This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Fast Anomaly Detection for Large Data Centers

Ang Li and Lin Gu
Hong Kong University of Science and Technology
Email: {laxab,lingu}@cse.ust.hk

Kuai Xu
Arizona State University
Email: kuai.xu@asu.edu

***Abstract*—Recent spates of cyber attacks towards cloud computing services running in large data centers have made it imperative to develop effective techniques to detect anomalous behaviors in the "clouds". In this paper, we propose to use the distributions of IP address octets and centroid based measures to characterize the inherent IP structure in high-volume data center traffic, and subsequently design a simple yet effective algorithm to detect abnormal traffic patterns caused by network attacks such as worms, virus, and denial of service attacks. We evaluate the effectiveness and efficiency of this algorithm with synthetic traffic that combines real data center traffic collected from a large Internet content provider with worm traces and denial of service attacks. The experiment results show that our algorithm consistently diagnoses the abnormal traffic from normal ones, and does so in a short time with a low false alarm rate. We believe that the proposed approach could be potentially deployed in real-time data center environments to enhance the security and high availability of cloud computing.**

I. Introduction

Cloud computing reflects a trend of integrating data, users, and logic on a vast scale, thus enabling a potentially global optimization of computing resources. At the center of this global view of computation are high-capacity data centers serving as a high-availability backbone for application services. The availability of abundantly provisioned data centers and the development of elastic cloud infrastructures bring new applications opportunities and business models, and may reshape the IT industry.

However, many challenges and obstacles remain for cloud computing. The number one obstacle identified in the Berkeley's view of cloud computing [4] lies in "availability of services" due to service outages and distributed denial of service (DDoS). The recent work by Ristenpart et al. [11] introduces the vulnerabilities with shared virtual machines (VM) from cloud computing providers and demonstrates the feasibility of mounting cross-VM side-channel attacks to gain information from the target VMs. Given the diversity and magnitude of security threats from outside and inside the cloud, it is crucial to develop sound technical measures to protect data centers.

In this work, we are mainly interested in protecting data centers from external threats, where the adversary is outside the data centers and attacks the cloud infrastructure or service through the Internet. Examples of outside attacks include Internet worms [13], viruses, system penetrations [14], and DDoS attacks [5]. Originating possibly from any obscure location in an uncontrolled environment (the Internet) at an unpredictable time, external threats are a major concern for network operators of large data centers. The recent attacks towards GMail and other cloud services illustrate the urgency of addressing such threats [14].

Because external attacks materialize in the form of network packets, one approach to protecting data centers is to closely monitor the network traffic and detect anomalous behaviors towards data centers. Anomaly detection based approaches have been studied in the literature, and used for personal systems, corporate firewalls, and backbone networks. However, existing solutions encounter serious issues in performance and effectiveness in large data center environments. The reasons are manyfold, including the scalability issues at the data center scale, the dynamic and hybrid workload, the requirement of extremely high sensitivity, the stringent time constraints, and the unknown and evolving external environment. Hence, as cloud computing and Internet-scale computation continue to grow, how to monitor network traffic and detect anomalies for large data centers become an urgent research topic.

Towards this end, this paper develops a simple and fast approach to provide on-line traffic anomaly detection for large data centers. Our work begins with the traffic characterizations of Internet data centers using network flow traces collected from several data centers in Yahoo!'s global network [1]. We design a lightweight calculation to summarize the structural characteristics of IPs into a value representing the *centroid* of selecte octets in the 32-bit IP addresses.

Based on the *centroid*, which is very easy to compute and makes continuously monitoring of traffic flow in very large data center feasible, we further develop methods for fast anomaly detection for large data centers. We validate our approach using synthetic traffic that combines real data center traffic collected on Yahoo!'s regional network exchanges and packet traces from real-world worm epidemics as well as denial of service attacks. Our experiment results demonstrate that the algorithm indeed is able to differentiate abnormal traffic patterns containing malicious traffic from normal ones in a short time. Our work makes the following contributions.

