ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/224195023

CALD: Surviving Various Application-Layer DDoS Attacks That Mimic Flash Crowd

Conference Paper · October 2010

DOI: 10.1109/NSS.2010.69 · Source: IEEE Xplore

CITATIONS
28

READS
175

5 authors, including:

Sheng Wen
Deakin University
127 PUBLICATIONS **4,153** CITATIONS
SEE PROFILE

Wei Zhou
UNSW@Canberra
8 PUBLICATIONS **247** CITATIONS
SEE PROFILE

Wanlei Zhou
Deakin University
531 PUBLICATIONS **11,149** CITATIONS
SEE PROFILE

Chuan Xu
Chongqing University of Posts and Telecommunications
58 PUBLICATIONS **576** CITATIONS
SEE PROFILE

All content following this page was uploaded by Chuan Xu on 28 July 2016.

The user has requested enhancement of the downloaded file.

Deakin Research Online

Deakin University's institutional research repository

This is the published version (version of record) of:

Wen, Sheng, Jia, Weijia, Zhou, Wei, Zhou, Wanlei and Xu, Chuan 2010, CALD : surviving various application-layer DDoS attacks that mimic flash crowd, *in NSS 2010 : Proceedings of the 4th International Conference on Network and System Security*, IEEE, Piscataway, N.J., pp. 247-254.

http://hdl.handle.net/10536/DRO/DU:30033643

©20**[10]** IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright : 2010, by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved

CALD: Surviving Various Application-Layer DDoS Attacks That Mimic Flash Crowd

Sheng Wen, Weijia Jia, Wei Zhou
Central South University
Changsha, Hunan, China, 410083
swen@deakin.edu.au

Wanlei Zhou
Deakin University
Melbourne, Australia, 3125
wanlei@deakin.edu.au

Chuan Xu
Chongqing University
Chongqing, Sichuan, China, 40044
chuanxu@gmail.com

***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. In this paper, we present the design and implementation of CALD, an architectural extension to protect Web servers against various DDoS attacks that masquerade as flash crowds. CALD provides real-time detection using mess tests but is different from other systems that use resembling methods. First, CALD uses a front-end sensor to monitor the traffic that may contain various DDoS attacks or flash crowds. Intense pulse in the traffic means possible existence of anomalies because this is the basic property of DDoS attacks and flash crowds. Once abnormal traffic is identified, the sensor sends ATTENTION signal to activate the attack detection module. Second, CALD dynamically records the average frequency of each source IP and check the total mess extent. Theoretically, the mess extent of DDoS attacks is larger than the one of flash crowds. Thus, with some parameters from the attack detection module, the filter is capable of letting the legitimate requests through but the attack traffic stopped. Third, CALD may divide the security modules away from the Web servers. As a result, it keeps maximum performance on the kernel web services, regardless of the harassment from DDoS. In the experiments, the records from www.sina.com and www.taobao.com have proved the value of CALD.**

Keywords-DDoS; Application-layer; Kalman Filter; Information Theory

1. INTRODUCTION *(HEADING 1)*

For many years, distributed denial of service (DDoS) attack has caused severe damage to victims and still constitutes one of the major threats in current internet. A popular form of DDoS today is the application-layer floods that overwhelm the Web server with a large number of GET requests. To circumvent detection, the attackers increasingly move away from pure bandwidth floods to stealthy DDoS attacks that masquerade as flash crowds. The successful cases in the early history included MyDoom [1], Code Red [2] and FBI case involving DDoS-for-hire [3]. In recent years, we frequently heard the news and complaint about application-layer DDoS harassment [4-7]. In fact, the situation is much worse than we can expect because the botnet is booming. The China CCTV program 'Economy Half Hour' broadcast that the botnet has formed an industry chain and the capital concerned went beyond 10 billion Chinese Yuan in 2009 [8]. Although it is not just DDoS attacks that are associated with botnet, current DDoS attacks are mainly launched by it. For an investigation into DDoS crime, the BBC program 'Click' brought a medium sized Website and demonstrated only 60 broadband connections were enough to make this Website unusable [9]. The experts in the TV show also said that the high-traffic sites were potential victims for application-layer DDoS attacks. The criminals got into contact with the Websites and threatened them in DDoS attacks. All kinds of high-traffic Websites that generate lots of revenue relied on the Websites to be online, so a lot of the Websites paid up to avoid the DDoS attacks.

Countering application-layer DDoS attack becomes a great challenge because they can be mounted with legitimate requests from legitimate connected machines. The requests originating from the compromised computer are indistinguishable from the requests generated by legitimate users. In fact, they differ from the legitimate ones only in intent but not in content. The malicious requests arrive from a large number of geographically distributed machines; thus they cannot be filtered on the IP prefix. Also, many Websites do not use passwords or login information, and even when they do, passwords could be easily stolen from the hard disk of a compromised machine. Further, checking the site-specific password requires establishing a connection and allowing unauthenticated clients to access socket buffers and worker processes, making it easy to mount an attack on the authentication mechanism itself. Moreover, the method of using computational puzzles, which requires the clients to recognize a figure and submit the result, is not suitable sometimes because its demand of artificial engagement annoys people. Finally, in contrast to bandwidth attacks [11], it is difficult to detect big resources consumers when the attack targets higher-layer bottlenecks such as CPU, database and disk because commodity operating systems do not support fine-grained resource monitoring. Further, an attacker can resort to mutating attacks which cycle between different bottlenecks.

