attack', numeral three as the 'repeated workload AL-DDoS attack' and numeral four as the flash crowds. Based on the previous definitions of each type of AL-DDoS attacks and flash crowds, we have

$$\frac{En(S)_2}{En(T)_2} > \frac{En(S)_1}{En(T)_1} > \frac{En(S)_3}{En(T)_3} > \frac{En(S)_4}{En(T)_4} \tag{10}$$

wherein the entropy $En(S)$ or $En(T)$ is calculated by Eq. (6). Theoretically, as shown in Eq. (10), flash crowds generally have the smallest ratio of entropy values. Thus, we are able to distinguish the flash crowds from various AL-DDoS attacks.

4.3. Filter module (FM)

As shown in Fig. 3, when the abnormal traffic is proven to be an AL-DDoS attack, the DADM will send the malicious source IP addresses as the parameters to the FM, and then stop the AL-DDoS attack. Current AL-DDoS attacks are generally launched by more than 20 000 compromised computers [9]. Therefore, it is a tough task to block such a large list of IP addresses. The simply searching method will definitely decrease the efficiency of the whole system. In order to filter most malicious traffic, we can apply filtering technologies in network routers [34,35], but the routers need to look for the prefix of an IP address based on its mask code, while the FM in our system can cover the whole IP address. In this paper, we adopt the Bloom Filter [36]. An empty Bloom Filter is a bit array of m bits. All items are set to be 0. There are also k different hash functions, each of which maps or hashes some elements to one of the m positions in an array with a uniform random distribution. In our system, the length of the bit array m is 2^{20} and two hash functions are implemented inside ($k = 2$). Suppose the IP addresses are described by the dotted decimal notation '$A.B.C.D$', then the hash functions are:

$$(A^3 + B^3 + C^3 + D^3) \bmod 2^{20} \tag{11}$$

$$(A \times B \times C \times D) \bmod 2^{20}. \tag{12}$$

In order to add an element, we need to pass the IP address to each of the two hash functions and get two array positions. Set the bits at all these positions to 1. To query for an element (test whether the IP address is in the set), we also pass it to each of the two hash functions to get two positions. If any of the bits at these positions is equal to 0, the IP address does not exist in the set. As is shown in our experiment, it is capable of limiting the conflict below 16×10^{-4}. If all the hash functions return to 1, either the IP address is in the set, or the bits have been set to 1 by inserting other IP addresses. The latter situation will be considered as a collision. We set the filter has two hash functions and the length of the hash table is 2^{20}. The estimated quantity of malicious IP addresses is 20,000. Then, the collision possibility (CP) can be calculated as in

$$\begin{aligned} \mathrm{CP} &= \lim_{m \gg n} \left[1 - \left(1 - \frac{n}{m}\right)^k\right]^k = \lim_{m \gg n} \left(1 - e^{-\frac{kn}{m}}\right)^k \\ &< \left(-\frac{kn}{m}\right)^k < \left(\frac{2 \times 2 \times 10^4}{10^6}\right)^2 = \frac{16}{10^4}. \end{aligned} \tag{13}$$

The result means that the number of 2^{20} bits occupies 128 kB memory for two hash tables. This memory cost is trivial for most current machines.

5. Experiment

We evaluate the AL-DDoS detection method and the system architecture in two aspects: firstly, we use the discrete random series to simulate the traffic and validate our system using the simulated data. Secondly, we use the real-world Sina web traffic from a backbone network.

5.1. Evaluate real-time processing on backbone traffic

We performed an evaluation of our system using the real-world traffic. The traffic data came from the backbone of China Telecom in Southwest China with a bandwidth of 40 Gb/s. We used an 8 Gb/s TAP for collecting the traffic. The sample rate is $\frac{1}{5}$. We show the Sina traffic statistics from 3rd May to 7th May in Fig. 6. During this period of time, we can see three suspicious periods indicating possible attacks.

