SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks

Chenxu Wang, Tony T. N. Miu, Xiapu Luo, and Jinhe Wang

***Abstract*— Application layer distributed denial of service (DDoS) attacks have become a severe threat to the security of web servers. These attacks evade most intrusion prevention systems by sending numerous benign HTTP requests. Since most of these attacks are launched abruptly and severely, a fast intrusion prevention system is desirable to detect and mitigate these attacks as soon as possible. In this paper, we propose an effective defense system, named SkyShield, which leverages the sketch data structure to quickly detect and mitigate application layer DDoS attacks. First, we propose a novel calculation of the divergence between two sketches, which alleviates the impact of network dynamics and improves the detection accuracy. Second, we utilize the abnormal sketch to facilitate the identification of malicious hosts of an ongoing attack. This improves the efficiency of SkyShield by avoiding the reverse calculation of malicious hosts. We have developed a prototype of SkyShield and carefully evaluated its effectiveness using real attack data collected from a large-scale web cluster. The experimental results show that SkyShield can quickly reduce malicious requests, while posing a limited impact on normal users.**

***Index Terms*— Application layer DDoS attacks, sketch data structure, intrusion prevention system.**

I. Introduction

DISTRIBUTED denial of service (DDoS) attacks have been a severe threat to the Internet for decades and numerous defense schemes have been proposed to mitigate DDoS attacks at the network layer [1]–[8]. Recently, application layer DDoS (app-layer DDoS) attacks against web servers grow rapidly and bring victims with great revenue losses [9]–[12]. App-layer DDoS attacks attempt to disrupt legitimate access to application services by masquerading flash crowds with numerous benign requests. Flash crowd refers to the situation when many users simultaneously access a popular website, producing a surge in traffic to the website and causing the site to be virtually unreachable [13]. The stealthiness of app-layer DDoS attacks makes most signature-based intrusion prevention systems ineffective.

Since most DDoS attacks are launched abruptly and severely, it is desirable to design a defense system that can detect and mitigate app-layer DDoS attacks as soon as possible to minimize the losses [14]. Turing test schemes based on graphical puzzles have been proposed to address the above problem on the cost of additional delays [15], [16]. Unfortunately, since a few milliseconds extra delay may cause users to abandon a web page early [17], [18], applying such mechanism to all users will negatively affect the Quality of Experience (QoE). Therefore, an effective defense system should mitigate app-layer DDoS attacks as soon as possible while posing a limited impact on the access of normal users.

The increasingly high-speed network links demand an efficient data structure to process a huge volume of network traffic efficiently, especially under HTTP flooding attacks. The sketch data structure can efficiently estimate the original signals by aggregating high dimensional data streams into fewer dimensions, making it very suitable for DDoS attack detection [19]. A series of sketch-based approaches have been proposed for anomaly detection in large-scale network traffic [20]–[25]. Since sketches contain no direct information about the malicious hosts, they cannot be directly used for the mitigation of attacks. To tackle this problem, several efficient reverse hashing schemes have been proposed to infer the IP addresses of malicious hosts from reversible sketches [22], [26], [27]. These studies attempt to retrieve the anomalous keys either by using reverse hashing methods or by storing parts of the keys. However, these methods are either computation-intensive or storage-demanding, limiting their applications in intrusion prevention systems.

The challenge of designing a sketch-based defense system lies in the coordination between the detection and mitigation of attacks. First, since network traffic is inherently dynamic in the real environment, a proper measure of network traffic is essential to accurate anomaly detection. Second, when an attack occurs, it needs to identify malicious hosts accurately

Manuscript received February 14, 2017; revised August 12, 2017; accepted September 16, 2017. Date of publication October 2, 2017; date of current version December 19, 2017. This work was supported in part by Hong Kong ITF under Grant UIM/285, in part by the Hong Kong GRF under Grant PolyU 5389/13E, in part by the Shenzhen City Science and Technology Research and Development Fund under Grant JCYJ20150630115257892, in part by the National Natural Science Foundation of China under Grant 61602370, Grant 61672026, Grant 61772411, in part by the Postdoctoral Foundation of China under Grant 201659M2806, in part by the Fundamental Research Funds for the Central Universities of China under Grant 1191320006, in part by the 111 International Collaboration Program of China, and in part by the Natural Science Foundation of Shaanxi Province under Grant 2016JM6040. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Guevara Noubir. *(Chenxu Wang and Tony T. N. Miu contributed equally to this work.) (Corresponding authors: Chenxu Wang; Xiapu Luo.)*

C. Wang is with the School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and also with the MoE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: cxwang@mail.xjtu.edu.cn).

T. T. N. Miu is with Nexusguard Ltd., Hong Kong.

X. Luo is with the Department of Computing, The Hong Kong Polytechnic University, and also with the Shenzhen Research Institute, The Hong Kong Polytechnic University Hong Kong.

J. Wang is with the School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2758754

without affecting the access of legitimate users. Third, since most attacks are launched abruptly, the defense system should adopt an effective light-weight process to detect and mitigate the attack as soon as possible.

To address these issues, in this paper, we propose SkyShield, a novel defense system based on sketches to defend against app-layer DDoS attacks. Akin to previous studies, SkyShield exploits the random aggregation property of sketches to improve its capability. Differently, SkyShield mitigates the attacks without retrieving the exact IP addresses of malicious hosts, and thus avoids the intensive computation process. The rationality of this scheme is the fact that attacks are usually persistent. Therefore, the abnormal sketch could be reused to facilitate the identification of malicious hosts by examining whether an incoming host is the one that caused the anomaly in the previous detection cycle. This avoids the reverse calculation of malicious hosts and thus greatly improves the efficiency of the system. In summary, we make the following major contributions in this paper:

- We propose a sketch-based anomaly detection scheme for app-layer DDoS attacks. The scheme utilizes the divergence of sketches in two consecutive detection cycles to detect the occurrence of an anomaly. We design a new variant of Hellinger Distance [28] to measure the divergence for mitigating the impact of network dynamics.
- We design an effective attack mitigation scheme without the need of reverse calculation or storage of malicious hosts. The scheme exploits the detected abnormal sketch to identify malicious hosts directly. This avoids the computation-intensive process to infer the malicious hosts and thus greatly improves the efficiency of the system.
- We develop an adaptive mitigation scheme which dynamically determines the number of malicious hosts according to the volume of requests. Basically, it will discriminate more hosts as suspicious if the load of the server is heavy and regard fewer hosts as suspicious if the load is moderate and tolerable. This scheme well balances the trade-off between attack mitigation speed and accuracy.
- We develop a prototype of SkyShield and evaluate its effectiveness using real attack data collected from a large-scale web server cluster. The experimental results show that SkyShield can quickly mitigate app-layer DDoS attacks with a limited impact on legitimate users.

The rest of the paper is organized as follows. Section II introduces the background knowledge. Following an overview of the system in Section III, we detail the mitigation and detection schemes in Section IV and V, respectively. Section VI describes the implementation of SkyShield. Section VII presents the evaluation results. Section IX reviews relevant literature. After a brief discussion in Section VIII, we conclude the work in Section X.

II. Background

A. Sketches

A sketch is a type of data structure composed of H hash tables of size K. It is used to efficiently estimate the original signals by aggregating high dimensional data streams into fewer dimensions. Fig. 1 shows a diagram of the sketch data structure. The incoming data stream is composed of pairwise items encompassing a key and an associated $value$. Each row of the sketch is associated with an independent hash function to index the incoming keys. When a pairwise item $(key, value)$ arrives, the data in the bucket corresponding to the key is updated by the $value$.

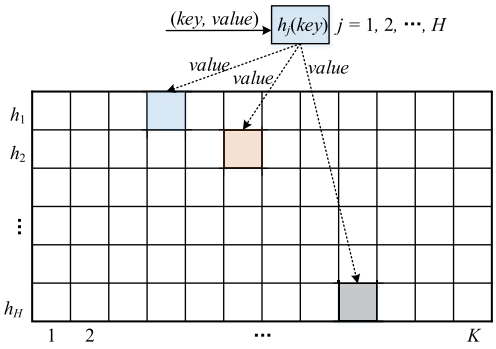


Fig. 1. A diagram of the sketch data structure.

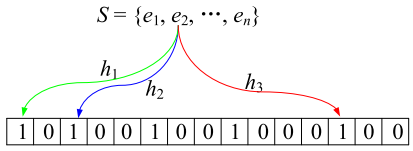


Fig. 2. A diagram of the Bloom filter data structure.

A sketch is an approximation tool to efficiently estimate a signal by sacrificing tolerable accuracy. Due to the randomization of hash functions, the distribution of values in each hash table is relatively stable for normal network traffic. Therefore, sketches are capable of detecting significant changes in massive data streams, such as high-volume network traffic [20]. SkyShield adopts the sketch techniques for anomaly detection and malicious hosts identification.

B. Bloom Filters

A Bloom filter is a space-efficient data structure for set membership queries. It employs an array of m bits to represent a set. Fig. 2 illustrates a diagram of the Bloom filter data structure. A Bloom filter employs k independent hash functions $h_1, h_2, \cdots, h_k$ with a range $\{1, 2, \cdots, m\}$ to represent a set $S = \{e_1, e_2, \cdots, e_n\}$. For each element $e \in S$, the bit at the location indicated by $h_i(e)$, $1 \leq i \leq k$ are set to 1. A bit can be set to 1 for multiple times. To query if an element e is in the set S, the bits at locations indicated by $h_i(e)$, $1 \leq i \leq k$ are checked. The element is supposed to be in the set with high probability if all checked bits equal 1 and not if otherwise.

A Bloom filter may also lead to a negligible false positive rate. A false positive means that an element being checked is mistakenly determined by the above criteria whereas it is actually not in the set. It is caused by conflicts of keys that occasionally share common hashing results for all hash functions. The false positive rate can be decreased by carefully adjusting the number of hash functions based on the cardinal of the set and the size of the bit-array [29]. SkyShield employs Bloom filters to implement the black and white lists that are utilized to filter requests from malicious hosts.

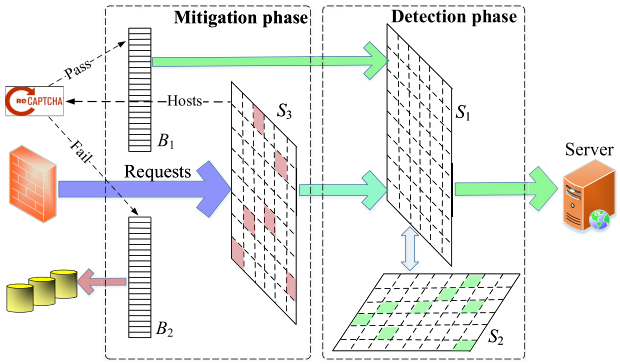


Fig. 3. The process of SkyShield.