This paper proposes CALD, an architectural extension to protect Web servers against application-layer DDoS attacks that masquerade as flash crowds. It is targeted towards large-scale online businesses as well as non-commercial portal Website. CALD combines three functions: abnormal traffic detection, DDoS attack detection, filter.

a) Abnormal traffic detection: The abnormal traffic detection is a real-time time series analyser. This function is deployed in the front-end sensor. According to our idea, the underlying assumption is that regular traffic behaviour is related to normal system use, or at least that only changes in any regular behaviour are interesting. Thus we aim to detect any abrupt change in the HTTP Get request traffic. The basic idea of this function is that once the modelling is done, the output of the model corresponds to normal system

978-0-7695-4159-4/10 $26.00 © 2010 IEEE
DOI 10.1109/NSS.2010.69

IEEE computer society

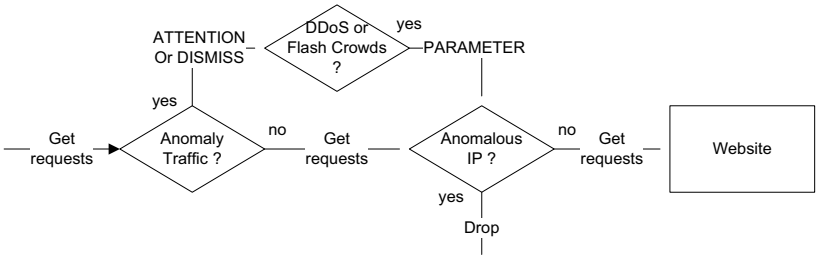

Figure 1: CALD Overview.
Note that DDoS Attack Detection component is only activated when front-end sensor reports ATTENTION signal.

behaviour, and the difference between observed behaviour and model output gives us the anomalous signature. We report this signature as a signal to DDoS attack detection component and diversify whether flash crowd or DDoS really happens. We have seen lots of successful application of such idea on network traffic analysis [12, 13]. In this paper, we drew AR model to describe and predict the Get request rate, and introduce the dynamical Kalman filter to calibrate the prediction result because of intensely fluctuation. In current status of our research, we only considered stationary AR model which could be extended to non-stationary process i.e. the even using Kalman filter for estimating time-dependent AR coefficients.

b) DDoS attack detection: when receives ATTENTION signal from front-end sensor, the DDoS attack detection component is activated. It traces each incoming source IP address as well as each visiting Webpage, and records the average frequencies in a vector. In an intense traffic, the average frequency can be considered as the possibility of each source IP address. Based on the vector, this component calculates the entropy. The entropy called mess extent describes the distribution of incoming sources and target WebPages. We define the mess extent of source IP addresses as A, the mess extent of target WebPages as B and the rate between A and B as R. Theoretically, the variant R in flash crowd is smaller than the one in the application-layer DDoS attack. Thus, we are able to set several thresholds and pick out the anomalous source IP addresses.

c) Filter: when the abnormal traffic is justified as containing DDoS attack, the DDoS attack detection component sends the anomalous source IP addresses to filter so that it can release the flooding. Current application-layer DDoS attack is generally launched by more than 20,000 compromised computers [14]. Implementing such a large list of IP addresses for block is not very easy. Simply searching method will definitely decrease the efficiency of the whole system. In this paper, we adopted Bloom Filter [15], where we implemented two hash functions inside and was capable of limiting the collision below sixteen ten-thousandth.

Fig.1 summarizes CALD. It has a few important characteristics. Firstly, CALD addresses various application-layer DDoS attack. Prior work usually has a bias against the Get request flooding the homepage. As will be described in section 2, CALD is concerned with three types of DDoS attacks; the latter is stealthier than the former. They are all easily achieved in the Internet and CALD can be capable of dealing with them. Next, the design of CALD draws essential properties against the application-layer DDoS attack. In the front-end sensor, the essential property is the large quantity of Get request from the attackers. In the DDoS detection model, the essential property is the difference in the rate of mess extents. No matter which type of DDoS is launched by the attackers, CALD is able to trace the attack. At last, CALD is simple. The simplicity in the algorithm increases the efficiency which is particular important in famous Website. CALD also divides the security modules away from the Web servers. As a result, it keeps maximum performance on the kernel web services, regardless of the harassment from DDoS. Even the update and deployment become more convenient.

The rest of the paper is organized as follows. We start by describing the three types of threat models in Section 2. Section 3 describes the details of CALD. More specially, here we analyse the essential properties of the application-layer DDoS attacks and flash crowds. Section 4 elaborates the experiments. The last two sections are concerned with some discussion and related work about countering application-layer DDoS attacks.

