Abusing Public Third-Party Services for EDoS Attacks

Huangxin Wang, Zhonghua Xi, Fei Li, and Songqing Chen

Department of Computer Science
George Mason University
VA 22030, USA

Abstract

Cloud computing has been widely adopted nowadays as it provides economical, elastic and scalable services to customers. The cloud resources are offered in an on-demand manner and the consumers are charged based on their resource usage, known as "pay-as-you-go". Such a cloud utility scheme opens the door to Economic Denial of Sustainability (EDoS) attacks in which the cloud consumers would suffer from financial losses. In this paper, we uncover the severity of EDoS attacks through demonstrating that EDoS attacks can be easily conducted at very low costs. In specific, we show that attackers can launch amplification attacks against the cloud consumers by abusing the free public third-party services provided by the Internet giants such as Google, Microsoft, Facebook and LinkedIn. Through studying the characteristics of 10 main public third-party services, we reveal that all of them can be abused to launch EDoS attacks and the amplification factor can reach up to 135K. To combat against the uncovered attacks, we propose several mitigation strategies for the third-party service providers as well as the cloud consumers.

1 Introduction

As cloud computing offers customers convenient access to affordable, flexible and scalable computing resources, nowadays more and more small to medium enterprises have moved their IT infrastructure to the cloud. The cloud consumers are charged under a "pay-as-you-go" scheme in which they only need to pay for the resources that they have used. With this utility charging scheme, traditional Distributed Denial of Service (DDoS) attacks have evolved to a new type of attack called Economic Denial of Sustainability (EDoS) attack [6, 30]. In EDoS attacks, attackers aim to drain the financial budget of the cloud consumers by making them consume more resources for fraudulent traffic requests and thus pay more to the cloud providers. Cost-effectiveness, which is one of the main reason for consumers to adopt cloud computing, however, will diminish under EDoS attacks.

In traditional DDoS attacks, attackers aim to make the victim servers unavailable for legitimate clients by congesting the victim servers with bogus traffic. While in EDoS attacks, attackers seek to bring economic burden to the victims and thus eventually make the victims unable to afford their services. Hence, such an attack is named Economic Denial of Sustainability (EDoS) attack [6]. Specifically, the EDoS attackers can fraudulently consume the computational or bandwidth resources of the victim servers, and thus incur high bills to the victims under the "pay-as-you-go" cloud utility model. Recent events and studies reveal that the EDoS attack is a severe threat to the cloud consumers [26, 39]. For example, GreatFire.org that hosts its services in Amazon Cloud suffered from an attack with more than 2.8 billion requests per hour at the peak in 2015. The attack makes GreatFire.org experience more than 2600 times of the traffic than its usual and pushes the bandwidth cost to as high as $\$30,000$ per day [26].

In this paper, we systematically and experimentally uncover how severe the EDoS attacks are to the cloud consumers by demonstrating that EDoS attacks can be conducted easily and at very low cost. The attack can be achieved by abusing the free public third-party services made available by a lot of IT companies, such as Google [10], Microsoft [15], Facebook [7] and LinkedIn [14]. These public services are originally intended to be used by their social networks, mail services or other web-based services to improve user experience. For example, Facebook uses its fetching services to prefetch the URLs contained in a user's post in order to preview landing pages of the URLs. Similarly, Microsoft Outlook email service prefetches the URLs contained in the emails for the purpose of previewing contents. These service APIs, however, are not well protected and thus can be abused by the attackers to fraudulently consume

the resources of their attack targets. Imagine the scenario in which an attacker would like to attack Alice who hosts her public accessible website in the cloud. The attacker can use the public fetching services to make requests to Alice's website, e.g, requesting images, videos, documents etc. As a result, it will cause large bandwidth consumption and incur a high bill to Alice.

We also demonstrate that it is more beneficial for the attackers to leverage the public third-party services to conduct amplification attacks rather than directly attack the victims. The reasons are multiple folds: (1) The attackers can hide their IPs from the victims; (2) The public third-party services have numerous IPs, thus even if the attacker only has a single IP, it can appear to have multiple IPs as seen from the victim's perspective, thus makes it difficult to block the attackers by IP addresses; (3) It has large bandwidth amplification factors such that the attackers only experience light traffic volume while the victims suffer from much larger bandwidth consumption. More details will be shown in Section 4.