We replay the real traffic data to the proposed system of this paper. After we train the model set of the DADM using the traffic collected during 4th and 5th May, we mainly get two models in the model set V. We show the results in Table 2. In this experiment, the MST is set to be 0.8. The first model represents the traffic from 0 am to 5 pm on 4th May, followed by 0 am to 4 pm on 5th May. The second model represents the traffic from 6 pm to 11 pm on 4th May and 5 pm to 11 pm on the following day. Table 3 shows the SSM

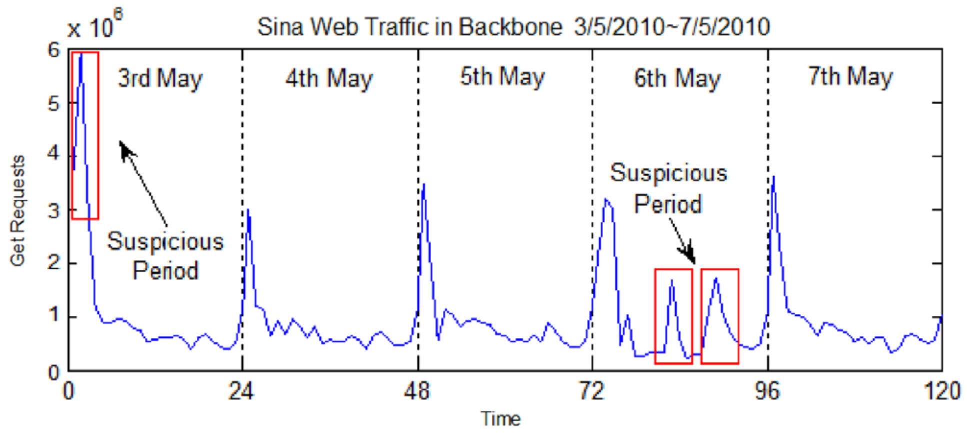


Fig. 6. Sina traffic of 5 days by hour.

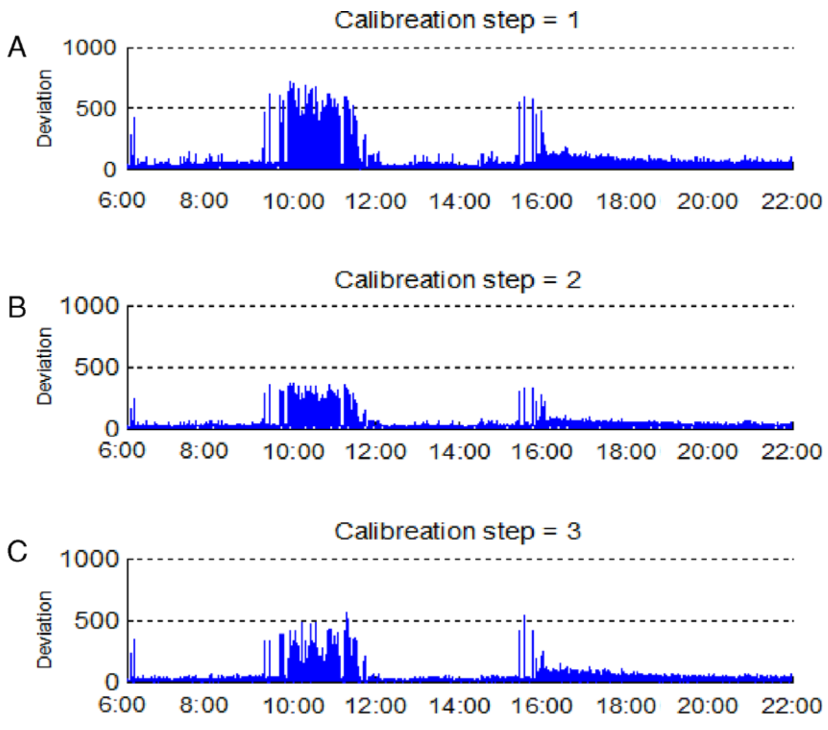


Fig. 7. The deviations when Kalman step is set to be 1–3 (6th May).

and the related bModels. In this experiment, there is one bModel 'M1 $\rightarrow$ M2', which is indicative that the M1 has an exclusive possibility for it to precede M2 in the sequence. Therefore, the problem becomes whether those three suspicious periods of traffic can be captured by the ATDM and then identified as an AL-DDoS attacks rather than flash crowds in the DADM.