2. THREAT MODEL

This paper mainly concerns with three types of application-layer DDoS, of which each one is stealthier than the former. (1) The attackers organize lots of compromised computer bombing the homepage of the Website, which we call as 'repeated request DDoS'. MyDoom [1] and Code Red [2] all belong to the DDoS attack of this kind. (2) The attackers pick several premium pages out into a list and stochastically choose one as the target for each sending of the HTTP Get requests. Even this type of attack can perform like 'Website Spider' [10], which in an automated manner and orderly fashion browses the pages of the target Website. We have not seen any news or reports talking about such case happened in the wild as an application-layer DDoS attack. However, it is quite easy to be achieved and definitely hard to be tracked because the abnormal traffic is leveraged into a group of targets and acts more like a legitimate

visiting. We call it 'recursive request DDoS'. (3) There may exist some large files like image or download resources in the Website. There may also contain some operations that are concerned with heavy workload like database search. The attacker sends Get request to aim at these resources. This 'repeated workload DDoS' can be invoked at a lower request rate, thereby requiring less work from the attacker and making detection increasingly difficult.

3. THE DESIGN OF CALD

According to our idea, a front-end sensor is used to detect the abrupt change in the traffic. If abnormal traffic exists, the DDoS attack detection component will be activated and make advanced inspection for a decision. The DDoS traffic from the malicious IP addresses will be ultimately blocked and flash crowded continues. In this section, we will describe how the modeling is done for abnormal traffic detection, how the anomalies are diversified as DDoS or flash crowd, and how the DDoS traffic is filtered.

3.1 Abnormal Traffic Detection

The traffic we will target is a stream of successive HTTP Get requests. Traffic intensity measurement taken at fixed, discrete time intervals from a time series $\{y_t\}$. The intense change in the value of measurement means possible existence of application-layer DDoS attacks or flash crowds. Therefore, at instant t, the difference between measured value y_t and model output y_t' represents the abnormal constituent of the traffic. The abnormal constituent is called also the residual series or the model error, and is defined as $d_t = |y_t - y_t'|$.

As an early stage of our work, we use the stationary AP (p) model for abnormal traffic detection. The model is defined as:

$$y_t'' = \sum_{k=1}^{p} a_t^k x_{t-k} + e_t \quad (1)$$

Here y_t" is the prediction. x_t is the observation at instant t. a_t^k are the stationary model parameter, and e_t is observation error. In other words, the model uses a weighted sum of p previous values to estimate the current observation value, the weights being the AR coefficients a_t^k, k=1...p. The stationarity means that the weights a_t^k are time independent. The idea is that normal system usage causes sufficiently regular and smooth traffic, that the current value can be predicted as a linear combination of p past values. The part of the traffic behaviour that cannot be predicted in this fashion is sufficiently anomalous to be reported to the next component. In current stage of design, we use stationary AR model, which means the parameters will not adapt to changes of the monitored traffic. We simply assign a_t^k to be 2^{-k} and the sum of the a_t^k is equal to 1. In the future research, we will extend this component. The model degree p needs to be defined. However, its value depends strongly on the type of Websites. We will discuss it in the Section 4.2 with the experimental setting.

The interval is 1 second and the sensor counts the HTTP Get requests during the interval. Based on the previous p records, the AR model calculates the prediction y_t" for the next step. We need to calibrate the y_t" because of intense fluctuation in the network traffic. We fetch the deviation d_t between calibrated value y_t' and realistic measured value to make decisions. Fig. 2 describes an episode of the HTTP Get requests on April 10th 2010. We considered the cases in both Fig. 2A and Fig. 2B as normal traffic. However, if we set the threshold of abnormal traffic decision to be a little small for more sensitivity, the fluctuation will greatly noise the prediction as we see in Fig. 2C and Fig. 2D.

Therefore, we need to calibrate the value after prediction. Kalman filter is an adaptive and recursive data processing algorithm that is suited for on-line estimation. Its essence is to compute the covariance and make a calibration between the measured value and observed value. Filtering can be seen as a process when a new observation arrives. The prediction and measurement equations can be put in vector form:

$$y_t'' = AX_{t-1}^{t-p} + e_t \quad (2)$$

$$y_t = MX_t^{t-p+1} + w_t \quad (3)$$

With the coefficients as

$$A = (a_t^1 .....a_t^p) = (2^{-1}, 2^{-2} .....2^{-p+1}, 2^{-p+1}) \quad (4)$$

$$M = (m_t^1 .....m_t^p) = (1,0.....0)_p \quad (5)$$

For representing the Kalman filter equations, we use $X_{t|t-1}$ to denote the estimated value at time t using observations accumulated at instant t - 1. In addition, we use C_t to denote the covariance at time t. Now the Kalman filter equations for calibrating the prediction are:

$$x_{t|t-1} = y_t'' = AX_{t-1}^{t-p} = A(y')_{t-1}^{t-p} \quad (6)$$

$$C_{t|t-1} = C_{t-1} + \sigma_x^2 \quad (7)$$

$$x_{t|t} = x_{t|t-1} + Kg(y_t - x_{t|t-1}) \quad (8)$$

$$Kg(t) = C_{t|t-1}(C_{t|t-1} + \sigma_y^2)^{-1} \quad (9)$$

$$C_t = (1 - Kg(t))C_{t|t-1} \quad (10)$$

Among the equations, we fetch previous p values of x and initial the covariance C_0 as:

$$C_0 = (X_1^p - avg(y_1^p))(X_1^p - avg(y_1^p))^T \quad (11)$$

And the one-step calibration result will be (12). As an example, Fig. 2E and Fig. 2F depict a 2-steps calibration.

$$y_t' = x_{t|t} \quad (12)$$

Fig. 2(E) and Fig. 2(F) depict an example of 2-steps calibration. Moreover, Kalman filter is a recursive algorithm that may bring a little delay. This is because of the N-steps recursion. Thus the total complexity for abnormal traffic detection is N*p. In fact, we have to use observations x_t - x_{t+N} to estimate an improved prediction. The delay increases when N is enlarged and the accuracy inversely decreases. Therefore it should be chosen for a desired balance between prediction accuracy and suitable delay. We will discuss this value in Section 4.2 and 4.3.