C. Comparison Between Sketches and Bloom Filters

The main difference between sketches and Bloom filters is their data representations, which are determined by their different applications. Sketches are used to store a summary of large-scale data streams in situations where it is costly to store the whole data. The storage unit of a sketch is determined by the types of values to be stored (e.g. an unsigned integer for frequency count). By contrast, Bloom filters are used for efficient set membership queries. The storage unit is one bit and multiple true-value bits combined together indicate the existence of an element in the set with high probability.

The common of the two data structures lies in the usage of hash functions. The principle behind both data structures is the power of randomness that stems from hashing algorithms. Specifically, we employ sketches to detect the occurrences of attacks and Bloom filters to serve as black and white lists. Both of them have IP addresses as input keys.

III. System Overview

Fig. 3 depicts the process of SkyShield. It is deployed behind a network firewall that will filter out malformed HTTP requests, and the process consists of two phases, namely, mitigation and detection.

In the mitigation phase, SkyShield employs two Bloom filters, including a whitelist (B_1) and a blacklist (B_2) to filter incoming requests. The whitelist (resp. blacklist) contains the legitimate (resp. malicious) hosts that are confirmed by the CAPTCHA techniques. Normal requests verified by the whitelist are passed to the detection phase directly whereas malicious requests verified by the blacklist are filtered and logged. The remaining requests are inspected based on the abnormal sketch S_3. The rationality behind this scheme is that malicious hosts need to send numerous requests persistently to launch effective app-layer DDoS attacks. Therefore, SkyShield can identify malicious hosts without reversely calculating their IP addresses. Detailed mitigation method is described in Section IV.

For a suspicious request, SkyShield first examines whether its origin is in the whitelist. If not, the host will be checked by the CAPTCHA module. If the host passes the CAPTCHA test, it will be added to the whitelist. Otherwise, it will be added to the blacklist. Since only suspicious hosts are tested by the CAPTCHA, only parts of legitimate users might be affected. Additionally, to prevent blacklisted users from being blocked forever, both the blacklist and whitelist are emptied periodically. Initially, B_1, B_2 and S_3 are set to be empty and no hosts are suspected and filtered.

In the detection phase, SkyShield exploits the divergence between two sketches S_1 and S_2 as a signal to detect anomalies that are caused by numerous requests originated from malicious hosts. SkyShield conducts the detection cyclically with a fixed time interval ΔT, which is an adjustable parameter. By adjusting ΔT, the system can balance the trade-off between the attack mitigation speed and the detection accuracy. In each detection cycle, all incoming requests are aggregated into S_1, with the source IP addresses as input keys. The backup sketch S_2 stores the results of S_1 in the last normal detection cycle. At the end of each detection cycle, the divergence $d(S_1, S_2)$ between S_1 and S_2 is calculated. If $d(S_1, S_2)$ exceeds a threshold θ_t, the system is supposed to suffer an attack and an alarm is raised. If an anomaly is detected, S_2 will not be updated anymore. This guarantees that the current sketch is always compared with a normal pattern. Alternatively, S_3 will be updated by S_1 and the abnormal buckets are calculated. When the alarm is lifted, S_2 will be updated by S_1 again at the end of each detection cycle and S_3 is emptied. The detection method is detailed in Section V.

IV. Attack Mitigation

A. Locating Abnormal Buckets

When an anomaly is detected, SkyShield needs to locate the abnormal buckets that cause the sharp change in the divergence between S_1 and S_2. Since incoming requests are mapped to every row of a sketch, the abnormal buckets for every row of the sketch are calculated.

SkyShield classifies the top g buckets with the largest request volumes as abnormal ones. Since attackers usually employ more compromised hosts (or bots) to generate a larger volume of requests in a short time, we define the value of g as a function of the volume of the total requests. Denote the i-th row of S_3 as a vector $\langle n_{i1}, n_{i2}, \ldots, n_{iK} \rangle$, where n_{ij} is the value of the j-th bucket in the i-th row. Let $N_i = \sum_{j=1}^{K} n_{ij}$ represent the volume of total requests in a detection cycle. Then the value of g is defined as:

$$g = \lfloor (\ln N_i)^r \rfloor, \tag{1}$$

where r is an adjustable parameter. When g is calculated, the indexes of abnormal buckets in the i-th row are obtained as:

$$A_i = \{j | n_{ij} \geq n'_{ig}\}, \tag{2}$$

where n'_{ig} is the g-th largest value in v_i. The abnormal buckets in other rows are obtained similarly.

B. Identifying Malicious Hosts

SkyShield further employs these obtained abnormal buckets to identify malicious hosts. We denote the abnormal buckets set for row i as A_i and the bucket index of a specific IP address in row i as $h_i(\texttt{IP})$, respectively. If $h_i(\texttt{IP}) \in A_i$ is true for all $i = 1, 2, \ldots, H$, then we label the IP address as suspicious and the host will be verified by the CAPTCHA module.

Fig. 4 demonstrates the flow chart of the identification procedure. The mitigation module first checks if the host

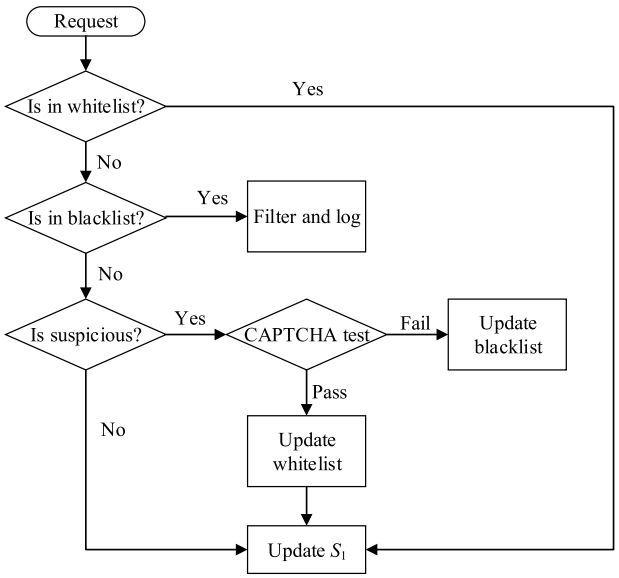


Fig. 4. The flow chart of the identification procedure.

is in the whitelist or blacklist. If not, the host is examined against S_3. A suspicious host is then tested by a graphical puzzle. If the host passes the test, it is added to the whitelist and the associated requests are passed to the detection phase. Otherwise, the corresponding host is added to the blacklist and the associated requests are filtered and logged. The employment of the whitelist guarantees that legitimate users will not be checked by the CAPTCHA module repeatedly. Since the compromised hosts may become normal and the legitimate hosts may be controlled by attackers some time later, we clean the blacklist and the whitelist periodically with a longer interval than the detection cycle. In this paper, we set $100\Delta T$ (i.e., 2000s) as the default clean period. This value is selected because it has been reported that 80% of the DDoS attack intervals last less than 1081 seconds [14]. It is worth noting that the cleaning of the blacklist and whitelist will not affect the detection accuracy because the clean period is much longer than the detection cycle. Even if an attack lasts longer than the clean period, SkyShield will recapture the attacking hosts, just like how it does at the start of a new attack. We can also choose a much longer clean period (e.g., a day or even a week) in order to block the malicious hosts longer.

The rationality behind the mitigation scheme is that a malicious host detected in current detection cycle has a high probability to appear in the next detection cycle. An attacker may be able to evade the system by violating this assumption. However, such attempts will greatly increase the cost of an effective attack. (see detailed discussion in Section VIII-B).

V. Anomaly Detection

A. Divergence of Sketches

We employ the divergence between S_1 and S_2 as a signal to detect the occurrence of an attack. This metric is selected according to the observation that the distribution of bucket values in a sketch for normal network traffic is steady. It is worth noting that such steadiness does not contradict with the implication of network dynamics that describes the uncertainty of individual requests, including the source IP address, request time, body size, etc. Although the aggregate network traffic of all normal users is often steady in a short time scale, the request rates of individual users may have large variations. Since sketches map the request rates of individual users into single buckets, network dynamics will influence the measure of divergence between sketches.

To mitigate the impact of network dynamics, we propose a novel calculation of sketch divergences. We denote the i-th row of a sketch as a vector $\langle n_{i1}, n_{i2}, \ldots, n_{iK}\rangle$, where n_{ij} is the value of the j-th bucket in the i-th row of a sketch. Let $N_i = \sum_{j=1}^{K} n_{ij}$ represent the volume of all requests. We define a probability vector as $P_i = \langle p_{i1}, p_{i2}, \ldots, p_{iK}\rangle$ for the corresponding row, where $p_{ij} = n_{ij}/N_i$ measures the probability that an incoming request is mapped into the j-th bucket by the i-th hash function. The divergence between two probability vectors could be measured by their Hellinger Distance (HD). Given two discrete probability vectors $P_i = \langle p_{i1}, p_{i2}, \ldots, p_{iK}\rangle$ and $Q_i = \langle q_{i1}, q_{i2}, \ldots, q_{iK}\rangle$, the Hellinger distance is defined as

$$d(P_i, Q_i) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{K}(\sqrt{p_{ij}} - \sqrt{q_{ij}})^2}. \tag{3}$$

However, it is improper to use such distance to measure the divergence between two sketches, because the uncertainty of source IP addresses in the dynamic network traffic makes the probability distribution in each row of a sketch vary dramatically and consequently the unpredictable divergences will result in high a false positive rate. What's worse, since the last normal sketch is stored as a baseline of normality when an anomaly is detected, using the original Hellinger distance between two probability vectors may lead to continuous false alarms even when the actual attack has stopped.

To solve this problem, we propose computing the Hellinger distance of two *sorted* probability vectors instead of the original vectors. Specifically, we first sort the probability vectors in a descending order, and then we have $P_i' = \langle p_{i(1)}, p_{i(2)}, \ldots, p_{i(K)}\rangle$ and $Q_i' = \langle q_{i(1)}, q_{i(2)}, \ldots, q_{i(K)}\rangle$, where $p_{i(1)} \geq p_{i(2)} \geq \cdots \geq p_{i(K)}$ and $q_{i(1)} \geq q_{i(2)} \geq \cdots \geq q_{i(K)}$. The improved divergence between P_i and Q_i is calculated by the Hellinger distance between P_i' and Q_i':

$$\begin{aligned} d'(P_i, Q_i) &= d(P_i', Q_i') \\ &= \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{K}(\sqrt{p_{i(j)}} - \sqrt{q_{i(j)}})^2}. \end{aligned} \tag{4}$$

According to Equation 4, we can calculate the divergence between the corresponding hash tables of two sketches.

