DEFENDING AGAINST ADVANCED DDOS ATTACKS

MIU TUNG NGAI

MPhil

The Hong Kong Polytechnic University

2022

The Hong Kong Polytechnic University

Department of Computing

Defending Against Advanced DDoS Attacks

Miu Tung Ngai

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

Aug 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

____________________________ (Signed)

Miu Tung Ngai (Name of student)

Abstract

Distributed denial of service (DDoS) attacks have been a severe threat to the Internet for decades. Although many detection and defense mechanisms have been proposed, the attackers always attempt to evade the detection by adopting various sophisticated approaches. In this thesis, we investigate such advanced DDoS attacks from three aspects. First, we inspect application layer DDoS attacks because their attack requests can be the same as benign ones for evasion and exhaust the computational resources of target servers. Specifically, we first design a new approach to model users' browsing behaviors and use it to differentiate between attacks and benign visits at both session and page level. Then, we develop an effective defense system named SkyShield that leverages the sketch data structure to detect and mitigate application-layer DDoS attacks quickly. Second, network layer volumetric attacks are becoming even more popular with the emergence of the DDoS-as-a-service economy, and most attacks are launched abruptly. Hence, a defense system should adopt an effective process to detect and mitigate the attacks as soon as possible. Since different DDoS protection services (DPS) adopt diverse defense strategies, we characterize the Border Gateway Protocol (BGP)-based DPSes by proposing a machine learning based approach to analyze BGP update messages. Third, to better understand the trends of DDoS amplification attacks, we deploy DDoSTrap, a high-performance honeypot to collect data and report interesting observations after analyzing 4-year data. We conducted extensive experiments to evaluate the proposed approaches, and the experimental results demonstrate their effectiveness. Moreover, our findings shed light on the trends of DDoS attacks and the design of effective DDoS attack mitigation schemes.

Acknowledgements

I would like to thank my supervisor, Dr. Daniel Luo, for his advice and guidance throughout the study. I am very grateful that he took me on as a student and has confidence in me over the years. Thank Dr. Luo for understanding my difficulties as a part-time student. Understanding in difficult time is the biggest support to me.

Thanks to my co-supervisor, Dr. Dan Wang. I have significantly benefited from his detailed feedback.

Thanks to Dr. Chenxu Wang. He helped me find my footing as I started this research and gave me invaluable assistance and suggestions.

Thanks also to Lei Xue and Muhui Jiang. Thank you for listening to my complaints. I know everyone has worked so hard.

Thanks to ELC Instructor Marshall Yin, your class is what impressed me the most.

Thanks to all staff in the General Office, and thank you for clearly explaining my inquiry.

Thank you to my wife and friends for the endless support.

Contents

List of Figures

List of Tables

1 Introduction

Distributed denial of service (DDoS) attacks have been a severe threat to the Internet for decades. Although many detection and defense mechanisms have been proposed[68, 78, 89, 100], the attackers always attempt to evade the detection by adopting various sophisticated approaches. For example, the application layer DDoS (app-layer DDoS) attacks against web servers grow rapidly and bring victims with great revenue losses [68, 78, 89, 100]. They attempt to disrupt legitimate access to application services by exhausting the target's resources and masquerading flash crowds with numerous benign requests. Flash crowd refers to the scenario that a popular website service is unavailable due to massive visitors simultaneously accessing it[96]. The stealthiness of app-layer DDoS attacks makes most signature-based intrusion prevention systems ineffective. Moreover, compared to network-layer DDoS attacks, app-layer DDoS attacks usually consume less bandwidth and thus are more stealthier[20].

In this thesis, we first investigate how to detect and defend against app-layer DDoS attacks. Generally speaking, there are four types of app-layer flooding attacks [97]. First, session flooding attacks exhaust bandwidth resources by high session connection request rates. Second, request flooding attacks consume bandwidth resources with massive requests per session. Third, asymmetric attacks exhaust computational resources through high-workload requests. Fourth, slow request/response attacks consume connection resources by forcing the server to hold massive established connections. The inherent difference between DDoS flooding attacks and the flash crowd is the intention of users' behaviors reflected by their access patterns. Therefore, accurately estimating users' intentions through access pattern analysis is the key to detecting HTTP flooding attacks. Hence, we propose a new approach to model users' browsing behaviors based on the access logs recorded at the webserver and use it to detect app-layer DDoS attacks at both the session level and the page level for the sake of reducing the false positive rate and improving the performance.

Moreover, to defend against app-layer DDoS attacks, the increasingly high-speed network links demand an efficient data structure to efficiently process a huge volume of network traffic. 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 [26]. A series of sketch-based approaches have been proposed for anomaly detection in large-scale network traffic [36, 73, 81, 80, 35, 25]. Since sketches contain no direct information about the malicious hosts, they cannot be directly used to mitigate attacks. To tackle this problem, several efficient reverse hashing schemes have been proposed to infer the IP addresses of malicious hosts from reversible sketches [73, 79, 48]. These studies attempt to retrieve 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 for accurate anomaly detection. Second, when an attack occurs, it needs to identify malicious hosts without affecting legitimate users' access accurately. Third, since most attacks are launched abruptly, the defense system should be able to detect and mitigate the attack as soon as possible. To address these challenges, we propose SkyShield, a new 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, thus avoiding 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.

Besides examining app-layer DDoS attacks, we also investigate network layer volumetric attacks, which have been threatening the infrastructure of the Internet for decades and become even more popular with the emergence of the DDoS-as-a-service economy [60, 34, 71], from a new perspective. Specifically, the DDoS protection services (DPS) providers [32] usually conduct traffic cleansing through traffic diversion, which allows traffic to be routed through the DPS infrastructure, either always-on or on-demand. There are two main approaches to divert traffic, including the Domain Name System

(DNS)-based method and Border Gateway Protocol (BGP)-based method. The former diverts network traffic through proper configuration of Domain Name Servers or anycast techniques. The latter diverts traffic towards the DPS infrastructure for scrubbing by announcing/withdrawing specific IP subnets. Although recent studies have measured the DNS-based approach [15], little is known about the behavior of BGP-based DPS. To fill the gap, we first study BGP anomalies by collecting a number of DDoS attacks and disaster events that are reported to cause abnormal BGP dynamics as well as the BGP update messages during the reported period from the Route views project. Then, we design a machine learning-based method to determine whether or not the BGP anomalies are caused by DDoS attacks. If a DDoS attack event is identified, we conduct an in-depth analysis of the BGP traffic to characterize how BGP-based DPS leverages BGP to mitigate the attack.

In addition to investigate known app-layer DDoS attacks and volumetric attacks, we also conduct an empirical study on the DDoS amplification attacks by deploying a honeypot named DDoSTrap to collect data. By analyzing the data collected over the past four years, we obtain a number of new observations, such as the distribution of attack duration and victims, and the new bit-and-piece attack, etc.

In summary, we have made the following contributions in this thesis:

- We propose a new approach to model users' browsing behaviors and use it to differentiate between application-layer attacks and benign visits at session level and page level. We evaluate the performance of our detection method based on a dataset of real DDoS attacks collected from a busy commercial web server.

- We propose a sketch-based system to detect and mitigate application-layer DDoS attacks, which employs the abnormal sketch to facilitate the detection of malicious hosts of an ongoing attack. We evaluate its effectiveness using actual attack data collected from large-scale web server clusters.

- We propose a machine learning-based approach to identify DDoS events by analyzing BGP update messages and investigate the behaviors of BGP-based DPS after DDoS attacks occur in terms of BGP manipulation.

- We deploy a high-performance honeypot named DDoSTrap to capture DDoS amplification attacks and obtain a number of new observations by analyzing the data collected in the past 4 years.

The thesis is organized as follows. Chapter 2 gives an overview of DDoS attacks and defense mechanisms. Chapter 3 presents our approach to model users' browsing behaviors and differentiate between application-layer attacks and benign visits. Chapter 4 introduces our sketch-based system for detecting and mitigating application-layer DDoS attacks. Chapter 5 describes our study on the behaviors of BGP-based DPS. Chapter 6 reports our honeypot for capturing amplification DDoS attacks and a number of new observations. Charter 7 concludes the thesis with future work.

Literature Review 2

We first introduce DDoS attacks and then describe the related studies on the detection of application layer and network layer DDoS attacks.

2.1 DDoS attacks

DDoS attacks could be roughly classified into two categories: network-layer DDoS attacks and app-layer attacks. Network-layer DDoS attacks abuse the compromised devices and computers to generate a massive volume of traffic to exhaust the bandwidth resource, causing network congestion to impact victims' services. According to the latest record from Cloudflare, the peak size of DDoS attacks was up to 2.5Tbps in 2022 Q3 [13]. Such kind of traffic volume impacts a victim's services, damages network infrastructure and causes a regional outage of internet services. For example, on 2016 October 21, a 1Tbps DDoS attack hit the Domain Name Service(DNS) of Dyn company that provides services on Internet intelligence, domain services, traffic management, message management, and Domain Name Service(DNS). It caused the unavailability of the DNS service of Dyn. This further results in difficulties connecting numerous websites (such as Amazon, BBC, CNN, etc.) for many users. During the attack, the traffic going to other DNS providers increased dramatically and thus caused widespread network congestion [74]. Besides, attackers abuse public access services to launch reflection attacks. We divide such attacks into amplification attacks and non-amplification attacks. By launching amplification attacks, attackers abuse amplifiers, , such as DNS, CHARGEN, SSDP, and NTP servers, to enlarge the attack traffic to congest the victim's network[76, 37, 28, 82, 18]. For many of these protocols, attackers can use an internet-wide scan to find millions of amplifiers [37, 28]. Non-amplification attacks will not magnify the reflected traffic, such as TCP SYN-ACK reflection attacks [1]. The attacker sends a TCP SYN packet with a spoofed IP address to a server. After receiving it, the server replies to the spoofed IP address with a TCP SYN-ACK packet.

Application layer (app-layer) attacks affect online services and also cause victims with great revenue losses[68, 78, 89, 100]. They usually exploit the victim's vulnerabilities to exhaust application resources. For example, when launching slow POST attacks, the attackers exploit the HTTP header Content-Length field by setting it to a large value but sending data very slowly. Since the target web server will wait for the data, its memory will be consumed [91]. Meanwhile, such attacks exhaust connection resources by forcing the server to hold massive established connections. Finally, the victim's online service either reaches out of memory or fails to establish a new TCP connection causing a denial of service.

The stealthiness of app-layer DDoS attacks makes most signature-based intrusion prevention systems ineffective. Compared with the botnet-induced volumetric attacks that generate a significant amount of traffic, app-layer DDoS attacks usually consume less bandwidth and are stealthier[20]. For instance, HTTP attacks against a web application. The attackers attempt to exhaust server resources by generating valid, countless HTTP requests or sessions [91]. The most commonly used method to launch such attacks is HTTP GET flooding. Attackers can either initialize a large number of valid sessions or send a large number of requests in a single session to inundate the victim's web servers with answer requests. The process forces servers to allocate maximum traffic resources so legitimate requests cannot reach them. What's worse, masquerading flash crowds with numerous benign requests. Flash crowd refers to the scenario that a popular website service is unavailable due to massive visitors simultaneously accessing it[96]. Unlike HTTP GET flooding example, due to a valid HTTP request needing to establish a complete TCP connection so that we can identify the source IP addresses, it will generate an HTTP request to the victim service with a single IP address fastly. For instance, over a thousand requests with a single source IP address in a minute that identify as abnormal web browsing behavior. However, flash crowd uses massive source IP addresses to send valid HTTP requests simultaneously. As a result, victims become challenging to filter malicious requests but easy to block legitimate traffic correctly [33].

2.2 Detection of application layer DDoS attacks

The methods for detecting app-layer DDoS attacks can be roughly classified into three categories. The first scheme employs the Turing test to automatically tell computers and humans apart. Kandula et al. [33] proposed an anomaly detection method by graphical test to identify malicious hosts quickly. This method can effectively protect web servers from DDoS attacks that masquerade flash crowds. Rangasamy et al. [67] designed a puzzle authentication mechanism to determine whether a client is suspicious or not. However, these methods bring users extra burdens, which reduce the QoS [64]. Moreover, they prevent the access of legitimate robot crawlers, thus going against Search Engine Optimization (SEO). What's worse, the CAPTCHA itself may become the attack target of adversaries [27, 87, 93]. To reduce the impacts of the Turing test, Sivabalan et al. [75] encouraged using one of the above methods to detect suspicious users only when the server load exceeds a predefined threshold. The second scheme employs machine-learning methods to detect the anomalies caused by DDoS attacks [44]. Such methods are shown as practical and light-weight for real-time web server anomaly detection. However, the computational cost for the genetic algorithm is expensive, which results in a high training time for the model. To solve this problem, the authors developed an extended fuzzy C-means (E-FCM) algorithm to fulfill clustering tasks [45].The third scheme focuses on modeling the profiles of normal user behavior. Then any deviations from the normal profiles are determined as an anomaly. Xie et al. use the hidden semi-Markov model (HSMM) to characterize the access pattern of normal behaviors [88, 90, 92, 89]. This scheme can detect novel types of attacks. However, the implementation of the HSMM model is very complex, and it needs some prior knowledge of the website page structure, which is unavailable or hard to collect. Yatagai et al. propose HTTP-GET flood detection techniques based on analysis of page access behavior [95]. They offer two detection algorithms, one focusing on a browsing order of pages and the other focusing on a correlation with browsing time to page information size. Lee et al. introduce a sequence-order-independent method to detect potential application layer attacks [40].

2.3 Detection of network layer DDoS attacks

Network-layer DDoS attacks with Tbps traffic have become the new normal on the internet [14]. It is easy to detect such attacks due to their huge volume of traffic but difficult to defeat them due to limited bandwidth resources. How to distribute the attack traffic is one of the major issues to handling massive volumetric attacks. DPS providers with sufficient bandwidth resources will distribute attack traffic to different sites by leveraging BGP protocol [32].

Sketch techniques have already been widely used in the detection of DDoS attacks. Barford et al. [6] 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. [25] proposed a novel sketch-based data-streaming algorithm for robust and real-time DDoS attack detection in large ISP networks. Tang et al. [81, 80] developed an efficient online flooding attack detection scheme by integrating sketch techniques with Hellinger distance. Su et al. [77] 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. Schweller et al. [73] proposed an efficient reverse hashing scheme to infer the IP addresses of malicious hosts from reversible sketches. Salem et al. [79] proposed a flooding attack detection method using a multiple layer reversible sketch. Liu et al. [48] 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, either computation-intensive or storage consumptive.

There are a number of studies on the mitigation of DDoS attacks. Filter-based approaches use ubiquitously deployed filters to block unwanted traffic [36, 33, 75]. Capability based mechanisms focus on controlling resource usage by clients [4, 49, 90, 94]. Clients have to obtain servers' explicit permissions before transmitting packets. Traffic from authorized or privileged clients with valid capability permissions is served with a higher priority during an attack. Liu et al. [50] 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 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. [85] 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.

Many studies have been conducted to detect the instability or pathological behaviors of the BGP dynamics. Labovitz et al. [38] investigated the BGP routing messages and found that the volume of routing updates is more redundant than expected. Besides, they revealed several unexpected trends of both forwarding instability and routing policy fluctuations. Deshpande et al. [21] proposed an online instability detection architecture that applies statistical pattern recognition techniques to detect the instabilities of BGP dynamics. They found that features like AS path length and AS path edit distance are very effective in modeling the behaviors of the Internet topology. Chang et al. [12] proposed an algorithm to identify inter-domain path-change events from streams of BGP updates. Feldmann et al. [24] proposed a methodology to identify the origin of routing instability from BGP updates. Several studies utilize statistical pattern recognition techniques to detect the instabilities of BGP routing dynamics [38, 42, 21]. Al-Rousan et al. [3] employed the support vector machine (SVM) and Hidden Markov Models (HMMs) to detect and classify BGP anomalies. Their method achieves good performance in detecting BGP anomalies. Some studies examine the impacts brought by historical events such as blackouts, cable cuts, worms, prefix-hijacking attacks, etc. Cowie et al. [17] analyzed the global BGP routing instabilities caused by the Code Red II and Nimda worms that occurred in July and September 2001, respectively. They also examined the impacts of the blackouts in 2003 on Internet connectivity and traffic routing in the region [16]. They found that the impact was more severe than publicly revealed in the blacked-out region. Li et al. [43] analyzed the BGP behavior during large-scale power outages from a perspective of both the global and prefix levels. They found there was an increase in the number of withdrawals at the global level. Consequently, there was a sharp decrease in the number of edges and nodes at the prefix level. LaPerriere [39] studied the effect of Taiwan Earthquake fiber cuts from a service provider view. Different from them, we focus on the disruptions caused by DDoS attacks and the impacts of different DPS policies. There are also studies on classifying BGP traffic data into normal and abnormal. Prakash et al. [65] developed an analysis tool named BGP-lens which can find patterns in BGP updates and identify anomalies in these patterns. Zhang et al. [99] proposed a signature-based detection methods to detect BGP anomalies in BGP UPDATES. Zhang et al. [98] proposed an instance-learning framework to identify BGP traffic anomalies based on wavelets. Li et al. [41] proposed a measurement tool called I-seismograph to measure the deviation of BGP dynamics from its normalcy. Caesar et al. [11] designed a BGP health inference system to localize the root causes of routing changes using only BGP update information, which can determine the BGP routing dynamics on the location of potential ASes and the types of routing events. Noroozian et al. [61] performed an in-depth investigation and explanation of DDoS attacks victimization patterns. They found that the bulk of the victims is users in access networks rather than in hosting networks. However, they fail to provide a uniform approach to analyzing the causes of these observed anomalies in BGP traffic. We employ the detection method proposed in [41] to detect abnormal BGP dynamics.

3 A Multilevel Detection of Application Layer DDoS Attacks

This chapter introduces our approach to model users' browsing behaviors and to differentiate between app-layer attacks and benign visits at session level and page level as well as the evaluation result.

3.1 User access patterns

The intention of the visits can be used to distinguish illegitimate users from normal ones [88]. It could be well inferred from user access patterns on the website. We first design a page chain data structure, based on which we model user access patterns. Then, we calculate the likelihood of a session to measure the normality of the browsing behavior.

In an HTTP session, the user browses the website by jumping from one web page to another. Hence, the order of the main pages clicked by a user reflect some relations between these pages. We assume that in a single session the next page a user will browse only depends on the current browsing page, and employ the Markov Chain Model to model user access patterns. The Markov property of user access patterns has been validated in [46]. We further use a directed weighted graph to represent the Markov Chain, where each node represents a main page and the weights of the edge represent the transition probabilities from one page to another. Formally, the transition probability from page i to page j is defined as

$$\sum_{j=1}^{N} p(i|j) = \frac{n_{ij}}{\sum_{j=1}^{N} n_{ij}}, \tag{3.1}$$

where n_{ij} is the number of observations that page i is followed by page j in a single session; N is the total number of pages. Notably, the user access patterns are used to characterize the browsing behavior of aggregated users in a website. The user access patterns are trained based on the page chain data structure to be introduced as follows.

When a browser loads a page, it firstly resolves a dependency graph to determine the request order of associated objects [9, 47, 59, 86]. Therefore, a web request sequence is usually composed of main pages followed by series of associated objects. We adopt the HTTP ON/OFF model to characterize the web browsing behavior. In the "ON" state, the browser sends out a number of requests for the main page and its associated objects. The "OFF" state represents users' page reading period and no request is sent. A schematic diagram of the ON/OFF model is presented in Fig. 3.1. In a browsing session, the user follows a series of hyper-links either provided by the current browsing page or fetched from outside the page such as the navigation tools or the favorites of the browser. We also observe that some object request sequences (mainly images) do not follow main pages. In this case we use a Null page to represent the main page.

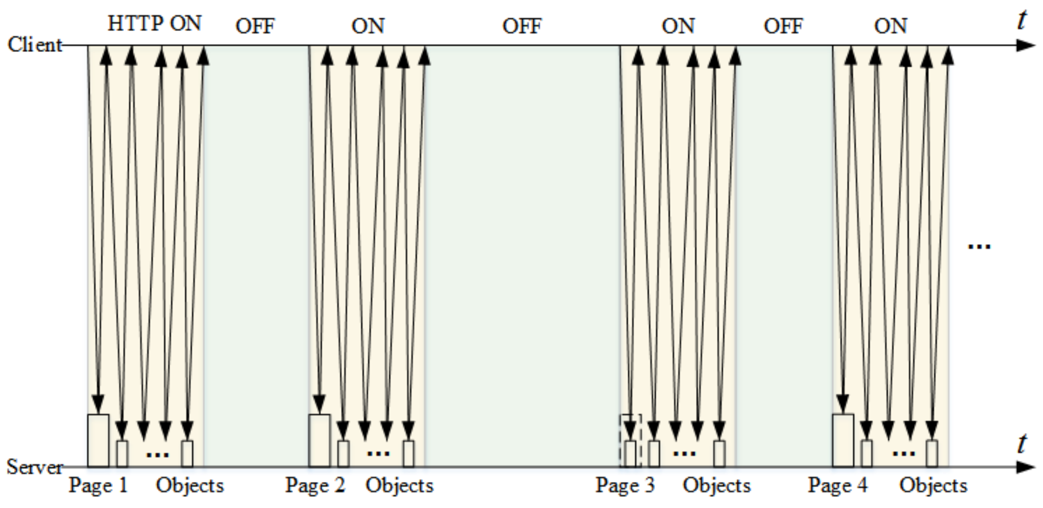


Fig. 3.1: The HTTP ON/OFF model

Based on the ON/OFF model, we design a page chain data structure to profile the click-streams in an HTTP session. We define an HTTP session as the complete browsing process of a user in the website. A closing indicator of an HTTP session is that the maximum HTTP OFF time is greater than a predefined threshold T_{max} (1800s in this thesis). In addition, the main pages are followed by a series of associated objects (AOs), which are requested by the browser automatically. Besides main pages, all the objects are indexed in advance and only the indexes are stored in the data structure for memory efficiency. Since the time intervals between consecutive clicked pages also reflect some aspects of users' browsing patterns, we also store the starting and end times of a browsing session in the data structure. Accordingly, as shown in Fig. 3.2, the page chain structure is defined as below:

1. A main page (MP) is primarily an *html* document whose content type is text/html with links to other objects.

2. The time interval between two consecutive main pages is the difference between the requests' timestamps and equals to the sum of the HTTP ON time and the HTTP OFF time of the preceding page.

3. A page chain is a series of main pages whose time intervals are not greater than T_{max}. That is, $\tau_i < T_{max}, i = 1, 2, \ldots, k-1$.

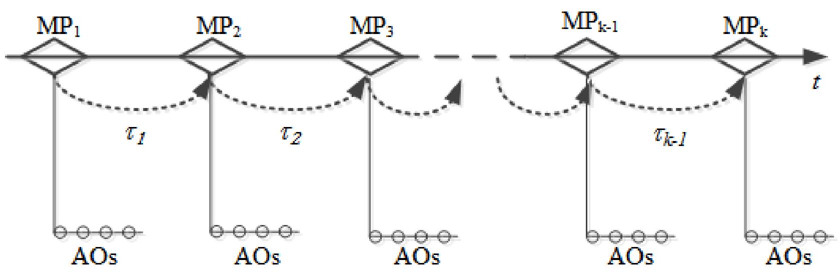


Fig. 3.2: The page chain data structure

The data structure is constructed based on the access logs recorded at the server end. It allows us to learn user access patterns for detection.

3.2 Multilevel Detection

3.2.1 Session level

Since user access patterns are extracted from the aggregate browsing behaviors recorded at the server end, we assume that attackers could not know the access pattern of users. Consequently, the page orders of attacking sequences are quite different from that of normal ones. To evaluate the divergence of attacking sequences from normal ones, we define the likelihood of a session and use it to evaluate the normality of a request sequence. We denote a session as $\{MP_1, MP_2, \ldots, MP_n\}$, where n is the length of the session representing the number of main pages. Then, the likelihood of the session is defined as

$$L = p(MP_1) \prod_{i=1}^{n-1} p(MP_i|MP_{i+1}), \tag{3.2}$$

where $p(MP_1)$ is the probability of page MP_1, and $p(MP_i|MP_{i+1})$ is the transition probability from the i_{th} to $(i+1)_{\text{th}}$ page. For computation convenience, we use the logarithmic likelihood as the metric to determine whether a session is normal or not. That is,

$$\ln L = \ln p(MP_1) + \sum_{i=1}^{n-1} \ln p(MP_i|MP_{i+1}) \tag{3.3}$$

3.2.2 Page level

The likelihood of a session detects abnormal behaviors at session level. However, there are many sessions containing only one main page. The likelihood metric is ineffective to these sessions. Moreover, the likelihood method will take a long time to detect an anomaly if comprised hosts request a large quantity of associated objects (e.g. images). Thus, we propose a detection method at the page level, and each main page usually contains several associated objects. Although a browser resolves a dependency graph to determine the request order of associated objects, browsers do not load image objects synchronously. Thus the order of associated objects is not important to the evaluation of main pages. Lee et al. proposed an sequence order independent method to detect anomaly [40]. However, they also emphasized that the sequence order is unsuitable for profiling browsing behaviors.