We replay the traffic of 6th May to our system. We record the process of detecting abnormal traffic in the ATDM. As shown in Fig. 7, in order to choose an optimized value of the Kalman calibration step, we evaluate this parameter from 1 to 3 on the traffic of 6th May. Firstly, the experiment results show that it is able to detect noticeable deviations with all the settings when the abnormal traffic on 6th May appears. Secondly, the deviations have been better restrained and smoothed by setting k to be 2. Later, we will set k to be 2 in the experiment.

We also investigate the DADM by using the traffic of 6th May. As shown in Table 2, we find these three suspicious attacks are highly correlated to two models in the model set. Since we consider the models from the normal traffic on 4th and 5th May, we conclude that the two suspicious periods of traffic on 6th May are not AL-DDoS attack traffic. They are occasional high volume normal traffic.

5.2. *Evaluate detection accuracy and false positive*

Previous experiments have proven our system is able to distinguish between normal traffic and abnormal traffic. In this section, we will focus on evaluating the detection accuracy of AL-DDoS attacks and flash crowds in the DADM. The experiments use both the simulated method and the real-data method.

5.2.1. *Simulated method*

We assume that less than 10% of most popular web pages account for more than 90% of the requests. In this paper, we introduce Gaussian discrete time series [17] to represent this character. An alphabetic set $A = \{A\text{–}Z, a\text{–}z\}$ is used for simulating HTTP web traffic. In Fig. 8, the 52 letters are numbered from 0 to 51. With different average values (MU) and variances (SIGMA), the whole procedure forms 5 models, each of which delegates one special case

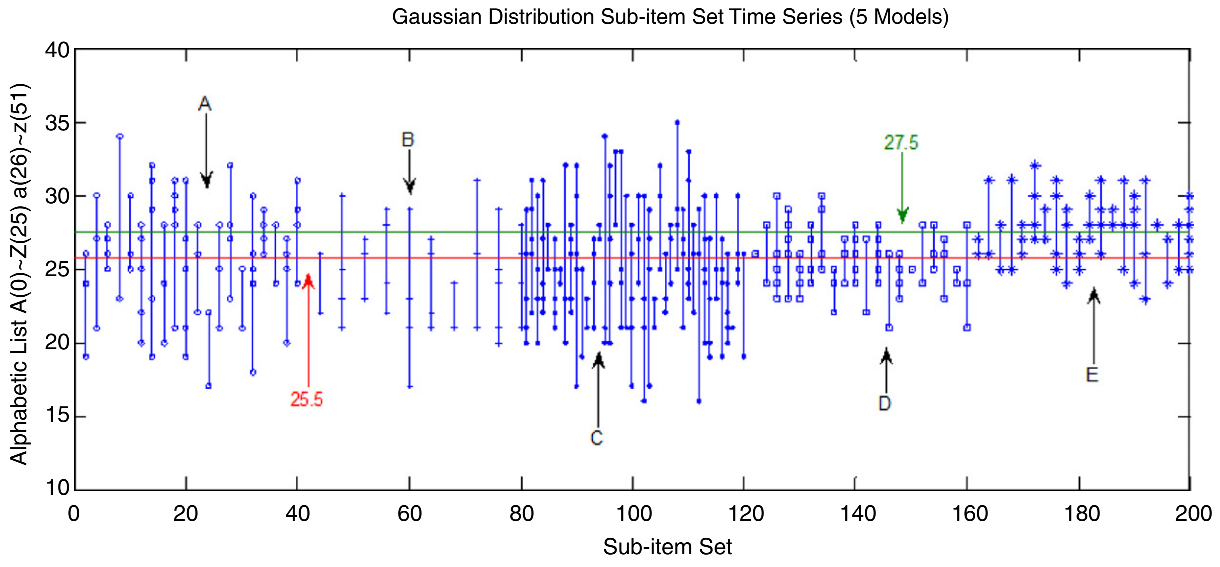


Fig. 8. The setting of parameters in simulated environment. Each sub-item set in this figure denotes an abnormal traffic event.

Table 2
Correlation result (M: Model, H: Hour, MST = 0.8).