Fig. 5 shows the comparisons between the original and the improved Hellinger distances of a hash table for one of our datasets. We can see that the improved divergence is much lower and more stable than the original one. This benefits from the randomness of hashing algorithms and the sorted probability distributions of hash tables are more predictable for normal network traffic.

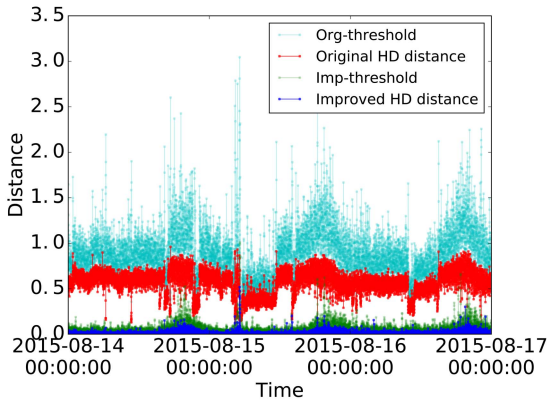


Fig. 5. Different divergences of sketches versus the time.

B. Dynamic Threshold Estimation

Since network traffic is inherently dynamic and fluctuates persistently in long time scales, using a constant threshold for detection will result in many false positives. Hence, we employ the Exponential Weighted Moving Average (EWMA) method [30] to obtain an adaptive threshold. However, if we use the averaged divergence of hash tables in the two sketches for detection, a large deviation of any individual hash table pair may cause a large deviation of the average divergence, which leads to high false alarm rates. Practically, we prefer to a conservative criterion. Therefore, we propose a Multiple Independent Exponential Weighted Moving Average (MIEWMA) method to compensate the fluctuations caused by individual hash tables. Specifically, denote $d_t^{(i)}$ as the calculated divergence between the i-th hash tables in S_1 and S_2 at time t, $\hat{d}_t^{(i)}$ as the estimated divergence for time t according to historical observations, and $\hat{d}_{t+1}^{(i)}$ as the estimated divergence for time $t+1$. Then we have

$$\hat{d}_{t+1}^{(i)} = \alpha d_t^{(i)} + (1-\alpha)\hat{d}_t^{(i)}, \tag{5}$$

$$e_t = |\hat{d}_t^{(i)} - d_t^{(i)}|, \tag{6}$$

$$\sigma_{t+1}^2 = \beta e_t^2 + (1-\beta)\sigma_t^2, \tag{7}$$

$$\theta_{t+1}^{(i)} = \hat{d}_{t+1}^{(i)} + \lambda\sigma_{t+1}, \tag{8}$$

where $\theta_{t+1}^{(i)}$ is the calculated threshold at time $t+1$ for the i-th hash table, α, β, and λ are adjustable parameters. Since the app-layer DDoS attacks will overwhelm the victim server in seconds with numerous malicious requests, it will greatly disturb the probability vectors of S_1. Therefore, if $d_{t+1}^{(i)} > \theta_{t+1}^{(i)}$ for all $i = 1, 2, \ldots, H$, then an alarm is raised. Fig. 5 also shows the obtained thresholds of a hash table for different calculations of sketch divergences. We can see that the new definition of Hellinger distance derives more stable thresholds than the original one.

When an alarm is raised, SkyShield performs two actions to protect the normal baseline. On one hand, S_2 will not be updated by S_1, thus the sketch of the next detection cycle will always be compared with the normal pattern. On the other hand, the threshold will not be updated until the alarm is lifted. This protects the threshold from the impact of attacks.

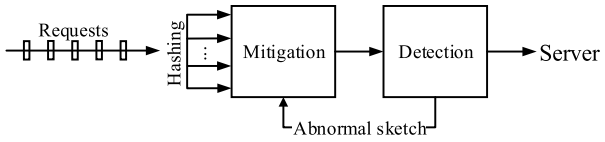


Fig. 6. The implementation architecture of SkyShield.

VI. Implementation

A. Architecture

Fig. 6 shows the implementation architecture of SkyShield. We adopt parallel techniques to hash the hosts of incoming requests with multiple cores in order to improve the efficiency. When the detection module discovers an anomaly, it passes the abnormal sketch to the mitigation module. On receiving an abnormal sketch, the mitigation module identifies the malicious hosts and filters out all requests from these hosts. We have developed a prototype of SkyShield using Python 2.7 and evaluated it on a 64-bit Windows 10 system with an Intel(R) Core(TM)2 Quad CPU Q9550 @2.83GH and 8.0GB RAM.

B. Selection of Hash Functions

We use the modular hash functions which are randomly selected from a universal family $\mathbb{H} = \{h_{a,b}\}$, where $h_{a,b}$ is defined as

$$h_{a,b}(x) = (ax + b) \mod p, \tag{9}$$

where a, b are randomly selected positive integers, and p is the largest prime less than K and m for the sketches and Bloom filters, respectively. For example, for sketches with $K = 2^{12}$, we set $p = 4093$. For Bloom filters with $m = 2^{22}$, we set $p = 4,194,301$. This hashing method generates uniformly distributed values of input keys, and thus yields negligible impact on the accuracy of the system.

C. Parameter Configurations

Since the effectiveness of SkyShield depends on the proper configurations of the parameters, we provide guidelines for determining the proper configuration of the parameters.

1) Sketch's Parameters: A sketch has two parameters, namely the number of hash functions H and the hash table size K. We configure H and K to satisfy two constraints. The first constraint stems from the requirement of a low FPR. Let the total number of IP addresses be N and the numbers of abnormal buckets detected in each row of S_3 be $|A_1|, |A_2|, \ldots, |A_H|$, respectively. Then the FPR for completely random hashing is $N\frac{\prod_{i=1}^{i=H}|A_i|}{K^H}$ approximately, where $|A_i| \ll K$ is the cardinal of set A_i. Since we take the top g buckets with the largest number of requests as abnormal ones in each hash table, the ratio between the number of abnormal buckets for each row and the size of hash tables K is fixed. Denoting the fixed ratio by τ, we can obtain the false positive rate as $N\tau^H$, where $0 < \tau \ll 1$. For a given constant false positive rate bound $\epsilon \geq N\tau^H$, we have

$$H \geq \frac{\ln N - \ln \epsilon}{-\ln \tau} \tag{10}$$

For example, if $N = 2^{32}$, $\epsilon = 10^{-5}$, and $\tau = 0.01$, we have $H \geq 7.32$, and thus $H = 8$ is a suitable choice for the above condition. The second constraint is to choose a proper value of K such that the size of the sketch $H \times K$ is small enough to fit the sketch into fast memory in order for updates to be performed at high speed. In this paper we set $K = 2^{12}$ following [22]. With an unsigned integer as a bucket, each sketch consumes 64KB memory.

2) Bloom Filter's Parameters: A Bloom filter has two parameters, namely the number of hash functions k and the hash table size m. Given a set S containing n elements and a Bloom filter with parameters k and m, after all elements of S are hashed into the Bloom filter, the probability that a specific bit is still 0 is

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m} \tag{11}$$

The probability of a false positive is

$$f = (1-p)^k = \left(1 - e^{-kn/m}\right)^k \tag{12}$$

By minimizing f as a function of k, we obtain

$$k = \frac{m}{n} \ln 2 \tag{13}$$

Then, the false positive rate $f = (1/2)^k = (0.6185)^{m/n}$. Above analyses show that the false positive can be significantly reduced by a sufficiently large m, which consumes more memory storage. Given a positive rate $\epsilon > f$, we have

$$f = (1/2)^k = (1/2)^{\frac{m}{n} \ln 2} \leq \epsilon \tag{14}$$

Solving the above inequality we get

$$m \geq \frac{n \log_2(1/\epsilon)}{\ln 2} = n \log_2 e \log_2(1/\epsilon) \tag{15}$$

According to the fact that most botnets have tens of thousands of compromised hosts, we assume that $n = 10^5$. Given a false positive rate $\epsilon = 10^{-5}$, we have that $m \geq n \log_2 e \cdot \log_2(1/\epsilon) \approx 2396264.6$. Therefore, $m = 2^{22}$ is a proper choice. Since $k = \frac{m}{n} \ln 2 \approx 14.5$, we set $k = 15$. The parameters are also applicable to the whitelist Bloom filter. Each Bloom filter consumes about 16KB memory.

In practice, the intrinsic FPR of sketches and Bloom filters are relatively small compared to the FPR caused by the detection mechanism of the system itself. For instance, the intrinsic FPR of a sketch with parameters $H = 8$ and $K = 2^{12}$ is expected to be two or three orders of magnitude smaller than that caused by the detection system (10^{-5} vs. 10^{-2}).

3) MIEWMA's Parameters: Each EWMA method has three parameters, namely the damping coefficient α, the variance damping coefficient β, and the threshold damping coefficient λ. The parameter α determines the memory of the EWMA model (i.e., the weight of "elder" data in the calculation of EWMA). A larger α implies that the most recent data is more important in the estimate of next values. The value of α is suggested between 0.2 and 0.3 [31]. The parameter β is used to smooth the estimates of the variances and the value is suggested to be smaller than 0.5 [31].

TABLE I
DEFAULT VALUES OF THE PARAMETERS IN SKYSHIELD

	Parameters	Description	Default
Sketch	H	Number of hash functions	8
	K	Size of hash tables	2^{14}
Bloom filter	k	Number of hash functions	15
	m	Size of Bloom filters	2^{22}
EWMA	α	Damping coefficient	0.3
	β	Variance Damping coefficient	0.4
	λ	Threshold damping coefficient	6.0
Others	ΔT	Detection time intervals	20s
	r	An intermediate parameter	2.0

Different from the parameters α and β, the parameter λ directly influences the obtained thresholds. Therefore, the value of λ is more important than that of α and β. It is suggested that the value of λ is either set to 3 or 1.96 in order to obtain the X-sigma control limits. We evaluate the impact of λ in a much wider range as we prefer a lower FPR.

4) Other Parameters: There are two other important parameters, namely the detection time interval ΔT and the parameter r that determines the number of abnormal buckets g in a hash table. The parameter ΔT determines the response time of the system. Small ΔT empowers fast detection of attacks on the cost of frequent divergence calculations between S_1 and S_2. However, this may result in high computation consumption. In addition, short detection intervals may also result in insufficient statistics of network traffic and thus increases the false alarm rate.