- By analyzing real-world traffic traces, we show several characteristics of the IP address distribution observed on large data centers for cloud computing services.
- We reveal a stable concentration on selected octets of both source and destination IP addresses, and derive *centroid* based statistics to characterize data center traffic.
- We propose an IP-structural approach to fast anomaly detection based on the changes of the *centroid* over time, and design a lightweight anomaly detection algorithm that

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

can be used for very large data centers.
- We evaluate the effectiveness of our methodology using real-world data sets collected from a large Internet content provider and a variety of trace-driven simulations.

The remainder of this paper is organized as follows. Section II describes related work. Section III presents traffic characteristics of large data centers and our approach of fast anomaly detections. In Section IV, we evaluate our approach via experiments. Section V concludes this paper and outlines the future work.

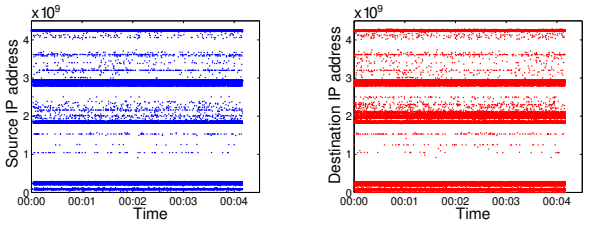
II. Related Work

As the infrastructure of cloud computing, the data centers play an increasingly important role in today's computation. Meanwhile the security of data centers is still a challenging area with many open problems [4]. While there exist a number of novel solutions to developing scalable and energy-efficient network architecture for data centers, very few recent developments focus on network security for data centers and how to protect them from cyber attacks. Our work is to develop efficient and scalable algorithms to detect anomalous behavior towards data centers.

Many techniques have been proposed to detect anomalies in traffic volume. Snort [12] and Bro [10] are two widely used signature-based intrusion detection systems that examine preconfigured and predetermined attack patterns in network traffic of end hosts to detect attack patterns. Much of the work in anomaly detection has been restricted to specific types of anomalies, e.g., portscans, worms, and DOS attack. However, these systems are inefficient when detecting unknown attacks. When a new worm spreads, data centers cannot afford to wait several hours or days in a vulnerable state for a security fix. For this reason, anomaly detection is advantageous in handling new, unknown, and unanticipated attacks in the network flow. Our work is one approach to anomaly detection, and has been verified in Section IV under multiple types of attacks without *a priori* knowledge on the adverse traffic patterns.

A number of existing techniques treat anomalies as deviations in the overall traffic volume [8]. Such volume based schemes are effective for large traffic changes, such as bandwidth flooding attacks, but often have difficulty in detecting anomalies when the spurious packets constitute only a small fraction of the overall traffic volume.

Closer to our approach are anomaly detection systems using IP address information. In [9], Lakhina et al. use entropy as a summarization tool, and implement automatic classification of anomalies via unsupervised learning. [6] represents multiple pieces of measurements as different colors of an image, enabling uniform processing of multidimensional packet header data. However, they only consider the number of packets for different values of the header field of a packet, e.g., source address, destination address, source port, destination port, etc. Our method uses only the source and destination addresses, but uses sorting and weighted average to extract IP-structural information instead of simply using the packet counts. As shown in Section IV, our method is much more sensitive than the earlier ones, and is resilient to normal traffic fluctuation.



(a) Source IP distribution (b) Destination IP distribution

Fig. 1. Source and destination IP addresses observed on Hong Kong data center over 4 minutes. IPs are represented as 32-bit numeric values.