The rest of the paper is organized as follows. Section 2 provides some background knowledge of amplification attacks and public third-party services. The threat model is described in Section 3. Section 4 characterizes the third-party services and demonstrates how they can be abused for EDoS attacks. An EDoS attack is conducted in Section 5 to demonstrate the power of the third-party services when they are used for attacks. We discuss several EDoS mitigation strategies in Section 6. Section 7 reviews the related work and concluding remarks are given in Section 8.

2 Background

In this section, we introduce amplification attacks and provide a brief overview of the public third-party services that can be abused for EDoS attacks.

2.1 Amplification attacks

An amplification attack is an attack in which the attackers use fewer resources to cause the victims to suffer from much higher resource stress. Smurf attacks [33], DNS amplification attacks [28] are two notorious examples of amplification attacks. As an example, in DNS amplification attacks [28], an attacker sends DNS lookup messages to a public accessible DNS server with a spoofed IP address of the victim. As a result, the DNS server floods the victim with DNS response traffic. In this amplification attack, the payload of the packets that the attacker sends to the DNS server is much smaller than the payload of the packets the victim receives from the DNS server. The EDoS attack discussed in this paper often also belongs to the family of *Amplification Attacks*, as what will be shown in Section 4. More specifically, in an EDoS attack, the attackers experience much less traffic than what the victims have suffered.

Bandwidth Amplification Factor (BAF) is a widely used metric in measuring the amplification effects [33, 28]. It is defined as below.

Definition 1 (Bandwidth Amplification Factor (BAF))**.**

$$BAF = \frac{Traffic_{victim}}{Traffic_{attacker}}, \tag{1}$$

where $Traffic_{attacker}$ is the traffic volume the attacker sends out per request, and $Traffic_{victim}$ is the traffic volume received and responsed by the victim correspondingly.

In an EDoS attack, the attackers aim to impose a financial burden on the victims. We take into account the variety of charging rates and define the economic amplification factor (EAF) as below.

Definition 2 (Economic Amplification Factor (EAF))**.**

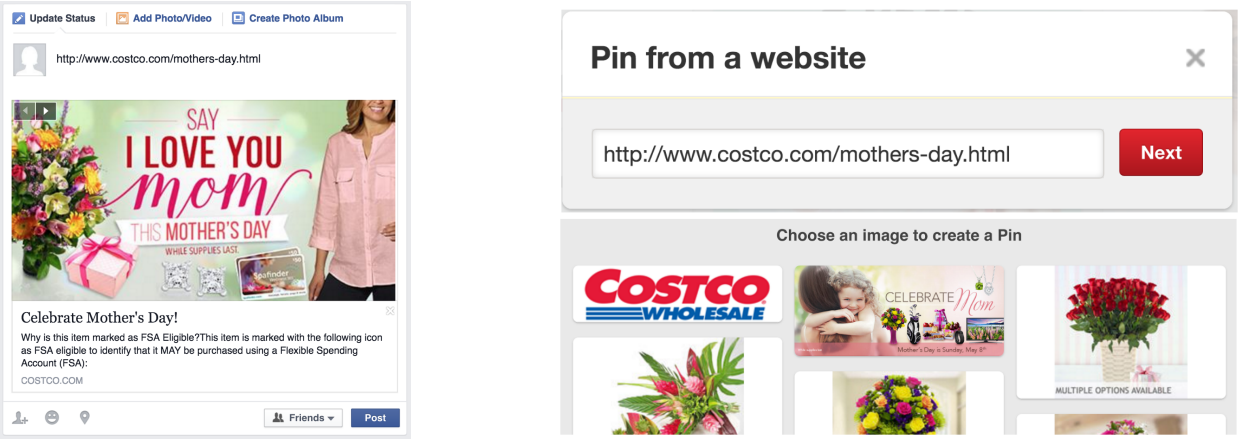
$$EAF = \frac{\lambda_{victim} Traffic_{victim}}{\lambda_{attacker} Traffic_{attacker}}, \tag{2}$$

where $\lambda_{attacker}$ and λ_{victim} are the resource charging rate for the attacker and the victim, respectively.

The resource charging rate depends on the charging scheme of the service providers. For example, if a victim is hosting her services in the Amazon Cloud, the bandwidth resource charging rate will be \$0.09/GB for outgoing traffic as shown in Section 3 which is relatively high. On the contrary, there are many low-cost options for the attackers besides clouds as their operations are rather simple (see Section 5). For example, Zare [27] provides KVM servers with 3TB transfer limit for £5/month (i.e., \$7.30/month), the bandwidth charging rate will be \$0.0024/GB which further amplifies the bandwidth amplification factor by 37.5 times.