We employ the parameter r and use Equation (1) to dynamically adjust the number of abnormal buckets in a hash table. The parameter r determines the number of abnormal buckets to be selected when an anomaly is detected. It influences the true positive rate (TPR), false positive rate (FPR), and the detection accuracy in the following detection cycles. Since the value of g is much smaller than the hash table size K, the impact of the parameter r is similar to that of g. To lower the FPR, we can set the number of abnormal buckets g to be sufficiently small by decreasing the value of r. However, it may increase the number of false negatives. When we increase the value of g, the probability of falsely discriminating a legitimate host increases. Moreover, when the value of K and the detection interval ΔT are fixed, a small g will lead to a long mitigation phase as we only filter out a limited number of malicious hosts in a single detection cycle. We empirically determine the values of ΔT and r based on the collected datasets. Table I summarizes the parameters of SkyShield and their default values, which are optimized by empirical experiments.

VII. EXPERIMENTS

In this section, we first describe the collection of datasets, and then report the extensive evaluation results of SkyShield using the real datasets.

A. Datasets

We collect the datasets from a large-scale web cluster that manages the traffic of about 200 customer websites. Fig. 7 illustrates the architecture of the cluster that employs

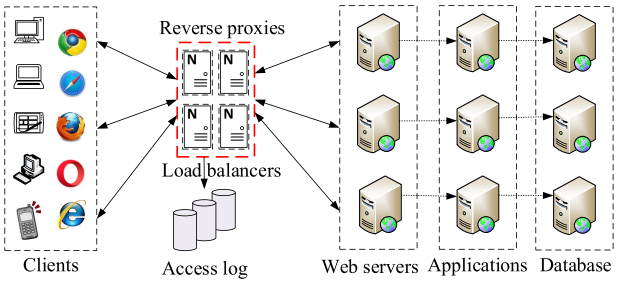


Fig. 7. The architecture of the reverse proxy cluster.

TABLE II
SUMMARY OF THE DATASETS

Dataset	# of requests	# of hosts	# of mal-hosts
Dataset1508	84,992,781	329,827	53,387
Dataset1603	131,778,807	408,119	92,446
Dataset1604	56,725,591	297,780	43,274

NGINX servers as reverse proxies to serve the customer websites. The load of reserve proxies is scheduled by IP-hash based balancers, and thus requests from a specific source IP address are processed by deterministic balancers. All processed requests are recorded in the access log by the balancers. The cluster also includes a CAPTCHA module to test whether an incoming request is malicious. To alleviate the impact on the QoE, the CAPTCHA module is configured working on a sampling mode and incoming requests are tested with the probability of 0.01. Requests that did not pass the test are recorded in the mitigation logs. Since attacking hosts usually send a lot of requests, the probability of testing a malicious host by the CAPTCHA module is very high. For instance, the probability of a host which sent more than 200 requests to be recorded in the mitigation logs is greater than $1-(1-0.01)^{200}=0.886$. Therefore, most of the attacking hosts are contained in the mitigation logs. We extracted the IP addresses in the mitigation logs and treat them as the ground truth of attacking hosts.

We obtained three datasets from the cluster, each of which contains three days of access logs with an attack reported in the middle day according to the reports of customers. We also extracted the mitigation logs in the corresponding periods of these datasets. In the following experiments, we refer to the dataset of dates from 2015/08/14 to 16 as Dataset1508, that of dates from 2016/03/16 to 18 as Dataset1603, that of dates from 2016/04/13 to 15 as Dataset1604, respectively. Table II presents a brief summary of the datasets. The second column lists the number of requests in each dataset. The number of the total requests are obtained by combining requests in the normal access logs with those in the mitigation logs. The large differences of the total requests are caused by seasonal tides and different attack traffic rates. The 3rd and 4th columns list the numbers of distinct hosts and distinct malicious hosts (mal-hosts), respectively.

B. Parameter Evaluation

Q1: How do the parameters impact the performance of SkyShield?

We employ the true positive rate (TPR), false positive rate (FPR), and the fraction of filtered malicious requests over the total attacking requests (Fraction) as criteria to evaluate the impact of parameters on the performance of SkyShield. True positive rate (TPR) measures the proportion of attacking hosts that are correctly identified as malicious by SkyShield. False positive rate (FPR) measures the proportion of benign hosts that are mistakenly identified as malicious by SkyShield. The impact of a parameter is evaluated by adjusting its value in a proper range while keeping others as their defaults.

1) Impact of MIEWMA's Parameters: Fig. 8 presents the evaluation results of α based on the three datasets. Each subfigure contains three lines, with the red representing the FPR, the black representing the TPR, and the blue representing the fraction of filtered malicious requests, respectively. It is shown that the parameter α has a negligible impact on the detection results because attacks usually result in sharp changes in the divergence between the normal and abnormal sketches and hence the detection results are not sensitive to the weights of history data in the calculation of the EWMA statistic. The evaluation results of the parameter β are shown in Fig. 9. According to the evaluation results, we set α to 0.3 and β to 0.4 as their defaults.

Fig. 10 demonstrates the evaluation results of λ. We can see that the detection results are sensitive to the change of λ. Both TPR and FPR decrease with the increase of λ. This is due to the fact that a larger λ leads to a higher threshold, which consequently results in a greater degree of tolerance to the change of divergence. Subsequently, the system becomes more conservative to raise an alarm and might let more real attacking hosts pass, leading to a lower TPR. Meanwhile, it will also give the green light to more benign hosts that might be mistakenly discriminated as malicious once an alarm is raised. This interprets the decreasing trends of FPR. In practice, the operator usually prefers to a lower FPR to maintain a high QoE providing that an attack does not impact the availability of service significantly. Hence, we select $\lambda=6.0$ as the default value in SkyShield.

2) Impact of ΔT and r: Fig. 11 shows the impact of ΔT. The results show that the FPR monotonically decreases with the increase of ΔT for all datasets. With the increase of ΔT, TPR first monotonically increases when $\Delta T \leq 20$s, and then decreases sharply when $\Delta T > 20$s. However, the fraction of filtered malicious is always significant and robust to the change of ΔT. Comprehensively taking all these factors into account, we select $\Delta T = 20$s as the default detection interval.

Fig. 12 depicts the evaluation results of r. It demonstrates that both TPR and FPR increase with the increase of r. With the increase of r, more buckets will be discriminated abnormal. Thus, more malicious hosts, as well as legitimate ones, are filtered out, leading to the increase of both TPR and FPR. As analyzed above, we would like to find a balance between a higher TRP and a lower FPR. According to the evaluation results, we select $r=2.0$ as its default in SkyShield.

It is worth noting that the fractions of filtered malicious requests are always above a significant percentage in all experiments. Therefore, SkyShield is effective in preventing

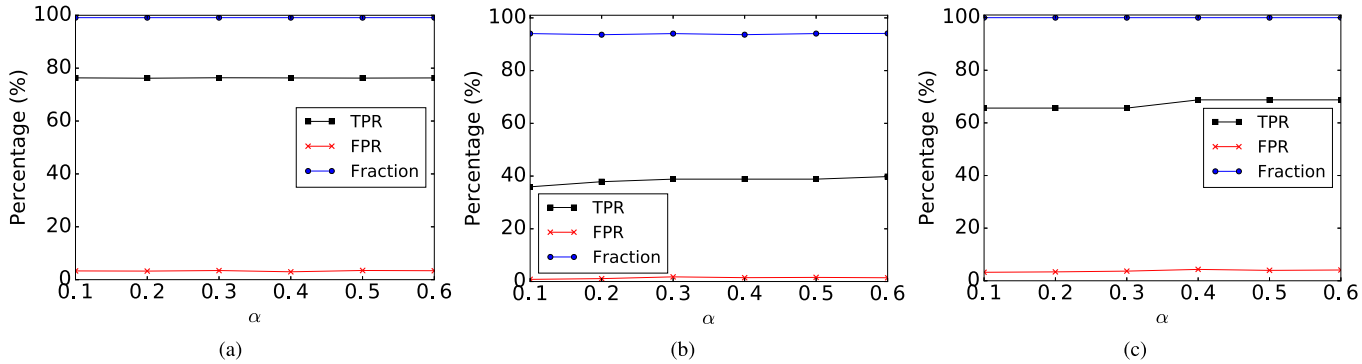


Fig. 8. Impact of parameter α. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

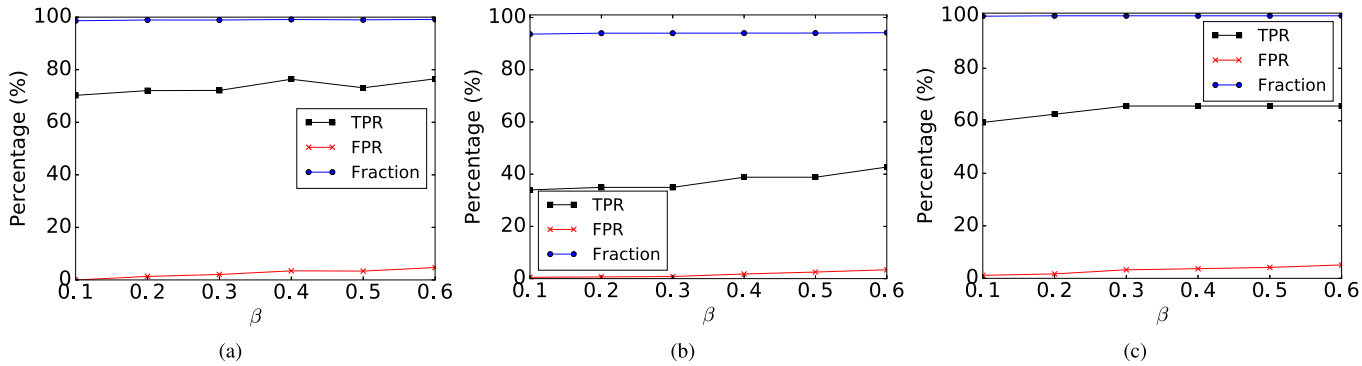


Fig. 9. Impact of parameter β. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

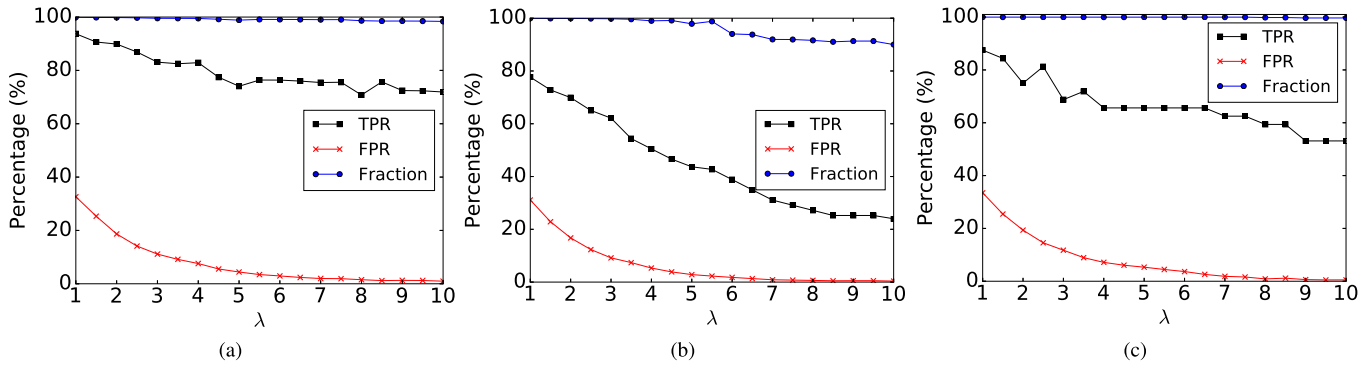


Fig. 10. Impact of parameter λ. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

the protected system from being overwhelmed by numerous malicious requests.