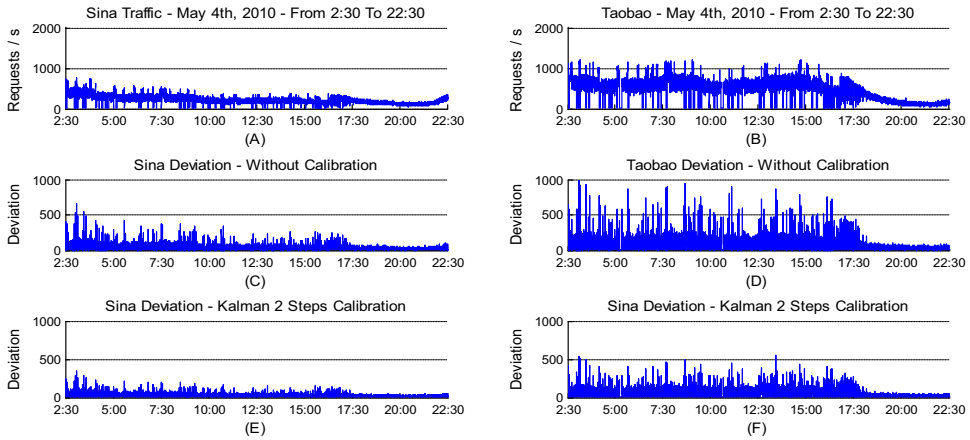

Figure 2. Normal Traffic Example and Abnormal Traffic Detection.
Note that this paper uses Kalman Filter to calibrate the deviation so that the fluctuation in the normal traffic will not noise the abnormal traffic detection.

3.2 Attack Inspection

3.2.1 Activate & Stop DDoS Detection Component

When the abnormal traffic is detected, the front-end sensor will send ATTENTION signal to the DDoS attack detection component and activate this component. This means the DDoS attack detection does not work for the whole traffic. It only runs when abnormal traffic arrives. In correspondence, when the front-end sensor finds the traffic has changed back to normal series, it will send DISMISS signal to the DDoS attack detection component and stop its work. To achieve above functions, we aggressively choose the threshold for abnormal traffic as:

$$d_t > k\sigma_d^2 \tag{13}$$

$$\sigma_d^2 \approx (\sum_{i=t-p}^{t}(d_t - avg(d_{t-p}^t))^2)/p \tag{14}$$

After Kalman filter smoothes the deviations, the value k in (13) becomes the only parameter to adjust the sensitivity for activating the DDoS attack detection. This is another balance value because smaller value k will bring more sensitivity. We will discuss this value in Section 4.2.

3.2.2 Properties of DDoS Attacks and Flash Crowds

In this subsection, we will analyse the basic properties of application-layer DDoS attacks and flash crowds so that no matter what stealth mechanisms the attackers adopt, the detection cannot be circumvented. As is mentioned in the Section 2, our work is concerned with three types of attacks. During the analysis on real dataset of www.taobao.com and www.sina.com, we noticed that:

- *Repeated Request DDoS*: this type of attack mainly focuses on the homepage or a hot Webpage, so the targets of the traffic converge to one or two point. We also suppose the attack is launched by a certain quantity of bots. Thus the sources of the traffic converge to a group of points.
- *Recursive Request DDoS*: this type of attack has a character that the traffic is scattered into different WebPages. 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.
- *Repeated Workload DDoS*: this type of attack is able to use less bots but even larger damage to the Website. However, the traffic will be similar to the case of 'repeated request DDoS'. A group of bots continually sends requests for a large image or database searching operations.
- *Flash Crowd*: an unexpected surge in visitors to a Web site, which is typically because of some newsworthy event that just took place. It may also be due to the announcement of a new service or free software download. The sources of flash crowds are definitely scattered because they are legitimate HTTP Get requests and 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, which comply with the definition of flash crowd.

3.2.3 Distinguish DDoS Attacks and Flash Crowds

Because the dispersion extent of the application-layer DDoS attacks and flash crowds are different in their targets and sources, we are able to use this basic property to distinguish them. However, such a statistical resembling method sounds intractable in real-time process as the statistical algorithms generally have great complexity in both space and time. In CALD, we firstly introduce a recursive method to calculate the possibilities for each sources and targets:

$$favg_n = (\frac{1}{T_2 - T_1} + \frac{1}{T_3 - T_2} + ... + \frac{1}{T_n - T_{n-1}})/n \tag{15}$$

$$favg_{n+1} = (favg_n \times n + \frac{1}{T_{n+1} - T_n})/(n+1) \tag{16}$$

According to equations (15) and (16), two dynamic vectors should be implemented in DDoS detection component. When a new HTTP Get request arrives, CALD will compute the time difference between the two arrival times and update the average frequencies in the source and target vectors. Particularly, we identify each target Webpage and source with special serial number so that searching the position for updating or inserting will not cost too much CPU time. The complexity extent of the whole algorithm keeps in O(3), which is acceptable for the efficiency of CALD.