Different from their method, we propose a clustering based method to find abnormal page requests. Considering the cache mechanism which is supported by most browsers, there may be some absences of associated objects in a page request. Therefore we represent a request page by an object vector which indicates the absence of associated objects. Formally, suppose there are N_k page requests for page k. Each page request contains a series of associated objects with arbitrary orders. We represent page k by a binary object vector $M_k^{\{0,1\}} = \{w_1, w_2, \ldots, w_n\}$, where the terms in the vector are the objects following the page and w_i is the corresponding weight of the i_{th} objects, $w_i = 1$ if the object i is requested following the request of the main page, otherwise $w_i = 0$. All the object vectors corresponding to the requests of page k are stacked to form the feature matrix M. Since the object matrix is highly redundant, we adopt the principal component analysis (PCA) to transform the samples to new coordinates consisting of principal component. To do PCA analysis, we first calculate the mean vector μ_0 and the covariance matrix $\mathbf{C}$, where $\mu_0 = (\sum_{i=1}^{N} w_i)/n$ and $\mathbf{C} = XX^T/n$, where $X = (w_1 - \mu_0, \dots, w_n - \mu_0)$. Let $\mathbf{u}_j$ denote the j_{th} most significant eigenvector and $\mathbf{U}$ the significant principal components. We form the matrix $\mathbf{T}$ by normalizing the rows of $\mathbf{U}$, that is to set $t_{ij} = u_{ij}/f\sqrt{\sum_k u_{kj}^2}$. Each row of $\mathbf{T}$ corresponds to a point in the high dimensional space. We then employ the DBSCAN clustering method [7] to classify the points into different categories, and regard the outliers of these points as anomalies.

3.3 Experiments

3.3.1 Dataset

Table 3.1: Summary of the dataset

Date	Requests	Users	Max. RR^1	Min. RR^1	Suspected IPs
2015/12/29	30,933,159	30,242	283	20	845
2015/12/30	32,202,986	32,886	290	18	1023
2015/12/31	30,850,731	31,063	341	19	1139
Total	93,986,876	74,773	-	-	1270

RR^1 is the abbreviate of request rate with a time unit of second.

In stead of using simulated attacking data, we conduct experiments to evaluate our detection method based on real attacking data recorded by a targeted commercial web server. The dataset contains three days (Dec. 29-31, 2015) access logs. Table 3.1 lists a brief summary of the dataset. We can see that the victim experienced attacks with similar strength in the observed three days. The total number of unique users is much less than the sum of the number of unique users observed in each day. This indicates that a large number of compromised hosts persistently attacked the victim in all three days. The request rate is defined as the number of requests received by the server in a time unit. The maximum and minimum of the request rates are also listed in the table. The server suffered the strongest attack on December 31, which has a maximum request rate as high as 341 requests per seconds. Different from all previous reported DDoS attacks, the attack only persisted for one minutes. However, such attack comes out repeatedly (see Fig. 3.6). We also list the number of suspected IPs blocked by the operator due to the high consumption of resources such as bandwidth, CPUs, and memory.

3.3.2 Performance evaluation

Since the dataset contains both normal and attack requests, it is hard to differentiate them for training the model, and thus we select the guaranteed normalized data to prevent deviation. Specifically, we obtain the guaranteed normal data using the following criterion: users who browse the website only in one session and the session length is between 2 to 100; Moreover, the repetition of a single page should not exceed 2. This results in a moderate dataset with 15,730 users and 10,089,497 requests. As expected, the access patterns are closely related to the web structure, which exhibits hierarchical clusters. We apply hierarchical cluster methods on the transition matrix of the trained access patterns, and the results are shown in Fig. 3.3. We only present the results of the top 80 most accessed pages, which dominate 90% of the total requests. The website has a total of 8464 pages and 14036 objects. Then, we use the transition matrix to calculate the likelihood of all sessions, and the results versus the session length are shown in Fig. 3.4. It shows that there are some outliers for different session lengths, demonstrating that our methods can distinguish the abnormal sessions from the normal ones.

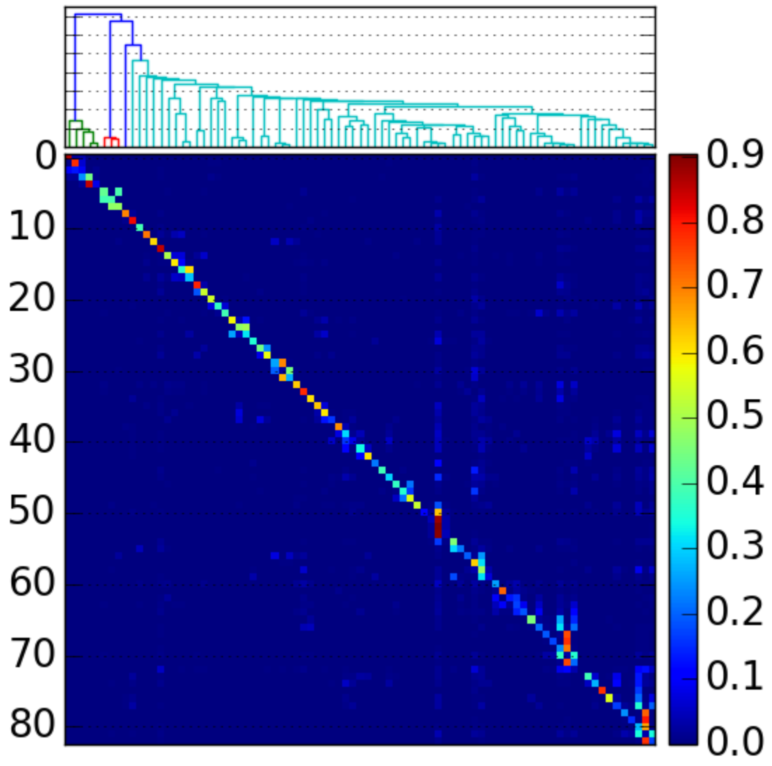


Fig. 3.3: Hierarchical structure of the website

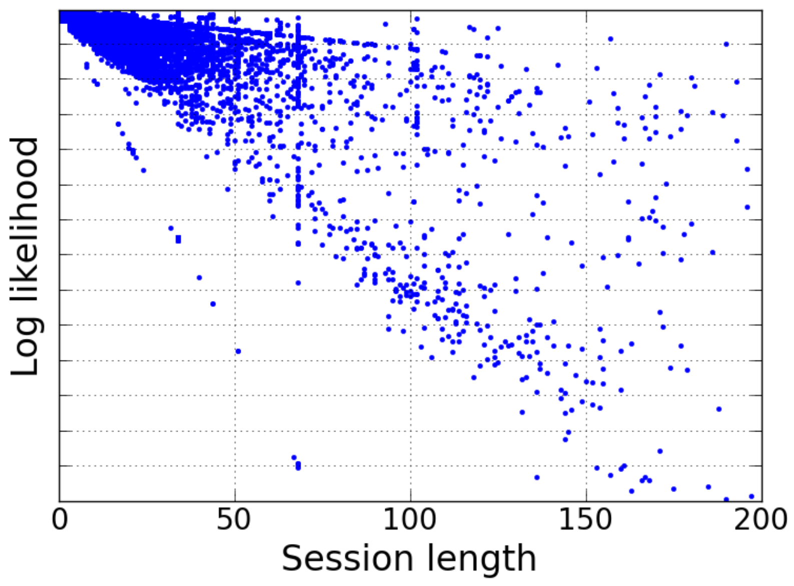


Fig. 3.4: Likelihoods of sessions versus session length

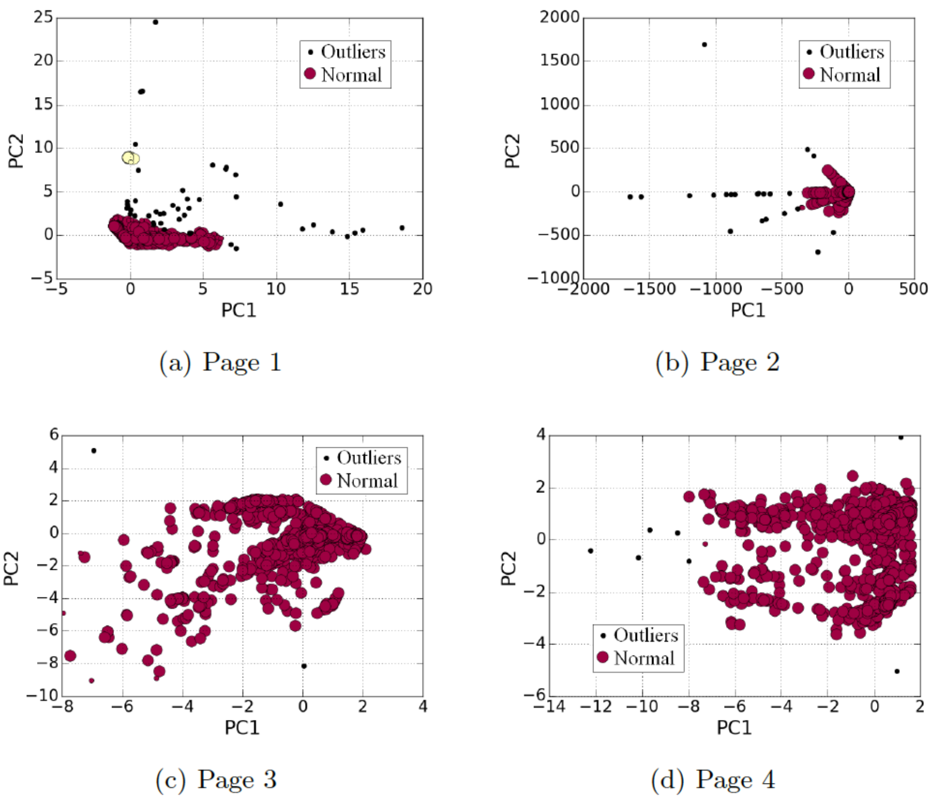


Fig. 3.5: Anomaly detection at the page level

For each request of the main page, we construct an object vector according to the following objects. For presentation convenience, the object vectors are decomposed into 2D points by principal component analysis. Fig. 3.5

Table 3.2: Detection results

Detection methods	TP rate (%)	FP rate (%)
Likelihood method	99.32	2.69
Clustering method	97.16	2.52
Combined method	99.81	2.71
HsMM	95.13	1.92
SOI method	93.7	3.86

presents the detection results of four selected pages. It is shown that Page 1 and 2 are under heavy attacks. Compared to Page 1 and 2, Page 3 and 4 have more diverse access patterns. This indicates that Page 3 and 4 contains much more cache-able content. These results are consisted with the insight analysis of the pages.

We use the true positive rate(TP) and false positive rate(FP) to evaluate the detection method. Table 3.2 shows that TP rate has the highest rate at the session level, indicating it can detect most of the suspected IPs blocked by the operator of the server. However, this method also results in a higher false-positive rate compared to the clustering method. We also combined the likelihood and the clustering method to trigger an alarm, which could achieve a higher accuracy on the cost of a high FP rate. We also compared our method with the state-of-art methods. It shows that the hidden semi-Markov model (HsMM) detection method [90] has both lower TP and FP rates. The HsMM uses a hidden state to estimate the browsing page while we directly obtain the browsing page by the file type of requested content. The sequence-order-independent (SOI) method [40] achieves a relatively lower detect accuracy but a higher FP rate because the SOI method is unable to detect the anomaly at the session-level.

Following [75], we conducted statistical experiments to further evaluate the effectiveness of our method further. Let n_t be the number of requests received by the server in a time unit, which is plotted in Fig. 3.6. We observed that the server suffered periodic pulsing DDoS attacks, which result in the comb-shape. Fig. 3.7 shows the request rate after filtering out the attack traffic based on the detection results of the combined method. We can see that the detection method is effective in reducing the burden of the server. In addition, it is noticeable that the request rate varies periodically, suggesting

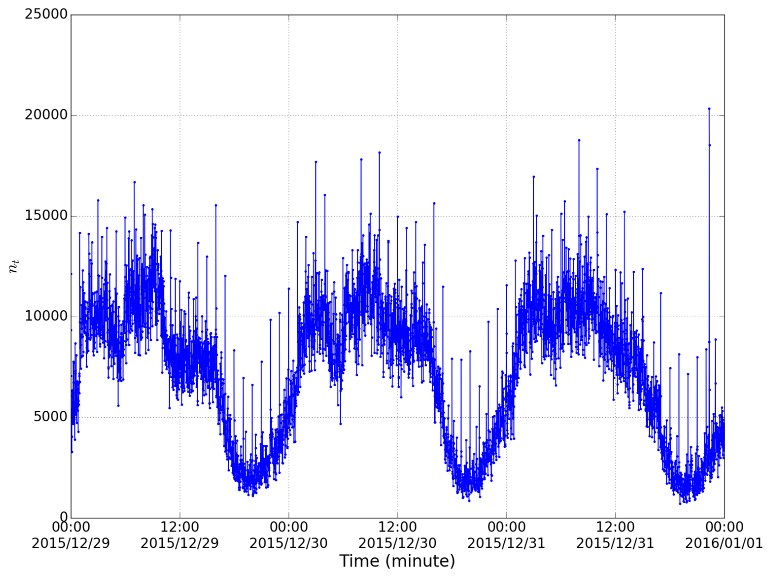


Fig. 3.6: Original request rate

that detection methods should avoid the impacts of fluctuations raised by such periodicity.

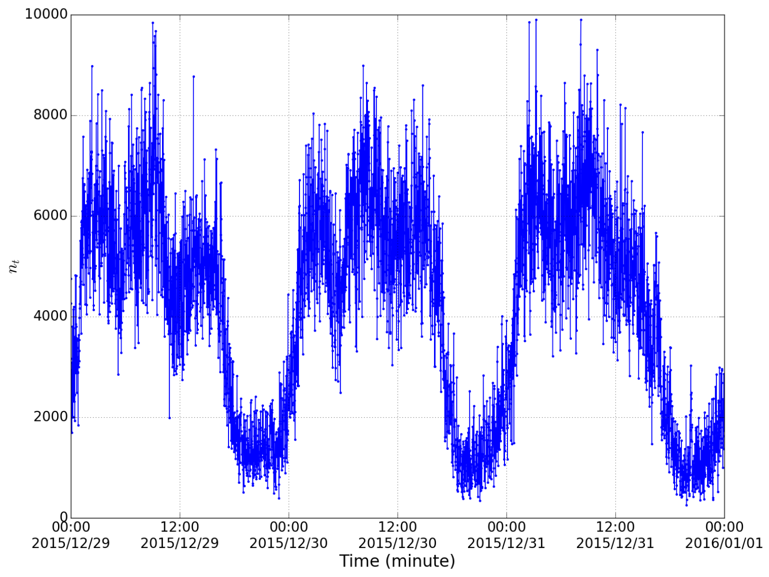


Fig. 3.7: The request rate after the attack traffic is removed.

It has been reported that the access frequency of pages follows the Zipf distribution. We compute some statistics of the collected data, and the results are shown in Fig. 3.8. The original data violates the Zipf distribution at both the head and tail of the distribution. The filtered data obeys the Zipf distribution at the tail after we filter the attacking request. Hence, the violation at the tail may be caused by attacking requests, and the violation at the head may be due to the requests to some extremely popular web pages.

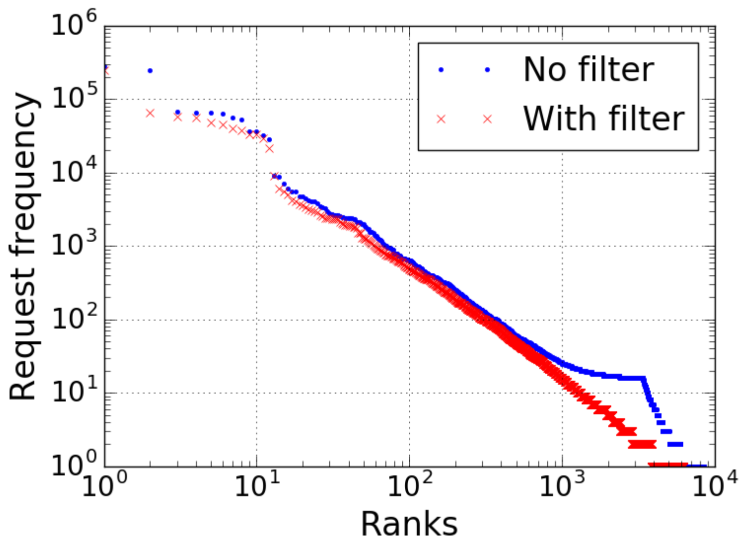


Fig. 3.8: Page access frequency distributions

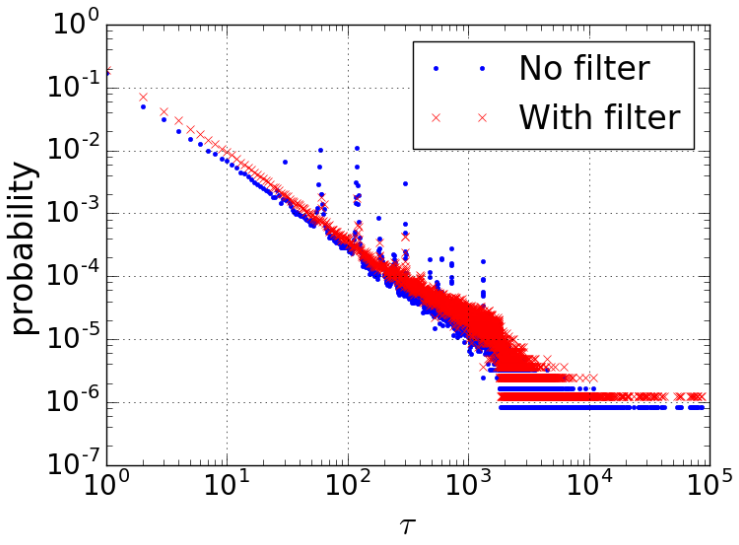


Fig. 3.9: Time interval distributions

Another common characteristic of web browsing behavior is that the browsing time for each page follows Pareto distribution. Fig. 3.9 shows the distribution of the inter-request times between two consecutive accessed pages. We can see that there are some fluctuations for the unfiltered data, indicating that the server suffered DDoS attacks by a large volume of requests with common time intervals. However, the inter-access time of the filtered data indeed follows Pareto distribution, demonstrating the effectiveness of our methods.

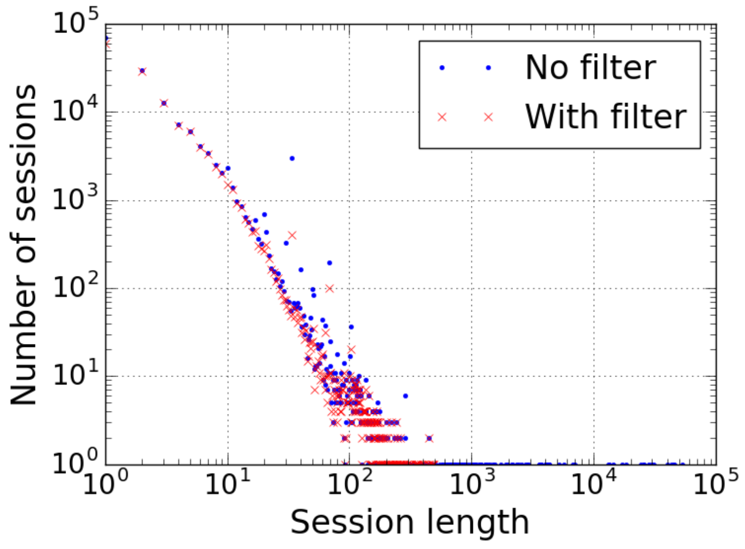


Fig. 3.10: Session length distributions

The session length, defined as the number of requested pages in the session, affects the likelihood of the session. Fig. 3.10 plots the distribution of session length. It is shown that the unfiltered data contains very long sessions, which are caused by periodically repeated attacks. The unfiltered data also contains some unexpected high probability session lengths around 40, indicating that the attackers may also use sophisticated access patterns in the attack besides repeating the requests. The session length of the filtered data follows Pareto distribution, which is consistent with the results in [31].

3.4 Summary

We propose a new mechanism to detect app-layer DDoS attacks by modeling users' browsing behaviors according to the access log at the sever end and to differentiate between app-layer attacks and benign visits at session level and page level. The experimental results on the real dataset show the effectiveness of our approach. The content of this chapter is published in [56].

4 A Sketch-Based Defense Against Application Layer DDoS Attacks

This chapter introduces our sketch-based system to detect and mitigate app-layer DDoS attacks. We first introduce the background knowledge of sketches and bloom filters and then describe the architecture and major components. After that, we present the implementation and the evaluation result.

4.1 Sketches and bloom filters

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. 4.1 shows a diagram of the sketch data structure.

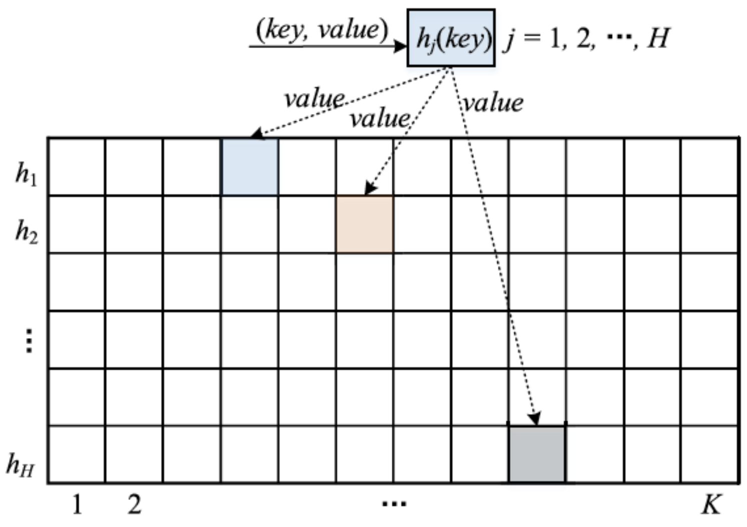


Fig. 4.1: A diagram of the sketch data structure

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 $value$.

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 [36]. We adopt the sketch techniques for anomaly detection and malicious hosts identification.

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. 4.2 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.

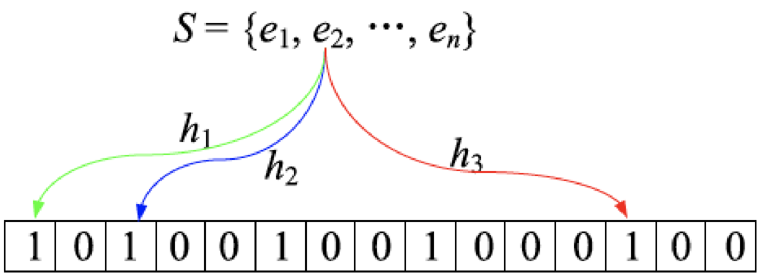


Fig. 4.2: A diagram of the sketch data structure

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 [8]. We employ Bloom filters to implement the black and white lists that are utilized to filter requests from malicious hosts.

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.

4.2 System overview

We propose SkyShield, an effective defense system to quickly detect and mitigate application layer DDoS attacks. Fig. 4.3 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.