M	H	4th May		5th May		6th May	
	1	1		0.9077	0.6690	0.9009	0.6430
	2	0.9511		0.8729	0.5353	0.8707	0.5302
	3	0.9113		0.8511	0.5504	0.8339	0.5269
	4	0.8799		0.8549	0.5649	0.8212	0.5058
	5	0.8755		0.8599	0.6075	0.8306	0.5692
	6	0.8576		0.8493	0.6275	0.8278	0.5947
	7	0.8664		0.8600	0.6368	0.8325	0.5967
	8	0.8792		0.8618	0.6166	0.8369	0.5717
1	9	0.8733		0.8597	0.5987	0.8395	0.5765
	10	0.8725		0.8630	0.6088	0.8450	0.5986
	11	0.8896		0.8599	0.7071	0.8547	0.6968
	12	0.8819		0.8636	0.7083	0.8511	0.6963
	13	0.8788		0.8629	0.6665	0.8493	0.6475
	14	0.8635		0.8629	0.6486	0.8283	0.5859
	15	0.8657		0.8556	0.6427	0.8327	0.6072
	16	0.8767		0.8535	0.7723	0.8447	0.7505
	17	0.8012		0.7981	0.9206	0.7884	0.9052
	18	0.6798	1	0.6666	0.9706	0.6556	0.9682
	19	0.5977	0.9830	0.6077	0.9702	0.5871	0.9679
	20	0.5509	0.9564	0.5706	0.9618	0.5484	0.9589
2	21	0.5396	0.9477	0.5631	0.9639	0.5381	0.9541
	22	0.5349	0.9269	0.5698	0.9662	0.5366	0.9509
	23	0.5913	0.9372	0.6039	0.9663	0.5804	0.9475
	24	0.7826	0.9296	0.7973	0.9255	0.7324	0.9458

Table 3
Special sequence matrix (M: MODEL).

	M1	M2		Support	Confidence	Possibility
M1	17	1	M1 → M2	0.0556	0.04	0.68
M2	0	7	M2 → M1	0	0	0.28

Table 4
Model meaning in simulated experiment.

Model	MU	SIGMA	DENSITY	Meaning
A	25.5	4	2	General web traffic
B	25.5	4	4	Traffic at deep night
C	25.5	4	1	Heavy traffic
D	25.5	2	2	Possible DDoS attack
E	27.5	2	2	Possible DDoS attack

Table 5
The number of models in the simulated traffic (the values in the first row: MST, the values in the first column: MCP.)

	0.95	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.55	0.50
10	95	67	40	28	12	7	3	4	2	1
20	42	27	12	3	2	1	1	1	1	1
30	29	18	8	5	4	2	1	1	1	1
40	21	13	6	3	1	1	1	1	1	1
50	18	9	5	1	1	1	1	1	1	1
60	15	6	4	1	2	1	1	1	1	1
70	11	9	3	2	1	1	1	1	1	1
80	11	8	3	2	1	1	1	1	1	1
90	9	7	5	2	1	1	1	1	1	1
100	9	7	3	2	1	1	1	1	1	1

Table 6
Correlation results from model A to model E.

Model	A	B	C	D	E
A	1	0	0	0	0
B	0.8469	1	0	0	0
C	0.7887	0.8607	1	0	0
C	0.6407	0.7016	0.5987	1	0
C	0.6392	0.6797	0.6776	0.6608	1

observed in web traffic. We list their meanings in Table 4 and implement this part of the evaluation in Matlab7.14. The 'DENSITY' describes the arrival rate of HTTP sessions. We assume models D and E are the potential AL-DDoS attacks because HTTP sessions are more integrated in these two models which may be caused by AL-DDoS attacks or flash crowds. In order to choose the best MCP and MST, we test a range of MCPs and MSTs for each model. As an example, Table 5 shows the result of model A's traffic. We choose the best MCP and MST by identifying the largest 1-Rectangular. In Table 6, we set MCP to be 40 (using the number of program loops as MCP) and MST to be 0.7. After the analysis of each model, we set MCP to be 40 and MST to be 0.725 (average value) for the simulated evaluation.

Table 6 shows the correlation results with 5 models. We can see that models A, B and C are highly correlated even though their density deviates from each other. This proves the normal traffic can be determined from one identical model. Conversely, models D and E have lower correlation to normal traffic models. It is evident that we can use correlation to separate the two classes.