2.2 Public third-party services

The world nowadays is filled with a rich set of Internet services including social networks (e.g., Facebook [7], Fotolog [8], LinkedIn [14], Pinterest [19], Tencent [22], Tagged [20]), mail services (e.g., Microsoft Outlook [15]), repository hosting services (e.g., Github [9]), online document services (e.g., Google Docs [10]) etc. We show some examples of the front-end functionalities in these third-party services in Figure 1. These front-end functionalities are supported by some back-end resource fetching services. In Figure 1(a), when a Facebook user is composing a post which contains URLs, Facebook URL fetching service API will be



(a) Facebook, composing a post

(b) Pinterest, "pin" images

Figure 1: Examples of the front-end UIs which call the back-end URL fetching service APIs.

called to fetch the URLs contained in the post to preview the corresponding landing pages. This function gives users a more vivid view of what is contained in a URL when they are composing a post. In Figure 1(b), the Pinterest [19] URL fetching service is designed to allow users to "pin" the images contained in a given URL. In particular, the Pinterest fetching service will fetch all the images contained in a given URL and allow the users to choose which image to "pin". This greatly improves the user experience as the user no longer needs to download the images by herself and then upload them to Pinterest.

These back-end fetching service APIs are originally designed to be used by their own products only. Unfortunately, they are not well guarded and attackers can gain access to them and abuse them to conduct attacks. For example, the attackers can use these service APIs to make requests to the victims' servers and thus to consume the resources from the victims.

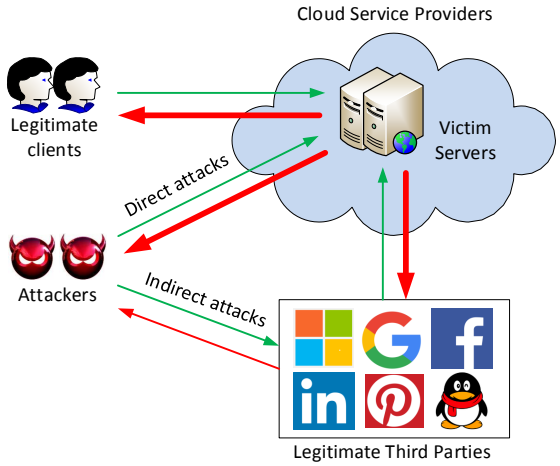


Figure 2: A Cloud EDoS attack scenario. The green and red arrows correspond to the request and the response, respectively. The line weight indicates the traffic volume.

3 Threat Model

In this section, we describe a threat model of cloud EDoS attacks in which attackers use free public third-party services to conduct attacks. An overview of the EDoS attack scenarios is shown in Figure 2. The components include cloud service providers (CSP), victim servers hosted by cloud consumers, legitimate clients, malicious clients (i.e., attackers), and legitimate third parties to be abused for conducting attacks. We walk through each component in more details in the following.

Cloud service providers (CSP). The cloud service providers offer computing and network resources that can be leased by cloud consumers. The cloud service providers will charge the cloud consumers based on the cloud utility model which will be explained in the following. Some widely used cloud service providers are Amazon EC2 Cloud [4], Google Cloud [11], Microsoft Azure [16], etc.

Cloud utility model. The cloud utility model refers to the charging scheme specified by the cloud service providers. The CSP meters the resource consumption of each customer and charges her a fee based on her exact resource consumption. This charging scheme is known as "pay-as-you-go".

Most of the cloud providers charge their customers based on the consumed server hours and the volume of outgoing network traffic (i.e., traffic flow from the servers to the public internet or to other servers in the same cloud). The incoming traffic, i.e., traffic sent to the cloud server instance, is free. Table 1 shows an example of the utility pricing of the cloud service providers including Amazon Cloud [4], Google Cloud [11] and Microsoft Azure [16] as of May 12, 2016. Note that for the same cloud provider, the charging rate varies based on server instance types and the cloud regions. The server prices in Table 1 refer to server instances with 2 vCPUs and located in the US region.

CSP	Server Price	Traffic Egress to Internet
Amazon Cloud	\$0.12/hr	\$0.09/GB
Google Cloud	\$0.10/hr	\$0.12/GB
Microsoft Azure	\$0.12/hr	\$0.087/GB

Table 1: The charging model of different cloud service providers.