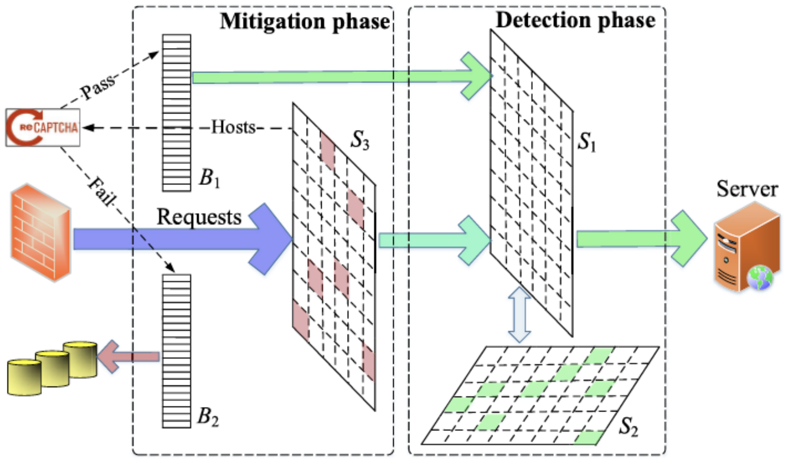


Fig. 4.3: The process of SkyShield

In the mitigation phase, SkyShield employs two Bloom filters, including a whitelist ($B1$) and a blacklist ($B2$) 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 S3. 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 chapter 4.3.

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 chapter 4.4.

4.3 Attack mitigation

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 malicious hosts (or bots) to generate a larger number 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 $\langle n_i 1, n_i 2, ..., n_i K\rangle$, 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:

$$g = \lfloor (\ln N_i)^r \rfloor, \tag{4.1}$$

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

$$A_i = \{j | n_{ij} \geq n'_{ig}\}, \tag{4.2}$$

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

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 $h_i(IP)$, respectively. If $h_i(IP) \in A_i$ is true for all $i = 1, 2, ..., H$, then we label the IP address as suspicious and the host will be verified by the CAPTCHA module.

Fig.4.4 demonstrates the flow chart of the identification procedure. The mitigation module first checks whether the host is in the whitelist or blacklist. If not, check the host against S_3. Then test the suspicious host through graphic puzzles. 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 will be 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 after a period of time, we clean the blacklist and the whitelist periodically with a longer interval than the detection cycle. In this report we set 100 ΔT (i.e., 2000s) as the default clean period. The

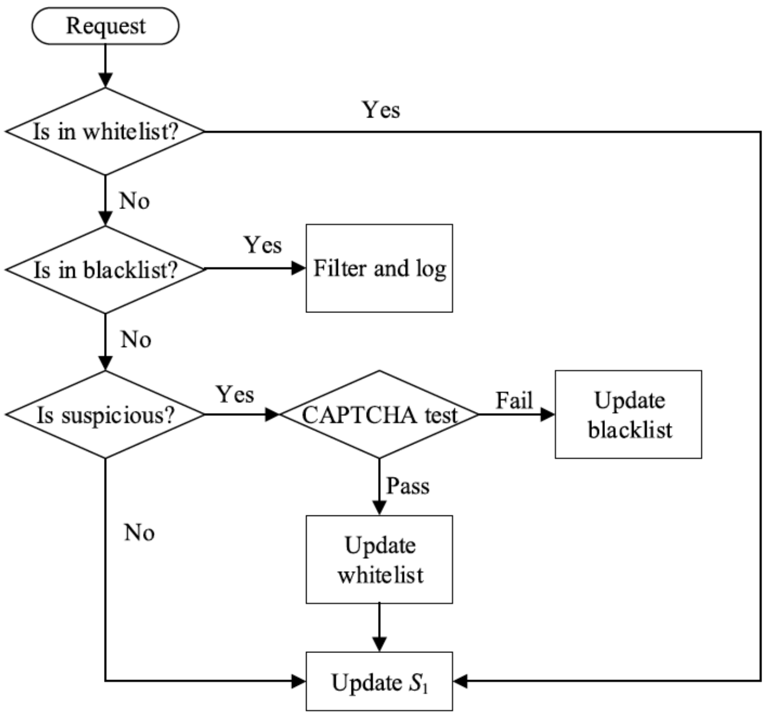


Fig. 4.4: The flow chart of the identification procedure

reason for selecting this value is that 80% of the DDoS attack intervals last less than 1081 seconds [83]. 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 at the start of a new attack. We can also choose a much longer clean period (e.g., a day or even a week) to block malicious hosts longer period of time.

The rationale behind the mitigation scheme is that malicious hosts detected in the current detection cycle are likely to appear in the next detection cycle. Attackers can evade the system by violating this assumption. However, such an attempt will greatly increase the cost of an effective attack.

4.4 Anomaly detection

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 stable. It is worth noting that this stability does not contradict the dynamic meaning of the network describing the uncertainty of a single request (including source IP address, request time, body size, etc.). Although the total network traffic of all normal users is usually stable in a short period of time, the request rate of each user may vary greatly. Since the sketch maps the request rate of a single user into a single bucket, network dynamics will affect the measure of divergence between the 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 $\langle n_{i1}, n_{i2}, ..., n_{iK}\rangle$, 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 $P_i = \langle p_{i1}, p_{i2}, ..., p_{iK}\rangle$ 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 $P_i = \langle p_{i1}, p_{i2}, ..., p_{iK}\rangle$ and $Q_i = \langle q_{i1}, q_{i2}, ..., q_{iK}\rangle$, the Hellinger distance is defined as

$$d(P_i, Q_i) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{K}(\sqrt{p_i j} - \sqrt{q_i j})^2}, \tag{4.3}$$

However, it is not appropriate to use such a distance to measure the divergence between the two sketches, because the uncertainty of the source IP address in the dynamic network traffic makes the probability distribution in each row of the sketch change drastically, therefore, the unpredictable divergences Will cause a high false positive rate. False alarm rate. What's worse, since the last normal sketch is stored as the baseline of the normality when an anomaly is detected, using the original Helinger distance between the two probability vectors may result in continuous false alarms even after the actual attack stops.

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 $P'_i = \langle p_i(1), p_i(2), ..., p_i(K)\rangle$ and $Q'_i = \langle q_i(1), q_i(2), ..., q_i(K)\rangle$, where $p_i(1) \geq p_i(2) \geq \cdots \geq p_i(K)$ and $q_i(1) \geq q_i(2) \geq \cdots \geq q_i(K)\rangle$. The improved divergence between Pi and Qi is calculated by the Hellinger distance between $P'i$ and $Q'i$:

$$d'(P_i, Q_i) = d(P'_i, Q'_i) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{K}(\sqrt{p_i(j)} - \sqrt{q_i(j)})^2}, \tag{4.4}$$

According to Equation (4.4), we can calculate the divergence between the corresponding hash tables of two sketches.

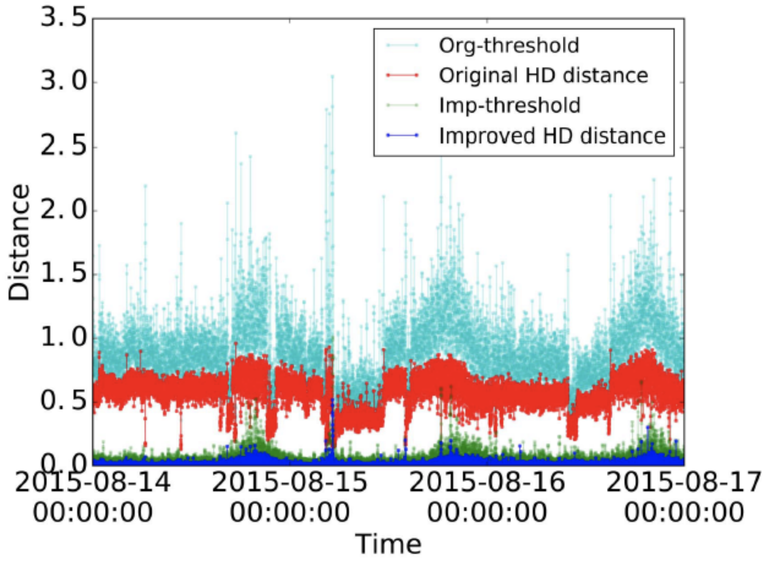


Fig. 4.5: Different divergences of sketches versus the time.

Fig.4.5 shows the comparison between the original and improved Hellinger distance of the hash table of one of our datasets. We can see that the improved divergence is much lower and more stable than the original divergence. This benefits from the randomness of the hash algorithm, the sorting probability distribution of the hash table is more predictable for normal network traffic.

Dynamic Threshold Estimation
Since network traffic is dynamic in nature and fluctuates continuously over 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 [58] to obtain the adaptive threshold. However, if we use the average divergence of the hash tables for detection in the two sketches, a large deviation of any single hash table pair may result in a large deviation of the average deviation, resulting in a higher false alarm rate. In fact, we

tend to be conservative. 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

$$\hat{d}_{t+1}^{(i)} = \alpha d_t^{(i)} + (1 - \alpha)\hat{d}_t^{(i)}, \tag{4.5}$$

$$e_t = |\hat{d}_t^{(i)} - d_t^{(i)}|, \tag{4.6}$$

$$\sigma_{t+1}^2 = \beta e_t^2 + (1 - \beta)\sigma_t^2, \tag{4.7}$$

$$\theta_{t+1)}^{(i)} = \hat{d}_{t+1}^{(i)} + \lambda\sigma_{t+1}, \tag{4.8}$$

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.4.5 also shows the thresholds of the hash tables used for different calculations of the hash divergence. We can see that the new definition of Hellinger distance is more stable than the original threshold.

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

4.5 Implementation

Architecture

Fig.4.6 show the implementation architecture of SkyShield. We adopt parallel techniques to hash incoming requests from hosts with multiple cores to improve efficiency. When the detection module finds an abnormality, it passes the abnormal sketch to the mitigation module. After the mitigation module receives the abnormal sketch, it will identify malicious hosts and filter out all requests from these hosts.

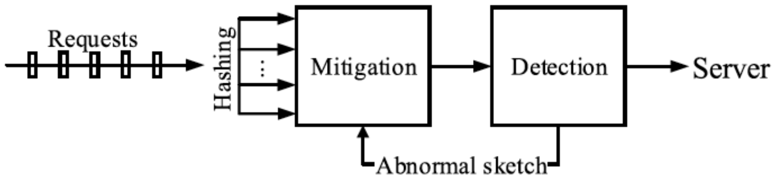


Fig. 4.6: The implementation architecture of SkyShield

Parameter Configurations

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

$$h_{a,b}(x) = (ax + b) \bmod p, \tag{4.9}$$

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.

Selection of Hash Functions

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 $|A_1|$, $|A_2|$, . . . , $|A_H|$, respectively. Then the FPR for completely random hashing is $N\frac{\prod_{i=1}^{i=H}|A_i|}{K^H}$ approximately, where $|Ai| \ll K$ 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_{τ^H}, where $0 < \tau \ll 1$. For a given constant false positive rate bound $\epsilon \geq N_{\tau^H}$, we have

$$H \geq \frac{\ln N - \ln \epsilon}{-\ln \tau}, \tag{4.10}$$

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 x K is small enough to fit the sketch into fast memory in order for updates to be performed at high speed. In this report we set $K = 2^{12}$ following [73]. 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

$$p = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m}, \tag{4.11}$$

The probability of a false positive is

$$f = (1 - P)^k = (1 - e^{-kn/m})^k, \tag{4.12}$$

By minimizing f as a function of k, we obtain

$$k = \frac{m}{n} \ln 2, \tag{4.13}$$

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

$$f = (1/2)^k = (1/2)^{\frac{m}{n}\ln 2} \leq \epsilon, \tag{4.14}$$

Solving the above inequality we get

$$m \geq \frac{n \log_2(1/\epsilon)}{\ln 2} n \log_2 e \log_2(1/\epsilon), \tag{4.15}$$

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 $m \geq n \log^2 e \log^2(1/\epsilon) \approx 2396264.6$. 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 [30]. The parameter β is used to smooth the estimates of the variances and the value is suggested to be smaller than 0.5 [30].

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 (4.1) 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 4.1 summarizes the parameters of SkyShield and their default values, which are optimized by empirical experiments.

Table 4.1: Default Value 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	0.6
Others	ΔT	Detection time intervals	20s
	τ	An intermediate parameter	2.0

4.6 Experiments

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

Datasets

We collect the datasets from a large-scale web cluster that manages the traffic of about 200 customer websites. Fig.4.7 illustrates the architecture of the cluster that employs NGINX servers as reverse proxies to serve the customer websites.

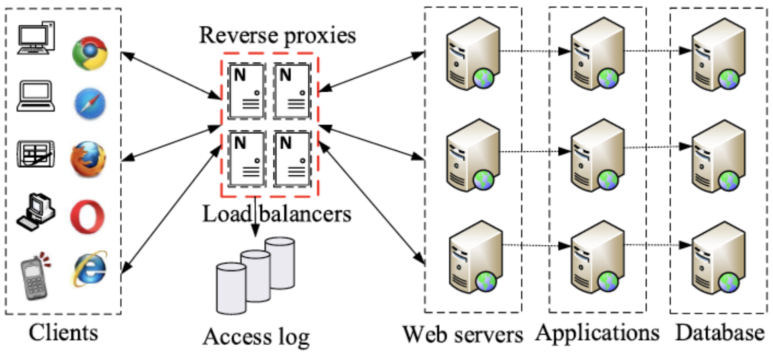


Fig. 4.7: The architecture of the reverse proxy cluster

The load of reserve proxies is scheduled by IP-hash based balancers, and thus the deterministic balancers will handle requests from a specific source IP address. The balancers record all processed requests in the access log. The cluster also includes a CAPTCHA module to test whether an incoming request is malicious. In order to reduce the impact on QoE, the CAPTCHA module is configured to work in sampling mode, and the incoming request is tested with a probability of 0.01. Requests that fail the test will be recorded in the mitigation log. Since attacking hosts usually send a large number of requests, the possibility of CAPTCHA module testing malicious hosts is very high. For instance, the probability that the host has sent more than 200 requests to be recorded in the mitigation log 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 regarded it as the basic fact of the attacking host.

We obtained three datasets from the cluster, each of which contains access logs for three days, and reported an attack in the middle day based on the customer's report. We also extracted mitigation logs during the corresponding period of these datasets. In the following experiments, we refer to dataset date from 2015/08/14 to 16 as Dataset1508, the date dataset from 2016/03/16 to 18 as Dataset1603, and the date dataset from 2016/04/13 to 15 Called Dataset1604, respectively. Table 4.2 briefly describes these datasets. The second column lists the number of requests in each dataset. By combining the requests in the normal access log with the requests in the mitigation log, the total number of requests can be obtained. The huge difference in the total number of requests is caused by seasonal tides and different attack traffic. The 3rd and 4th columns respectively list the number of different hosts and different malicious hosts (mal-hosts).

Table 4.2: 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	$56,725,591$	$297,780$	$43,274$

Parameter Evaluation

We employ the true positive rate (TPR), false positive rate (FPR), and the ratio of filtered malicious requests to the total number of attack requests (Fraction) as the criteria for evaluating the impact of parameters on SkyShield performance. The True Positive Rate (TPR) measures the proportion of attacking hosts that are correctly identified as malicious by SkyShield. The false positive rate (FPR) measures the proportion of benign hosts that are incorrectly identified as malicious by SkyShield. By adjusting the value of a parameter within an appropriate range, while keeping other parameters as default values, the influence of the parameter can be evaluated.

1) Impact of MIEWMA's Parameters: Fig.4.8 shows the α evaluation results based on three data sets. Each subfigure contains three lines, where red represents FPR, black represents TPR, and blue represents the fraction of malicious requests that have been filtered. The results show that the impact of the parameter α on the detection result is negligible, because the attack usually causes the divergence between the normal sketch and the abnormal sketch to change sharply and hence the detection results are not sensitive to the weights of history data in the calculation of the EWMA statistic.

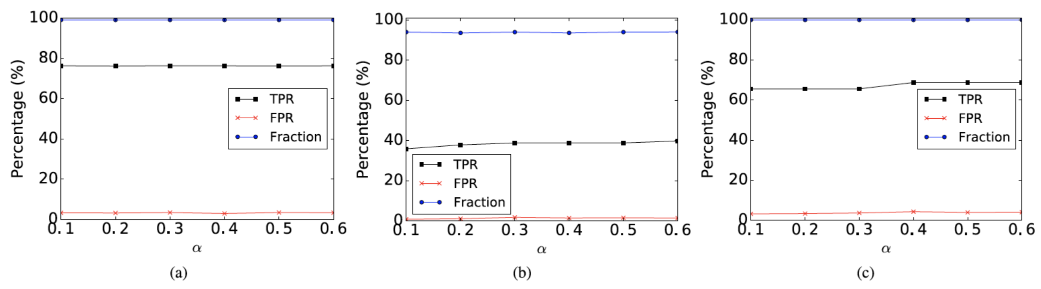


Fig. 4.8: Impact of parameter α. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

The evaluation results of the parameter β are shown in Fig.4.9. According to the evaluation results, we set α to 0.3 and β to 0.4 as their defaults.

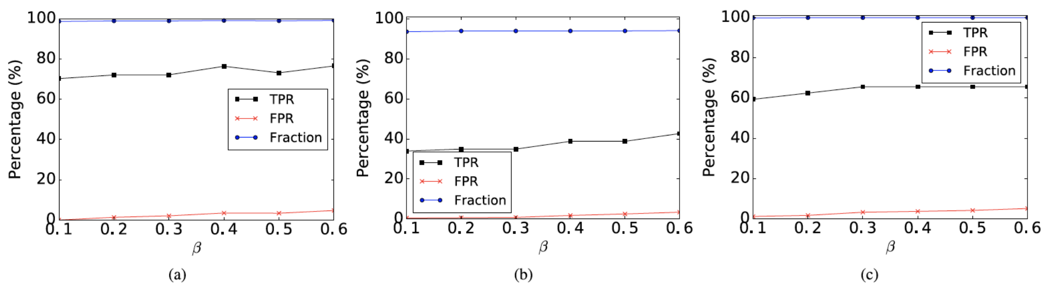


Fig. 4.9: Impact of parameter β . (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

Fig.4.10 demonstrates the results of λ's evaluation. We can see that the detection results are very sensitive to changes in λ. Both TPR and FPR decrease with the increase of λ. This is due to the fact that a larger λ will result in a higher threshold and therefore a higher tolerance for divergence changes. Subsequently, the system became more conservative to issue alarms and could allow more real attacking hosts to pass, resulting in a decrease in TPR. Meanwhile, once an alert is issued, it will also give green light to more benign hosts that may be mistaken for malicious. This explains the downward trend of FPR. In practice, operators usually want a lower FPR to maintain a higher QoE, provided that the attack does not significantly affect the availability of the service. Therefore, we select $\lambda = 6.0$ as the default value in SkyShield.

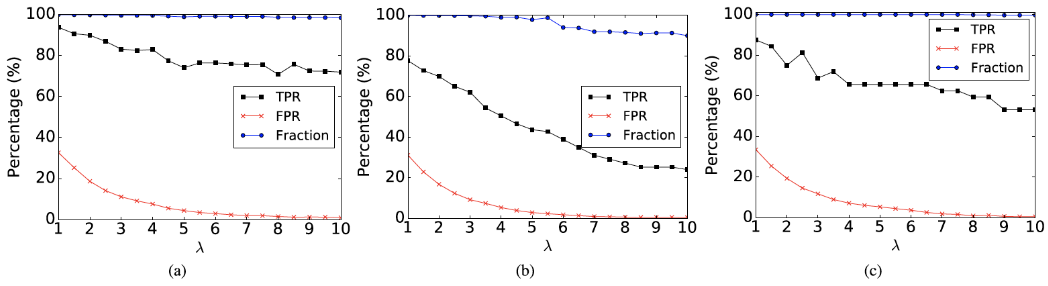


Fig. 4.10: Impact of parameter λ. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

2) The impact of ΔT and r : Fig.4.11 shows the impact of ΔT. The results show that the FPR of all datasets decreases monotonously with the increase of T. As ΔT increases, TPR first increases monotonically when $\Delta T \leq 20$s, and then drops sharply when $\Delta T > 20$s. However, the fraction of filtered malware is always significant and robust to changes in ΔT. Considering all these factors, we select $\Delta T = 20$s as the default detection interval.

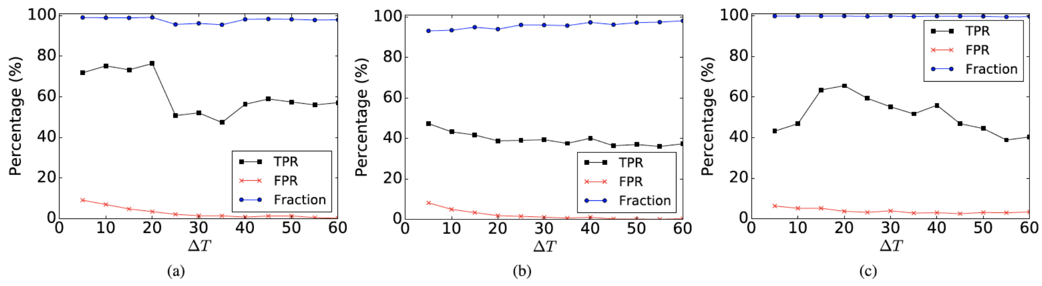


Fig. 4.11: Impact of parameter ΔT . (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

Fig.4.12 depicts the evaluation results of r. The results demonstrates that both TPR and FPR increase with the increase of r. As r increases, more buckets will be classified as abnormal. Thus, more malicious hosts and legitimate hosts are filtered out, resulting in an increase in 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 the default value in SkyShield.

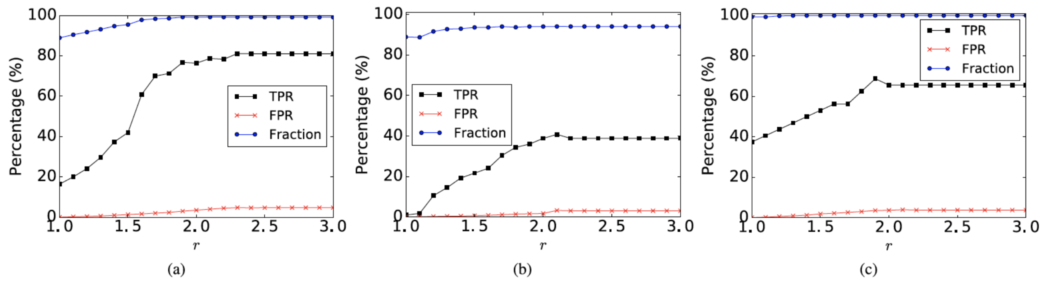


Fig. 4.12: Impact of parameter r. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

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 the protected system from being overwhelmed by numerous malicious requests.

Evaluation of the effectiveness of mitigating various DDoS attacks
Since SkyShield aims to reduce attack traffic as soon as possible when there is an app-layer DDoS attack, we have evaluated its effectiveness in filtering malicious requests. We simulate the real network environment by replaying the original request flow. The request rate is defined as the number of requests in the detection cycle. Fig. 4.13 to 4.15 present the relationship between request rates and time for the three datasets. In each subfigure, the blue line represents the original request rate, and the red line represents the request rate filtered by SkyShield.

Fig.4.13 (a) shows that the cluster suffered a flood attack at 2015-08-15 04:21, and was quickly overwhelmed by a large number of requests. Fig.4.13 (b) illustrates the detailed information during the attack from 04:00 to 06:00 on 2015-08-15. We can see that the cluster has suffered five attacks during this period. The peak request rate of attack traffic was around 290,000 request per second. 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.4.13 (c) display the detailed information at the beginning of Dataset1508. The filtered curve will follow the original curve for a long time, which indicates that the false alarm rate of SkyShield is low.

Fig.4.14 (a) illustrates an overview of the request rate of Dataset1603. The peak request rate of attack traffic was around 18,000 request per second.