III. Analysis of Traffic Flows

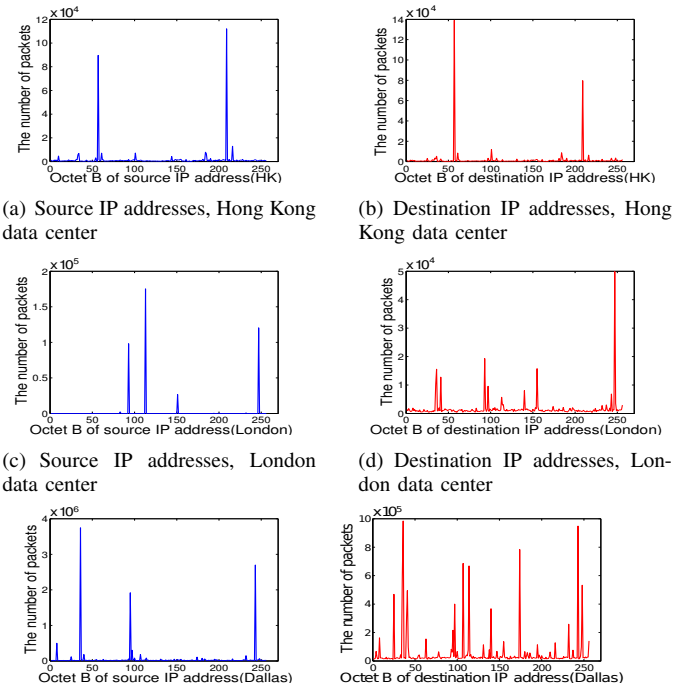
In this section, we first analyze the characteristics of IP addresses in packet traces obtained from several data centers, which forms a solid basis for developing the centroid based algorithm. We then present the details of our anomaly detection algorithm and its implementation in realistic settings where the "normal" traffic is not known.

A. IP-structural characteristics of data center traffic

We start the analysis by examining the distribution of IP addresses in the sequence of packets observed on a data center. Representing IPv4 addresses as numbers between 0 to $2^{32}-1$, Figure 1 illustrates the distribution of source and destination IP addresses observed on the Hong Kong data center [1]. Obviously, the IP addresses concentrate in a few bands. Due to the high packet rate, we only plot the IPs for 4 minutes, but the same regularity in the distribution of IP addresses is generally evident in all windows of different sizes, and, not surprisingly, a natural result of block-based IP allocation. However, the stableness of the concentration suggests a potential "invariant" for data center traffic – the structure of the collection of IP addresses on a data center is a relatively stable variable resilient to temporal traffic fluctuations.

To inspect the IP addresses distribution more precisely, we examine the octets in the 32-bit IP addresses. Suppose we use the dotted quad notation *A.B.C.D* to denote an IP address, and call the octets in the IP octet *A*, octet *B*, octet *C*, and octet *D*, respectively. Following the strong concentration of IP addresses, the values of octets *A*-*D* are not uniformly distributed. For example, Figures 2(a)–2(f) show the number of packets for each octet *B* value in the source and destinations IP addresses in a 15-minute interval. It is interesting to observe that a very small number of values dominate the distribution of octet *B*. The statistics for other octets in the IP addresses and other time intervals are similar. In conclusion, the *concentrated* distribution of octets in the IP addresses is a common characteristic of traffic flows in data centers.

In Figure 2, we present empirical results from the Hong Kong, London and Dallas data centers to have a broad coverage. As the figures show, the characteristics of the distributions on the three data centers are the same, and the difference



(a) Source IP addresses, Hong Kong data center

(b) Destination IP addresses, Hong Kong data center

(c) Source IP addresses, London data center

(d) Destination IP addresses, London data center

(e) Source IP addresses, Dallas data center

(f) Destination IP addresses, Dallas data center

Fig. 2. Distribution of values of octet B in the IP addresses in packets sampled from the Hong Kong, London, and Dallas data centers.

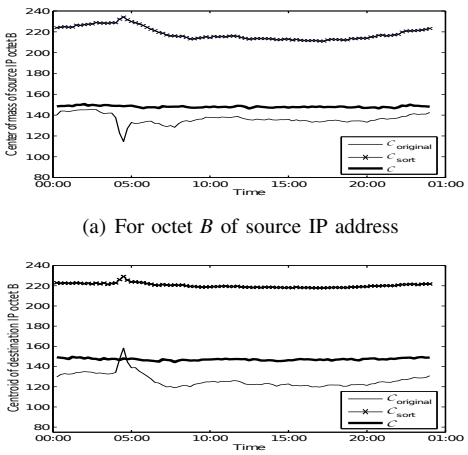
is only the specific values that dominates the distribution. This generality across different data centers also applies to other empirical results in this study. To avoid clutter, we will consistently use the Hong Kong data center for presenting results henceforth.