Summary: The configuration of system parameters has limited impact on the performance of SkyShield in terms of filtering out malicious requests.

C. Effectiveness Validation

Q2: How is the effectiveness of SkyShield in mitigating different kinds of DDoS attacks?

Since SkyShield aims at reducing the attacking traffic as soon as possible in the presence of an app-layer DDoS attack, we evaluate its effectiveness in filtering malicious requests. We simulate the real network environment by replaying the original request flows. The request rate is defined as the number of requests in a detection cycle. Fig. 13 to 15 present the request rates versus the time for the three datasets. In each subfigure, the blue line represents the original request rate and the red one represents the filtered request rate by SkyShield.

Fig. 13a shows that the cluster suffered from a flooding attack at 2015-08-15 04:21 and it is overwhelmed by a large volume of requests quickly. Fig. 13b illustrates the detail during the attacking period from 2015-08-15 04:00 to 06:00. We can see that the cluster suffered from five waves of attacks in this period. For all attack waves, SkyShield can reduce the overwhelming request rate to a reasonable level in about two or three detection cycles (i.e., less than 1 minute). Fig. 13c displays the detail at the start of the Dataset1508. The filtered

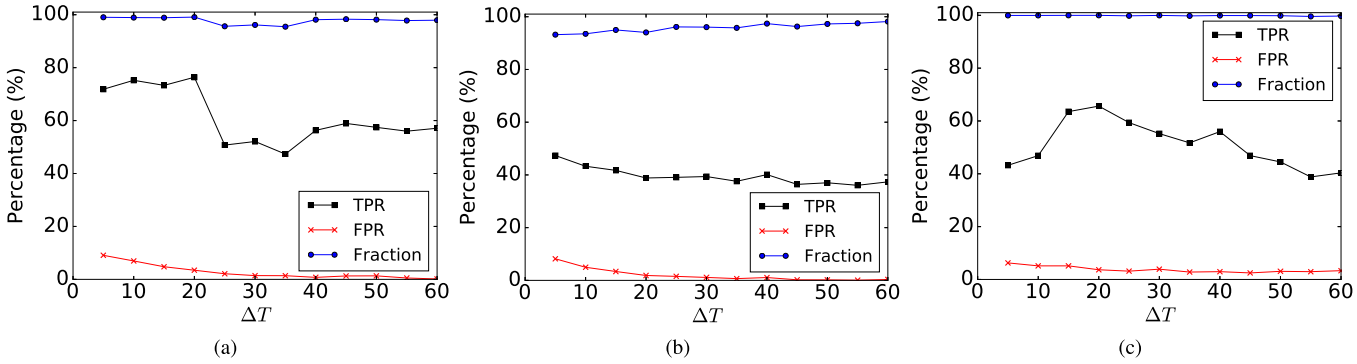


Fig. 11. Impact of parameter ΔT. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

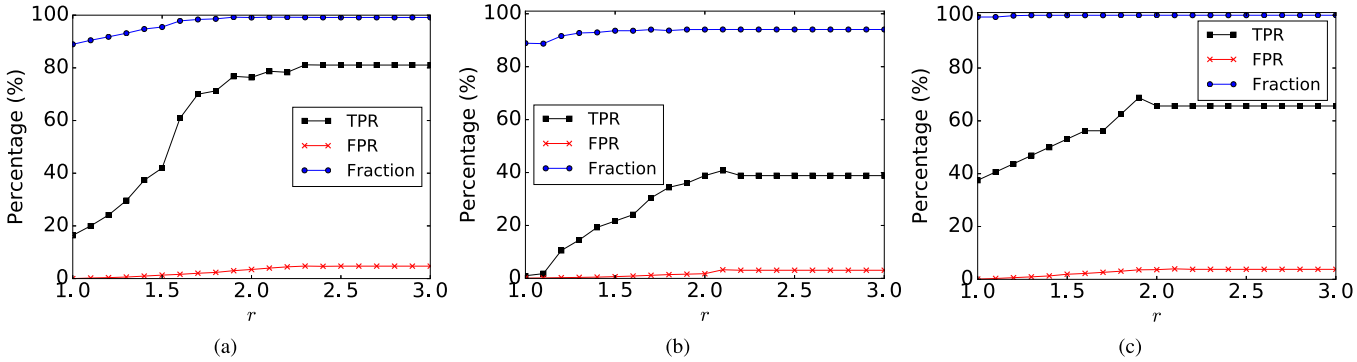


Fig. 12. Impact of parameter r. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

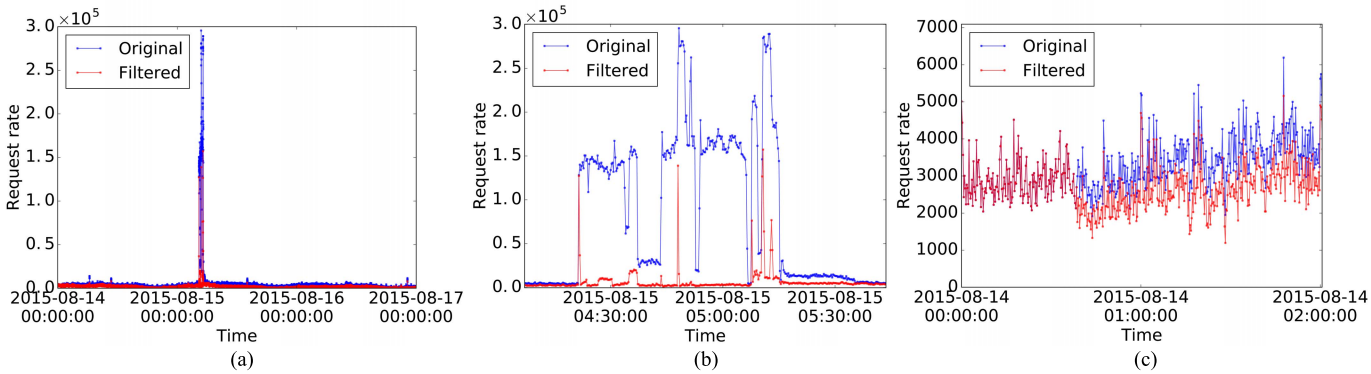


Fig. 13. Experimenal results of Dataset1508. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start.

curve follows the original one for a long time, indicating a low false alarm rate of SkyShield.

Fig. 14a demonstrates the overview of the request rate for Dataset1603. Different from the short-term overwhelming attack demonstrated for Dataset1508, the attack in Dataset1603 persisted much longer. In addition, the volume of total requests is relatively small and volatile, indicating a challenging scenario for SkyShield. It is shown that the system suffered from slow rate attacks in all three days. However, SkyShield could still effectively reduce the request rate to a reasonable level. Fig. 14b presents the detail of the attack for Dataset1603. It can be seen that there are no obvious spikes in the filtered request rate, indicating that the attacking hosts are already detected before they increase their request rates. Fig. 14c illustrates the start of the Dataset1603. As illustrated by the black frame, we can see that the attacking request rate increases very slowly. However, SkyShield still detects the anomaly, and thus performs the mitigation of malicious requests. This is because SkyShield detects the distribution changes of request numbers in the sketch and thus can identify the malicious hosts which cause the abnormal increase of the request rate.

Fig. 15a shows the request rates for Dataset1604. It is seen that the system encountered a low request rate attack at an interval of 30 minutes in all three days. Insight into the original access log shows that this attack is against one of the web servers hosted in the cluster. Such attack is speculated to be caused by slow rate crawlers which access the website in a fixed time interval. Fig. 15b is the detail of an attack

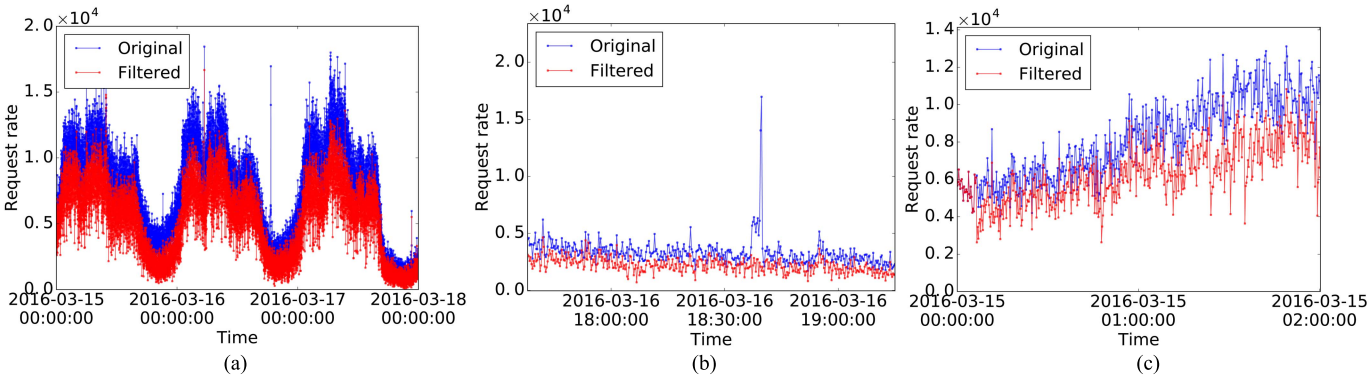


Fig. 14. Experimenal results of Dataset1603. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start.