Note that the price of the traffic going out to the same cloud is much cheaper than to the Internet [3, 18, 13]. For example, for Amazon Cloud, the traffic outgoing to the same cloud region is charged at \$0.01/GB, and the traffic outgoing to a different region of the same cloud is charged at \$0.02/GB [3]. Both prices are much lower than the price of traffic outgoing to the Internet as shown in Table 1.

Cloud consumers. The cloud consumers refer to the users who lease resources from cloud service providers. The cloud consumers will be billed by the resources they have consumed under the aforementioned "pay-as-you-go" cloud utility model.

In the EDoS attack model, we assume the cloud consumers host websites or any computing resources that are public accessible to all clients without requiring extra authentication. Note that many online shopping websites (e.g., Target [21], Walmart [25]), news websites (e.g., The New York Times [23]), social websites (e.g., Pinterest [19], Tumblr [24]) are open to the public without requiring logging-in for browsing the websites.

Legitimate clients. The legitimate clients are the clients who make requests of the cloud consumers' resources purely for their interests, but not intended to cause financial losses (e.g., the bandwidth cost) to the cloud consumers.

Attackers. The attackers are also called *malicious clients*. Different from legitimate clients, the purpose for an attacker to visit a cloud consumer's website is to consume the bandwidth resource of the cloud consumer and to impose a financial burden on her. The behaviors for an attacker to fraudulently consume resources from her attack target is called attacks. In particular, there are two types of attacks, namely **direct attacks** and **indirect attacks**.

For direct attacks, the attacker requests the resource directly from the victim. This is less profitable especially when the resource charging fee is expensive for the attackers. Besides, the attackers will have risks to reveal their identity to the victims. The economic inefficiency and the security risk may prohibit the attackers to conduct direct attacks. While under indirect attacks, the attacker abuses public third-party services for consuming the victim's resource. We show in Section 4.5 that under this type of attacks, the amplification factor could be extremely high, i.e., up to 135K. The legitimate third parties that can be abused for indirect attacks are introduced in the following.

Legitimate third parties. The legitimate third parties refer to the companies or organizations who provide public available services that can be abused by the attackers to launch EDoS attacks to the cloud consumers.

Based on the cloud utility model, one of the most effective forms of EDoS attacks is to fraudulently consume a victim's bandwidth resource. Therefore, the services that can be abused to launch the EDoS attacks should exhibit the bandwidth amplification characteristics, i.e., the traffic volume between the attackers and the third parties should be much less than that between the third parties and the victims. The third-party services introduced in Section 2 have such characteristics as what will be demonstrated in Section 4.

4 Characterizing Third-party Services

In this section, we conduct experiments to study the characteristics of 10 popular third-party services. We first demonstrate how attackers can make use of these legitimate third-party services to consume resources from their attack targets. Then we go ahead to study the power of these third-party services when they are used as attack weapons.

4.1 Setup

Third-party services. We choose a set of popular network services for investigation according to their popularity and types in Alexa [1]. The network services chosen for investigation cover several types, including social

networks, email services and other web-based services. These service providers we study are: Facebook [7], Fotolog [8], Github [9], Google [10], LinkedIn [14], Microsoft [15], Pinterest [19], Tagged [20], Tencent [22] and Tumblr [24].

Website hosting. In order to study the characteristics of the third-party services when they are used for attacks, we build a victim website in the Amazon Cloud with the t2.micro instance type, which has 1 v-CPU and 1GB memory. The website hosts various recourses including images with sizes from 38KB to 40MB, videos with sizes from 2MB to 171MB and documents with size from 10KB to 10MB. Our deployed website is a typical one. The types of resources it offers and the way it offers the resources are similar to various types of websites, including online-shopping websites (e.g., Walmart [25], Target [21] etc.), online-social websites (e.g, Facebook [7], Tumblr [24], Pinterest [19] etc.), news websites (e.g., The New York Times [23]), etc.