5.2.2. *Real-data method*

We also insert an artificial traffic period as flash crowds into the traffic of 4th May and an artificial traffic period as an AL-DDoS attack into the traffic of 7th May. Both additions are made at 12:00 am for a period of 10 min. Fig. 9 describes the traffic by minutes. The AL-DDoS attack is launched by a scale of 2000 nodes. Flash

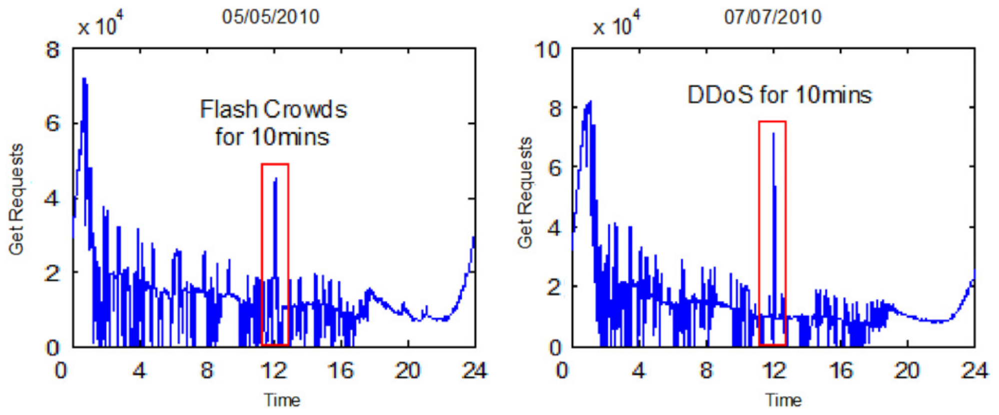


Fig. 9. Sina traffic of 2 days by minute (5th and 7th May).

Table 7
Detection of flash crowds and AL-DDOS attack.

	Model 1	Model 2	Source IP entropy
Flash crowds	0.0445	0.0344	3 008 975.8508
AL-DDoS	0.0419	0.0351	1 452 942.2055

Table 8
Comparison with previous work (low, medium, high, unknown).

	Efficiency	Accuracy	Scalability	Complexity
HsMM [1,3]	Low	High	Low	High
Human Behavior [13]	Low	High	Low	High
Botz-4-Sale [5]	Medium	Unknown	Medium	Medium
Traffic Analysis [17]	High	Medium	High	Low
Wavelet [6,30]	Low	Medium	Low	Medium
Covariance [33]	High	Medium	Low	Medium
Our System	High	High	High	Medium

crowds is launched by a scale of 20 000 nodes. The entire sources' IP addresses are random.

Table 7 indicates that they are easily detected by our system. Moreover, through calculating the entropy of the source IP addresses, we are able to classify them into different traffic. As is discussed in Section 3.3, the entropy of a flash crowd generally has larger value than that of an AL-DDoS attack. The problem is how to set the thresholds to distinguish them. In our system, according to our experience, we set it to be 2×10^6, and this threshold works well till now.

5.3. Comparison with other AL-DDoS detection methods

The detection of the AL-DDoS attack is a 'yes' or 'no' question. Lots of methods in previous works [1,3,5,6,13,30,33] can also achieve good performance on detecting AL-DDoS attacks. In this paper, we mainly focus on proposing an AL-DDoS detection system which is capable of being deployed in the traffic of the backbone. Therefore, we compare our AL-DDoS detection method and system architecture with previous works in four key aspects: efficiency, accuracy, scalability and complexity.

Because the backbone carries heavy traffic, it requires the proposed system has great efficiency. Besides, modern AL-DDoS attacks are similar to flash crowds and are usually concealed in the heavy backbone traffic. Therefore, in order to reduce the false positive, the proposed system also needs to identify normal traffic, AL-DDoS attacks and flash crowds. Some of the previous works only suit a small or medium scaled environment rather than on a backbone, so we also evaluate the scalability of our system. Besides, several previous methods have good performance on detection, but it is hard for them to be deployed in a backbone environment. This concerns the complexity of the system. We show the comparison results in Table 8.