To summarize the structural characteristic, we compute the *centroid* of the collection of IPs. Generally, the *centroid* $\mathcal{C}_{general}$ of a collection of K items $\{r_i, i = 1, 2, ..., K\}$ is defined as follows:

$$\mathcal{C}_{general} = \frac{\sum_{i=1}^{K} m_i \cdot r_i}{\sum_{i=1}^{K} m_i} \quad (1)$$

For each i in $[0, 255]$, if we let $r_i = i$ and m_i be the number of packets whose octet *B* is i, we can calculate *centroid* based on Eqn.(1). We call it the original *centroid*, denoting it as $\mathcal{C}_{original}$. As shown in Figure 3, $\mathcal{C}_{original}$ shows small fluctuation, but it is not stable enough to capture the IP-structural "invariant" in the traffic flow.

We calculate a more stable *centroid* by *sorting* the numbers of packets. The *sorting* is equivalent to re-numbering the octet *B* values so that the largest packet count is associated with 255, the second largest count is associated with 254, and so on. After *sorting*, we calculate the *centroid*, and denote it as $\mathcal{C}_{sort}$. As shown in Figure 3, the fluctuation of $\mathcal{C}_{sort}$ is much smaller – changes between consecutive data points are within 3. Yet we can obtain a more stable *centroid*, denoting as $\mathcal{C}$, based on Algorithm 1, which describes the calculation of $\mathcal{C}$ from N_x. Different from the mathematical definition, we use two configurable parameters, S and T, in this algorithm, so



(a) For octet *B* of source IP address

(b) For octet *B* of destination IP address

Fig. 3. Centroid of source and destination IP address for octet *B*

that system operators can adjust the impact dominating values have on the overall centroid. As shown in Figure 3, $\mathcal{C}$ is very stable, with differences between adjacent data points no more than 1, which is resilient to the temporal fluctuations of normal traffic. Hence, $\mathcal{C}$ can be used as a very effective metric to approximate the IP-structural characteristics of network flows when detecting anomalous traffic disturbances.

B. Anomaly detection based on the centroid $\mathcal{C}$

Based on the centroid $\mathcal{C}$, we construct a solution to detecting abnormal traffic patterns. Intuitively, significant anomalous traffic disturbances, such as worm spread, or DDoS attacks, disrupt the distribution of the IP addresses observed on a data center, and make the *centroid* deviate from the normal value. Hence, a significant change in $\mathcal{C}$ indicates a suspicious traffic pattern that the data center operators need to investigate.

As a packet arrives at an entry point of a data center, the router or a monitoring station at the entry point examines the source and destination IP addresses in the packet, and increment the packet counter corresponding to the value of a particular octet of interest. We use a small array N_x of 256 entries to record the packet counts for specific octet values, with x being *A, B, C or D*. The anomaly detection algorithm periodically computes the centroid $\mathcal{C}$ from N_x, and clears N_x to zero. The centroids thus calculated provide raw measurements based on which the anomaly detection function is implemented. The per-packet cost is an array indexing operation and an integer increment, and no packet header, payload, or other detailed information needs to be analyzed or retained. With such low run-time overhead, this anomaly detection method can scale to any large data centers.

To measure the significance of changes in the *centroid*, we define

$$ratio_i = \frac{\Delta_i}{\Phi_i} = \frac{|\mathcal{C}^i_{Mix} - \mathcal{C}^i_{Normal}|}{|\mathcal{C}^i_{Normal} - \mathcal{C}^{i-1}_{Normal}|} \quad (2)$$

where $\mathcal{C}^i_{Mix}$ represents the centroid of traffic flows that contains abnormal traffic at the time period i, and $\mathcal{C}^i_{Normal}$