4.2 How to abuse third-party services

Many daily used functionalities of legitimate websites, such as uploading a profile image, previewing the changes of a readme file or even pasting a URL into a post composer, can be easily abused to launch EDoS attacks. In Table 2 we show such vulnerable services exist in the products of the service providers. Most of these products involve image uploading or webpage previewing. We demonstrate here that even non-expert malicious users can easily abuse these services to launch EDoS attacks with the following simple steps:

- Step 1: Login into the website (optional for some services)
- Step 2: Start to record the networking logs using browser's developer tool (e.g., Chrome DevTools [5])
- Step 3: Do the operation (e.g., paste the target URL into the post composer or email draft)
- Step 4: Extract the command of the operation from the recorded networking logs (i.e., an HTTP request with access token in cookie)
- Step 5: Replay the HTTP request got from Step 4

An example of the Pinterest's HTTP requests got from Step 4 that can be replayed to attack the victim is shown in Figure 3. Table 2 shows the services from the legitimate third parties that can be abused for EDoS attacks. In particular, it lists the related products and the using scenarios. We can see that the service APIs that can be

```
curl "https://www.pinterest.com/resource/FindPinImagesResource/get/?sou
rce_url=%2Fpin%2Ffind%2F%3Furl%3Dhttp%253A%252F%252F52.24.85.100%252Fim
gs%252F$name%253Fpin%253D$qs&data=%7B%22options%22%3A%7B%22url%22%3A%22
http%3A%2F%2F52.24.85.100%2Fimgs%2F$name%3Fpin%3D$qs%22%7D%2C%22context
%22%3A%7B%7D%2C%22module%22%3A%7B%22name%22%3A%22ImagesFeedPage%22%2C%2
2options%22%3A%7B%22url%22%3A%22http%3A%2F%2F52.24.85.100%2Fimgs%2F$
    name%3Fpin%3D$qs%22%7D%7D%2C%22render_type%22%3A1%2C%22error_strate
gy%22%3A0%7D&module_path=App%3EFooterButtons%3EDropdownButton%3EDropdow
n%3EAddPin%3EShowModalButton(module%3DAddPinURL)%23Modal(state_isVisibl
e%3Dtrue%2C+showCloseModal%3Dtrue%2C+state_showModalMask%3Dtrue%2C+stat
e_showPositionElement%3Dtrue)&_=1461556313210" -H 'Cookie:
    sessionFunnelEventLogged=1; _b="ARsn6EzhbN5JxrjhL4XpafCyEWbBEYjvwfn
Tfl+dR9d0hE6hLy87pqUVO1EcrcXqX00="; _auth=1;
    csrftoken=GLoIGA9mJjto4ecJLNktSIhcwa0SsTgC; "; extensionNag=true;
    c_dpr=1; cm_sub=none' -H 'X-Pinterest-AppState: active' -H 'X-APP-
    VERSION: 8f54b1b'  -H 'X-Requested-With: XMLHttpRequest' --
    compressed -m $to -w 'request = %{size_request} upload =
    %{size_upload} download = %{size_download} header = %{size_header}'
```

Figure 3: An example of Pinterest's HTTP request that can be replayed by attackers. The *source-url* can be filled with the target victim's resource address.

Provider	Product	Scenario
Facebook	Composer	pasting a URL in the post
Fotolog	Photo	'upload' a remote image
Github	Markdown	inserting a URL, previewing the changes
Google	Docs	inserting images by URLs
LinkedIn	Blog	adding medias by URLs
Microsoft	Outlook	pasting a URL
Pinterest	Pin	'pin' images by URLs
Tagged	Profile	'upload' a remote image
Tencent	Blog	inserting a URL
Tumblr	Link	inserting a URL, saving as a draft

Table 2: Public services from legitimate third parties with potential EDoS attack capability.

abused all have the functionality about resource fetching. Specifically, the attackers can use these service APIs to fetch resources from their target victims. When the attacker calls a service API, a response message will be sent back to the attacker. For most of the API calls, the size of the response message is small (less than 7K, see Table 4) which usually contains the result of the API operation (e.g., resource fetching success or failure).

There are two exceptions: Google and Github, both of which are proxy services (the latter one is a lazy proxy service, two requests are needed to get the actual content) which will return exact the fetched content to the attackers. To achieve high bandwidth amplification factors, the attackers can specify the request as a HEAD request such that only the header of the fetched resource will be sent back to the attackers.