The first work in the comparison is the hidden semi-Markov method introduced in [1,3]. This method records each visiting sequence of users and is able to accurately detect AL-DDoS attacks and flash crowds. However, due to the complexity of their algorithms, we consider the efficiency and the scalability are 'low'. The second work [13] is to model human behavior to defend against AL-DDoS attacks. This method also needs to record the visiting sequence of users on the website. For most popular business websites, since the number of web pages is considerable, this method is almost impossible to be applied. The Botz-4-Sale [5] needs the collaboration of CAPTCHA. The cost in CPU processing time leads to Botz-4-Sale only being deployed in medium-scaled websites. Paper [17] characterizes abnormal traffic to detect AL-DDoS attacks. This traditional method can be easily deployed in large-scaled websites. However, modern AL-DDoS attacks are quite stealthy and the characters summarized in this paper may not attain a good accuracy. Wavelet methods proposed in [6,30] are generally post-mortem methods which are not suited for real-time AL-DDoS detection in the backbone traffic. Paper [33] adopts a covariance-matrix for the detection of AL-DDOS attacks, but implementing such a matrix for each user in a popular website has a high cost. Our proposed system outperforms previous works because of the simple and efficient algorithm used for AL-DDoS detection. According to the previous experiments, it achieves good performance on distinguishing between AL-DDoS attacks and flash crowds. Moreover, the architecture is organized by modules. Therefore, the system is isolated from web servers and the detection will not affect the performance of them.

6. Discussion and open issues

A few limitations and open issues are worth discussing. Firstly, because the real traffic in backbone changes rapidly, the AR model may be rough to present the HTTP Get traffic. We adopt it in our system for three reasons: (1) in this paper, the AR model is not used for accurate presentation of traffic. What we focus on is to capture significant deviations in this traffic. (2) By using the Kalman filter online, the noise of rapid changes can be mitigated and the prediction accuracy will be improved. (3) The AR model is simple and can work efficiently.

Secondly, sophisticated attackers may increase the AL-DDoS traffic slowly, so that they can avoid detection by our system. In order to overcome this limitation, we have set a threshold inside the ATDM. When the real-time traffic rises beyond this threshold, even if the ATDM does not detect any abnormal traffic, it will also send out an 'ATTENTION' signal and activate the DADM to start the AL-DDoS detection.

Thirdly, the proposed method and system involve many variables and parameters. Some of them are determined by experience.

For example, we set the parameter of the AR model p to be 5 in the experiments. However, there is nothing special about 5. We use 5 instead of other values because it is neither too large nor too small. Actually, we can train these unknown variables and parameters and find their optimized values in real traffic.

Fourthly, because the compromised zombies may be recovered to be legitimate clients, we need to reset bloom filters in the FM after AL-DDoS attacks. We can schedule this process by either resetting all entries simultaneously or removing some entries at a specific rate. We will implement this process in our future work.

7. Conclusion

This paper proposes a method of model mining on heavy backbone traffic to detect application-layer DDoS attacks. We also propose a defense architecture including a novel detection method in the related module. Evaluation using simulation and real data demonstrates that our algorithm is effective. When AL-DDoS attacks are detected, we can identify the malicious sources and use an efficient filter to stop the abnormal traffic. An open issue is how to improve the reaction rate. We will investigate and implement this problem in the future.