Algorithm 1 Algorithm for calculating $\mathcal{C}$

1: Initialization: an array $N_x[i], i = 0, 1, ...255$, where $N_x[i]$ represents the number of packets whose IP address has an octet x value equal to i.
2: Sort the array $N_x[i]$ in ascending order, i.e., $N_x[0] \leq N_x[1] \leq ... \leq N_x[255]$
3: $S = 0; T = 150$; // Configurable parameters
4: $m = 0$; $z = 0$;
5: **for each** $i \in [S, 255]$ **do**
6: **if** $N_x[i] \leq N_x[T]$ **then**
7: $W_x[i] = N_x[i]$;
8: **else**
9: $W_x[i] = N_x[T] + log(N_x[i] - N_x[T])$;
10: **end if**
11: $m_i = W_x[i]$; $m = m + m_i$; $z = z + i \cdot m_i$
12: **end for**
13: $\mathcal{C} = \frac{z}{m}$; // Compute the center of mass $\mathcal{C} = \frac{\sum_{i=S}^{255} i \cdot m_i}{\sum_{i=S}^{255} m_i}$;

represents the centroid of traffic flows without abnormal traffic at the time period i. Similarly, $\mathcal{C}_{Normal}^{i-1}$ denotes the centroid without abnormal traffic at the period $i-1$. The ratio compares the *current* change of centroid with the *normal* change of centroid. A large ratio indicates that the current change of center of mass is likely driven by abnormal network events.

C. Improvements for realistic environments

The definition of ratio in Eqn.(2) assumes *a prior* knowledge of what packets are normal or abnormal during the time period i, which is not achievable in a real case when the normal and abnormal packets are mixed together. Without *a priori* knowledge on the properties of attacks, it is also very difficult, in some cases impossible, to distinguish spurious packets from normal ones. Therefore, a realistic question is *"how to define Φ and Δ for computing the ratio?"*. We solve this and other realistic issues in the rest of this section.

1) Estimation of Δ and Φ: We use exponentially weighted moving average (EWMA) to estimate $\mathcal{C}_{Normal}^{i}$ with $\mathcal{C'}_{Normal}^{i}$ as follows:

$$\mathcal{C'}_{Normal}^{i} = (1-\alpha) \cdot \mathcal{C'}_{Normal}^{i-1} + \alpha \cdot \mathcal{C}_{Mix}^{i}$$
$$\mathcal{C'}_{Normal}^{0} = \mathcal{C}_{Mix}^{0}$$

$$\Delta_i' = |\mathcal{C}_{Mix}^{i} - \mathcal{C'}_{Normal}^{i}| \tag{3}$$

Similarly, $\Phi_i = \mathcal{C}_{Normal}^{i} - \mathcal{C}_{Normal}^{i-1}$ would measure the movement of centroid of current traffic flows in a realistic cloud environment. Our solution to this challenge is to consider $W_x[i]$ only if $N_x'[i] \geq N_x[T]$ while calculating the centroid for Φ. Note that, in Algorithm 1, the condition $N_x'[i] \geq N_x[T]$ determines the dominant traffic flows. As a result, even though the current traffic flows may contain anomalous traffic, when we only consider the arrays $i \geq T$, the change of the centroid mainly reflects the insignificant changes of the normal traffic flows.

Formally, we define

$$\mathcal{C}_{Mix}^{*i} = \frac{\sum_{i=T}^{255} W_x[i] \cdot i}{\sum_{i=T}^{255} W_x[i]}$$
$$\mathcal{C}_{Normal}^{*i} = (1-\alpha) \cdot \mathcal{C}_{Normal}^{*i-1} + \alpha \cdot \mathcal{C}_{Mix}^{*i}$$
$$\mathcal{C}_{Normal}^{*0} = \mathcal{C}_{Mix}^{*0}$$
$$\Phi_i' = |\mathcal{C}_{Normal}^{*i} - \mathcal{C}_{Normal}^{*i-1}| \tag{4}$$

In conclusion, ratio in real cases is defined as follows. In the case that $\Phi_i' = 0$, we set $\Phi_i' = 0.0001$ (the minimum definition precision).

$$ratio_i' = \frac{\Delta_i'}{\Phi_i'} \tag{5}$$

2) Choice of threshold: Our approach to detecting anomalous situation uses a threshold and a parameter, η, to control when an alert should be sent to system administrators about a potentially hazardous network anomaly. We believe that the threshold and η should be flexible and adaptive to the environment and traffic characteristics of different data centers. In our experiment, we employ the adaptive threshold defined in Eqn.(6). and set $\eta = 0.1$ by default. In the next section, we will show the advantages of the adaptive threshold over the fixed ones. When $ratio_i' > (1+\eta) \cdot threshold_i$, the anomaly detection system dispatches an alert to the administrators, and the administrators may take further actions to investigate and fix the problem. The anomaly detection system ifself does not dictate what remedy actions to take upon reception of alerts.