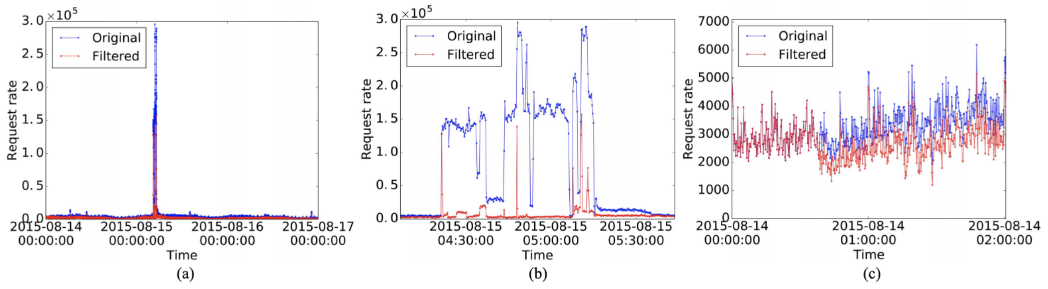


Fig. 4.13: Experimenal results of Dataset1508. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start

Unlike the short-term overwhelming attack shown against Dataset1508, the attack in Dataset1603 persisted much longer. In addition, the total request volume is relatively small and volatile, which shows that SkyShield faces challenges. The results showed that the system suffered a slow attack in three days. However, SkyShield can still effectively reduce the request rate to a reasonable level. Fig.4.14 (b) display the details of the attack on Dataset1603.

Fig.4.14 (c) shows the beginning of the dataset 1603. As shown in the black frame, we can see that the rate of attack requests increases very slowly. However, SkyShield will still detect anomalies, thus mitigating malicious requests. This is because SkyShield can detect changes in the distribution of request numbers in the sketch, so that it can identify malicious hosts that have caused an abnormal increase in the request rate.

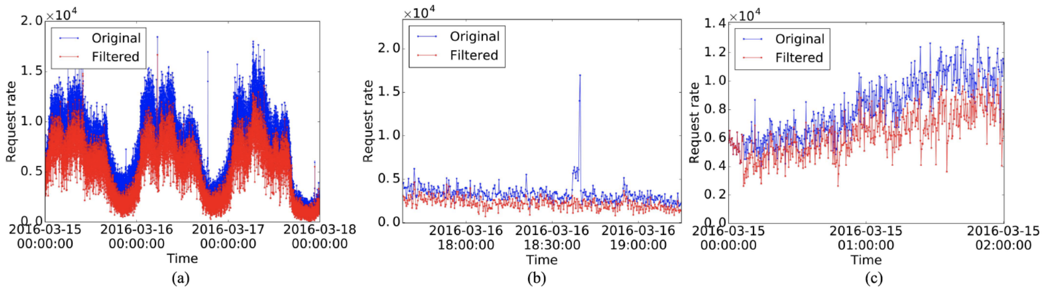


Fig. 4.14: Experimenal results of Dataset1603. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start

Fig.4.15 (a) shows the request rate of Dataset1604. The peak request rate of attack traffic was around 23,000 request per second. It can be seen that the system encountered a low request rate attack within a 30 minute interval of three days. In-depth knowledge of the original access log indicated that the attack was targeted at one of the web servers hosted in the cluster. Presumably, this attack was caused by slow rate crawlers that visited the site at regular intervals. Fig.4.15 (b) is the detail of the attack on Dataset1604. There is an obvious spike on the filtered request rate curve, and within only one detection cycle, the curve returns to its normal level. Fig.4.15 (c) illustrates the start of details of Dataset1604. The results verify the effectiveness of our system in mitigating such suspense traffic. However, these crawlers may be benign users, such as search engines or even partners. The deployment of SkyShield forces these users to change their access policies to avoid being filtered.

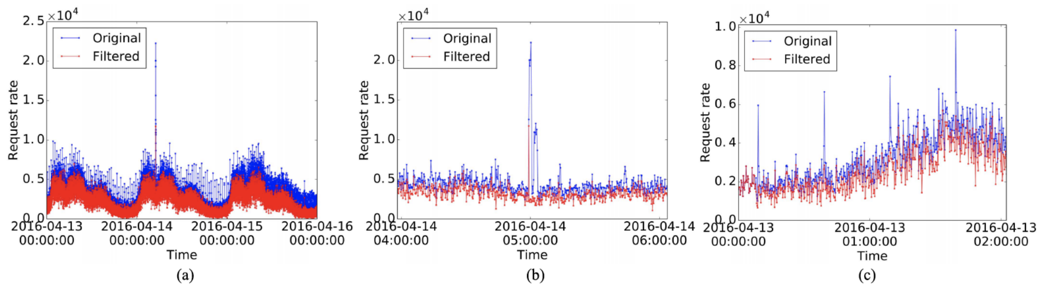


Fig. 4.15: Experimenal results of Dataset1604. (a) Request rate overview. (b) Detail of the attack. (c) Detail of the start

Fig.4.16 depicts the relationship between the number of hosts in the blacklist and time. In this experiment, we block each detected malicious host with $100\Delta T$ (i.e., 2000s) instead of clearing the blacklist, which may provide us an insight into how bots cooperate with each other to launch an attack. The results showed that during the attack, the number of hosts blacklisted by Dataset 1603 was much smaller than the number of hosts blacklisted by Dataset 1508. Compared with other datasets, Dataset1604 has the least number of blacklisted hosts and fluctuates greatly. This is because most blacklisted hosts are crawlers. These crawlers simultaneously send a large number of requests to specific Web servers at regular intervals of 30 minutes. Since the blacklist is periodically cleared, fluctuations are caused by the release and recapture of these crawlers. We also conducted experiments to evaluate the number of blocked hosts with another cleanup period of $150\Delta T$ (i.e., 3000s). We obtain very similar results but with higher number of blocked hosts. In addition, the scores of TPR, FPR, and filtered malicious requests are also close to the results shown in Table 4.3, indicating that the choice of clearing time has a limited impact on the performance of SkyShield. We ignored these results because they are very similar to the results shown in Fig.4.16 and Table 4.3.

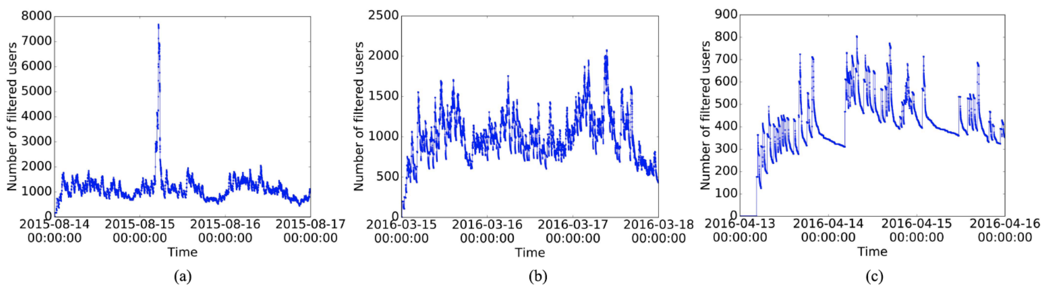


Fig. 4.16: The number of hosts in the blacklist versus the time. (a) Dataset1508. (b) Dataset1603. (c) Dataset1604

Table 4.3: PERFORMANCE OF SKYSHIELD WITH DEFAULT PARAMETERS

Dataset	TPR%	FPR(%)	Fraction (%)
Dataset1508	76.4	3.45	99.1
Dataset1603	38.9	1.77	94.0
Dataset1604	65.6	3.67	99.9

The above experimental results validate that SkyShield can quickly detect and mitigate various types of app-layer DDoS attacks. Table 4.3 lists the percentages of TPR, FPR, and filtered malicious requests for different datasets with default optimal parameters. It shows that the TPR of Dataset1603 is much lower than the TPR of the other two datasets. This is because compared with the other two data sets, the attack duration in Dataset1603 is longer and the attack intensity is weaker. This results in a more even distribution of requests in the detection cycle than other requests. In addition, the number of abnormal buckets is a function of the number 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 compared the performance of SkyShield with the original HD distance and the improved one. Fig.4.17 demonstrates the obtained results. Fig.4.17(a) and Fig.4.17(c) show that with the new HD distance, both TPR and fractions are improved. Although the new HD distance may cause an increase in FPR, as shown in Fig.4.17(b), this increase is negligible compared to the benefits of TPR (Note the placement of the decimal points in Fig.4.17(b)). This increase is due to the new HD distance sorting the request numbers in the sketch before calculating the distance, so it is more sensitive than the original distance.

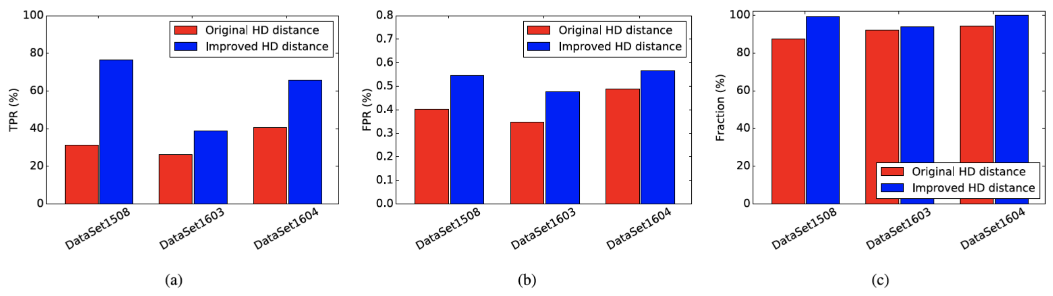


Fig. 4.17: Performance comparison between the original and the improved Hellinger Distances. (a) TPR. (b) FPR. (c) FRA

Evaluation of the efficiency of processing the large volume of requests
The efficiency of SkyShield is mainly affected by the number of hash functions.

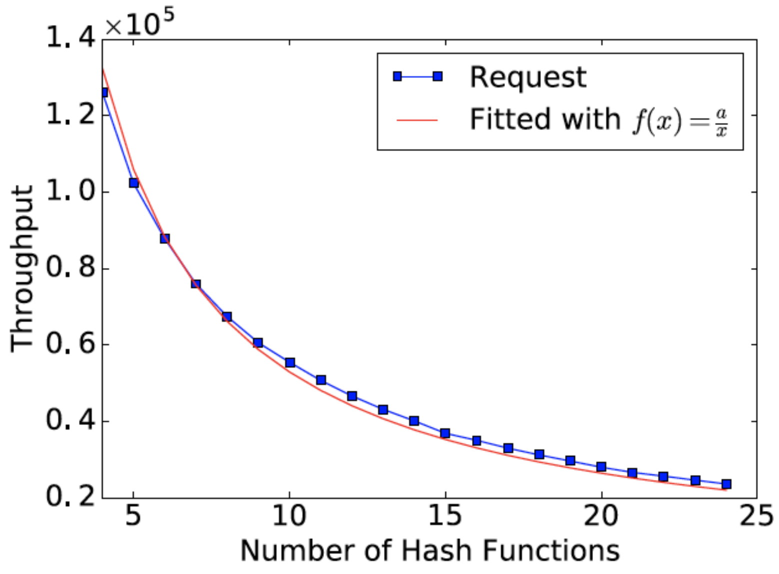


Fig. 4.18: The throughput of the system in a single thread

To evaluate SkyShield's capability to handle large-scale flooding attacks, we defined throughput as the number of requests processed by the system per second. The processing time is mainly consumed by the calculation of the hash function. Fig.4.18 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:

$$f(x) = \frac{a}{x}, \tag{4.16}$$

Where a is the parameter to be estimated.

The least squares fitting method 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 wellformed HTTP requests and thus such a request rate is relative high. Therefore, the system is capable of handling large-scale flooding attacks.

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

Performance of Mitigating Flash Crowd Mimicking Attacks
App-layer DDoS attacks utilize legitimate HTTP requests to overwhelm victims. To make matters worse, attackers prefer to launch attacks during flash crowd events or mimicking flash crowd events to avoid detection. We employ WorldCup98 data [5] to validate the effectiveness of SkyShield. The reason for using this data set is that it has a similar scenario with our application, and it is guaranteed that all requests in this data set are normal. We invite readers to refer to [5] for a detailed description of the data.

We conduct two experiments. The first goal is to test whether SkyShield interferes with normal user access during flash crowd incidents. In the experiment, we applied SkyShield to WorldCup98 data and checked whether an alarm was issued. Fig.4.19(a) shows that when the website experienced four waves of flash crowds, SkyShield dose not issue any alarms on the original WorldCup98 data from 1998/6/28 to 30. Even in the case of a sharp increase in the number of requests, SkyShield will not detect any anomalies, because the sharp increase in the number of flash crowds is caused by the simultaneous access of many normal users, and these hosts are evenly mapped into the sketch data structure. Since all hosts send a similar number of requests, the values in the buckets of the sketch are evenly distributed, therefore the divergence between the sketches in two consecutive detection cycles is small and steady, resulting in no alarms in the flash crowds.

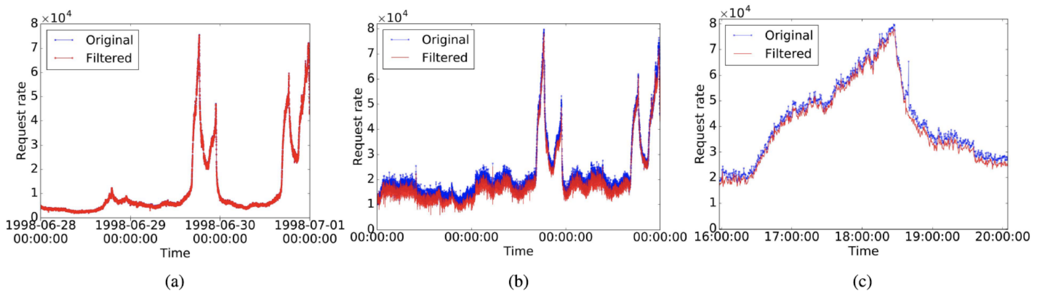


Fig. 4.19: Performance of SkyShield in flash crowd mimicking attacks. (a) The original data. (b) The combined data. (c) Insight of the combined data

A second experiment is conducted to test whether SkyShield can effectively detect attacks that occur during flash crowd events. In this experiment, we combined Dataset1603 data with WorldCup98 data for 3 days from 1998/6/28 to 30 to simulate app-layer DDoS attacks that occurred during a Flash crowd event. Fig.4.19(b) demonstrates the detection result of the synthetic data. The results show that SkyShield can effectively mitigate attack requests. Fig.4.19(c) 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 results showed that there was an obvious attack from Dataset1603 at 18:40:00. However, SkyShield will still detect anomaly and mitigate the attack within a short period of time, which indicate that SkyShield can effectively mitigate DDoS attacks even during flash crowd. We also combine other types of attacks with flash crowd, and all experiments have proven the effectiveness of SkyShield. We omit these results for space concerns.

Comparison with the State-of-the-Art Methods

We qualitatively compare SkyShield with two state-of-the-art methods, namely the approach based on the Hidden semi-Markov Model (HsMM) [90, 89, 92] and the approach based on the Transductive Confidence Machines for K-Nearest Neighbors (TCM-KNN) [45].

The HsMM-based method profiles behavior of normal users, and regards any deviation from the normal profile as anomaly. Entropy is used to detect potential app-layer DDoS attacks. By correctly configuring the parameters, the HsMM method can achieve a FPR as low as 1.5% and a detection rate of about 90%. However, this model requires frequent updates with a stable and low-volume Web workload. In addition, the training of the model is computationally intensive, and the training data should be recollected to maintain its freshness. This greatly limits the application of the HsMM method in real-time DDoS attack detection. Additionally, the HsMM model requires priori knowledge of website page structure, which can be daunting at times. The wide use of dynamic web pages makes this problem even more worse. Compared with the HsMM method, SkyShield can mitigate the attack faster, and the parameter configuration of SkyShield is much easier.

The TCM-KNN method is designed as a light-weight DDoS attack detection scheme for Web servers. The reported TPR and FPR were 99.53% and 1.93%, respectively. This method employs a new objective measurement as the input features, and utilizes the Genetic Algorithm based instance selection method to improve real-time detection performance. However, the training of the model is still expensive, although the author subsequently developed an extended fuzzy C-means algorithm to solve this problem, even for the improved version [45]. What's worse, the model cannot adapt to network dynamics. A model trained on data in a specific time period may not be suitable for detecting anomalies that occur in other time periods. Compared with the TCM-KNN model, SkyShield is also lightweighted and detect DDoS attacks against Web servers in real time. SkyShield uses a novel measure of sketch divergence which is stable to network dynamics. The number of abnormal buckets is adapted to the amount of requests, thus in an intensive attack more malicious requests will be filtered.

Compared with the above method, SkyShield's TPR is relatively low. However, the main goal of SkyShield is to guarantee the availability of services in the event of a sudden overwhelming attack on the system. Although only partial malicious hosts are detected, the attack traffic can be reduced by more than 94%. Therefore, SkyShield can effectively defend against app-layer DDoS attacks, and can well prevent Web servers from being overwhelmed by severe flooding attacks.

4.7 Summary

We design and implement a new defense system named SkyShield by taking advantages of the sketch techniques to identify malicious hosts efficiently and leveraging other techniques including Bloom filters and the CAPTCHA techniques to guarantee the effectiveness of SkyShield. The experimental results demonstrate that SkyShield can effectively mitigate application layer

DDoS attacks and pose a limited impact on normal users. The content of this chapter is published in [84].

5 Understanding the Behaviors of BGP-based DDoS Protection Services

This chapter describes our study on the behaviors of BGP-based DDoS Protection Services (DPS) providers. We first introduce our analysis process and describe the implementation of our prototype. After that, we present the experimental results.

5.1 Analysis process

Our analysis aims to detect abnormal BGP dynamics caused by DDoS attacks and identify the behaviors utilized by DPS providers to mitigate the attacks. Fig. 5.1 shows the analysis process, which consists of a training phase and a monitoring phase. Several features are extracted from the BGP update data within a fixed time interval of one minute in both phases, and then they are grouped into a vector referred to as a databin.

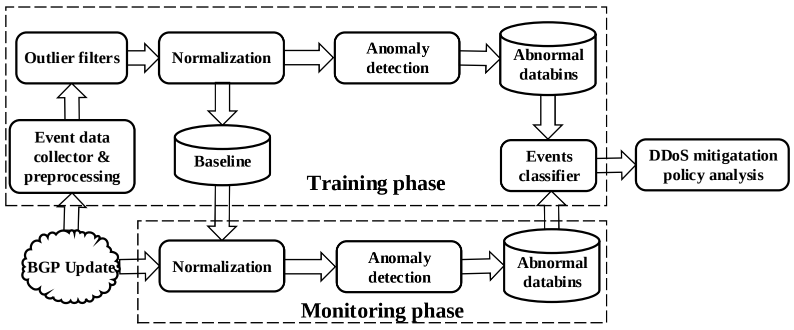


Fig. 5.1: The analysis process

In the training phase, we first collect a sufficient number of different events, such as a hurricane, blackouts, earthquakes, cable cuts, and DDoS attacks, to train a classifier. We manually searched the related news about these events to determine the occurrence times of these events. Then we collect the BGP update data in a time period that can cover the occurrence of the events. Since the dynamics of BGP routing may persist much longer than the actual duration of an event, there may be some deviations in the reports or news related to those events. Thus, we collect BGP traffic data in a longer period of time to ensure a reference period in the reports or news is included. It is supposed that the reference period contains more normal BGP traffic data than that in the occurrence period, thus facilitating the normalization of databins in the occurrence period.

Since the BGP traffic is inherently dynamic and there are some outliers even during the normal state, we employ the k-means method to filter out the outlier databins in the reference period. Specifically, the databins in the reference period are clustered into two groups based on the Euclidean distances between databins. The group of the majority is expected to contain only normal databins. It is used as the normality baseline, which is used to normalize the incoming databins in the monitoring period. We utilize a Z-score normalization method to normalize the databins. The Z-score value of a feature is calculated as $z = \frac{x-\mu}{\sigma}$, where μ is the mean of the obtained normal databins and σ is the standard deviation. The calculated mean and deviation in the training phase are used to normalize the databins in the monitoring phase.

We then mix the normal databins extracted from the reference period with those in the occurrence period. Again, the k-means method is employed to cluster the mixed databins into two groups. One of the majority is regarded as normal and the other one as abnormal. Li et al. utilized the k-medoids method for the same task [41]. We employ the k-means method instead of the k-medoids method as the latter spends more time finding the centroids (e.g. $O(n)$ for k-means while $O(n^2)$ for k-medoids). We further group the obtained abnormal databins by timestamps to obtain the consecutive abnormal databins. By "consecutive", we mean that the intervals of detected abnormal databins are less than 3 minutes and at least three consecutive databins in the cluster. The obtained groups of databins are referred to as "incidents". After that, the obtained abnormal databins are labeled as the types of incidents. We only distinguish between disaster events and DDoS attacks. Then a random forest classifier is trained based on the labeled databin. The random forest classifier is selected as it can make a considerably high prediction accuracy and be well interpreted by the features. We also analyze the mitigation policy adopted by DPS providers.

The newly incoming BGP traffic data is normalized in the monitoring phase using the previous baseline obtained in the training phase. The anomaly detection module will check whether there is an anomaly in the BGP dynamics. We use the trained classifier to identify whether DDoS attacks cause the abnormal event if an anomaly is detected. If so, further analysis is performed to determine whether a BGP-based mitigation policy is utilized. It is worth noting that our analysis process allows practitioners to utilize their experience knowledge to improve the system's performance. When an alarm is raised, the practitioner could judge the result based on other external information sources. If the prediction agrees with the practitioner's judgment, the newly incoming databins will be added to the training databins. Otherwise, we rejected the prediction.

Features

BGP routing information are exchanged between BGP routers through BGP update messages. A route is announced by a BGP speaker when it is chosen as the preferred forwarding path. On the contrary, a route is withdrawn when the BGP speaker has chosen a new route, and the old one is no longer available or reachable. Besides, there are many implicit withdrawals, which means the prefix is implicitly withdrawn by sending the same prefix with new attributes. We extract 9 features from BGP update messages to character the fluctuation of the BGP traffic. Table 5.1 shows the features.

Table 5.1: Description of features

Feature name	Definition
Ann	Number of announcements generalized by BGP speakers
Udt	Number of BGP update messages
WADup	Number of duplicate announcement after withdrawal to the same IP prefix
AW	Number of Withdrawal after announcement to the same IP prefix
WADiff	Number of new announced paths after an explicit withdrawal
AADiff	Number of new announced paths without explicit withdrawals
AADupType1	Number of duplicate announcements to the same IP prefix
AADupType2	Number of duplicate announcements to the same IP prefix with only AS-PATH and NEXT-HOP fields unchanged
Unq_pfx_as	Number of unique prefixes originated by an AS
Max_AS_path_len	The maximum length of AS-PATHs
pfx_org_chg	Number of Prefix origin change

Ann is the number of paths announced by BGP speakers in a detection cycle. BGP update messages deliver the reachability information. Base on various situations, such as the length of AS path or a new route, the router will announce to their peers; however, if this route keeps up and down, the router withdrawals it by protection policies. Otherwise, the router needs to send numerous BGP updates to peers.

Udt is the number of updates sent by BGP speakers to share routing information with other peers. It is the sum of announcements and withdrawals. The distribution of Udt is similar to that of Ann. Since the number of withdrawals caused by abnormal events is much smaller than that of announcements, the Udt feature provides valuable information that can help us distinguish the event data between DDoS events and Disaster events.

WADiff is the number of newly announced paths after an explicit withdrawal. We distinguish between explicit or implicit withdrawals based on whether a withdrawal message is sent or not. The former means that the corresponding BGP update message is received. The latter does not receive such a message. Since disaster events usually result in the unreachability of some BGP routes, they will trigger peer routers to send more explicit withdrawals.

AADiff is the number of newly announced paths without an explicit withdrawal sent by the BGP speakers. It happens when a previously preferred route is not available, or a newly preferred route is announced. This feature reflects possible exogenous network events, such as router failures or link disconnection [42]. Since disaster events usually damage physical devices, such as BGP routers the cyber links, the edge routers will announce more new paths.

AADupType1 is the number of duplicate announcements to the same prefix with all fields unchanged. Since the router will receive the BGP update of any established BGP peer, it is not synchronized between iBGP, and meanwhile, it updates BGP with the eBGP peer be a problem of repeated announcements. [63]. Hence, the more alternative paths an AS has, the more internal path explorations, the more duplicate announcements. During DDoS attacks, the victims will repeatedly announce a fixed amount of the affected paths, which will result in more duplicate announcements to the same prefix.