In an intense traffic, we simply consider average frequency as the possibility. To describe the dispersion extent of the application-layer DDoS attacks and flash crowds, here we introduce the concrete measurement:

Definition (Mess Extent): suppose there is a set {n}, if the elements in set {n} are positioned dispersedly, then the Mess Extent is higher. Otherwise, if the elements in set {n} are converged in some points by any organized form, the Mess Extent is lower and close to 0.

In CALD, the different mess extents of the sources' or targets' possibilities are capable of describing the right distribution of attackers and targets. In fact, the mess extent denotes the distribution of contained information. This is essentially equivalent to the related concept in information theory. More concretely, CALD periodically calculates the mess extent of a vector using Shannon formula:

$$Me(t) = H(p_t^1, p_t^2, ..., p_t^n) = \sum_{i=1}^{n} p(x_t^i)\log(x_t^i)$$

$$= \sum_{i=1}^{n} favg_i \log(favg_i) \quad (17)$$

According to the analysis in Section 3.2.2, we can use the rate of mess extents between sources and targets to distinguish various application-layer DDoS attacks and flash crowds. We deduce the following conclusion (S is source vector; T is target vector; 'repeated request DDoS' as 1; 'recursive request DDoS' as 2; 'repeated workload DDoS' as 3 and flash crowd as 4):

$$\frac{Me(S)_2}{Me(T)_2} > \frac{Me(S)_1}{Me(T)_1} \geq \frac{Me(S)_3}{Me(T)_3} > \frac{Me(S)_4}{Me(T)_4} \quad (18)$$

In CALD, the DDoS detection component implements two thresholds, one for mess extent calculation period and another for the boundary between DDoS attack and flash crowd. We put the details of the former in Section 4.3 and the latter in Section 4.2. When the DDoS detection component finds a real-time rate fallen in the range DDoS attack, it will post the source IPs, which has a higher frequency, to the filter component.

3.3 Filters

When the DDoS detection component has made a decision that current traffic contains application-layer DDoS attack, the DDoS detection component will set the parameters of the filter for blocking the HTTP Get requests from those malicious IP addresses. In fact, current application-layer DDoS attacks are mainly launched from botnet, there is a tendency that the quantity of bots concerned in the DDoS attack becomes larger and larger [14] that the number may reach more than 20,000. As described in Section 3.2(c), we simply choose the source IP addresses that emerges more than Q times (we set Q = 10 in CALD) in 1 second interval. However, it is not an easy task to implement such a list which more than 20,000 malicious IP addresses. We are able to find some related work in the technologies of network routers for packets relaying [16, 17]. However, routers only need to look up the prefix of an IP address based on mask code. When the cases come to the filter component of CALD, the looks-up should cover the whole IP address.

To avoid the filter becoming a bottleneck of the whole system, CALD similarly adopted Bloom Filter [15] as the basic IP looks-up and update technology. An empty Bloom Filter is a bit array of m bits, all set to 0. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions with a uniform random distribution. Here CALD implements two hash functions inside (k = 2) and the length of the bit array m is set to be 2^{20}. Suppose the IP address is in dotted decimal notation 'A.B.C.D', and then the hash functions are:

Hash function 1:

$$(A^3 + B^3 + C^3 + D^3) \mod 2^{20} \quad (19)$$

Hash function 2:

$$(A * B * C * D) \mod 2^{20} \quad (20)$$

To add an element, feed the IP address to each of the two hash functions to get two array positions. Set the bits at all these positions to 1. To query for an element (test the IP address is in the set), feed it to each of the two hash functions to get two array positions. If any of the bits at these positions are 0, the IP address is not in the set. The reason is if it were, then all the bits would have been set to 1 when it was inserted. If all are 1, then either the IP address is in the set, or the bits have been set to 1 during the inserting of other IP addresses. The latter situation is called collision. Because we set the filter with two hash functions, the length of hash table is 2^{20} and estimated quantity of malicious IP addresses is 20,000, the collision possibility CP becomes:

$$CP = \lim_{m>>n}[1-(1-\frac{n}{m})^k]^k = \lim_{m>>n}(1-e^{-\frac{kn}{m}})^k$$

$$< (-\frac{kn}{m})^k < (\frac{2\times 2\times 10^4}{10^6})^2 = \frac{16}{10^4} \quad (21)$$

For CALD, the CP of sixteen ten-thousandth is not very big and the number of 2^{20} bits means 128KB memory is needed for hash table.

4. PERFORMANCE EVALUATION

4.1 Experiment Environment

Fig. 3 illustrates the key component of CALD. Our design allows different components assembled in different

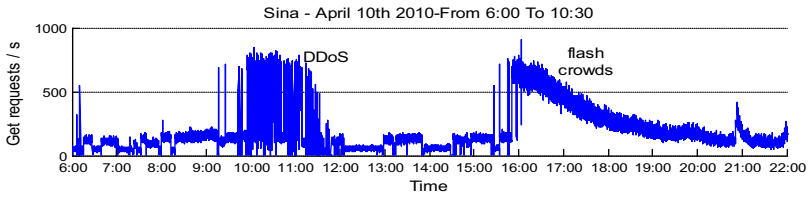

Figure 4: The Quantity of HTTP Get Requests per Second.