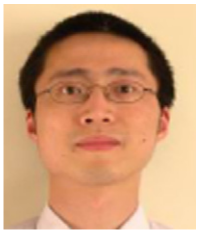
References

[1] Y. Xie, S. Zheng Yu, Monitoring the application-layer ddos attacks for popular websites, IEEE/ACM Trans. Netw. 17 (1) (2009) 15–25.
[2] Arbor. Networks, Worldwide network infrastructure security report, Tech. Rep., Arbor Networks, 2011.
[3] Y. Xie, S. Zheng Yu, A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors, IEEE/ACM Trans. Netw. 17 (1) (2009) 54–65.
[4] L. von Ahn, M. Blum, N.J. Hopper, J. Langford, Captcha: using hard ai problems for security, in: EUROCRYPT, 2003, pp. 294–311.
[5] S. Kandula, D. Katabi, M. Jacob, A. Berger, Botz-4-sale: surviving organized ddos attacks that mimic flash crowds, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation, NSDI'05, USENIX Association, Berkeley, CA, USA, 2005, pp. 287–300.
[6] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, IMW '02, ACM, New York, NY, USA, 2002, pp. 71–82.
[7] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, S. Shenker, Ddos defense by offense, ACM Trans. Comput. Syst. 28 (1) (2010) 1–54.
[8] S. Ranjan, R. Swaminathan, M. Uysal, E. Knightly, Ddos-resilient scheduling to counter application layer attacks under imperfect detection, in: Proceedings. INFOCOM 2006. 25th IEEE International Conference on Computer Communications, 2006, pp. 1–13.
[9] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the dos and ddos problems, ACM Comput. Surv. 39 (1) (2007).
[10] G. Thatte, U. Mitra, J. Heidemann, Parametric methods for anomaly detection in aggregate traffic, IEEE/ACM Trans. Netw. 19 (2) (2011) 512–525.
[11] Y. Xie, S. Tang, Y. Xiang, J. Hu, Resisting web proxy-based http attacks by temporal and spatial locality behavior, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1401–1410.
[12] Y. Xie, J. Hu, S. Tang, X. Huang, A structural approach for modelling the hierarchical dynamic process of web workload in a large-scale campus network, J. Netw. Comput. Appl. 35 (6) (2012) 2081–2091.
[13] G. Oikonomou, J. Mirkovic, Modeling human behavior for defense against flash-crowd attacks, in: IEEE International Conference on Communications, 2009, pp. 1–6.
[14] B. Krishnamurthy, J. Wang, On network-aware clustering of web clients, in: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM'00, ACM, New York, NY, USA, 2000, pp. 97–110.
[15] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, F. Tang, Discriminating ddos attacks from flash crowds using flow correlation coefficient, IEEE Trans. Parallel Distrib. Syst. 23 (6) (2012) 1073–1080.
[16] S. Khanna, S.S. Venkatesh, O. Fatemieh, F. Khan, C.A. Gunter, Adaptive selective verification: an efficient adaptive countermeasure to thwart dos attacks, IEEE/ACM Trans. Netw. 20 (3) (2012) 715–728.
[17] J. Jung, B. Krishnamurthy, M. Rabinovich, Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites, in: Proceedings of the 11th International Conference on World Wide Web, WWW'02, ACM, New York, NY, USA, 2002, pp. 293–304.
[18] D. Das, U. Sharma, D.K. Bhattacharyya, Detection of http flooding attacks in multiple scenarios, in: Proceedings of the 2011 International Conference on Communication, Computing and Security, ICCCS'11, ACM, New York, NY, USA, 2011, pp. 517–522.
[19] S. Byers, A.D. Rubin, D. Kormann, Defending against an internet-based attack on the physical world, ACM Trans. Internet Technol. 4 (3) (2004) 239–254.
[20] T. Yatagai, T. Isohara, I. Sasase, Detection of http-get flood attack based on analysis of page access behavior, in: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2007, pp. 232–235.
[21] J. Yu, Z. Li, H. Chen, X. Chen, A detection and offense mechanism to defend against application layer DDoS attacks, in: Proceedings of the Third International Conference on Networking and Services, ICNS'07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 54–66.
[22] S. Khattab, S. Gobriel, R. Melhem, D. Mosse, Live baiting for service-level DoS attackers, in: INFOCOM 2008. The 27th Conference on Computer Communications, IEEE, 2008, pp. 171–175.
[23] M. Srivatsa, A. Iyengar, J. Yin, L. Liu, Mitigating application-level denial of service attacks on web servers: a client-transparent approach, ACM Trans. Web 2 (3) (2008) 1–49.
[24] G. Yan, R. Lee, A. Kent, D. Wolpert, Towards a Bayesian network game framework for evaluating ddos attacks and defense, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS'12, ACM, 2012, pp. 553–566.
[25] M.E. Crovella, A. Bestavros, Self-similarity in world wide web traffic: evidence and possible causes, IEEE/ACM Trans. Netw. 5 (6) (1997) 835–846.
[26] M. Handley, Internet architecture wg: DoS-resistant internet subgroup report, Tech. Rep., University College London, 2007.
[27] H. Project, Know your enemy: tracking botnets, Published on the Web, URL: http://www.honeynet.org/papers/bots/.
[28] A. Sharad, D. Travis, T. Christos, Ddos mitigation via regional cleaning centers, Tech. Rep., Sprint Labs, 2004.
[29] J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, M. Tarvainen, Processing intrusion detection alert aggregates with time series modeling, Information Fusion 10 (4) (2009) 312–324.
[30] W. Lu, A.A. Ghorbani, Network anomaly detection based on wavelet analysis, EURASIP J. Adv. Signal Process. 4 (2009) 1–16.
[31] CERT, Incident note in-2004-01 w32/novarg.a virus, Tech. Rep., CERT, 2004.
[32] CERT, Code red worm crashes iis 4.0 servers with url redirection enabled, Tech. Rep., CERT, 2001.
[33] D.S. Yeung, S. Jin, X. Wang, Covariance-matrix modeling and detecting various flooding attacks, IEEE Trans. Syst. Man. Cybern. Part A: Systems and Humans 37 (2) (2007) 157–169.
[34] I. Ioannidis, A. Grama, M. Atallah, Adaptive data structures for ip lookups, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, Vol. 1, 2003, pp. 75–84.
[35] M. Mitzenmacher, A. Broder, Using multiple hash functions to improve ip lookups, in: IN Proceedings of IEEE INFOCOM, IEEE, 2000, pp. 1454–1463.
[36] A. Broder, M. Mitzenmacher, Network applications of bloom filters: A survey, in: Internet Math., 2002, pp. 636–646.