AADupType2 is the number of duplicate announcements to the same IP prefix with only AS-PATH and NEXT-HOP fields unchanged. One or more of the other attributes (such as community) is different from the former announcement the same prefix, which could reflect the routing policy change. Community in messages is used to make routes that share common property and thus

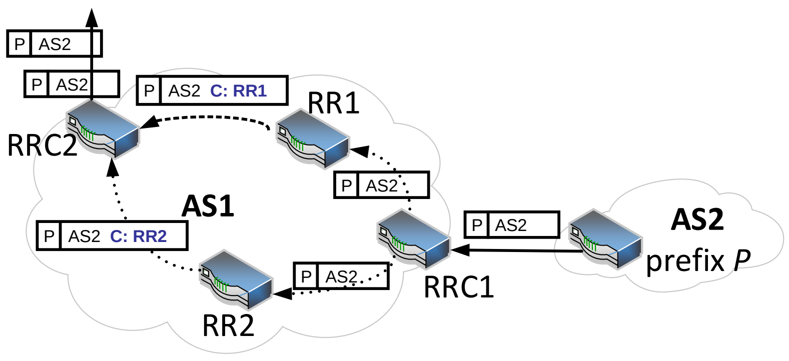


Fig. 5.2: Inferred cause of duplicates

undergoes a specific treatment. Some providers also allow their customers to control the redistribution of their routes via communities. There will be more policy fluctuations during disaster events than that during DDoS events, because practitioners need to make a policy change when the destination is unavailable through a ruined region due to disasters. However, these policy changes happen less in DDoS attacks as the duration of DDoS attacks is much shorter than that of disasters.

Unq_pfx_as is the number of unique prefixes originated from an AS in a given time window. The number of announcements and withdrawals exchanged by neighboring peers is an essential feature during instability periods. We utilize this feature to model the stable situation of the normal state. This feature is more stable during the disaster event period than that in the DDoS event period. The reason is that when DDoS attacks occur, the DPS provider may utilize a BGP-based approach by announcing the prefix that belongs to the victim to mitigate the DDoS attack traffic, leading to the increase of the number of a unique prefix.

Max_AS_path_len is the maximum length of AS paths announced by BGP routers in a specific time window. In the normal state, the AS paths announced by BGP routers usually have a limited number of hops, since the BGP protocol prefers short paths. However, when an AS suffers from attacks, the operator might implicitly withdraw a pre-announced path by pre-pending many duplicated ASes in the AS-path field. It could significantly increase the lengths of AS paths. However, the distribution of the maximum AS path lengths for disaster events will be much evener than that for DDoS attacks, because disaster events usually cause outages of the Internet and thus result in more long paths during disaster events.

ASN (AS number) is a globally unique number that is used to identify an AS. It allows an AS to exchange exterior routing information between neighboring ASes. Asn_prfx_chg is the number of prefix changes of ASes in a time window. This feature is proposed based on the assumption that the Internet topology should not frequently change. It has been used as a single BGP feature to detect prefix hijacking attacks. However, it is also possible for an AS to alter the prefix to reroute the subnet traffic through the DPS AS. There are more prefix origin changes during DDoS attacks. When DDoS attack events occur, the DPS provider will announce the prefix that belongs to the victim and scrub the traffic.

Community analysis

We utilize the community attribute in BGP updates to analyze the behavior patterns used by DPS providers to mitigate DDoS attacks. Between BGP peers, they set specific community values to demand a modification of peer's local preference values.

First, we extract UTransaction = {prefix, $community_1$, $community_2$, ... , $community_M$, AS_1 , AS_2 , AS_3 , ... , AS_N }($N = |S|$, where S is the set of ASes in the AS Path) in a databin that belongs to normal traffic period, and one databin is responsible for one UTransaction. Let $UDB = \{UTransaction_1, UTransaction_2, ..., UTransaction_n\}$ represent all the normal databin UTransactions. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items(for each $i \in I$, i can be prefix, community and ASN).and then we conduct affinity analysis on the UDB. Here we adopt Apriori Algorithm, and the Apriori process will read the UDB for multiple passes to find all frequent normal updates itemsets. For the first pass, Apriori scans the UDB to get the frequent 1-itemsets that satisfies minimum support. In a subsequent kth pass, Apriori uses self-join rule to generates the candidate frequent k-itemsets with the help of (k-1)-itemsets, then it will scan the UDB to get frequent k-itemsets that satisfies the minimum support. Repeat this process from $k = 1$ until we can't apply self-join rule any more. At the end of Apriori process, we get the frequent normal updates itemsets, then we extract all the rules which are greater than the threshold support and the threshold confidence as the association rules. Besides, the association rule will be selected as our frequent normal updates pattern if it takes the form: {prefix, $community_1$, $community_2$, ... , $community_P$ } $\Longrightarrow$ { AS_1 , ... , AS_Q } ($P \leq M$ and $Q \leq N$), which indicated that if {prefix, $community_1$, $community_2$, ... , $community_P$ } appears in the databin, { AS_1 , ... , AS_Q } is also likely to appear in the databin. In order to find a bigger network scope behavior, our system will search in the frequent normal updates patterns to find the prefixes that belong to the same subset, then prefixes in the same subset will be replaced with the more generic prefix. For example, "152.113.0.0/16" and "152.113.32.0/24" belong to the same subset "152.113.0.0/16" and the prefix "152.113.32.0/24" will be replaced with to "152.113.0.0/16". And the process is shown in Fig. 5.3. Let AR_1 represent the frequent normal updates patterns. Second, we perform the same process using the databins that be-

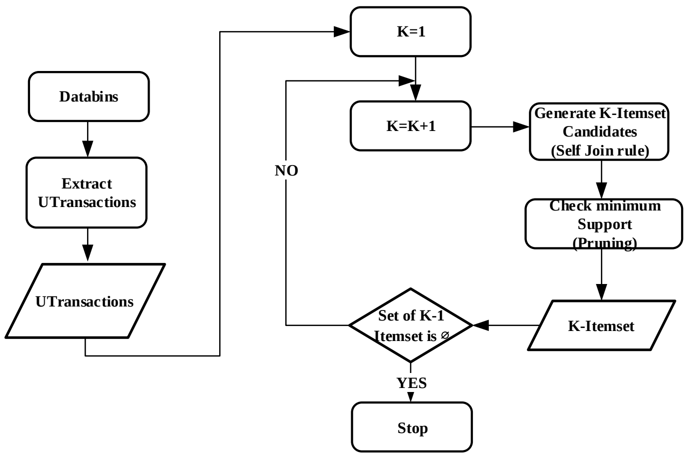


Fig. 5.3: Frequent UPDATES pattern's Mining process

long to the abnormal periods when the system sends a DPS mitigation signal, and we get frequent updates patterns AR_2. Finally, we use $AR_3 = AR_2 - AR_1$ to extract the abnormal patterns, which is the relative complement of AR_1 with respective to frequent updates patterns AR_2 (the difference of AR_1 and AR_2).

5.2 Implementation

Fig. 5.4 shows the architecture of our prototype. We utilize the BGP traffic data collected from Route Views to monitor the BGP routing dynamics. The system persistently sniffers the Internet dynamics and extracts features from the BGP traffic data with a one-minute fixed time interval. We encapsulate the extracted features in a databin and pass the databins to the detection module. When the detection module detects consecutive anomalies, the system determines that a disruptive event is going on. Then, the classification module classifies the abnormal databins to identify the type of event. If the event type is a DDoS attack, we will further analyze the DDoS mitigation policy adopted by the victim. We develop the prototype using Python 2.7.12 running on a 64-bit Windows 10 system with an Intel(R) Core(TM)2 Quad CUP Q9550 @2.83GH and 8.0GB RAM.

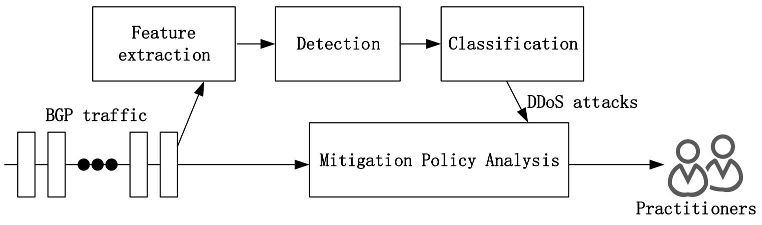


Fig. 5.4: The architecture of the system

The feature extraction module constantly retrieves BGP traffic data from Route Views and extracts features (e.g. Ann and AADiff) with a fixed time interval of one minute. The detection module takes the extracted features as input and detects abnormal databins with these features. Then we employ the k-means clustering method to remove the outliers in the reference period. After the preprocessing, the k-means method is employed to distinguish between the abnormal databins from the normal ones. The abnormal databins are tagged as 1, and the normal ones are tagged as 0. When the system detects 3 consecutive abnormal databins, it will raise the alarm, and the detected abnormal databins are passed to the classification module for event type identification. When a DDoS attack is identified, it will further analyze the BGP update traffic originating from DPS's ASes.

The classification module employs the random forest method to classify the abnormal databins into different categories. Random forest is an ensemble learning method for classification, regression. In this report, we focus on two categories: DDoS attacks and disaster events. We collect 4 different disaster events, including blackout, earthquake, hurricane and cable cut, respectively. We collect the BGP traffic data of 41 historical events which are summarized in Table. 5.2 and use these events to train the classification model. We use 5-fold cross validation method to evaluate the accuracy of the system. The detected abnormal databins in each category is divided into 5 folds and in each test we use one of the 5 folds as test data and the other 4 folds as training data. The clustering results are obtained by averaging the 5 results. Finally, we obtained an accuracy of 91.2%. The results produced by the classification module will help the practitioners fast analyze the abnormal dynamics of BGP traffics and thus take steps to eliminate the damage to the concerned local networks.

Table 5.2: A summary of the dataset

type	Event number	Detected databins
hurricane	4	30
black out	4	14
earthquake	4	152
cable cut	9	602
DDoS	20	889

When a DDoS attack event is identified, the mitigation policy analysis module will check whether a BGP-based DDoS mitigation method was adopted or not by the victim. In this process, the module constructs the AS graphs of the victim to check. Any ASes that belong to DPS providers are connected to the victim. The BGP messages from the DPS ASes are checked, and the mitigation policies are analyzed. We manually collected the top 15 DPS providers for this module, and their ASes are listed in Table 5.3.

5.3 Experiments

In this section, we show how the Mitigation Policy Analysis model works. We focus on community values in the BGP UPDATES to infer the behavior of DPS providers. We also conduct case studies on two DPS providers to validate the effectiveness of our method.

5.3.1 BGP routing policies

DPS providers may have many ways to mitigate DDoS traffic, including BGP-based routing policies, DNS-based routing policies, In-line filtering policies [72], and so on. Our work focuses on BGP-based routing policies. Many DPS providers adopt BGP-based mitigation methods. For example, Akamai provides a DDoS mitigation service named Prolexic Routed, which leverages the Border Gateway Protocol (BGP) to route all of an organization's network traffic through Akamai's globally distributed scrubbing centers [2]. Nexusguard provides an Origin Protection service which offers comprehensive protection of the entire network by routing the inbound traffic to their

Table 5.3: The top 20 DPS and their ASes

Rank	DPS	ASes
1	Akamai Technologies Inc.	22207,18717,23454,20189,16702, 18680,23455,35994,12222,35993
2	VeriSign, Inc.	29403
3	Incapsula, Inc.	19551
4	CloudFlare, Inc.	13335,132892
5	Arbor Networks, Inc.	20052
6	Sucuri Inc.	30148
7	F5 Networks, Inc.	22317
9	Check Point Software Technologies Ltd.	25046
9	Neustar, Inc.	32979,7786,32978,12008,19905, 19911,32980
10	NSFOCUS, Inc	8757
11	Radware Ltd.	15823,48851
12	Staminus Communications, Inc.	25761
13	Storm Systems LLC	59796
14	Corero Network Security, Inc.	395752
15	Zenedge Inc.	393676

worldwide scrubbing centers and the clean traffic will be routed back via a GRE tunnel [62].

An organization may have several routes to reach a particular destination, and operators can control the import and export policies by modifying preference, filtering and tagging to choose the best route [10]. The preferences attributes, such as LocalPref, AS path length and the multi-exit discriminator(MED) pose a great influence on the BGP routes decision process. LocalPref is an integer value which can be set and transmitted in the local AS and will be filtered before reaching the neighboring AS. A highest LocalPref value means the routes is the best among all the routes to a particular destination. The MED attribute is a non-transitive attribute which provides a way for an AS to negotiate with its external neighbors about the preferred path to enter the AS. A route with lowest MED value will be adopted first. Filtering can be used to control the import and export routes by configuring the routers to ignore the BGP updates advertisements with matching certain specified values or ranges. According to the condition we pre-set in tagging, routers use the iBGP to update each other [10]. The main method is grouping destinations into a single entity, named community [22]. The first 2 bytes represent an ASN and the last 2 bytes as a value with a predetermined meaning. The predetermined meaning is not standardized. Although MED and LocalPref are essential in the route selecting process, they are seen only by neighbor routers or the router itself. We can not use them directly to determine the DPS provider's mitigation behavior. The attributes that are available to us are AS Path and community values. Our system utilizes these two attributes to analyze the DPS provider's mitigation behavior.

Table 5.4 shows how the as-path changes with the community attribute during the period of an attack. When the new community value 712 and 801 appeared, the as-path changed from "3549 26769 45474 45599" to "3549 3491 45474 45599". Table 5.5 shows the routing tables before it. At 08:00, the community attributes were "732, 3114, 3210, 3220, 3314". However, at 14:00, the as-path changed to "3549 26769 45474 45599" when the community attributes added 712 and 801. It shows that DPS providers change their protection policies by modifying community values.

Table 5.4: updates: prefix 112.78.104.0/24, monitor 3549

time(UTC +8)	DPS'(45474) community attribute	as path
2011-05-10 08:48:42+08:00	732,3114, 3210,3220,3314	3549 26769 45474 45599
2011-05-12 13:33:53+08:00	712,801,3114,3211, 3221, 3314	3549 3491 45474 45599

Table 5.5: routing table: prefix 112.78.104.0/24, monitor 3549

time(UTC +8)	DPS'(45474) community attribute	as path
2011-05-10 08:00:05+08:00	732, 3114,3210,3220,3314	3549 26769 45474 45599
2011-05-12 14:00:07+08:00	712, 801,802	3549 3491 45474 45599

Besides, we find the relationship between community values and the as-path hops. We count the frequency of an AS appearing in an as-path associated with a community value H and the total frequency of the community value T. We refer to the confidence of the hops as H/T. The bigger the confidence value is, the more likely it is that an AS appears in the as-path. When we set a threshold of the confidence value of 0.6, we remove the community values corresponding with hop 45474. We observe that there are around 16 ASes that are related to the community value changes, and the hops information is shown in Table 5.6. The relationship between the number of related community values and shown hop in Fig. 5.5. From Fig. 5.5, we could see that as hop 4826 and 45599 have high confidence with only one community value, so we can infer the hops like 4826 and 45599 are more specific to the corresponding community value. While ASes like 3549 and 10107 has more than 20 related community values, the relation with the community values may be week compared with the former ones.

Table 5.6: ASes that highly related to the DPS provider's (45474) community values

ASN	URL	as name	as country
4635	hkix.net	HKIX-RS1	Hong Kong SAR China
4837	chinaunicom.cn	CHINA169-Backbone	China
4134	chinatelecom.com.cn	CHINANET-BACKBONE	China
45578	-	SPLUNKNET-PH	Philippines
10026	pacnet.com	PACNET	AP
7660	nic.ad.jp	APAN-JP	Japan
3549	level3.com	LVLT-3549	United States
26769	bandcon.com	BANDCON	United States
3491	pccwglobal.com	BTN-ASN	United States
15412	relianceglobalcom.com	FLAG-AS	United Kingdom
38325	-	WTP-AS-AP	Philippines
3320	dtag.de	DTAG	Germany
4826	vocus.com.au	VOCUS-BACKBONE-AS	Australia
10107	-	AASTARNET-PH	Philippines
12989	eweka.nl	HWNG	Netherlands
45599	-	EQUANTECH-TW-AP	Taiwan

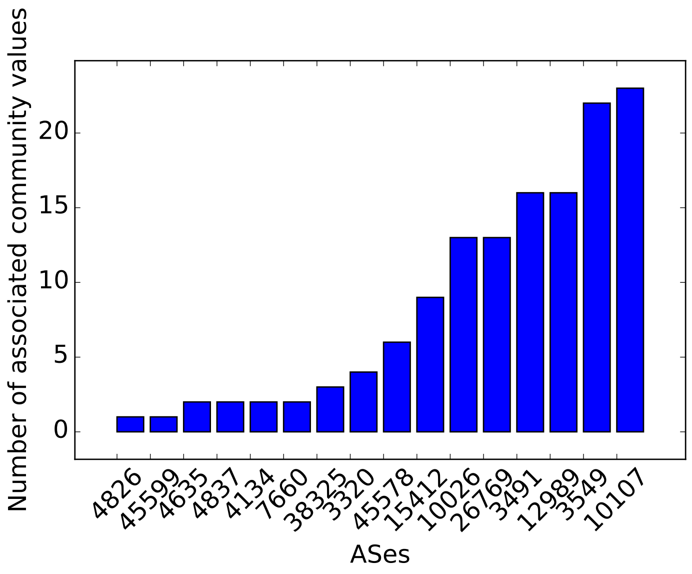


Fig. 5.5: Dyn DDoS attack overview

5.3.2 DDoS attack events

This section presents the identification of DDoS attack events from the perspective of a victim and a DPS provider.

Perspective from Victim
The first case is the DDoS attack against the Dyn in October 2016 [74]. On October 21, 2016, the Dyn suffered from many DNS queries from a considerable vast of clients, which consume the ability of the managed DNS network. It caused the unavailability of the DNS service of the Dyn. This further results in difficulties connecting numerous websites (such as Amazon, BBC, CNN, etc.) for many users. During the attack, the traffic going to the other DNS providers increased dramatically and thus caused the widespread congestion of network traffic. This congestion eventually results in the abnormal dynamics of BGP traffic, which enables us to detect the Dyn DDoS attack event through the BGP dynamics.

For the impact value, it is the sum of the differences between the normalized features, and the baseline represents the distance of a databin from the normal ones. Fig. 5.6 illustrates the impact values versus the time. Three red blocks illustrate the three periods of abnormal dynamics on October 21, 2016. Our anomaly detection module can identify these abnormal databins and correctly classified them as DDoS attacks. The detected abnormal periods are described as follows:

- The first period started at 04:30:22(PDT), and there were fluctuations of BGP traffics from that time point. Until around 06:16:00(PDT), the fluctuation diminished. Which coincides with the reported start and mitigation time of the incident [19]. During this period, the Dyn's DNS server platforms in the Asia Pacific and East Europe suffered from massive requests, and then the US-East region, resulting in the vast BGP route dynamics [74].

- According to our system detection results, the second period started at 08:41:44(PDT) and ended at around 10:32:00 (PDT), which also agrees to the reported DDoS attack period [29].

- Our system also detected the third period of abnormal BGP dynamics, which started at 13:19:28 and ended at around 14:08:00(PDT). We is also consistent with the DDoS attack period reported in the news [74].

We also found some noticeable additional fluctuations in the BGP traffic, which started at 01:22:23(PDT) and 17:47:1 respectively, as shown in Fig. 5.6 with green blocks. However, they are not reported by Dyn.com or other news media. We speculate these events were caused by the initiation and aftershocks of the DDoS attacks.

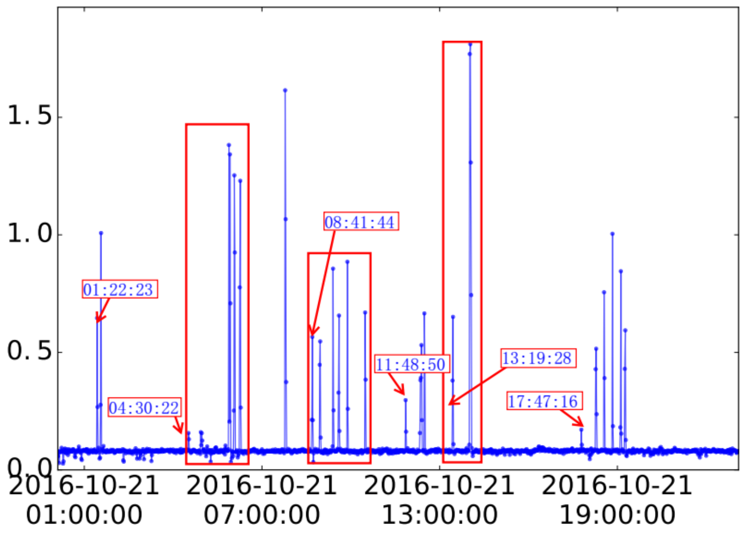


Fig. 5.6: Dyn DDoS attack overview

To look insight into the mitigation policy of the Dyn, we extract part of the BGP traffic that originated from or passed through AS33517 (Dyn.com) during the DDoS attack. We find that the peers AS2914 and AS6453 are both DPS providers. AS2914 is NTT-COMMUNICATIONS, which provides information and communications technology solutions in Japan and internationally. It is also one of the world's top ISP and DSP. AS6453 is TATA Communications, one of the leading providers in a new world of communications. It started launching its network-agnostic DDoS protection globally and delivering over a cloud-based infrastructure since 2011 Sept.07. Fig. 5.7 (a) presents the topology of AS33517 in the day of 2016/10/01, without DDoS attacks. Fig. 5.7 (b) presents the topology of AS33517 in the day of 2016/10/21. The circle around AS33517 represents the path where next hop is AS33517 itself. The red AS number is the DPS ASs AS2914 and AS6453, and the blue circle ASes are the missed ASes in the DDoS attack period. The missed ASes are a sign of the poor performance of Dyn.com to provide DNS query services.

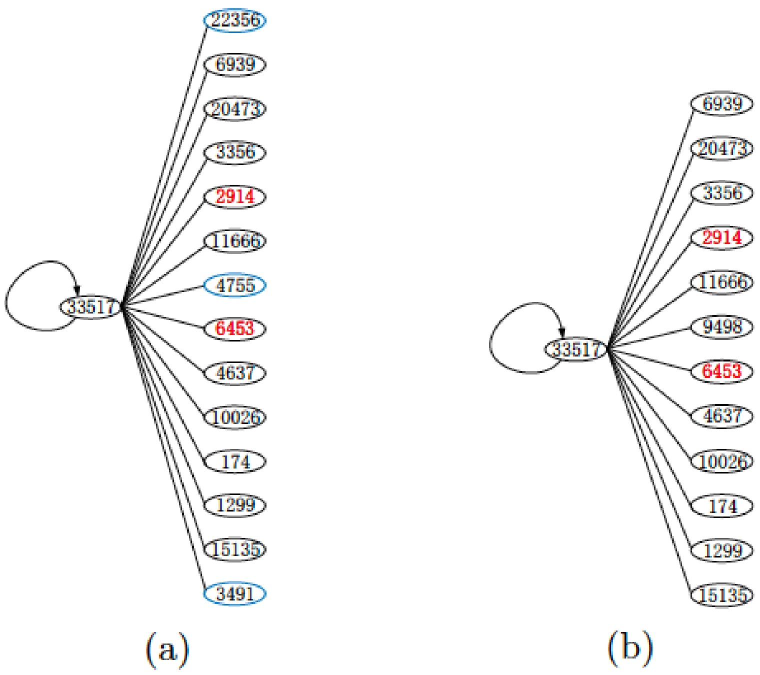


Fig. 5.7: Topology of AS33517 at different times

During the first DDoS period, most traffic originating from or to AS33517 is routed to AS2914 and AS6453. We summarize the paths during the first period of the DDoS attack from 04:30:22(PDT) to 06:16:00(PDT). AS33517 announced the route and passed through The DPS AS6453 and AS2914. A comparison between October 1 and October 21 is shown in Table 5.7. We demonstrated that the total number of paths decreased while the number passed through the DPS providers increased during the attack. It indicates that DPSs are trying to mitigate the traffic when the DDoS attack started and more traffic is routed through the two DPSes. Hence, it is clear that AS33517 was under the protection of the two big DPS providers like TATA Communications and NTT-COMMUNICATIONS during the period of the DDoS attack. This case also demonstrates that even a large DPS provider may fail to protect its customers completely.