$$threshold_i = (1-\beta) \cdot threshold_{i-1} + \beta \cdot ratio_i'$$
$$threshold_0 = ratio_0' \tag{6}$$

IV. Experiments

We have conducted a series of experiments to study how the anomaly detection algorithm performs under abnormal network traffic that may threat data centers. To have a broader coverage, three types of attacking traffic – UDP worms, TCP worms, and DDoS attacks – are used in our experiments.

We generate synthetic traffic by mixing the normal data center traffic (i.e., Yahoo! traffic traces [1]) with the worm or DDoS traffic. The attack traffic traces are collected from real-world worm cases or generated by the virus traffic generator. To obtain abnormal "mixed" network traces, the attack traffic is blended into the Yahoo! network trace with the latter multiplied by 1000 to compensate the sampling attenuation. The mixed trace is subsequently replayed to the anomaly detection algorithm to evaluate its performance.

A. Evaluations with UDP worms

We evaluate our algorithm with the UDP worm traffic using Witty worm traces collected by CAIDA through UCSD Network Telescope [2]. Witty is a UDP worm that exploits a buffer overflow vulnerability in several ISS products. Once a host is infected with Witty worm, it starts to send 20,000 packets to randomly selected destination IPs from source port 4000 [13].

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

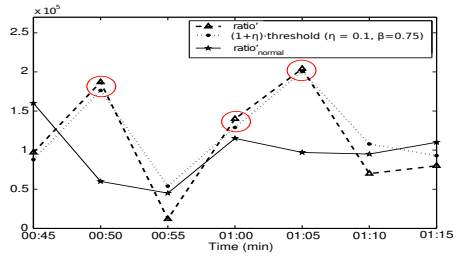


Fig. 4. Differentiate abnormal flows with Witty trace. The circles indicate successful detection of anomaly.

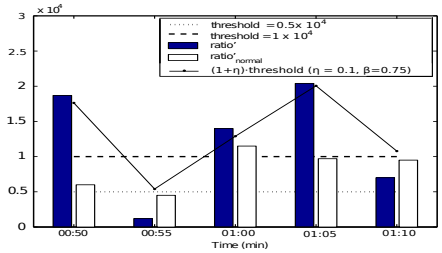
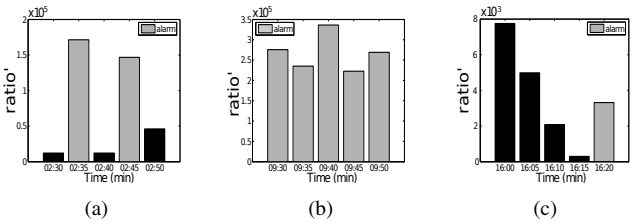


Fig. 5. Differentiate abnormal flows with Witty trace

We replay the traffic flows from Hong Kong data center from 00:00, and start to simulate Witty worm behavior at 00:15. We choose octet *B* for calculating the *center of mass*, and the anomaly detection runs every five minutes. As shown in Figure 4, in a number of periods, such as 00:50, 01:00, and 01:05, $ratio'$ is significantly larger than $ratio_{normal}$, successfully indicating abnormal traffic patterns.

The detection latency (the time between the start of the intrusion traffic and the generation of an alarm), is highly related to the value of threshold. Figure 5 compares three thresholds: the adaptive threshold ($\beta = 0.85$) and two other fixed thresholds. Small threshold improves the sensitivity, but increases the possibility of introducing false positives, e.g., it causes false positive at 00:50 and 00:55 when $threshold = 0.5 \times 10^4$. On the other hand, a larger threshold is likely to reduce the sensitivity of detection. There is a trade-off between sensitivity and precision.

To verify that our approach is resilient to temporal fluctuations, we perform a series of experiments in several time periods of a day. We choose three representative time periods with diverse traffic patterns, and use adaptive threshold ($\beta = 0.85$) as a criteria, when ratio is larger than adaptive threshold, an alarm is generated. As shown in Figure 6, our algorithms can effectively detect anomalies within 5 minutes for most of the time periods. For the case of Figure 6(c), where N_{norm} only contributes to 2% of the total traffic, our algorithms can still detect anomalies within 20 minutes.