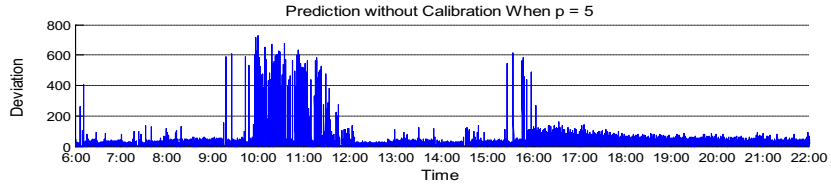

Figure 5: The Deviation when p = 5.

computers so that the functions of CALD will not affect the process of Website. However, in our experiment, we simply integrated various components together with the Web server. The computer for experiment was a standard 3GHz double kernels Pentium IV machine with 4GB memory and running window vista. We also implemented a discrete time series producer which replayed each HTTP Get request of www.sina.com in April 10th. These HTTP Get requests were collected from the backbone of China Southwest. The population concerned was more than 300 million. The bandwidth of the backbone was 40GB. Under current condition of technologies, we were able to collect the requests in 10GB wherein 2GB for management. Therefore the rate for collection was one fifth. As CALD never cares about the sequence of HTTP Get requests, such rough data is enough for the analysis. We believe this character is right an advantage against the work of [20-23, 26], because sometimes we can hardly gather the exact HTTP Get request series.

4.2 Sensitivity and Modular Delay

There are two aspects that may influence the performance of CALD: system sensitivity and modular delay. System sensitivity decides the ability to detect the application-layer DDoS attacks and modular delay decides the time for CALD to detect the attacks. There are three parameters which may affect these two kinds of performances.

The first one is the parameter p of AR model. The traditional assignment on parameter p should is directed by parameter estimation [27], which is not suitable to be dynamically used in CALD. As will be discussed in Section 6, CALD does not need accurate model for tracing the traffic. In fact, in current version, we set the parameter a_t^k as $\{2^{-k}\}$, so when k becomes larger, the effect becomes less, even nearly no effect on AR model. Fig. 4 shows an episode of application-layer DDoS attack from www.sina.com in April 10th 2010. Fig. 4 also describes a flash crowd which happened in the same day. Fig. 5 shows the corresponding prediction when we set p as 5.

The second one is the lag of Kalman filter. Larger lag brings more sensitivity as long as larger delay for DDoS attacks detection. Fig. 6 shows 1-step to 3-steps calibrations which will bring 1 second to 3 seconds delay respectively. In the figures, we noticed a strange phenomenon that 2-steps calibration performance better among the three. The 3-steps calibration rebounded to be more serrated in the amplitude of deviations. We will find the reason in the future work.

The third one is the boundary for distinguishing the attacks and flash crowds. Setting this threshold will also affect both the sensitivity and modular delay. In current work, we have not found any reasonable methods to decide the most suitable value for this boundary. We simply assigned it as 0.7 and successfully detected the DDoS attack of www.sina.com in the seventh second after five seconds for AR model report abnormal traffic and 2 seconds for calculation period of mess extent. CALD never triggered the block of source IPs against flash crowds.

5. RELATED WORK

Jaeyeon Jung and his collaborators [18] studied the properties of both application-layer DDoS attack and flash crowd with a special attention to characteristics that might distinguish the two. Identifying these characteristics allowed a formulation of a strategy for Websites to quickly discard malicious requests. However, as we can say, some of the declared characteristics are a series of phenomenon, not the essential properties. The attackers were capable of avoiding this appearance by simply adjusting attack scheme. CAPTCHA [19] and later another improved version by Scrianth Kandula, etc [20], almost perfectly solved the application-layer DDoS, even the flash crowds, but this method basically required the engagement of clients to recognize a puzzle figure and input the result for an authentication. Supranamaya Ranjan et al. [21] proposed a counter-mechanism that consists of a suspicion assignment mechanism and a DDoS-resilient scheduler. The former assigned a suspicion measure to a session in proportion to its deviation from legitimate behaviour and used the latter to decide whether and when the session is serviced. According to their experiments, the victims' performance could be improved from 40 seconds to 1.5 seconds. However, we

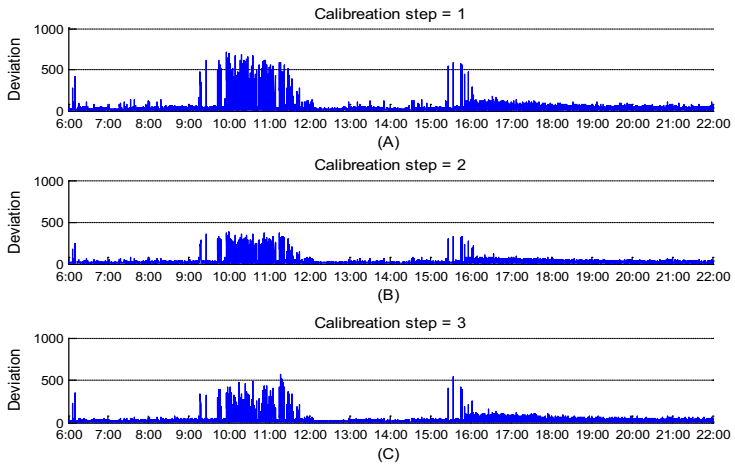

Figure 6. The Deviations when lag k is set to 1-3.

think DDoS Shield introduced in this paper had a drawback that their method could not actively block the malicious traffic. It only released the symptom of being attacked victims.