Table 5.7: A comparison of AS paths between 2016/10/01 and 2016/10/21

	# of Paths announced by Dyn.com	# of Paths passed through DPS
2016-10-01	3916	543
2016-10-21	1292	643

Perspective from a DPS Provider

In this case study, we analyze the policies of DPS adoption events through the BGP traffic. We detected three DDoS attack events against a victim AS owned by New World Telecom Inc. (NWT) from May 1 to June 30, 2012. The DPS provider confirms all these detected events. NWT, which is established to design, manufacture, and distribute telecommunications products, suffered from severe DDoS attacks from May 1 to June 30, 2012, and turned to the NexusGuard Ltd (NG) to seek protection from DDoS attacks.

Table 5.8 presents a summary of the three events. The AS Numbers of NWT and NG are 17444 and 45747, respectively. We then analyze the three

Table 5.8: A summary of the detected events

Prefixes	Protected AS#	DPS AS#	Mitigation period (UTC+8)
58.64.128.119/32	17444	45474	2012/06/21 08:25:17 to 06/28 09:58:21
58.64.138.186/32	17444	45474	2012/05/26 10:47:51 to 05/27 14:46:35
58.64.135.102/32	17444	45474	2012/06/20 17:29:10 to 06/20 18:23:59

prefixes “58.64.128.119/32”, “58.64.138.186/32”, and “58.64.135.102/32”. We found 24 BGP updates records in total, and the number of announcements and withdraws is 14 and 10, respectively. We have also analyzed the features mentioned in Table 5.1, and their values are listed in Table 5.9. There is neither AADupType1 nor AADupType2 update pattern, meaning that no identical announcements were sent from BGP routers. We track the prefixes and found that there is no community information in the BGP updates of these prefixes, resulting in zero AADupType2.

Table 5.9: Quantity of each update pattern

Pattern	number
WADiff	4
AADiff	4
WADup	6
AW	10
AADupType1	0
AADupType2	0

The first adoption event started on 2012/6/21 at 8:25:17 and ended at 2012/6/28 at 11:28:59. Fig. 5.8 shows the extracted BGP information that is relevant with the AS numbers 17444 and 45474. Let’s assume the former BGP updates record is a withdraw to the prefix “58.64.128.119.32”, then the update pattern are WADiff $\rightarrow$ AW $\rightarrow$ WADup $\rightarrow$ AW $\rightarrow$ WADup $\rightarrow$ AW $\rightarrow$ WADup $\rightarrow$ AADiff $\rightarrow$ AADiff $\rightarrow$ AW $\rightarrow$ WADup $\rightarrow$ AW. We could see that the update pattern sequence starts with a WADiff and ends with an AW, and there are several repeated AW to WADup change procedures. From Fig. 5.8, we could see that AS 17444 firstly attempted to announce a path through AS 45474 (NG LTD) to protect the IP address 58.64.128.119 from DDoS attacks. There are 4 such announcements with an average time interval of 6900s (about 2 hours). However, such an adoption policy seemed to take no effect, and at 13:57:28, AS 45474 announced the prefix “58.64.128.119/32” directly, indicating that a complete hosting policy is adopted and an AADiff update pattern occurred. After 2447s (about 40 minutes), the first protection policy is re-adopted, and we could capture another AADiff update pattern. The above analysis shows that the host whose IP address is 58.64.128.119 suffered from the DDoS attack from 2012/6/21 8:25:17 UTC+8, and the operator of the AS or host taken different DPS policies to defeat the attack.

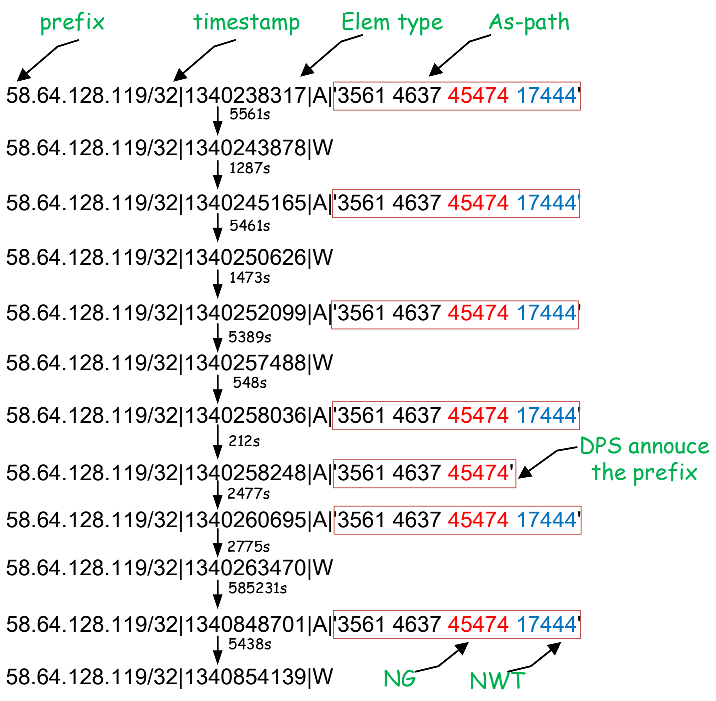
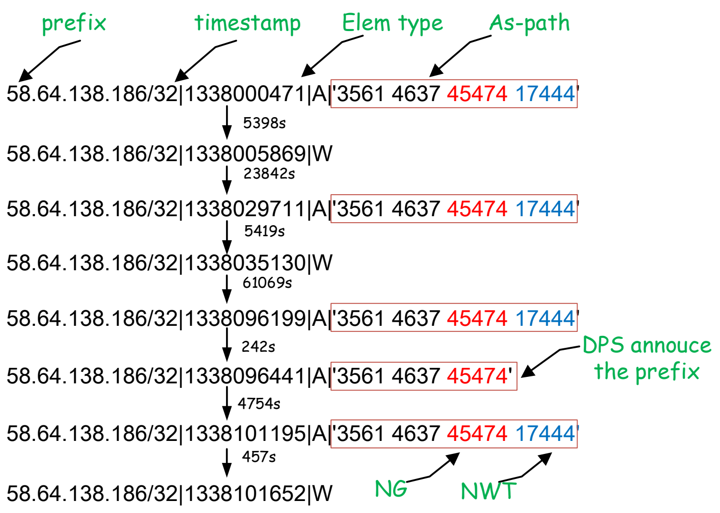

```
58.64.128.119/32|1340238317|A|'3561 4637 45474 17444'
58.64.128.119/32|1340243878|W
58.64.128.119/32|1340245165|A|'3561 4637 45474 17444'
58.64.128.119/32|1340250626|W
58.64.128.119/32|1340252099|A|'3561 4637 45474 17444'
58.64.128.119/32|1340257488|W
58.64.128.119/32|1340258036|A|'3561 4637 45474 17444'
58.64.128.119/32|1340258248|A|'3561 4637 45474'
58.64.128.119/32|1340260695|A|'3561 4637 45474 17444'
58.64.128.119/32|1340263470|W
58.64.128.119/32|1340848701|A|'3561 4637 45474 17444'
58.64.128.119/32|1340854139|W
```

Fig. 5.8: Extracted BGP information

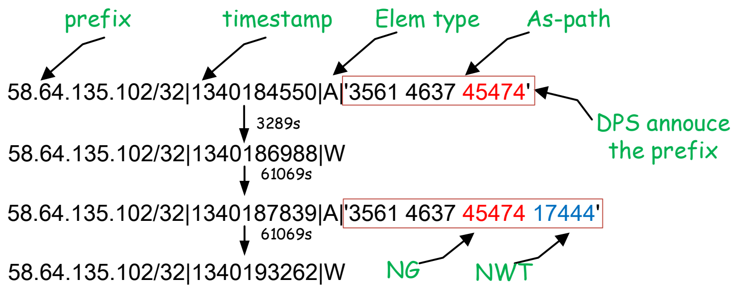
The second event started from 2012/5/26 10:47:51 and ended at 2012/5/27 14:46:35 UTC+8. Fig. 5.9 shows the extracted BGP information that is relevant with the two ASes. If we assume the former BGP updates record is a withdraw to the prefix “58.64.138.186/32”, then we could get the update pattern:“WADiff → AW → WADup → AW → WADup → AADiff → AADiff → AW”. The updated pattern also starts with a WADiff pattern and ends with an AW pattern. The repeated update pattern sequences are “AW → WADup”, which is similar to the DPS adoption process of prefix “58.64.128.119/32”. Firstly, the NWT started to mitigate the attack traffic by announcing a path to AS 45474 (NG LTD). Then the DPS provider completely took charge of the



```
58.64.138.186/32|1338000471|A|'3561 4637 45474 17444'
        5398s
58.64.138.186/32|1338005869|W
        23842s
58.64.138.186/32|1338029711|A|'3561 4637 45474 17444'
        5419s
58.64.138.186/32|1338035130|W
        61069s
58.64.138.186/32|1338096199|A|'3561 4637 45474 17444'
        242s
58.64.138.186/32|1338096441|A|'3561 4637 45474'
        4754s
58.64.138.186/32|1338101195|A|'3561 4637 45474 17444'
        457s
58.64.138.186/32|1338101652|W
```

Fig. 5.9: Extracted BGP information

protected hosts, directly announcing the prefix “58.64.138.186/32” and the AADiff update pattern enters our vision. As the path is much shorter than the previous one, more traffic would be rerouted to AS 45474 and filtered. After about one and a half hours, AS 17444 re-announced the prefix again when the attack relieved, and we could see another AADiff update pattern.



```
58.64.135.102/32|1340184550|A|'3561 4637 45474'
        3289s
58.64.135.102/32|1340186988|W
        61069s
58.64.135.102/32|1340187839|A|'3561 4637 45474 17444'
        61069s
58.64.135.102/32|1340193262|W
```

Fig. 5.10: Extracted BGP information

Table 5.10: update pattern sequences of three prefix

prefix	update pattern sequence
58.64.128.119/32	WADiff→AW→WADup→AW→WADup→AW→WADup →AADiff→AADiff→AW→WADup→AW
58.64.138.186/32	WADiff→AW→WADup→AW→WADup→AADiff→AADiff→AW
58.64.135.102/32	WADiff→AW→WADiff→AW

Fig. 5.10 shows the extracted BGP information from the third event. And here again, we assume the former BGP updates record is a withdrawal to the prefix “58.64.135.102/32”, then we could get the updated pattern: “WADiff → AW → WADiff → AW”. And the updated pattern also starts with a WADiff pattern and ends with an AW pattern. Unlike the previously analyzed two events, a

strict policy is directly adopted by the prefix "58.64.135.102/32" from the start, indicating it suffered an overwhelming DDoS attack. The protection persisted from 2012/6/20 17:29:10 to 2012/6/20 18:23:59 UTC+8 for about one hour. Then the prefix is re-announced by AS 17444 (NWT) with a longer path passing through AS 45474. Thus AS 17444 is still under the protection of DPS provided by NG.

From the analysis of the 3 prefixes, we could see some common DPS protection update pattern sequences, which means the response behavior to the DDoS attack. Here, we put the three prefixes updates pattern sequence together, as shown in Table 5.10. And if we represent the "WADiff.*AW" as B0, represent "AW → WADup" as B1, and represent "AADiff → AADiff" as B2, as shown in Table 5.11. Then we could represent the Table 5.10 as Table 5.12, from Table 5.12, the DPS protection behaviors are more clear. And it is obvious that DPS protected "58.64.128.119/32" and "58.64.138.186/32" once, respectively, and protected "58.64.135.102/32" twice.

Table 5.11: DPS protection behaviors

behavior abbreviation	update pattern sequence	meaning
B0	WADiff.*AW	protection behavior of start and end
B1	AW→WADup	DPS appended behavior
B2	AADiff→AADiff	DPS directly protection behavior

Table 5.12: DPS protection behaviors of update pattern sequences of three prefixs

prefix	update pattern sequence
58.64.128.119/32	WADiff→B1→B1→B1→B2→B1→AW
58.64.138.186/32	WADiff→B1→B1→B2→AW
58.64.135.102/32	B0→B0

Table 5.13: prefixs' the route table from monitor 3561

prefix	time(UTC +8)	route
58.64.128.119/32	2012/6/21 14:00:01	3561 4637 45474
58.64.128.119/32	2012/6/28 10:00:01	3561 4637 45474 17444
58.64.138.186/32	2012/5/26 12:00:00	3561 4637 45474 17444
58.64.138.186/32	2012/5/26 20:00:01	3561 4637 45474 17444
58.64.138.186/32	2012/5/27 14:00:00	3561 4637 45474
58.64.135.102/32	2012/6/20 18:00:01	3561 4637 45474

We referred to the BGP routing table from May 1 to June 30, and we extracted the routes dumped by the monitor 3561, as shown in Table 5.13, then we find that there are no routes that are related to them. On the contrary, they are unreachable from the monitor. And the number of them are quite a few. And if we combine BGP updates and the BGP ribs records (as shown in Fig: 5.11), we will have a comprehensive understanding of the behavior of the DPS AS 45474 and the normal AS 17444. In Fig. 5.11, we use different colors to represent the behavior of these ASes, and the colors are red and purple,

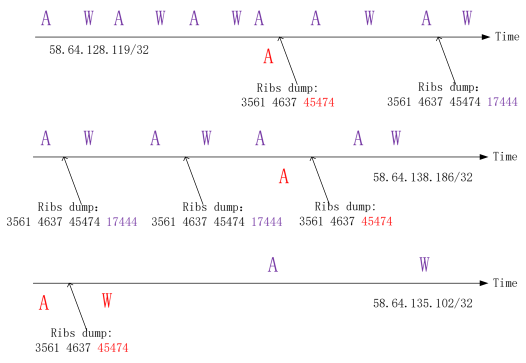


Fig. 5.11: Extracted BGP routing tables

representing NG and NWT, respectively. From Fig. 5.11 we could see that route tables dumped only a few times, and there is no more record in May and June, so the prefix may be backup and seek protection from Nexusguard to diverse the traffic.

5.3.3 Case studies on BGP community

This section investigates how the Border Gateway Protocol(BGP) community attribute can be used by Autonomous Systems (ASes) to control their routing policies through its upstream service provider networks.

Route Filter application in DPS
Table 5.14 and Table 5.15 shows the routing policy of Nexusguard, and Table 5.14 shows that the router announced 58.64.128.0/17 and 58.64.128.0/19 at 2011-05-22 08:20:28(UTC +8) and 2011-05-22 10:10:33(UTC +8) respectively, and announced a more specific and longer prefix 58.64.156.0/24 at 2011-05-22 10:10:33(UTC +8) and 2011-05-22 10:11:04. The as-path associated to the prefix 58.64.156.0/24 is changed to “7500 7660 4635 45474 17444”. Table 5.15 shows that there were two as-paths to the network 58.64.128.0, and the as-path to the more specific prefix 58.64.128.0/19 would be adopted. At 2011-05-22 12:00:00 there were three as-paths to the network 58.64.156.0, and the as-path to a more specific prefix 58.64.156.0/24 would be adopted.

Table 5.16 and Table 5.17 show the routing policy of Prolexic, we could see that it is similar to the Nexusguard routing policy. From Table 5.17 we could see that at 2011-09-10 14:00:18, there is two as path "11666 3257 702" and

Table 5.14: BGP update of AS7500

time(UTC +8)	prefix	DPS'(45474) community attribute	as path
2011-05-22 08:20:28	58.64.128.0/17	-	7500 2516 17444
2011-05-22 10:10:33	58.64.128.0/19	-	7500 2516 17444
2011-05-22 10:10:33	58.64.156.0/24	-	7500 2516 3356 1299 45474 17444
2011-05-22 10:11:04	58.64.156.0/24	1410 1420 1510 1710	7500 7660 4635 45474 17444

Table 5.15: AS path of AS7500

time(UTC +8)	prefix	DPS'(45474) community attribute	as path
2011-05-22 10:00:00	58.64.128.0/17	-	7500 2516 17444
2011-05-22 10:00:00	58.64.128.0/19	-	7500 2516 17444
2011-05-22 12:00:00	58.64.128.0/17	-	7500 2516 17444
2011-05-22 12:00:00	58.64.128.0/19	-	7500 2516 17444
2011-05-22 12:00:00	58.64.156.0/24	1410 1420 1510 1710	7500 7660 4635 45474 17444

"11666 3356 3209 8373" to the network 160.83.0.0/16 and 160.83.0.0/19 respectively, and 160.83.0.0/19 is the sub-network of 160.83.0.0/16, from Table 5.16 we could see AS 2824 announced a 32 bits prefix 160.83.95.1/32 which is more specific than 160.83.0.0/16 and 160.83.0.0/19, and we could see from Table 5.17 at 2011-09-10 16:00:19, there are three as paths to the network 160.83.0.0, and when the as path to the more specific prefix 160.83.95.1/32 will be adopted.

Table 5.16: BGP update of AS11666

time(UTC +8)	prefix	DPS'(32787) community attribute	as path
2011-09-10 15:48:22	160.83.95.1/32	35,65301	11666 32787 2824

Rollback application in DPS

Table 5.18 shows that a new as-path “4181 12989 45474 10107” was announced at 2012-05-10 08:01:26, and the community values were changed to “712 1210 1411 1510 1711”. However, it rolled back at around 2012-05-11 13:59. And we could verify this from the routing table shown in Table 5.19. The DPS community values were “712 1210 1411 1510 1711” and the as-path was “3549 3491 45474 10107” at 2012-05-10 08:00:10. However, the community values and the as-path replaced the old values at 2012-05-11 14:00:10. The whole process may be controlled by community values as shown in the updates and routing tables of the prefix 183.177.114.0/24, and AS 10107 used these inbound community values to set the LOCAL_PREF attributes in the AS that would receive the route message [22]. And we could see another example showed in Table 5.20 and Table 5.21, we could see that there was a fluctuation between the as path "6539 577 2914 32787 2824" and as path "6539 6453 32787 2824". From the updated BGP message

Table 5.17: AS path of AS11666

time(UTC +8)	prefix	DPS'(32787) community attribute	as path
2011-09-10 14:00:18	160.83.0.0/16	-	11666 3257 702
2011-09-10 14:00:18	160.83.0.0/19	-	11666 3356 3209 8373
2011-09-10 16:00:19	160.83.0.0/16	-	11666 3257 702
2011-09-10 16:00:19	160.83.0.0/19	-	11666 3356 3209 8373
2011-09-10 16:00:19	160.83.95.1/32	35,65301	11666 32787 2824

Table 5.18: BGP update of AS3549 and AS4181

time(UTC +8)	community attribute	as path
2012-05-10 08:01:26	732 3110 3120 3210 3220 3230	4181 12989 45474 10107
2012-05-11 13:59:48	-	4181 3356 2828 12989 45474 10107
2012-05-11 13:59:49	-	4181 3561 174 45474 10107
2012-05-11 13:59:54	-	4181 3491 45474 10107
2012-05-11 13:59:53	712 1210 1411 1510 1711	3549 3491 45474 10107

records, we could see that at 2011/10/03 15:58:23(UTC+8) AS 2824 announced an updates message telling the network that the route has changed, which resulted in the change of the routing table as recorded in Table 5.21. To look into this as path transformation, we could see that AS 2824's second upstream AS has changed from AS 2914 to AS 6453 (it is worth noting that the first upstream ASN is 32787, which belongs to prolexic). It may be caused by a link failure or a LOCAL_PREF change made by the Administrator, which means that as path "6539 6453 32787 2824" may be a temporary backup link, the route changed to the former link "6539 577 2914 32787 2824" again when the former link restored. This example is a bit different from the early one because we could see no community values neither in the updates message nor in the routing table message, which may be caused by the origin AS 2824 or due to output filtering of community values[69].

To look into the UPDATES behaviors of different DPS, we conduct experiments to see the time span between consecutive UPDATES of each prefix related to the DPS from the same monitor on RoutViews2. We select the top 15 monitors according to the total number of UPDATES messages of each monitor, and the results are shown in From Fig 5.12 and Fig 5.13. From

Table 5.19: AS path of AS3549 and AS4181

time(UTC +8)	community attribute	as path
2012-05-10 08:00:10	712 1210 1411 1510 1711	3549 3491 45474 10107
2012-05-10 08:00:01	-	4181 3491 45474 10107
2012-05-10 10:00:01	732 3110 3120 3210 3220 3230	4181 12989 45474 10107
2012-05-11 12:00:01	732 3110 3120 3210 3220 3230	4181 12989 45474 10107
2012-05-11 14:00:01	-	4181 3491 45474 10107
2012-05-11 14:00:10	712 1210 1411 1510 1711	3549 3491 45474 10107

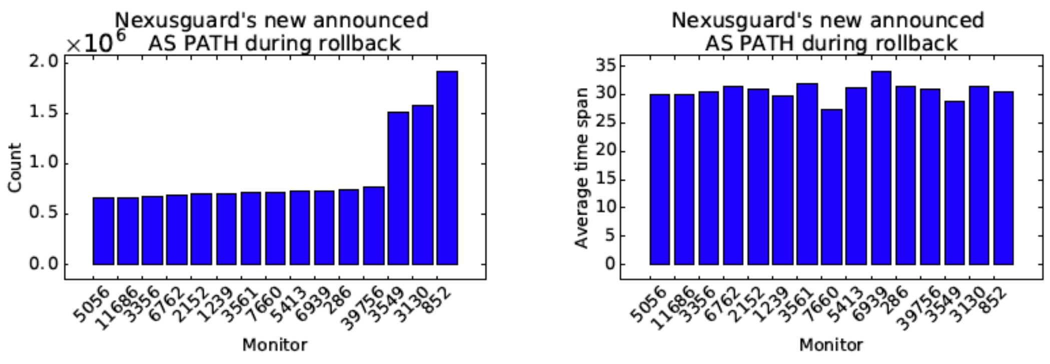
Table 5.20: BGP update of AS6539

time(UTC +8)	community attribute	as path
2011/10/03 13:27:39	-	6539 577 2914 32787 2824
2011/10/03 15:58:23	-	6539 6453 32787 2824
2011/10/03 16:41:30	-	6539 577 2914 32787 2824

Table 5.21: AS path AS6539

time(UTC +8)	community attribute	as path
2011/10/03 14:00:09	-	6539 577 2914 32787 2824
2011/10/03 16:00:09	-	6539 6453 32787 2824
2011/10/03 18:00:09	-	6539 577 2914 32787 2824

Fig 5.12, we can know the monitors receives a different number of UPDATES messages related to the DPS Nexusgurad. Still, the average time span between consecutive UPDATES is much similar and has a value of 30. Compared with Nexusguard, we can see the total number of UPDATES messages related to Prolexic is much smaller. In the fact that, as shown in the former, some routers did not re-announce any prefix solely with changed attributes, but other routers sent a considerable amount of these re-announced UPDATEs with changed attributes [69]. Besides, the average UPDATE time span of the Prolexic is scattered, which may be caused by the same reason as the total UPDATES number of each monitor. From Fig 5.14, we can see that a high proportion of simultaneous UPDATES that reaches about 40% appears both in the Nexusguard and Prolexic. Besides, the proportion value of UPDATES time spans is more than 90% at 20 Seconds, and it reaches more than 97% both in Prolexic and Nexusguard. And there are some slight differences between Nexusguard and Prolexic. For example, the simultaneous UPDATES is more than Prolexic; it lies in the more significant number of UPDATES messages related to Nexusguard.