(a) (b) (c)

Fig. 6. Detecting attacks in various time periods during the day

B. Evaluations with TCP worms

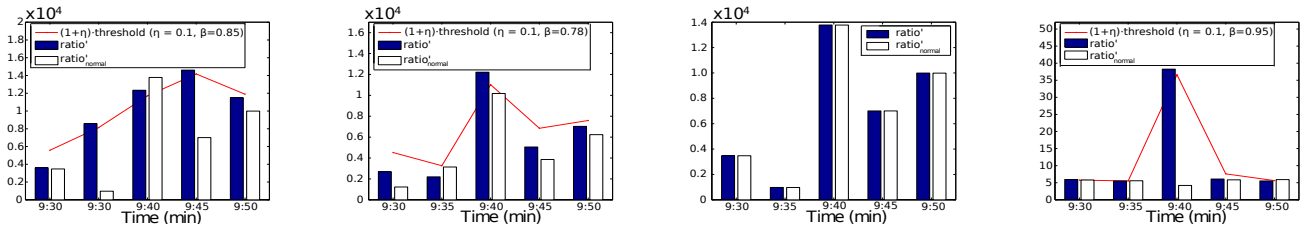
Next, we evaluate our approach with synthetic traffic of normal data center traffic and simulated TCP worms. To obtain TCP worm trace, we use a known worm traffic generation program on Georgia Tech Network Simulator (GTNetS) [3], and generate worm traffic that reflects a combination of CodeRedI and CodeRedII worms. We construct a topology consisting of over 83,400 nodes in two /16 subnets.

With this experiment setting, we consider three cases to evaluate the effectiveness of our anomaly detection method, with each case emulating one real-world scenario: *case 1*: worm packets in /16 subnets that contain the initial infection node, corresponding to the real-world case that one node in a data center has been infected; *case 2*: worm packets in /24 subnets that do not contain the initial infection node, while the /16 subnets that contain this /24 subnets do. This case emulates an infected node outside a data center, but the "distance" is not far away (in the same /16 network); *case 3*: worm packets in /16 subnets that do not contain the initial infection node. This is the most difficult case, emulating the situation when the infection is far away from the data center (in a different /16 network).

Figure 7(a) shows the experimental results for *case 1*. Since octets *A* and *B* are all the same in one /16 network, we choose octet *D* to calculate *centroid*. In this experiment, we take only 1/8 of total packets, assuming that the packets communicating within a same /16 subnet may not go through its main router, and only 1/8 of them go across the main router and has been recorded. The result shows that our approach can detect intrusion in 5 minutes when the worm began to attack from 09:30. We can see that our approach is effective in this case.

In *case 2*, the packets are collected by the edge router for a /24 subnet. Hence, we randomly choose a /24 subnet from the /16 subnet. As the quantity of worm packets has decreased significantly, the $ratio'$ becomes closer to $ratio_{normal}$ in average, so it becomes more difficult to detect intrusion. However, our approach is still effective to detect intrusion. For example in Figure 7(b), detection latency is only 15 minutes. For *case 2*, the average $\frac{N_{worm}}{N_{normal}+N_{worm}}$ is about 1.2%.

To simulate *case 3*, we treat nodes with the same octets *C* and *D* within a same /16 subnet as a /16 subnet with IP addresses as *C.D*.x.y. In fact, this case is more challenging than the real-world scenarios we are simulating, since we treat every /16 subnet as one single node, which decreases the number of infection packets. Since this case examines the anomaly detection when infected sources are relatively far away, it is less time-constrained to detect the intrusion. When the infection proliferates to the same /24 subnet or /16 network, the intrusion would be effectively detected. Nevertheless, slight improvement in our algorithm by adding:



(a) *case 1* (b) *case 2* (c) *case 3* (d) *case 3* after improvement of algorithm