In recent two years, Dr. Yi Xie adopted Hidden Semi-Markov model for the detection of application-layer DDoS attack. His method recorded the current user visiting sequences and calculated the diversities in possibilities of the sequences. The biggest problem of Hidden Semi-Markov method was the algorithm complexity. Although Dr. Yi Xie improved this drawback based on M-algorithm in [22] and independent component analysis in [23], we still found that tracking each user's visiting sequence was not a practical task. Georgios Oikonomou and Jelena Mirkovic [14] proposed defences against application-layer DDoS attacks via human behaviour modelling which differentiate DDoS bots from human users. Their method was achieved through three aspects: request dynamics, request semantics and ability to process visual cues. We found their work had several limitations. For instance, they tried to build a possibility graph for a Website, but as we can say, for most large Websites such as www.taobao.com, with too many dynamical Webpages inside, they can hardly finish the construction.

There are still other related works about detecting application-layer DDoS attacks. Simon Byers et.al [24] discussed the dangers that scalable Internet functionality may present to the real world. Their work focused upon an attack that is simple, yet could have great impact, which they believed might occur quite soon. This paper is an early published prospective work about application-layer DDoS attacks. Paul Barford et.al [25] used wavelet to distinguish the flash crowds and DDoS attacks. However, the method of using wavelet was only a post-mortem technology. Takeshi Yatagai et.al [26] proposed HTTP Get flooding detection techniques based on analysis of page access behaviour. They proposed two detection algorithms, one is focusing on a browsing order of pages and the other is focusing on a correlation with browsing time to page information size. We think the drawback of their work was somewhat the same with Dr. Yi Xie's and Jaeyeon Jung's, because the properties for detection could easily be circumvented by simply modifying the attack schemes.

6. DISCUSSION

It could be argued that AR model is too limited for modelling the type of HTTP Get traffic. For example, errors on rapid changes in observations are one manifestation of the AR model's limitations and non-linear models, for example, could provide better reactivity to rapid changes. We have three reasons to stay with linear models for front-end abnormal traffic detection: (1) given our underlying idea we do not even want to model the traffic exactly, especially the rapid changes should be left out of the model. We only pay attention to the amplitude of the changes. (2) Through the smoothing function by N-steps Kalman filter, the noise of rapid changes will be mitigated and the calibrated prediction is able to improve the accuracy. (3)Simple models mean simpler algorithms and this translates to faster and lighter implementations, which is an important consideration for deploying the method.

There is another methodological point which easily incurs argument. It also derives from the AR linear model. When the attackers slowly increase the abnormal traffic, they are able to circumvent the abnormal traffic detection. This is the key weakness of linear prediction model resembling machine learning technologies. In current version of CALD, in fact, we set a threshold inside. When the real-time traffic rises beyond that threshold, no matter whether abnormal traffic detection function is aware of the anomalies, the front-end will report the ATTENTION and activate the next DDoS detection component as well. However, if we induce the traffic of import as well as export, CALD will become capable of defending such slowly increasing DDoS attack. We will improve the CALD in the next stage of our work.

Third, CALD has a few parameters that we have assigned values based on experience. For example, we set the AR model parameter p to be 5 in the experiments. There is nothing special about 5, we only need a value that is neither too big nor too small. In fact, the prediction could become

more long-effecting if we set p a bit larger. Similarly, we set the mess extent of flash crowds to be under 20. There is also nothing special about this threshold, we just get it according to our experience by analysing the past records of application-layer DDoS attacks.

Fourth, the filter component needs to be flushed eventually since compromised zombies may turn into legitimate clients. The filter can be cleaned either by resetting all entries simultaneously or by decrementing the various entries at a particular rate. In the next stage of our work, we will implement such functions in CALD.

7. CONCLUSION

In this paper we have presented an approach for countering application-layer DDoS attack that mimics the normal users' behaviours. The basic assumptions are: (1) abrupt changes in the traffic mean a possible existence of abnormal traffic; (2) the mess extent of various DDoS attacks is larger than the one of flash crowds. Based on the first point, we introduce a front-end sensor to detect the abnormal traffic and report its existence when abnormal traffic arrives. The second component which is used for distinguish DDoS attack and flash crowd is activated or stopped by the signal ATTENTION or DISMISS from the front-end sensor. With parameters of malicious IP addresses, the filter blocks the abnormal traffic and leaves the Website to be safe. The analysis and experiments show that CALD perform well in countering various application-layer DDoS attacks and the delay for detection is kept in a range of N seconds.

This paper presents an early stage of our work. In the future, we will extend the work in the following aspects: (1) improve the stationary AR to be time-varying model where the parameters will denote the changes of the system status; (2) as is discussed in Section 6, CALD is not sensitive to the slowly increasing DDoS attack. Thus we may consider both the output and input traffic to deal with this situation.