(a) Number of new AS paths (b) Average time span

Fig. 5.12: Nexusguard's Extracted BGP information

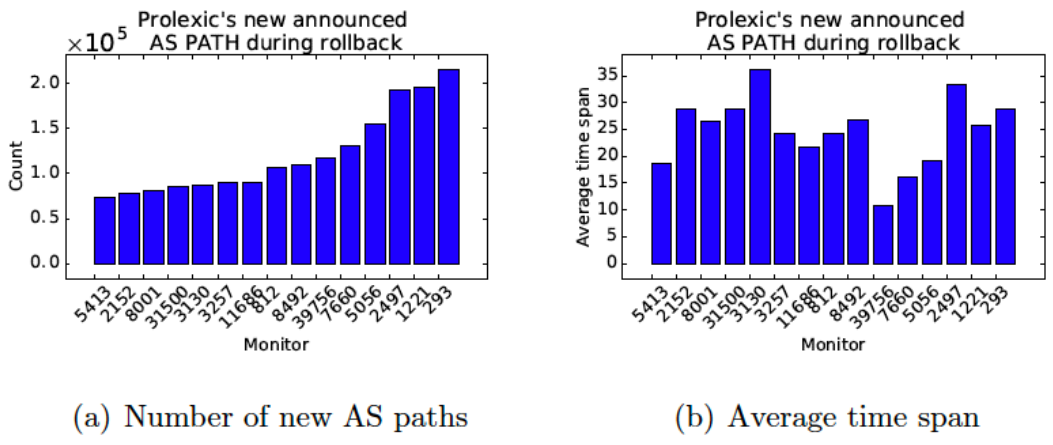


Fig. 5.13: Prolexic's Extracted BGP information

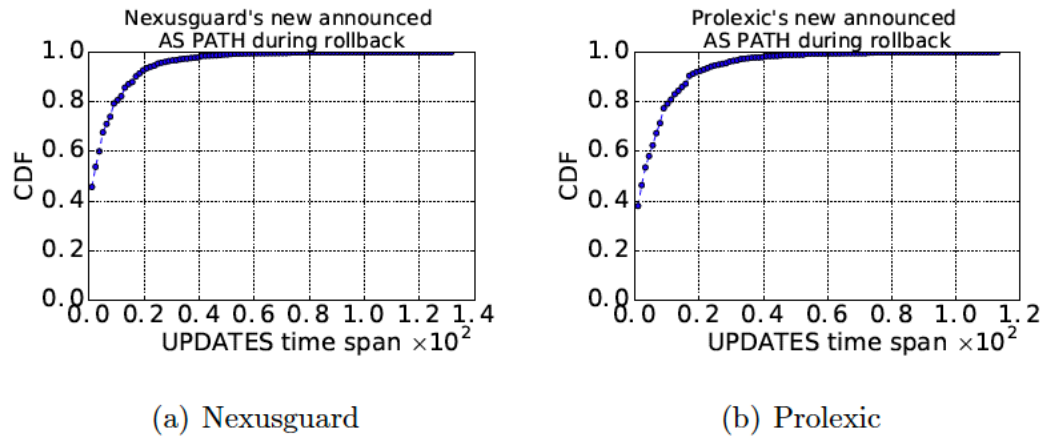


Fig. 5.14: Time span between consecutive UPDATES of each prefix from the same monitor

5.4 Summary

To identify the abnormal BGP dynamics caused by DDoS attacks, we train an accurate classifier based on a dataset of more than 40 manually collected events demonstrated to cause abnormal behaviors of BGP dynamics. We also develop a BGP monitoring and diagnosis system to analyze DDoS attacks with BGP-based DPS mitigation involved. The content of this chapter is published in [57].

6 An Empirical Study of DDoS Amplification Attacks

The DDoS Amplification attacks have an essential role in different global attack cases, and it has been shown that the DDoS amplification attacks accounted for 62.84% of all attacks in 2019 [51]. To better understand the trend and the evolution of DDoS Amplification Attacks, We deploy a honeypot named DDoSTrap to monitor such attacks. The contents of this chapter appear in [54, 52, 55, 53, 51].

6.1 DDoSTrap

DDoSTrap acts as a fake amplifier and listens on the following ports[70] for incoming UDP packets, including Echo(7), CHARGEN(19), TFTP(69), Portmapper(111), SNMP(161), XDMCP(177), LDAP/CLDAP(389), IKE(500), RIP(520), Mssql(1433), Mssql(1434), SSDP(1900), Mysql(3306), Sentinal-5093(5093), NAT-PMP(5351), MDNS(5353), UnrealTournament(7778), Elasticsearch(9200), TeamSpeak3(9987), Memcached(11211), Quake(27960). Once it receives a valid request, it will reply with a protocol-specific pre-generated packet.

Security policies deployment

To avoid DDoSTrap from being exploited to launch real attacks on the victim, we apply a rate limiting policy to it. Specifically, we will drop the response to a client IP address if the client sends more than 300 requests per minute. We evaluate the blacklist every hour and remove an IP address from the blacklist if it stops sending requests after one hour. We further limit the rate of response traffic to be less than 10Mbps. During the four-year operation, we only received two emails from the Google Cloud Platform(GCP) that host our honeypots, and we responded responsibly. After our clarification, no victim claimed that the honeypots caused damage.

Data Collection

Fig 6.1 shows the architecture of DDoSTrap, which includes a rabbitMQ

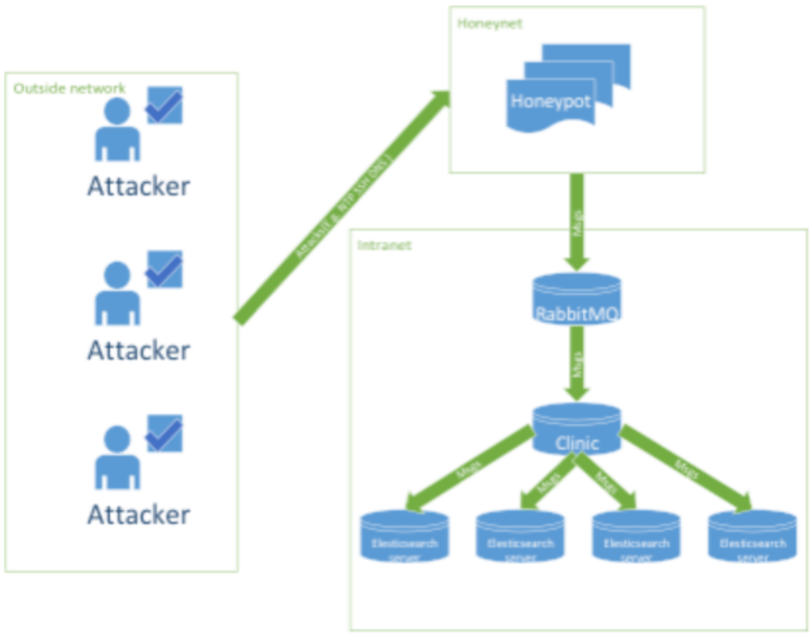


Fig. 6.1: Overview of the DDoSTrap

data broker, the logstash/elasticsearch servers, and honeypots. We use Elasticsearch[23] to process the collected data. RabbitMQ is an open source message broker supporting the Advanced Message Queuing Protocol[66]. The RabbitMQ server sends the collected data to Clinic server, a middleware, using logstash to convert and deliver messages from rabbitMQ(TSV text) to Elasticsearch servers(JSON format). When processing the captured data, we need to separate actual attacks from scanning packets. To this end, we filter out those IP sources that send at least 100 consecutive requests to our honeypot within 5 minutes and the intervals between two packets are less than 5 seconds. If a source keeps send the same requests for more than 15 minutes, we regard it as an attack. If the attack is paused for an hour and then resumed, we will divide the relevant traffic into two attacks.

Since 2016, we have installed 8 DDoSTrap instances on Google Cloud Platform (GCP) using static IP addresses to collect attack data. Table 6.1 summarizes the details. We try to spread the honeypots in different regions to contain more extensive real attacks.

Tool Sharing

The first version of DDoSTrap was developed by Terrence Gareau and sponsored by A10 Network, Nexusguard, and Cari.net in 2016 [70]. We share DDoSTrap data with trusted parties(Banking group, governments, Team

Table 6.1: Overview of honeypot deployments

HP	Geolocation	Deployed	IP Add.	Service
H01	United States	2016-12-28	Static	21
H02	Taiwan	2016-12-28	Static	21
H03	Belgium	2016-12-28	Static	21
H04	Hong Kong	2016-12-28	Static	21
H05	London	2016-12-28	Static	21
H06	Sao Paulo	2016-12-28	Static	21
H07	Singapore	2016-12-28	Static	21
H08	Sydney	2016-12-28	Static	21

Cymru, FBI, HK Policies, etc.) and make it accessible to fellow researchers, assuming that we can use the derived data. Source code is free to download from GitHub [70].

6.2 DDoS amplification attack analysis

Fig 6.2 summarizes the attacks our honeypots observed during the period from Jan 2017 to Dec 2020. It captured 10,458 attacks and 2,198,110,454 raw requests. We can see that most attacks exploit the DNS protocol. The second one is the SSDP protocol. Moreover, the raw requests of SSDP, CHARGEN, NTP, MDNS, NAT-PMP, RIP, Sentinal-5093, LDAP/CLDAP, Memcached, Mssql, TeamSpeak3, Elasticsearch, TFT, UnrealTournament, IKE, Quake, and SNMP occupy less than 5.8% of all raw requests.

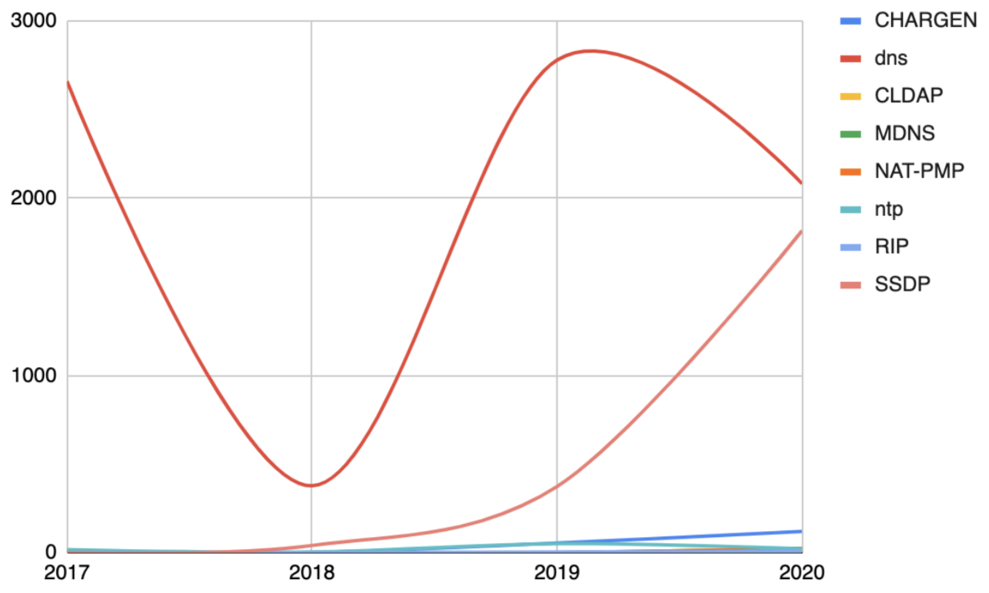


Fig. 6.2: Number of attacks per protocol between 2017 and 2020

Attack Duration

Figure 6.3 shows the distribution of attack duration, which is defined as the time interval between the first and last data packets involved in an attack on a particular victim. It shows that the amplification attacks are usually short-lived: 38.4% of the attack duration is shorter than 20 minutes, and 72.5% of the attack duration is shorter than 1 hour. Only 1.15% of attacks lasted for more than 23 hours.

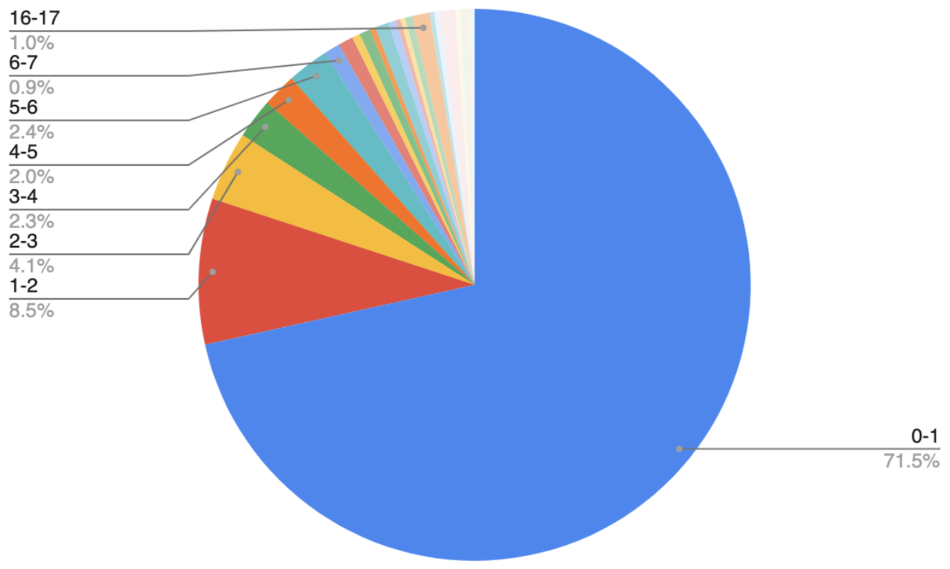


Fig. 6.3: Attack duration by hours

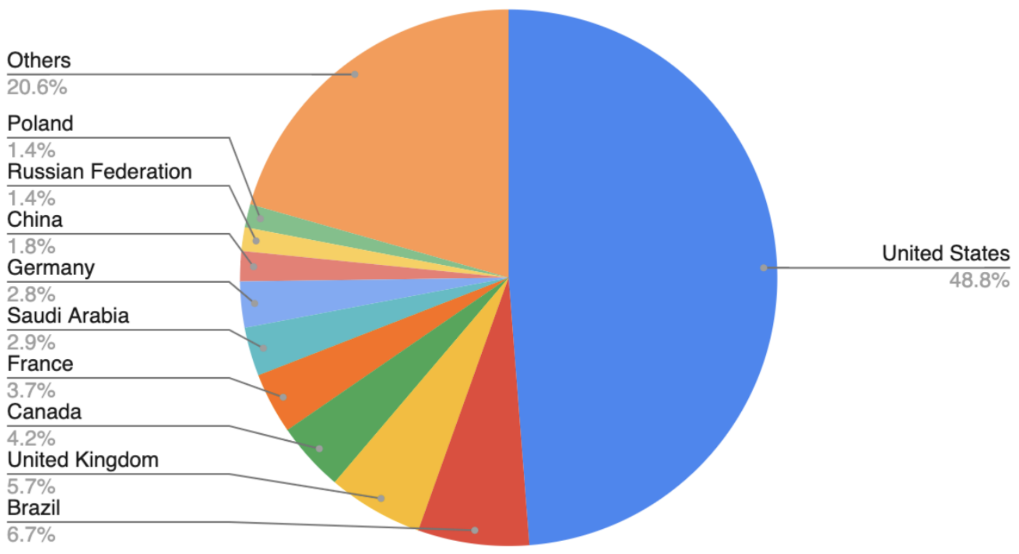


Fig. 6.4: Geolocation of victim by countries

Victim Analysis

According to the source IP addresses of the amplification attack, we analyze the Geolocation of Victims. Figure 6.4 shows the distribution of victims by their countries. In 2019, when looking at countries, the U.S. stands out, hosting half of the victims. There are many victims in Brazil (6.7%) and United Kingdom (5.7%), and all other victims in 231 countries have a long-tail distribution. We further investigated the attack peak period from Sep 2019 to Dec 2019. Most attacks targeted global communications service providers (ISP) — mobile networks, ISPs, and cloud service providers [54], such as Comcast, Virgin Media, Aliyun Computing, Amazon, and China Mobile. In particular, Comcast was attacked nearly every day. According to our findings, ASes in the U.S., the U.K., and China were hit the hardest, with the U.S. being hit by more than 42 million individual attacks. Reflection attacks that exploit open DNS resolvers were used most frequently, and the stealthy bit-and-piece attacks(detailed in section 6.3) continued to cause outages and remained as a threat.

United States - More than 1,000 ASes of most industries in the U.S. fell victim to DDoS attacks, the highest level of activities that has ever been seen.

United Kingdom - DDoS attacks against the U.K. networks were most likely politically driven as the high level of occurrence of attacks coincided with the country's election period. Virgin Media, a British company that provides telephone, television, and internet services., was hit hard, whose 244 PoPs were all attacked. The Bank of England was also one of the victims.

China - Though China has the world's largest internet population, attack activity was relatively subdued compared with the U.S. and the U.K. Most attacks targeted Alibaba and Aliyun Computing. That's likely due to the fact that most cloud services rely on China's Aliyun cloud, which is protected by its anti-DDoS platform.

Table 6.2: Overview of countries under large scope attacks between Sep 2019 to Dec 2019

Country	Amplification Attack	Bit-and-Piece Attack	Total of amplification raw request
United States	2,209	76	42,370,109
United Kingdom	295	12	4,696,498
China	261	18	11,528,380
Brazil	143	75	5,994,470
Hong Kong	162	5	4,830,051
Canada	155	0	3,060,355
Saudi Arabia	145	0	1,920,072
Germany	99	1	3,386,033

DNS Abuse Domain List

As shown in Fig. 6.2, amplification attacks exploiting DNS account for 75.5%. Therefore, for defending against such attacks, we create a list of domains abused for amplification attacks. Table 6.3 lists the ten most popular attack domains since December 2020, in descending order of the number of requests recorded in the honeypot.

Table 6.3: DNS domains, ordered by the number of request seem at the honeypot between 2017 and 2020

FQDN	Type	Request Count	First Seen	Last Seen	Days	Victim IPs
isc.org	ANY	918,461,633	2018-07-29	2021-07-31	1,098	829,789
arctic.gov	ANY	187,020,688	2019-03-01	2019-03-11	10	100,489
1x1.cz	ANY	86,924,677	2017-07-09	2021-07-31	1,483	843,786
doc.gov	ANY	78,644,456	2017-10-09	2021-07-31	1,391	654,324
wzb.eu	ANY	78,493,514	2019-03-13	2021-07-31	871	583,935
mz.gov.pl	ANY	76,233,583	2018-07-29	2021-07-31	1,098	810,373
peacecorps.gov	ANY	68,166,193	2019-10-15	2021-07-26	650	116,736
commerce.gov	ANY	61,231,866	2017-10-09	2021-07-31	1,391	695,112
aids.gov	ANY	57,689,436	2017-10-09	2021-07-31	1,391	819,588
paypal.com	ANY	46,657,045	2017-02-21	2021-07-31	1,621	776,187

6.3 Case studies

6.3.1 The bit-and-piece attack

Since July 2018, we have observed that 159 ASes and 527 Class C networks were targeted in a series of amplification attacks where attackers injected small bits and pieces of junk into legitimate traffic as a disguise [52]. Consequently, attack traffic in the space of each IP address was small enough to bypass detection but big enough to cripple the targeted site or even an entire ISP network once the traffic converged. Due to the negligible size of the junk traffic, typical security devices deployed by ISPs are unable to detect and mitigate the attack before it can cause any harm.

We investigated the honeypot data between 2018 and 2020 and found that bit-and-piece attacks impacted on average 318 ASes every year. As shown in Table 6.4, the maximum attack duration was 1439.67 minutes.

One attack case was presented in the report [52] prepared by us, where the orchestrated attacks generated only 33.2Mbps per destination IP, which is small enough to fly under the radar and will be easily regarded as legitimate

Table 6.4: Summary of Bit-and-Piece Attacks

	2018	2019	2020
No. of Targeted ASes	397	305	251
No. Target Geolocations	28	50	23
Total IP prefixes under attack(Class C)	892	1,207	1,188
Minimum no. of IP addresses under attack(per class c IP prefix)	32	30	30
Maximum no. of IP addresses under attack(per class c IP prefix)	252	256	256
Minimum Attack Duration(Minutes)	6.32	10.22	3.37
Maximum Attack Duration(Minutes)	1,439.67	1,391.38	1,433.85
Minimum Attack per IP	15	40	40
Maximum Attack per IP	45118	42946	70442
Minimum Attack per class c IP Prefix	191	200	937
Maximum Attack per class c IP Prefix	433,999	496,734	5,219,918

traffic delivered straight to the destination AS. Table 6.5 reveals that the campaign was significant. We found that attackers targeted networks within the exact geo-location, attempting to exhaust the capacity of transmission lines. In the worst-case scenario, the convergence of attack traffic spread across 38 IP prefixes, each loaded with 2.48Gbps of attack traffic — potent enough to overwhelm a 10Gbps ISP line.

Table 6.5: Summary of Actual Bit-and-Piece Attacks

Targeted ASes	159
Attack Types	DNS Amplification, SSDP, CHARGEN, NTP Amplification
Total IP Prefixes (Class C Networks) Under Attack	527
Maximum no. of Targeted IP Addresses per class c IP Prefix	252
Minimum no. of Targeted IP Addresses per class c IP Prefix	49
Average no. of Targeted IP Addresses per class c IP Prefix	131
Maximum Attack Durations(Minutes)	1,439.67
Minimum Attack Durations(Minutes)	5.12
Average Attack Durations(Minutes)	113.81
Maximum Attack Sizes per IP	300.1Mbps
Minimum Attack Sizes per IP	2.5Mbps
Average Attack Sizes per IP	33.2Mbps
Maximum Attack Sizes per class c IP Prefix	5.32Gbps
Minimum Attack Sizes per class c IP Prefix	285.4Mbps
Average Attack Sizes per class c IP Prefix	2.48Gbps

Since Feb 15, 2020, we have seen a shift in the attack tactics. That is, the attackers opt for a more deceptive and sophisticated approach by launching amplification and different types of UDP-based attacks to flood target networks[55]. In particular, smaller and more complex UDP-based and a variety of amplification attacks were often used to maximize the impact on target networks. Interestingly, for every wave of bit-and-piece attacks, a single IP address is selected in the same IP prefix to receive a large flood attack, namely, a UDP-based attack or amplification attack, in size range of 300Mbps to 21Gbps [55]. It may act as a smokescreen to distract in-house security teams from bit-and-piece attacks that are taking place in attempts to take down ISP infrastructures.

Table 6.6: Summary of Bit-and-Piece Attack Vectors

Distribution of Attack Vectors	Targeted Geo-locations
UDP Attack(44.57%), DNS Amplification Attack(35.68%), UDP Fragmentation Attack(6.8%), CLDAP Reflection Attack(6.24%), SSDP Amplification Attack(3.87%), CHARGEN Attack(2.19%), DNS Attack(0.56%), IP BOGONS(0.05%), SIP Flood(0.05%)	Argentina, Bangladesh, Brazil, Canada, China, Hong Kong, Islamic Republic of Iran, Japan, Lebanon, Netherlands, Poland, Romania, Russian Federation, Singapore, South Africa, Taiwan, Turkey, Ukraine, United States

In the past, attackers have utilized bit-and-piece attacks with a single attack vector such as a UDP amplification attack to launch UDP-based attacks. However, in 2020, there has been a tendency to employ a blend of attack vectors to launch a wider range of UDP-based attacks [55]. The combined effect of this tactic is to increase the difficulty for ISPs to detect and differentiate between attack and legitimate traffic. Table 6.6 illustrates the types of attack vectors employed in this actual attack. Note that 44.57% of attacks were attributed to UDP attacks, though in previous studies and instances of bit-and-piece attacks, they were not commonly used. Moreover, we found that UDP-based attacks were characterized in the 4Mbps to 21.64Gbps size range, which is smaller than previously observed [55]. UDP-based attacks are still widely adopted. Though smaller in size now, these types of UDP-based attacks can be enlarged by randomly crafting payloads to congest target networks.

6.3.2 Amplification attacks exploiting DNSSEC

The Domain Name System (DNS) is a fundamental element in Internet technology as it translates domain names into corresponding IP addresses. DNS servers are constantly bused to reflect DNS amplification attacks. The continued adoption of DNSSEC suggests that DNS Amplification will remain a mainstream attack method and continue to pose a significant threat to the Internet.

DNSSEC was designed to protect applications from forged or manipulated DNS data. The extra security provided by DNSSEC relies on a resource-intensive data verification process using public keys and digital signatures. Table 6.7 compares the amplification factors of the ten most frequently abused domains before and after DNSSEC adoption. Using aids.gov as an example, the domain's DNS server amplification power surged to more than 45.28X (up from 4.53X) after DNSSEC deployment.

Table 6.7: 10 Most Frequently Abused Domains and Query Counts of DNS Requests

Domain	Amp Factor(no DNSSEC)	Amp factor included DNSSEC
1x1.cz	8.19	72.55
edu.za	3.36	47.96
aids.gov	4.53	45.28
isc.org	3.92	58.89
eftps.gov	4.25	44.37
mz.gov.pl	2.31	48.31
paypal.com	3.96	42.24
leth.cc	4.96	53.52
dfafacts.gov	2.53	36.67
nel.gov	2.69	41.71

On Jan 5, 2019, we observed that multiple government domains (as well as paypal.com) fell victim to rampant attacks. Closer scrutiny, however, suggests that many of these domains had deployed DNSSEC to the top-level .gov domain as required by the U.S. government's OMB mandate. There is a strong causal relation between DNSSEC implementation and increased DNS Amplification because, due to the large size of responses they generate. DNSSEC-enabled servers are at risk of being targeted to reflect amplification attacks [53] as shown in Table 6.8.