Fig. 7. Detecting TCP attacks

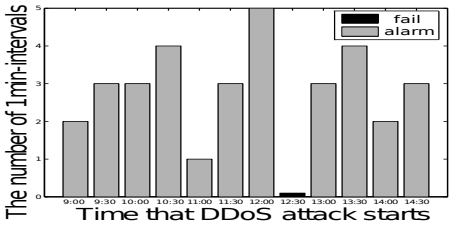


Fig. 8. Latency at different time when detecting DDoS attack

if $N'_x[i] >= N'_x[T], W_x[i]' = log(W_x[i])$ in step 9 (Algorithm 1) will further reduce the impact of dominant buckets on *centroid*. Hence, this extra step enables improved algorithm to detect even *case 3* when the average $\frac{N_{worm}}{N_{normal}+N_{worm}}$ is about 0.078%, as shown in Figure 7(d).

C. Evaluations with DDoS traffic

The attack traces we used are captured at Los Nettos, a moderate size ISP located in Los Angeles [7]. DDoS attacks have generated large quantities of infection packets in short time, and it is hence relatively easy to detect. Thus, the focus of experiments with DDoS traffic is to study how fast our approach can detect DDoS attack. If our approach cannot detect in five time intervals, we treat it as a failure to detect intrusion. Figure 8 shows that our approach is valid to detect DDoS attack within 5 minutes during most of time in the day.

In summary, the experiments results demonstrate that our algorithm is generally effective to detect a variety of attacks in a short time resilient to temporal fluctuations.

V. Conclusions and Future Work

This paper presented a simple yet effective anomaly detection algorithm based on an IP-structural approach for large data centers. Using the *centroid* analysis on the octets of IP addresses in data center traffic, we find that the octets in both source and destination IP addresses exhibit a strong concentrating distribution. More importantly, such concentrations are very stable during normal traffic patterns, but deviate noticeably under abnormal patterns caused by anomalous events such as worm outbreaks or denial of service attacks. Inspired by this interesting and unique traffic characteristic in data center traffic, we devise an effective algorithm to detect anomalous situations based on the change or movement on the *centroid* of some octets in IP addresses. We evaluate this algorithm with synthetic traffic that combines normal data center traffic and intrusion traffic. Our experiment results show that the approach can detect the intrusions under test even when worm traffic flows account for as low as 0.078% of the total traffic volume.

VI. Acknowledgement

We would like to thank Yahoo! Research for providing us Webscope data and support. Also, the work is supported in part by an Arizona State University New College SRCA grant, and HKUST grants DAG08/09.EG11 and REC09/10.EG06.

References

[1] Yahoo! Research and Academic Relations. G4: Yahoo! network flows data 1.0. http://research.yahoo.com/Academic_Relations.

[2] The CAIDA Dataset on the Witty Worm-March 19-24, 2004, Colleen Shannon, David Moore, and kc claffy. http://www.caida.org/data/passive/witty_worm_dataset.xml.

[3] GTNetS http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, Feb 2009.

[5] CNet News. DDoS attack hobbles major sites, including Amazon. *http://news.cnet.com/8301-30684_3-10421577-265.html*, Dec. 2009.

[6] R. Fontugne, T. Hirotsu, and K. Fukuda. An image processing approach to traffic anomaly detection. In *Proc. of the 4th Asian Conf. on Internet Engineering (AINTEC'08)*, pages 17–26, 2008.

[7] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proc. of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03)*, pages 99–110, 2003.

[8] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of the 2004 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04)*, pages 219–230, 2004.

[9] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proc. of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'05)*, pages 217–228, 2005.

[10] V. Paxson. Bro: a system for detecting network intruders in real-time. In *Proc. of the 7th Conf. on USENIX Security Symposium (SSYM'98)*, pages 3–3, 1998.

[11] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud! Exploring information leakage in third-party compute clouds. In *Proc. of the* 16^{th} *ACM Conf. on Computer and Communication Security (CCS'09)*, Nov. 2009.

[12] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proc. of the 13th USENIX conference on System administration (LISA'99)*, pages 229–238, 1999.

[13] C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.

[14] Wall Street Journal. Google introduces new security measures after cyber-attack. *http://online.wsj.com/article/BT-CO-20100112-716798.html*, Jan. 2009.