ACKNOWLEDGEMENTS

We thank Chongqin University for supporting the real traffic traces on the Internet backbone of China. We also thank Professor Guofeng Zhao for giving us lots of constructive comments on CALD.

REFERENCES

[1] CERT. Incident Note IN-2004-01 W32/Novarg.A Virus, 2004.
[2] CERT. Incident Note IN-2001-10 "Code Red" Worm Crashes IIS 4.0 Servers with URL Redirection Enables, 2001.
[3] Kevin Poulsen. FBI busts alleged DDoS Mafia, http://www.securityfocus.com/news/9411, 2004.
[4] J.Leyden. East European Gangs in Online Protection Racket, http://www.theregister.co.uk/2003/11/12/east_european_gangs_in_online/, 2003.
[5] Martyn Williams and Jeremy Kirk. Updated MyDoom responsible for DDOS attacks, http://www.networkworld.com/news/2009/070809-updated-mydoom-responsible-for-ddos.html?ap1=rcb, 2009.
[6] Chuck Miller. Russia Confirms involvement with Estonia DDoS Attacks, http://www.scmagazineus.com/russia-confirms-involvement-with-estonia-ddos-attacks/article/128737/ , 2010.
[7] Dancho Danchev. The DDoS Attack against CNN.com, http://ddanchev.blogspot.com/2008/04/ddos-attack-against-cnncom.html, 2008.
[8] SOHU News. Botnet has formed industry chain, http://news.sohu.com/20091126/n268475118.shtml, 2009.
[9] BBC News. How cyber criminals attack Websites. http://news.bbc.co.uk/2/hi/programmes/click_online/7940485.stm, 2009.
[10] B.Krishnamurthy and J.Wang. On Network-Aware Clustering of Web Clients, in Proceedings of the ACM SIGCOMM, Stockholm, Sweden, 2000.
[11] Ratul Mahajan et al. Controlling High bandwidth Aggregates in the Network, ACM Computer Communication Review, 2002.
[12] Jouni Viinikka, Herve Debar, Ludovic Me et al. Processing Intrusion Detection Alerts Aggregates with Time Series Modeling, Information Fusion, Elsevier, 2009.
[13] Wei Lu and Ali A. Ghorbani. Network Anomaly Detection Based on Wavelet Analysis, EURASIP Journal on Advances in Signal Processing, Hindawi Publishing Corporation, 2009.
[14] Georgios Oikonomou and Jelena Mirkovic, Modeling Human Behaviour for Defense against Flash-Crowd Attacks, IEEE International Conference on Communications, 2009.
[15] A.Broder and M.Mitzenmacher, Network Applications of Bloom Filters: A Survey. Internet Math, Volume 1, Number 4, 2003.
[16] Ioannis Ioannidis, Ananth Grama and Mikhail Atallah, Adaptive Data Structures for IP Lookips. In Proceedings of the IEEE Computer and Communications Societies (Infocom), 2003.
[17] Andrei Broder and Michael Mitzenmacher, Using Multiple Hash Functions to Improve IP Lookups. In Proceedings of the IEEE Computer and Communications Societies (Infocom), 2001.
[18] Jaeyeon Jung, Balachander Krishnamurthy and Michael Rabinovich, Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In Proceedings of the 11th International World Wide Web Conference (WWW), Honolulu, Hawaii, USA, 2002.
[19] L.von Ahn, Manuel Blun, Nicholas J.Hopper and John Lanford, CAPTCHA: Using Hard AI Problems for Security. In Proceedings of EUROCRYPT, pp.294-311, 2003.
[20] Srikanth Kandula, Dina Katabi, Matthias Jacob and Arthur Berger, Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In Proceedings of the 2nd Symposium on Networked Systems Design & Implementation (USENIX NSDI), 2005.
[21] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal and Edward Knightly. DDoS-Resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection. In Proceedings of the IEEE Computer and Communications Societies (Infocom), 2006.
[22] Yi Xie and Shun-Zheng Yu, A Large-Sclae hidden Semi-Markov Model for Anomay Detection on User Browsing Behaviours. IEEE/ACM Transactions on Networking, vol.17, 2009.
[23] Yi Xie and Shun-Zheng Yu, Monitoring the Application-Layer DDoS Attacks for Popular Websites. IEEE/ACM Transactions on Networking, vol.17, 2009.
[24] Simon Byers, Aviel D.Rubin and David Kormann, Defending Against an Internet-Based Attack on the Physical World. ACM Transactions on Internet Technology, vol.4, page.239-254, 2004.
[25] Paul barford, Jeffery Kline, David Plonka and Amos Ron, A Signal Analysis of Network Traffic Anomalies. In Proceedings of ACM SIGCOMM Internet Measurement Workshop, 2002.
[26] Takeshi Yatagai, Takamasa Isohara and Iwao Sasase, Detection of HTTP-GET Flood Attack Based on Analysis of Page Access Behaviour. In Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing, 2007.
[27] George Box, Gwilym M.Jenkins and Gregory C.Reinsel, Time Series Analysis: Forecasting and Control, Third Edition, Prentice Hall, 199

View publication stats