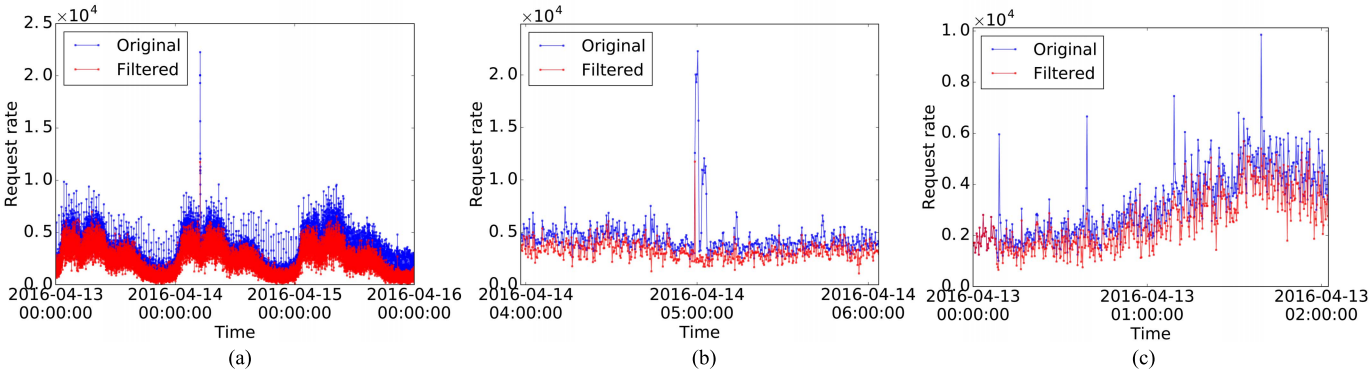


Fig. 15. Experimenal results of Dataset1604. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start.

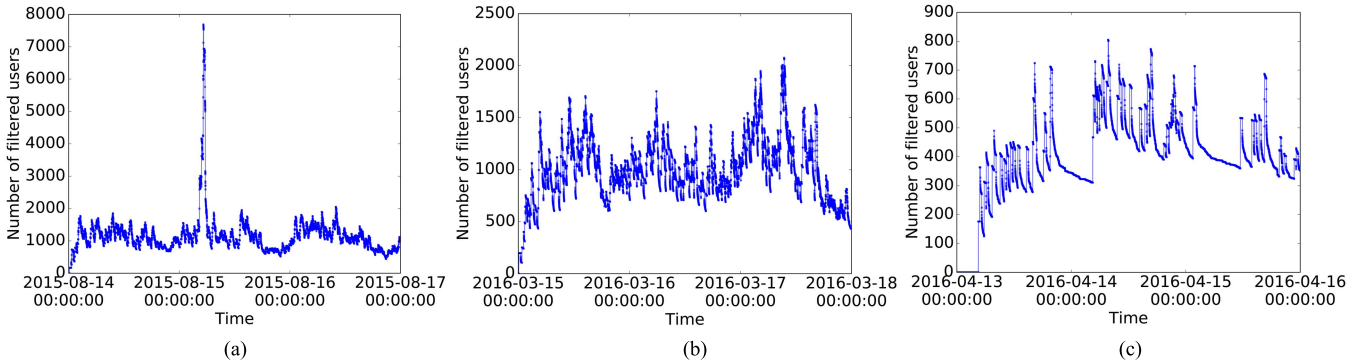


Fig. 16. The number of hosts in the blacklist versus the time. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604.

for Dataset1604. There is an obvious spike in the curve of the filtered request rate and in only one detection cycle the curve returns to its normal level. Fig. 15c demonstrates the detail of the start of the Dataset1604. The result validates the effectiveness of our system in the mitigation of such attacks. However, these crawlers may be benign users such as search engines or even partners. The deployment of SkyShield forces these users change their access policies to avoid being filtered.

Fig. 16 depicts the number of hosts in the blacklist versus the time. In this experiment, we block each detected malicious host for $100\Delta T$ (i.e., 2000s) instead of cleaning the blacklist, which may provide us an insight into how bots cooperate with each other to launch an attack. It is shown that the number of blacklisted hosts for Dataset1603 is much smaller than that for Dataset 1508 at the attacking period. The number of blacklisted hosts for Dataset1604 is the smallest and exhibits greater fluctuations than the other datasets. This is because that most of the blacklisted hosts are crawlers. These crawlers send plenty of requests to a specific web server simultaneously with a fixed interval of 30 minutes. As the blacklist is emptied periodically, therefore the fluctuations are caused by the release and recapture of these crawlers. We also conduct experiments to evaluate the number of blocked hosts with another clean period of $150\Delta T$ (i.e., 3000s). We obtain very similar results but with larger

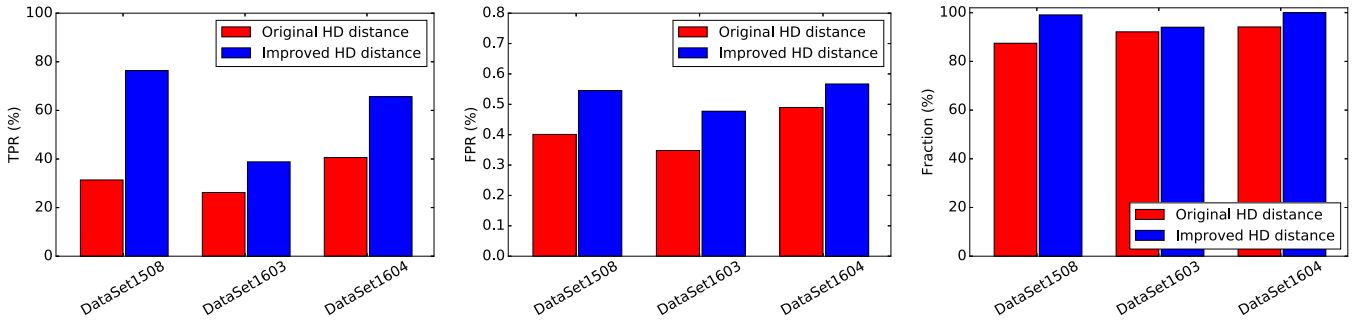


Fig. 17. Performance comparison between the original and the improved Hellinger Distances. (a) TPR. (b) FPR. (c) FRA.

TABLE III
PERFORMANCE OF SKYSHIELD WITH DEFAULT PARAMETERS

Dataset	TPR (%)	FPR (%)	Fraction (%)
Dataset1508	76.4	3.45	99.1
Dataset1603	38.8	1.77	94.0
Dataset1604	65.6	3.67	99.9

numbers of blocked hosts. Moreover, the values of TPR, FPR, and Fraction of filtered malicious requests are also close to the results shown in Table III, indicating that the selection of clean period has limited impacts on the performance of SkyShield. We omit these results because they are very similar to the results presented in Fig. 16 and Table III.

The above experimental results validate that SkyShield can quickly detect and mitigate different kinds of app-layer DDoS attacks. Table III presents the TPR, FPR, and the fraction of filtered malicious requests for different datasets with the default optimal parameters. It shows that the TPR for Dataset1603 is much lower than that for the other two datasets. This is because the duration of attack in Dataset1603 is much longer and the strength of the attack is much weaker than that of the other two datasets. This results in that the distribution of requests in a detection cycle is evener than others. Additionally, the number of the abnormal buckets is a function of the volume of requests. Hence, fewer buckets are discriminated suspicious and fewer hosts are blocked for Dataset1604 and Dataset1508 since their request volumes are smaller than that of Dataset1603. Overall, the fraction of filtered malicious requests for Dataset1603 is still above 94%.

We compare the performance of SkyShield between with the original HD distance and with the improved one. Fig. 17 demonstrates the obtained results. Fig. 17a and 17c show that both the TPR and the Fraction are improved with the new HD distance. Although the new HD distance may lead to an increase in FPR, as shown in Fig. 17b, such an increase is marginal compared to the benefits of TPR (Note the placement of the decimal points in Fig. 17b). Such an increase is due to the fact that the new HD distance sorts the request numbers in the sketches before calculates the distance, and hence is more sensitive than the original one.

Summary: The experimental results based on all datasets demonstrate the effectiveness of SkyShield in the mitigation of flooding attacks.

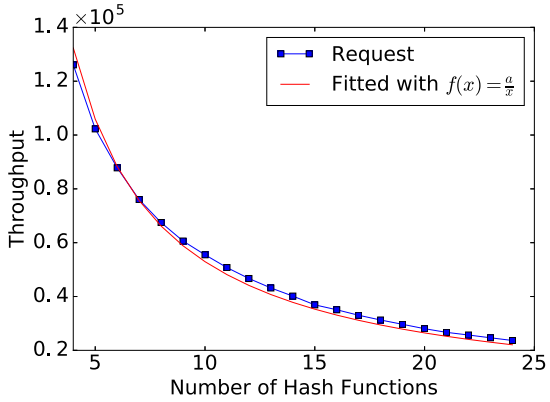


Fig. 18. The throughput of the system in a single thread.

D. Efficiency Evaluation

Q3: Can SkyShield efficiently process the large volume of requests during large flooding attacks?

The efficiency of SkyShield is mainly affected by the number of hash functions. To evaluate the capability of SkyShield to handle large flooding attacks, we define the throughput as the number of requests processed by the system per second. The processing time is mainly consumed by the calculations of hash functions. Fig. 18 shows the throughput as a function of the number of hash functions in a single CPU core. We can see that the throughput is a reciprocal function of the number of hash functions, which can be presented as:

$$f(x) = \frac{a}{x} \tag{16}$$

where a is a parameter to be estimated. The least square fitting method obtains an $a = 529,500$. According to the above analysis, SkyShield totally needs 23 hash functions, which results in a throughput slightly greater than 23,000 requests per second. The calculations of different hash functions are independent and thus can be paralleled with multiple cores. For instance, if we use four cores for hashing, each core processes six hash functions, and the throughput can be as high as 88,250 requests per second. Note that all incoming traffic are well-formed HTTP requests and thus such a request rate is relative high. Therefore, the system is capable of handling large-scale flooding attacks.

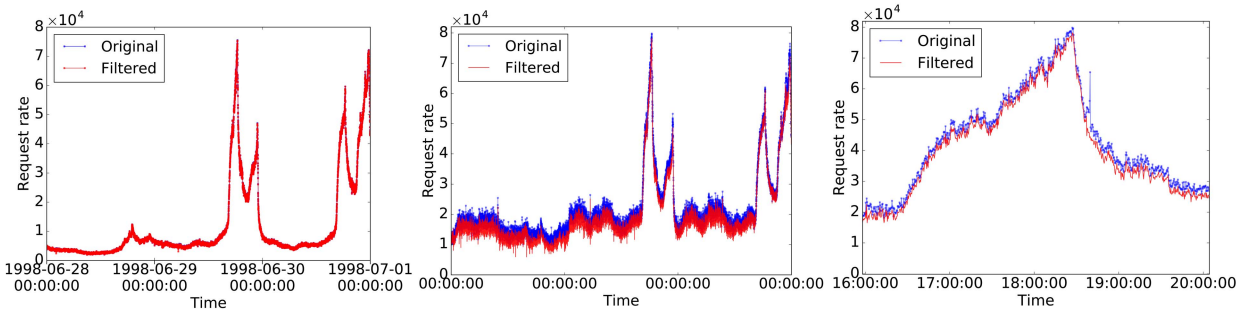


Fig. 19. Performance of SkyShield in flash crowd mimicking attacks. (a) The original data. (b) The combined data. (c) Insight of the combined data.