Wei Zhou received the B.Eng. and M.Eng. degrees from Central South University, Changsha, China, in 2005 and 2008, respectively, all in computer science. She is now a Ph.D. candidate in the School of Information Science and Engineering, Central South University and a visitor to the School of Information Technology, Deakin University. Her research interests include distributed systems, computer networks and network security.

Weijia Jia received the B.Sc. and M.Sc. degrees from Central South University, Changsha, China, in 1982 and 1984, respectively, and the Master of Applied Sci. and Ph.D. degrees from the Polytechnic Faculty of Mons, Mons, Belgium, in 1992 and 1993, respectively, all in computer science. He is currently a Full Professor with the Department of Computer Science and the Director of the Future Networking Center, ShenZhen Research Institute, City University of Hong Kong, Kowloon, Hong Kong. His research interests include next-generation wireless communication, protocols and heterogeneous networks, distributed systems, and multicast and anycast quality-of-service routing protocols.

Sheng Wen received the B.Eng. degree from Lanzhou Jiaotong University, Lanzhou, China, in 2003 in computer science. He is now a Ph.D. candidate in the School of Information Science and Engineering, Central South University and a joint training Ph.D. candidate in the School of Information Technology, Deakin University. His research interests include information fusion, computer networks and cyber security.



Yang Xiang received his Ph.D. in computer science from Deakin University, Australia. He is currently with the School of Information Technology, Deakin University. His research interests include network and system security, distributed systems, and networking. In particular, he is currently leading a research group developing active defense systems against large-scale distributed network attacks. He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 100 research papers in many international journals and conferences, such as IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Information Security and Forensics, and IEEE Journal on Selected Areas in Communications. He has published two books, Software Similarity and Classification (Springer) and Dynamic and Advanced Data Mining for Progressing Technological Development (IGI-Global). He has served as the Program/General Chair for many international conferences such as ICA3PP 12/11, IEEE/IFIP EUC 11, IEEE TrustCom 11, IEEE HPCC 10/09, IEEE ICPADS 08, NSS 11/10/09/08/07. He has been the PC member for more than 50 international conferences in distributed systems, networking, and security. He serves as the Associate Editor of IEEE Transactions on Parallel and Distributed Systems and the Editor of the Journal of Network and Computer Applications. He is a member of the IEEE.

Wanlei Zhou received the B.Eng. and M.Eng. degrees from Harbin Institute of Technology, China, in 1982 and 1984, respectively; the Ph.D. degree from the Australian National University, Canberra, in 1991; and the D.Sc. degree from Deakin University, Victoria, Australia, in 2002. He is currently the chair professor of information technology and the head of the School of Information Technology, Faculty of Science and Technology, Deakin University, Melbourne, Australia. His research interests include distributed and parallel systems, network security, mobile computing, bioinformatics, and e-learning. He has published more than 200 papers in refereed international journals and refereed international conferences proceedings. Since 1997, he has been involved in more than 50 international conferences as general chair, steering committee chair, PC chair, session chair, publication chair, and PC member. He is a senior member of the IEEE.