Table 6.8: 10 Most Frequently Abused Domains and Query Counts of DNS Requests

Domain	Query Count	Percentage	included DNSSEC or not
1x1.cz	16,605,666	11.49%	yes
edu.za	13,524,481	9.36%	yes
aids.gov	12,640,652	8.75%	yes
isc.org	12,541,244	8.68%	yes
eftps.gov	11,423,694	7.91%	yes
mz.gov.pl	10,811,274	7.48%	yes
paypal.com	9,403,514	6.51%	yes
leth.cc	9,118,943	6.31%	yes
dfafacts.gov	7,299,000	5.05%	yes
nel.gov	7,212,696	4.99%	yes
Others	33,884,389	23.45%	-

6.3.3 Amplification attacks exploiting Memcached

Attackers are constantly seeking to magnify the power of DDoS weapons by exploring new Amplification attack methods. Memcached is an open-source and distributed memory-caching system to caching data for webserver and assessed internally. Due to poor security deployment, attackers abuse it as an amplifier to generate DDoS. When attackers look for Memcached servers that can be exploited to launch attacks, the logs in DDoSTraip show that they tried different source IPs once or twice and typically used a “Version” or “Gets” scanning request. For example, when an attacker sent a “Version” request to confirm the existence of the target, packet size of the request was 17 bytes, whereas the returning packet was 14 bytes.

DDoSTrap collected a lot of information about Memcached attacks. It shows that most of targets’ IPs came from Cloud providers (82.05%), Telecoms (12.83%), DNS servers providers(2.56%) and Education network(2.56%). Table 6.9 summarizes the information of the observed attacks.

Table 6.9: Memcached Attack Summary in 2019

	Maximum	Minimum	Average
No. of Attacks (Count)	17339.00	68.00	1843.15
Frequency (per Second)	306.93	0.41	161.63
Duration (Seconds)	1440.98	0.54	69.98

Conclusions and Future Work 7

Conclusions. To understand and defend against advanced DDoS attacks, we investigate them from three aspects. First, we examine the application layer DDoS attacks that send requests similar to benign ones for evading the detection and consuming the computational resources of target servers. Specifically, we design a new approach to model users' browsing behaviors and use it to differentiate between attacks and benign visits at both session and page level. By using a dataset of real attacks to conduct the evaluation, the results showed that our approaches can detect application layer DDoS attacks. To defeat such attacks, we further develop an effective system named SkyShield that leverages the sketch data structure to detect and mitigate application-layer DDoS attacks quickly as well as address the limitations of previous studies. We also leverage other techniques, including Bloom filters and the CAPTCHA techniques, to improve the performance of SkyShield. By using a real attack dataset collected from a large-scale web cluster to evaluate SkyShield, we find that it can effectively mitigate app-layer DDoS attacks and pose a limited impact on normal users. Second, we characterize the BGP-based DDoS protection services (DPS) by analyzing BGP update messages. In particular, to identify the abnormal BGP dynamics caused by DDoS attacks, we train a classier by using a dataset of more than 40 manually collected events that have been demonstrated to cause abnormal behaviors of BGP dynamics. By applying our approach to actual DDoS attacks, we identify the policies used by DPS to mitigate the attacks and obtain interesting observations. Third, to understand the trends and the evolution of DDoS amplification attacks, we deploy DDoSTrap, a high-performance honeypot to collect data. By analyzing the data, we identify the majority threat in amplification attacks and report a new DDoS strategy called a Bit-and-piece attack. Moreover, we observe that multiple government domains fell victim to DNSSEC attacks and attackers exploited memcached to launch amplification attacks.

Future work. We will extend the current research in three ways. First, since many detection systems against application-layer DDoS attacks rely on machine learning techniques, we will investigate to what extent they can be evaded by generating adversarial samples and quantifying the cost of generating such adversarial samples. Based on this result, we will design more robust detect systems for capturing stealthy application-layer DDoS attacks. Second, since more and more ISPs will deploy programmable network elements, such as P4 switches, we will propose a distributed in-network detection system against both network layer and application layer DDoS attacks. By doing so, the service provider can detect and mitigate the attacks as soon as possible. Third, since attackers may use various approaches to figure out whether a compromised host is a honeypot or not, we will first investigate whether our DDoSTrap system could be identified or not. If that is the case, we will propose new approaches to enhance the fidelity of DDoSTrap for cheating the attackers. Moreover, we will try to deploy it to more places around the world to collect data for further studying the evolution of various DDoS attacks.

References

[1]Akamai. *Anatomy of a SYN ACK Attack*. `https://www.akamai.com/blog/security/anatomy-of-a-syn-ack-attack` (cit. on p. 5).

[2]Akamai. *Prolexic Routed | Product Brief | Akamai*. `https://www.akamai.com/site/en/documents/product-brief/prolexic-routed-product-brief.pdf` (cit. on p. 57).

[3]M. Al-Rousan and L. Trajković. "Machine learning models for classification of BGP anomalies". In: *2012 IEEE 13th International Conference on High Performance Switching and Routing*. IEEE. 2012, pp. 103–108 (cit. on p. 9).

[4]T. Anderson, T. Roscoe, and D. Wetherall. "Preventing Internet denial-of-service with capabilities". In: *ACM SIGCOMM Computer Communication Review* 34.1 (2004), pp. 39–44 (cit. on p. 8).

[5]M. Arlitt and T. Jin. *1998 World Cup Web Site Access Logs*. `http://ita.ee.lbl.gov/html/contrib/WorldCup.html` (cit. on p. 45).

[6]P. Barford, J. Kline, D. Plonka, and A. Ron. "A signal analysis of network traffic anomalies". In: *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment - IMW 02* (2002) (cit. on p. 8).

[7]D. Birant and A. Kut. "ST-DBSCAN: An algorithm for clustering spatial–temporal data". In: *Data & Knowledge Engineering* 60.1 (2007), pp. 208–221 (cit. on p. 15).

[8]A. Broder and M. Mitzenmacher. "Network Applications of Bloom Filters: A Survey". In: *Internet Mathematics* 1.4 (2004), pp. 485–509 (cit. on p. 24).

[9]M. Butkiewicz, D. Wang, Z. Wu, H. Madhyastha, and V. Sekar. "Klotski: Reprioritizing web content to improve user experience on mobile devices". In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. 2015, pp. 439–453 (cit. on p. 12).

[10]M. Caesar and J. Rexford. "BPG routing policies in ISP networks". In: *IEEE Netw.* 19.6 (2005), pp. 5–11 (cit. on p. 58).

[11]M. Caesar, L. Subramanian, and R. Katz. *Towards localizing root causes of BGP dynamics*. Citeseer, 2003 (cit. on p. 9).

[12]D. Chang, R. Govindan, and J. Heidemann. “The temporal and topological characteristics of BGP path changes”. In: *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*. IEEE. 2003, pp. 190–199 (cit. on p. 9).

[13]Cloudflare. *Cloudflare DDoS Threat Report 2022 Q3*. `https://blog.cloudflare.com/cloudflare-ddos-threat-report-2022-q3` (cit. on p. 5).

[14]Cloudflare. *DDoS Attack Trends for 2021 Q*. `https://blog.cloudflare.com/ddos-attack-trends-for-2021-q4/k` (cit. on p. 7).

[15]A. Cohen, Y. Gilad, A. Herzberg, and M. Schapira. “Jumpstarting BGP Security with Path-End Validation”. In: () (cit. on p. 3).

[16]J. Cowie, A. Ogielski, B. Premore, E. Smith, and T. Underwood. “Impact of the 2003 blackouts on Internet communications”. In: *Preliminary Report, Renesys Corporation (updated March 1, 2004)* (2003) (cit. on p. 9).

[17]J. Cowie, A. Ogielski, B. Premore, and Y. Yuan. “Internet worms and global routing instabilities”. In: *ITCom 2002: The Convergence of Information Technologies and Communications*. International Society for Optics and Photonics. 2002, pp. 195–199 (cit. on p. 9).

[18]J. Czyz, M. Kallitsis, M. Gharaibeh, et al. “Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks”. In: *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014) (cit. on p. 5).

[19]S. Daniel. *How Friday’s Massive DDoS Attack on the U.S. Happened*. `https://blog.radware.com/security/2016/10/fridays-massive-ddos-attack-u-s-happened/` (cit. on p. 61).

[20]DARPA. *Extreme DDoS defense*. `http://www.darpa.mil/program/extreme-ddos-defense`. Online, accessed 17-April-2016. 2015 (cit. on pp. 1, 6).

[21]S. Deshpande, M. Thottan, T. Ho, and B. Sikdar. “An online mechanism for BGP instability detection and analysis”. In: *IEEE Transactions on Computers* 58.11 (2009), pp. 1470–1484 (cit. on p. 9).

[22]B. Donnet and O. Bonaventure. “On BGP Communities”. In: () (cit. on pp. 58, 69).

[23]*Elasticsearch*. `https://aws.amazon.com/what-is/elasticsearch/` (cit. on p. 74).

[24]A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. “Locating Internet routing instabilities”. In: *ACM SIGCOMM Computer Communication Review* 34.4 (2004), pp. 205–218 (cit. on p. 9).

[25] S. Ganguly, M. Garofalakis, R. Rastogi, and K. Sabnani. “Streaming algorithms for robust, real-time detection of DDoS attacks”. In: *Proc. ICDCS* (2007), p. 4 (cit. on pp. 2, 8).

[26] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. “QuickSAND: Quick summary and analysis of network data”. In: *DIMACS, Piscataway, NJ, USA, Tech. Rep.* (2001) (cit. on p. 2).

[27] P. Golle. “Machine learning attacks against the Asirra CAPTCHA”. In: *Proceedings of the 15th ACM conference on Computer and communications security - CCS 08* (2008) (cit. on p. 7).

[28] H. Griffioen, K. Oosthoek, P. Knaap, and C. Doerr. “Scan, test, execute: Adversarial tactics in amplification ddos attacks”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021) (cit. on p. 5).

[29] *How Friday’s Massive DDoS Attack on the U.S. Happened*. `https://en.wikipedia.org/wiki/2016_Dyn_cyberattackcite_note-wired-5/` (cit. on p. 61).

[30] J. Hunter. “The exponentially weighted moving average”. In: *J. Quality Technol.* 18.4 (1986), pp. 203–210 (cit. on p. 34).

[31] N. Janevski and K. Goseva-Popstojanova. “Session reliability of web systems under heavy-tailed workloads: an approach based on design and analysis of experiments”. In: *Software Engineering, IEEE Transactions on* 39.8 (2013), pp. 1157–1178 (cit. on p. 21).

[32] M. Jonker, A. Sperotto, R. van, R. Sadre, and A. Pras. “Measuring the Adoption of DDoS Protection Services”. In: *Proceedings of the 2016 ACM on Internet Measurement Conference*. ACM. 2016, pp. 279–285 (cit. on pp. 2, 8).

[33] 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 (cit. on pp. 6–8).

[34] M. Karami and D. McCoy. “Understanding the Emerging Threat of DDoS-as-a-Service.” In: *LEET*. 2013 (cit. on p. 2).

[35] R. Kompella, S. Singh, and G. Varghese. “On scalable attack detection in the network”. In: *Proc. IMC* (2004), pp. 187–200 (cit. on p. 2).

[36] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. “Sketch-based change detection: Methods, evaluation, and applications”. In: *Proc. IMC* (2003), pp. 234–247 (cit. on pp. 2, 8, 24).

[37] J. Krupp, M. Backes, and C. Rossow. “Identifying the scan and attack infrastructures behind amplification ddos attacks”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016) (cit. on p. 5).

[38] C. Labovitz, G. Malan, and F. Jahanian. “Internet routing instability”. In: *IEEE/ACM transactions on Networking* 6.5 (1998), pp. 515–528 (cit. on p. 9).

[39] S. LaPerrière. “Taiwan earthquake fiber cuts: a service provider view”. In: *NANOG39, Febraury* 5 (2007) (cit. on p. 9).

[40] S. Lee, G. Kim, and S. Kim. “Sequence-order-independent network profiling for detecting application layer DDoS attacks”. In: *EURASIP Journal on Wireless Communications and Networking* 2011.1 (2011), pp. 1–9 (cit. on pp. 7, 14, 18).

[41] J. Li and S. Brooks. “I-seismograph: Observing and measuring Internet earthquakes”. In: *INFOCOM, 2011 Proceedings IEEE*. IEEE. 2011, pp. 2624–2632 (cit. on pp. 9, 10, 50).

[42] J. Li, M. Guidero, Z. Wu, E. Purpus, and T. Ehrenkranz. “BGP routing dynamics revisited”. In: *ACM SIGCOMM Computer Communication Review* 37.2 (2007), pp. 5–16 (cit. on pp. 9, 52).

[43] J. Li, Z. Wu, and E. Purpus. “CAM04-5: Toward Understanding the Behavior of BGP During Large-Scale Power Outages”. In: *IEEE Globecom 2006*. IEEE. 2006, pp. 1–5 (cit. on p. 9).

[44] Y. Li, L. Guo, Z. Tian, and T. Lu. “A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms”. In: *Computer Communications* 31.17 (2008), pp. 4018–4025 (cit. on p. 7).

[45] Y. Li, T. Lu, L. Guo, Z. Tian, and Q. Nie. “Towards lightweight and efficient DDOS attacks detection for web server”. In: *Proceedings of the 18th international conference on World wide web - WWW 09* (2009) (cit. on pp. 7, 46, 47).

[46] Z. Li and J. Tian. “Testing the suitability of Markov chains as Web usage models”. In: *Proceedings 27th Annual International Computer Software and Applications Conference. COMPAC 2003* () (cit. on p. 11).

[47] Z. Li, M. Zhang, Z. Zhu, et al. “WebProphet: Automating Performance Prediction for Web Services.” In: *NSDI*. Vol. 10. 2010, pp. 143–158 (cit. on p. 12).

[48] H. Liu, Y. Sun, and M. Kim. “Fine-Grained DDoS Detection Scheme Based on Bidirectional Count Sketch”. In: *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)* (2011) (cit. on pp. 2, 8).

[49] X. Liu, CX. Yang, and Y. Xia. “NetFence Preventing Internet denial of service from inside out”. In: *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM 10* (2010) (cit. on p. 8).

[50] X. Liu, X. Yang, and Y. Lu. “To filter or to authorize: Network-layer dos defense against multimillion-node botnets”. In: *ACM SIGCOMM Computer Communication Review* 38.4 (2008), pp. 195–206 (cit. on p. 8).

[51]T. Miu. *Annual Threat Report 2020*. https://blog.nexusguard.com/threat-report/annual-threat-report-2020 (cit. on p. 73).

[52]T. Miu. *DDoS Threat Report 2018 Q3*. https://blog.nexusguard.com/threat-report/ddos-threat-report-2018-q3 (cit. on pp. 73, 78).

[53]T. Miu. *DDoS Threat Report 2019 Q2*. https://blog.nexusguard.com/threat-report/ddos-threat-report-2019-q2 (cit. on pp. 73, 81).

[54]T. Miu. *DDoS Threat Report 2019 Q4*. https://blog.nexusguard.com/threat-report/ddos-threat-report-2019-q4 (cit. on pp. 73, 77).

[55]T. Miu. *DDoS Threat Report 2020 Q2*. https://blog.nexusguard.com/threat-report/ddos-threat-report-2020-q2 (cit. on pp. 73, 79, 80).

[56]T. Miu, C. Wang, X. Luo, and J. Wang. "Modeling User Browsing Activity for Application Layer DDoS Attack Detection". In: *Proc. 12th International Conference on Security and Privacy in Communication Networks (SecureComm)* (2016) (cit. on p. 21).

[57]T. Miu, C. Wang, and J. Wang. "Understanding the Behaviors of BGP-based DDoS Protection Services". In: *Proc. 12th International Conference on Network and System Security (NSS)* (2018) (cit. on p. 72).

[58]D. Montgomery. "Introduction to Statistical Quality Control". In: *Hoboken, NJ, USA: Wiley* (2007) (cit. on p. 30).

[59]R. Netravali, J. Mickens, and H. Balakrishnan. "Polaris: Faster page loads using fine-grained dependency tracking". In: *Proc. NSDI* (2016), pp. 123–136 (cit. on p. 12).

[60]A. Noroozian, M. Korczyński, C. Gañan, et al. "Who gets the boot? Analyzing victimization by DDoS-as-a-service". In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 2016, pp. 368–389 (cit. on p. 2).

[61]A. Norrozian, M. Korczynski, D. Makita, K. Yoshioka, and M. van Eeten. "Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service". In: *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*. Vol. 9854. Springer. 2016, p. 368 (cit. on p. 10).

[62]"Origin Protection What is Origin Protection?" In: () (cit. on p. 58).

[63]J. Park, D. Jen, M. Lad, et al. "Investigating occurrence of duplicate updates in BGP announcements". In: *International Conference on Passive and Active Network Measurement*. Springer. 2010, pp. 11–20 (cit. on p. 52).

[64]D. Pogue. "Time to Kill Off Captchas". In: *Scientific American* 306.3 (2012), pp. 23–23 (cit. on p. 7).

[65]B. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos. “BGP-lens: Patterns and anomalies in internet routing updates”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 1315–1324 (cit. on p. 9).

[66]*RabbitMQ*. `https://www.rabbitmq.com/` (cit. on p. 74).

[67]J. Rangasamy, D. Stebila, C. Boyd, and J. Nieto. “An integrated approach to cryptographic mitigation of denial-of-service attacks”. In: *Proc. ASIACCS* (2011), pp. 114–123 (cit. on p. 7).

[68]S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly. “DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks”. In: *IEEE/ACM Transactions on Networking* 17.1 (2009), pp. 26–39 (cit. on pp. 1, 5).

[69]P. Richter, A. Feldmann, and S. Uhlig. “Classification of origin AS behavior based on BGP update streams”. In: (2010) (cit. on pp. 70, 71).

[70]C. Rossow. “Amplification Hell: Revisiting Network Protocols for DDoS Abuse”. In: *Proceedings 2014 Network and Distributed System Security Symposium* (2014) (cit. on pp. 73–75).

[71]J. Santanna, R. van, R. Hofstede, et al. “Booters—An analysis of DDoS-as-a-service attacks”. In: *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE. 2015, pp. 243–251 (cit. on p. 2).

[72]C. Schutijser. *Comparing DDoS Mitigation Techniques*. 2016 (cit. on p. 57).

[73]R. Schweller, Z. Li, Y. Chen, et al. “Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications”. In: *Proc. INFOCOM* (2006), pp. 1–12 (cit. on pp. 2, 8, 33).

[74]H. Scott. *Dyn Analysis Summary Of Friday October 21 Attack*. `http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/` (cit. on pp. 5, 61, 62).

[75]S. Sivabalan and P. Radcliffe. “A novel framework to detect and block DDoS attack at the application layer”. In: *IEEE 2013 Tencon - Spring* (2013) (cit. on pp. 7, 8, 18).

[76]R. Sommese, K. Claffy, R. Van, et al. “Investigating the impact of ddos attacks on DNS infrastructure”. In: *Proceedings of the 22nd ACM Internet Measurement Conference* (2022) (cit. on p. 5).

[77]M. Su. “Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers”. In: *Expert Systems with Applications* 38.4 (2011), pp. 3492–3498 (cit. on p. 8).

[78] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. Liu. “A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.2 (2014), pp. 447–456 (cit. on pp. 1, 5).

[79] J. Tang, Y. Cheng, Y. Hao, and W. Song. “A scalable, efficient and informative approach for anomaly-based intrusion detection systems: Theory and practice”. In: *Int. J. Netw. Manage.* 20.5 (2010), pp. 271–293 (cit. on pp. 2, 8).

[80] J. Tang, Y. Cheng, Y. Hao, and W. Song. “SIP flooding attack detection with a multi-dimensional sketch design”. In: *IEEE Trans. Depend. Sec. Comput.* 11.6 (2014), pp. 582–595 (cit. on pp. 2, 8).

[81] J. Tang, Y. Cheng, and C. Zhou. “Sketch-based sip flooding detection using Hellinger distance”. In: *Proc. GLOBECOM* (2009), pp. 1–6 (cit. on pp. 2, 8).

[82] D. Wagner, D. Kopp, M. Wichtlhuber, et al. “UNITED WE STAND: Collaborative detection and mitigation of amplification ddos attacks at scale”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021) (cit. on p. 5).

[83] A. Wang, A. Mohaisen, W. Chang, and S. Chen. “Delving into Internet DDoS Attacks by Botnets: Characterization and Analysis”. In: *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (2015) (cit. on p. 28).

[84] C. Wang*, T. Miu*, X. Luo, and J. Wang. “SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks”. In: *IEEE Trans. Inf. Forensics Secur. (TIFS) 13(3): 559-573. (*equal contributions.)* (2018) (cit. on p. 48).

[85] H. Wang, Q. Jia, D. Fleck, et al. “A moving target DDoS defense mechanism”. In: *Computer Communications* 46 (2014), pp. 10–21 (cit. on p. 8).

[86] X. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. “Demystifying page load performance with WProf”. In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 2013, pp. 473–485 (cit. on p. 12).

[87] Y. Wu, Z. Zhao, F. Bao, and R. Deng. “Software Puzzle: A Countermeasure to Resource-Inflated Denial-of-Service Attacks”. In: *IEEE Transactions on Information Forensics and Security* 10.1 (2015), pp. 168–177 (cit. on p. 7).

[88] T. Xie and S. Yu. “A Novel Model for Detecting Application Layer DDoS Attacks”. In: *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS06)* (2006) (cit. on pp. 7, 11).

[89] Y. Xie, S. Tang, Y. Xiang, and J. Hu. “Resisting Web Proxy-Based HTTP Attacks by Temporal and Spatial Locality Behavior”. In: *IEEE Transactions on Parallel and Distributed Systems* 24.7 (2013), pp. 1401–1410 (cit. on pp. 1, 5, 7, 46).

[90]Y. Xie and S. Yu. “A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors”. In: *IEEE/ACM Trans. Netw.* 17.1 (2009), pp. 54–65 (cit. on pp. 7, 8, 18, 46).

[91]Y. Xie and S. Yu. “Monitoring the application-layer ddos attacks for popular websites”. In: *IEEE/ACM Transactions on Networking* (2009) (cit. on p. 6).

[92]Y. Xie and S. Yu. “Monitoring the Application-Layer DDoS Attacks for Popular Websites”. In: *IEEE/ACM Transactions on Networking* 17.1 (2009), pp. 15–25 (cit. on pp. 7, 46).

[93]J. Yan and S. Ahmad. “A low-cost attack on a Microsoft captcha”. In: *Proceedings of the 15th ACM conference on Computer and communications security - CCS 08* (2008) (cit. on p. 7).

[94]X. Yang, D. Wetherall, and T. Anderson. “TVA: A DoS-Limiting Network Architecture”. In: *IEEE/ACM Transactions on Networking* 16.6 (2008), pp. 1267–1280 (cit. on p. 8).

[95]T. Yatagai, T. Isohara, and I. Sasase. “Detection of HTTP-GET flood attack based on analysis of page access behavior”. In: *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*. IEEE. 2007, pp. 232–235 (cit. on p. 7).

[96]S. Yu, W. Zhou, W. Jia, et al. “Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient”. In: *IEEE Transactions on Parallel and Distributed Systems* 23.6 (2012), pp. 1073–1080 (cit. on pp. 1, 6).

[97]S. Zargar, J. Joshi, and D. Tipper. “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks”. In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 2046–2069 (cit. on p. 1).

[98]J. Zhang, J. Rexford, and J. Feigenbaum. “Learning-based anomaly detection in BGP updates”. In: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM. 2005, pp. 219–220 (cit. on p. 9).

[99]K. Zhang, A. Yen, X. Zhao, et al. “On detection of anomalous routing dynamics in BGP”. In: *International Conference on Research in Networking*. Springer. 2004, pp. 259–270 (cit. on p. 9).

[100]W. Zhou, W. Jia, S. Wen, Y.Xiang, and W. Zhou. “Detection and defense of application-layer DDoS attacks in backbone web traffic”. In: *Future Generation Computer Systems* 38 (2014), pp. 36–46 (cit. on pp. 1, 5).