Summary: SkyShield is efficient in handling a large volume of HTTP requests.

E. Performance of Mitigating Flash Crowd Mimicking Attacks

Q4: Can SkyShield distinguish flash crowd mimicking attacks from real flash crowds?

App-layer DDoS attacks utilize legitimate HTTP requests to overwhelm a victim. What's worse, attackers prefer to launch attacks during or mimicking a flash crowd event in order to evade the detection. We employ the WorldCup98 data [32] to validate the effectiveness of SkyShield. This dataset is used because it has similar scenarios with our application and all requests in this dataset are guaranteed to be normal. We refer readers to [32] for detailed descriptions of the data.

We conduct two experiments. The first aims at testing whether SkyShield disrupts the access of normal users in flash crowd events. In the experiment, we apply SkyShield to the WorldCup98 data and see whether an alarm is raised. Fig. 19a shows that SkyShield does not raise any alarms for the original WorldCup98 data from 1998/6/28 to 30 when the website experienced four waves of flash crowds. SkyShield does not detect any anomaly even at the sharpest increase of requests because the sharp increases in flash crowds are caused by the simultaneous access of numerous normal users and these hosts are evenly mapped into the sketch data structure. Since all hosts send similar numbers of requests, the values in the buckets of the sketch are evenly distributed and therefore the divergence of sketches in two consecutive detection cycles is small and steady, resulting in no alarms in the flash crowds.

The second experiment is conducted to test whether SkyShield can effectively detect the attacks occurred during a flash crowd event. In this experiment, we combine the data of Dataset1603 with selected three days of WorldCup98 data from 1998/6/28 to 30 in order to simulate an app-layer DDoS attack occurred during a flash crowd event. Fig. 19b demonstrates the detection results for the synthetic data. It is shown that SkyShield can effectively mitigate the attacking requests. Fig. 19c illustrates a detailed view of the time interval from 16:00 to 20:00 for the second day, which is the period of the largest flash crowd. The result shows that there is an obvious attack at the time 18:40:00. The attack is brought from the Dataset1603. However, SkyShield still detects the anomaly and mitigate the attack in a very short time, indicating that SkyShield is effective to mitigate DDoS attacks even during flash crowds. We also combine other types of attacks with the flash crowd, and all experiments evidence the effectiveness of SkyShield. We omit these results for space concerns.

Summary: SkyShield can distinguish malicious requests from normal ones even the attacks occur during flash crowds.

F. Comparison With the State-of-the-Art Methods

Q5: What are the advantages of SkyShield compared to other state-of-the-art methods?

We qualitatively compare SkyShield with two the state-of-the-art methods, namely the Hidden semi-Markov Model (HsMM) based approach [10], [33], [34] and the Transductive Confidence Machines for K-Nearest Neighbors (TCM-KNN) based approach [35].

HsMM based method profiles normal users' behaviors and regards any deviations from the normal profiles as an anomaly. The entropy is used to detect the potential app-layer DDoS attacks. With proper configuration of parameters, the HsMM method could achieve an FPR as low as 1.5% and a detection rate about 90%. However, the model needs to be frequently updated by stable and low-volume Web workload. Additionally, the training of the model is computation intensive and the training data should be newly collected in order to keep its freshness. This greatly limits the application of the HsMM method in real-time DDoS attack detection. In addition, the HsMM model needs a priori knowledge of the website page structure which sometimes is daunting to collect. The wide use of dynamic web pages makes this issue even worse. Compared to the HsMM method, SkyShield can mitigate attacks more quickly and the parameter configuration is much easier for SkyShield.

The TCM-KNN method is designed to be a light-weighted DDoS attacks detection scheme for web servers. The reported TPR and FPR are 99.53% and 1.93%, respectively. The method utilizes a new objective measurement as the input features and employs the Genetic Algorithm based instance selection method to boost the real-time detection performance. However, the training of the model is still expensive even for the improved version, although the authors then developed an extended fuzzy C-means algorithm to solve this problem [35]. What's worse, the model is not adaptive to network dynamics.

Models trained by data of specific periods may not be suitable for the detection of anomalies occurred in other time periods. Compared to the TCM-KNN model, SkyShield is also light-weighted and detects DDoS attacks against web servers in real time. SkyShield employs a novel measure of sketch divergence which is stable to network dynamics. The number of abnormal buckets is adaptive to the request volumes, thus in an intensive attack more malicious requests will be filtered.

Compared to the above methods, SkyShield has a relative lower TPR. However, the main goal of SkyShield is to guarantee the availability of service when the system suffers abrupt overwhelming attacks. Although only partial malicious hosts are detected, the attacking traffic can be reduced more than 94%. Therefore, SkyShield is effective to defend against app-layer DDoS attacks and can well prevent the web server from being overwhelmed when it suffers heavy flooding attacks.

Summary: Compared with the state-of-the-art methods, SkyShield can effectively handle HTTP requests and quickly mitigate attacks. Moreover, it is robust to network dynamics with easily configured parameters.

VIII. Discussion

A. Potential Improvement of Performance

The ground truth of the malicious hosts are extracted from the mitigation logs. However, it may contain some normal users who were annoyed by the CAPTCHA test and failed to pass the test. Besides, SkyShield tends to filter out the hosts with large request volumes. We speculate that these hosts may contain NAT or proxy users. Since we conduct the experiments based on the off-line data, the whitelist of the system cannot be activated. If the whitelist is functioned, the FPR can be even lower. Hence, we expect SkyShield to perform even better when it is applied in the real environment.

B. Possible Ways to Evade SkyShield

Although experimental results demonstrate that SkyShield is effective in mitigating flash crowd mimicking attacks, it is still possible for attackers to evade SkyShield. One approach is to produce a real flash crowd event using as many bots as possible. However, Yu et al. found that the number of active bots should be sufficiently large to mimic a genuine flash crowd [36]. Besides, it also requires the attackers have the ability to coordinate the bots accurately. However, due to diverse reasons such as regular power-off, the installation of anti-virus software, software patching, etc., the number of active bots at a specific time is limited. Therefore, this approach will greatly increase the cost of attackers. Recently, it has been observed that attackers can control compromised bots through social networks [37] and the Tor network [38], which provide attackers with possible ways to control bots more precisely. This brings another potential method to evade SkyShield by launching an attack with waves of new bots. Consequently, the malicious hosts detected by current detection cycle will not appear in the next cycle and thus will not be added to the blacklist. However, attacking with waves of new bots can only prevent malicious hosts from being added to the blacklist, but cannot prevent SkyShield from raising alarms. Therefore, even if the attackers have the ability to launch such an attack, SkyShield will raise consecutive alarms since the malicious requests are not blocked. SkyShield can easily recognize such an alarming pattern by counting the number of consecutive alarms, and then changes the detection interval deliberately to defend against such an attack. In addition, launching an attack with waves of new bots requires the attackers know the exact detection cycle and precisely controls the active bots, which will increase the cost of the attack.

Someone may argue that attackers might evade the detection by increasing the request rate very slowly because the sketch divergence between detection cycles is relatively small. However, it is difficult to launch such an attack because it requires attackers to precisely coordinate the compromised bots. Even if the attacker has such an ability, they cannot control the distribution of request numbers in the detection sketch due to the randomization of hash functions. Therefore, SkyShield can detect the malicious hosts which cause the abnormal increase in request rate even if it increases slowly. The experimental result in Fig. 14c illustrates the effectiveness of SkyShield against such attacks. More precisely, as shown by the black frame in Fig. 14c, the request rate increases very slowly. However, SkyShield can still detect the anomaly, and perform the mitigation of malicious requests. Moreover, launching an attack by increasing the request rate slowly also increases the cost of the attack and limits the influence of the attack.

In summary, SkyShield follows the DDoS attack countermeasure principle by forcing either an attacker to increase the costs or the possibility of attack traffic to be distinguished from legitimate traffic [39].

IX. Related Work

Sketch techniques have already been widely used in the detection of DDoS attacks. Barford et al. [41] found that the detection of a sharp increase in the local variance of the filtered network traffic is an effective way of exposing anomalies. Ganguly et al. [23] proposed a novel sketch-based data-streaming algorithm for robust and real-time DDoS attack detection in large ISP networks. Tang et al. [24], [25] developed an efficient online flooding attack detection scheme by integrating the sketch techniques with Hellinger distance. Su et al. [42] proposed a weighted k-NN clustering method to detect DoS attacks in real time. They employed a different genetic algorithm to select significant features to discriminate malicious requests. However, those studies focus on detecting anomalies without considering the mitigation of attacks.

To address the above issues, Schweller et al. [22] proposed an efficient reverse hashing scheme to infer the IP addresses of malicious hosts from reversible sketches. Salem et al. [26] proposed a flooding attack detection method using a multiple layer reversible sketch. Liu et al. [27] proposed a two-level approach for scalable and accurate DDoS attack detection by exploiting the asymmetry of the attack traffic. These methods attempt to retrieve the anomalous keys either by reverse hashing methods or by storing parts of keys, which are either computation intensive or storage consumptive. Compared with the above

literature, SkyShield avoids the reverse calculation process, which makes it efficient in real-time anomaly detection.

The CAPTCHA techniques are adopted in this work. Kandula et al. [15] proposed a system to protect web clusters from app-layer DDoS attacks by employing the CAPTCHA techniques [43]. Rangasamy et al. [16] designed a graphical puzzle authentication mechanism to determine whether a client is suspicious or not. Despite their effectiveness, these methods have several evident drawbacks. Firstly, the CAPTCHA techniques also bring normal users with extra burdens which harm the Quality of Experience (QoE) [44]. In addition, they also prevent the accesses of legitimate robot crawlers, thus go against the Search Engine Optimization (SEO). What's worse, the CAPTCHA itself may become the attack target of adversaries [45]–[47]. In this paper, we employ the CAPTCHA techniques to verifies the legitimization of only suspicious hosts, which greatly reduces the chance of normal users to be tested. The design of a secure and effective CAPTCHA technique is out of the scope of this paper.

Several studies address the mitigation of app-layer DDoS attacks. Filter-based approaches use ubiquitously deployed filters to block unwanted traffic [15], [20], [48]. Capability-based mechanisms focus on controlling resource usage by clients [33], [49]–[51]. Clients have to obtain servers' explicit permissions before transmitting packets. Traffic from authorized or privileged clients with valid capability permissions are served with a higher priority during an attack. Liu et al. [52] compared the effectiveness of filters-based methods to that of capabilities-based ones. They find both filters and capabilities are highly effective DDoS defense mechanisms, but neither is more effective than the other in all types of DDoS attacks. Several studies utilize proxy nodes between clients and protected hosts to absorb and filter out attack traffic. Wang et al. [53] proposed a moving target defense mechanism that defends authenticated clients against Internet service DDoS attacks. The scheme employs a group of dynamic proxy nodes that relay traffic between protected servers and authenticated clients. Joldzic et al. [54] proposed a fully transparent system monitoring network traffic for DoS attacks. The system offers an effective way of isolating the attack sources during a DDoS attack. SkyShield is orthogonal to these work and can benefit from the integration of these schemes.

X. Conclusion

To defend against application layer DDoS attacks, it is essential to have a fast response system that can automatically detect and mitigate malicious requests as soon as possible. In this paper, we design and implement such a system named SkyShield by taking advantages of the sketch techniques. First, to alleviate the impact of network dynamics, we propose a novel variant of Hellinger distance to calculate the divergence between sketches in two consecutive detection cycles. Second, to identify malicious hosts efficiently, we use the abnormal sketch obtained from the last detection cycle to avoid the reverse calculation of IP addresses. Third, we leverage other techniques including Bloom filters and the CAPTCHA techniques to guarantee the effectiveness of SkyShield. We have developed a prototype of SkyShield and carefully evaluated its performance using real attack datasets collected from a large-scale web cluster. The experimental results demonstrate that SkyShield can effectively mitigate application layer DDoS attacks and pose a limited impact on normal users.

References

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," in *Proc. SIGCOMM*, 2004, pp. 39–53.

[2] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Comput. Netw.*, vol. 44, no. 5, pp. 643–666, 2004.

[3] L.-C. Chen, T. A. Longstaff, and K. M. Carley, "Characterization of defense mechanisms against distributed denial of service attacks," *Comput. Secur.*, vol. 23, no. 8, pp. 665–678, 2004.

[4] J. Mölsä, "Mitigating denial of service attacks: A tutorial," *J. Comput. Secur.*, vol. 13, no. 6, pp. 807–837, 2005.

[5] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Comput.*, vol. 10, no. 1, pp. 82–89, Jan. 2006.

[6] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and DDoS problems," *Comput. Surv.*, vol. 39, no. 1, p. 3, 2007.

[7] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki, "Distributed denial of service (DDoS) detection by traffic pattern analysis," *Peer-Peer Netw. Appl.*, vol. 7, no. 4, pp. 346–358, 2014.

[8] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2519–2533, Sep. 2015.

[9] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "DDoS-shield: DDoS-resilient scheduling to counter application layer attacks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 26–39, Feb. 2009.

[10] Y. Xie, S. Tang, Y. Xiang, and J. Hu, "Resisting Web proxy-based HTTP attacks by temporal and spatial locality behavior," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1401–1410, Jul. 2013.

[11] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone Web traffic," *Future Generat. Comput. Syst.*, vol. 38, pp. 36–46, Sep. 2014.

[12] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 447–456, Feb. 2014.

[13] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, and F. Tang, "Discriminating DDoS attacks from flash crowds using flow correlation coefficient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1073–1080, Jun. 2012.

[14] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Delving into Internet DDoS attacks by botnets: Characterization and analysis," in *Proc. DSN*, 2015, pp. 379–390.

[15] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," in *Proc. NSDI*, 2005, pp. 287–300.

[16] J. Rangasamy, D. Stebila, C. Boyd, and J. G. Nieto, "An integrated approach to cryptographic mitigation of denial-of-service attacks," in *Proc. ASIACCS*, 2011, pp. 114–123.

[17] R. Netravali, J. Mickens, and H. Balakrishnan, "Polaris: Faster page loads using fine-grained dependency tracking," in *Proc. NSDI*, 2016, pp. 123–136.

[18] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" *J. Assoc. Inf. Syst.*, vol. 5, no. 1, pp. 1–28, 2004.

[19] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "QuickSAND: Quick summary and analysis of network data," DIMACS, Piscataway, NJ, USA, Tech. Rep. 2001-43, 2001.

[20] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proc. IMC*, 2003, pp. 234–247.

[21] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in *Proc. IMC*, 2004, pp. 187–200.

[22] R. Schweller *et al.*, "Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications," in *Proc. INFOCOM*, Apr. 2006, pp. 1–12.

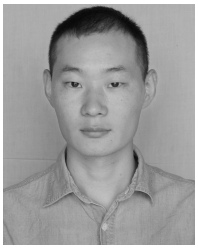
[23] S. Ganguly, M. Garofalakis, R. Rastogi, and K. Sabnani, "Streaming algorithms for robust, real-time detection of DDoS attacks," in *Proc. ICDCS*, Jun. 2007, p. 4.

[24] J. Tang, Y. Cheng, and C. Zhou, "Sketch-based sip flooding detection using Hellinger distance," in *Proc. GLOBECOM*, Nov./Dec. 2009, pp. 1–6.
[25] J. Tang, Y. Cheng, Y. Hao, and W. Song, "SIP flooding attack detection with a multi-dimensional sketch design," *IEEE Trans. Depend. Sec. Comput.*, vol. 11, no. 6, pp. 582–595, Nov./Dec. 2014.
[26] O. Salem, S. Vaton, and A. Gravey, "A scalable, efficient and informative approach for anomaly-based intrusion detection systems: Theory and practice," *Int. J. Netw. Manage.*, vol. 20, no. 5, pp. 271–293, 2010.
[27] H. Liu, Y. Sun, and M. S. Kim, "Fine-grained DDoS detection scheme based on bidirectional count sketch," in *Proc. ICCCN*, 2011, pp. 1–6.
[28] L. Le Cam and G. L. Yang, *Asymptotics in Statistics: Some Basic Concepts*. Verlag, NJ, USA: Springer, 2012.
[29] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.
[30] D. C. Montgomery, *Introduction to Statistical Quality Control*. Hoboken, NJ, USA: Wiley, 2007.
[31] J. S. Hunter, "The exponentially weighted moving average," *J. Quality Technol.*, vol. 18, no. 4, pp. 203–210, 1986.
[32] M. Arlitt and T. Jin. *1998 World Cup Web Site Access Logs*. Accessed: Dec. 4, 2016. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/WorldCup.html
[33] Y. Xie and S. Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 54–65, Feb. 2009.
[34] Y. Xie and S. Z. Yu, "Monitoring the application-layer DDoS attacks for popular websites," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 15–25, Feb. 2009.
[35] Y. Li, T.-B. Lu, L. Guo, Z.-H. Tian, and Q.-W. Nie, "Towards lightweight and efficient DDoS attacks detection for Web server," in *Proc. WWW*, 2009, pp. 1139–1140.
[36] S. Yu, S. Guo, and I. Stojmenovic, "Fool me if you can: Mimicking attacks and anti-attacks in cyberspace," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 139–151, Jan. 2015.
[37] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: Emerging threats and countermeasures," in *Proc. ACNS*, 2010, pp. 511–528.
[38] A. Sanatinia and G. Noubir, "OnionBots: Subverting privacy infrastructure for cyber attacks," in *Proc. DSN*, 2015, pp. 69–80.
[39] M. S. Kang, V. D. Gligor, and V. Sekar, "SPIFFY: Inducing cost-detectability tradeoffs for persistent link-flooding attacks," in *Proc. NDSS*, 2016, pp. 1–15.
[40] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting distributed denial of service attacks: Methods, tools and future directions," *Comput. J.*, vol. 57, no. 4, pp. 537–556, 2014.
[41] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. IMC*, 2002, pp. 71–82.
[42] M.-Y. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3492–3498, 2011.
[43] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard ai problems for security," in *Proc. EUROCRYPT*, 2003, pp. 294–311.
[44] D. Pogue, "Time to kill off captchas," *Sci. Amer.*, vol. 306, no. 3, p. 23, 2012.
[45] J. Yan and A. S. El Ahmad, "A low-cost attack on a Microsoft CAPTCHA," in *Proc. CCS*, 2008, pp. 543–554.
[46] P. Golle, "Machine learning attacks against the Asirra CAPTCHA," in *Proc. CCS*, 2008, pp. 535–542.
[47] Y. Wu, Z. Zhao, F. Bao, and R. H. Deng, "Software puzzle: A countermeasure to resource-inflated denial-of-service attacks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 168–177, Jan. 2015.
[48] S. Sivabalan and P. J. Radcliffe, "A novel framework to detect and block DDoS attack at the application layer," in *Proc. TENCON*, 2013, pp. 578–582.
[49] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet denial-of-service with capabilities," in *Proc. SIGCOMM*, 2004, pp. 39–44.
[50] X. Yang, D. Wetherall, and T. Anderson, "TVA: A DoS-limiting network architecture," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1267–1280, Dec. 2008.
[51] X. Liu, X. Yang, and Y. Xia, "NetFence: Preventing Internet denial of service from inside out," in *Proc. SIGCOMM*, 2010, pp. 255–266.
[52] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer dos defense against multimillion-node botnets," in *Proc. SIGCOMM*, 2008, pp. 195–206.
[53] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A moving target DDoS defense mechanism," *Comput. Commun.*, vol. 46, pp. 10–21, Jun. 2014.
[54] O. Joldzic, Z. Djuric, and P. Vuletic, "A transparent and scalable anomaly-based DoS detection method," *Comput. Netw.*, vol. 104, pp. 27–42, Jul. 2016.

Chenxu Wang received the B.S. degree in communication engineering and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, in 2009 and 2015, respectively. He was a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University. He is currently an Assistant Professor with the School of Software Engineering, Xi'an Jiaotong University. His current research interests include data mining, network security, online social network analysis, and information diffusion.

Tony T. N. Miu received the B.S. degree in computer science from The Hong Kong Polytechnic University. He is currently a Security Researcher with Nexusguard Ltd. His current research focuses on network security, especially DDoS detection, and defense.

Xiapu Luo received the Ph.D. degree in computer science from The Hong Kong Polytechnic University. He was a Post-Doctoral Research Fellow with the Georgia Institute of Technology. He is currently a Research Assistant Professor with the Department of Computing and an Associate Researcher with the Shenzhen Research Institute, The Hong Kong Polytechnic University. His current research focuses on smartphone security and privacy, network security and privacy, and Internet measurement.



Jinhe Wang received the B.S. degree in software engineering from the Dalian University of Technology. He was a Research Assistant with the Department of Computing, The Hong Kong Polytechnic University. He is currently pursuing the degree with the School of Software Engineering, Xi'an Jiaotong University. His current research focuses on network security and privacy, and traffic engineering.