Provider	User-Agent	Fetch file type	Cache Resource	IPs*	In the public cloud
Facebook	facebookexternalhit/1.1	any	✓	175	-
	visionutils/0.2	any	✓	32	-
Fotolog	fotolog	images**	✗	1	-
Github	Camo Asset Proxy 2.2.0	images**	✓	40	Amazon Cloud
Google	Feedfetcher-Google	any	✓	3	-
LinkedIn	Embedly/0.2	any	✓	6	Amazon Cloud
Microsoft	BingPreview/1.0b	any	✓	409	-
Pinterest	Pinterest/0.2	any	✓	728	Amazon Cloud
Tagged	-	any	✓	2	-
Tencent	Mozilla/5.0 AppleWebKit/537.36 Chrome/31.0.1650.57 Safari/537.36	any	✓	1	-
Tumblr	Tumblr/14.0.835.186	any	✓	2	-

Table 3: Characteristics of the fetching services. *unique number of IPs used to serve 2000 requests. **file types based on extensions.

4.3 Identities and unique IP addresses

The identities of the third-party services and their IP addresses are critical factors in measuring the attack power as well as designing mitigation mechanisms. Thus we uncover these features in the following. We conduct an experiment in which the attacker calls the service APIs to request resources (by URLs) from the victim website that we have built. Specifically, the attacker calls the service APIs of each service provider 2000 times with an average delay of 10s between requests. Note that in our experiment the attacker only has a single IP and a single account. We log each request in the victim website.

The identities and unique IPs of each third-party fetching service are summarized in Table 3. We find that most of the providers have unique User-Agent fields which can be used for whitelisting or blacklisting. We also find that some providers, especially those cloud-based, e.g., Github, LinkedIn, Pinterest, tend to use multiple IPs to fetch resources even when the requests are from a single user (with a single IP). For example, Pinterest uses 728 different IPs in serving 2000 requests. It can be inferred that the third-party services have employed load balancing mechanisms to distribute the requests to be served on different machines. It is likely that requests from different users may be assigned to the same machine. This indicates that it is difficult for the victims to employ an IP-based blocking mechanism to filter out the attackers since an attacker is not bind to a single IP and legitimate users may share the same IP with the attackers.

4.4 Capability of the fetching services

We are interested in the maximum content-length that a service provider would fetch in a single request as the amplification factor is proportional to the maximum fetched content-length.

We conduct an experiment to study the abortion criteria of each service API by calling it to fetch resources from the victim's website. For testing the timeout, we use *mod_bw* to limit the bandwidth of the victim's website to 1KB/s and let service providers fetch a large resource (5MB) from the website we built and use tcpdump to record the time when the connection started and reset.

Not surprisingly, we find that most of the service providers have a certain level of protections on their APIs that they limit either the maximum content-length or the maximum fetching time as shown in Table 4. When the request exceeds the limit enforced in the service APIs, it will be aborted.

We summarize the abortion criteria of the service APIs in the following:

- Content Length: a HEAD request is made before the GET request to check the content-length of the resource. If it is greater than the threshold, the fetching will be skipped. (Fotolog)
- Stream Length: a GET request is sent to fetch the victim's resource, the connection is reset when the length of transferred content exceeds a certain length. (Google, Linkedin, Github, Tumblr)

Provider	Attacker		Fetching Service		
	Sent (byte)	Received (byte)	Max Fetching Size (MB)	Timeout (sec)	Max Bandwidth Amplification Factor
Facebook	1901	4689	10	10	5998
Fotolog	1915	120	14	120	7726
Github	1773*	3179*,**	10	60	5482
Google	225	460**	12 - 14	10	63296
LinkedIn	2094	1072	20	45	9697
Microsoft	2987	1193	10	>3600	3555
Pinterest	1149	6691	>140	7	>134682
Tagged	1258	965	>130	45	>108298
Tencent	1409	438	>20	20	>14413
Tumblr	881	397	12 - 13	60	33873

Table 4: Sent, received size of the attacker and maximum fetching size (MFS), timeout and max amplification factors of the each fetching service. *sum of two requests. **HEAD only request.

- Metadata: some providers peek the first several kilobytes of the resource to check its file type, file size and resolution and abort the fetching if any criteria are not met (e.g., not an image). Providers that check metadata include Facebook, Github, Fotolog and Tencent.

- Timeout: if the transmission can not be finished within the time budget, connections are reset. All providers except Microsoft's BingPreview have this limit.

One interesting case is about Pinterest whose API does not have restrictions on the maximum content-length, instead, it has a relative shorter timeout which is about 7 seconds. From tcpdump we find that Pinterest also deploys its fetching service in Amazon Cloud, the bandwidth between our victim server and Pinterest's server can be as high as 167 Mbps even in a single request, 148 MB was transferred in 7 seconds which leads to the highest amplification factor we ever measured, over 134k as shown in Table 4.

4.5 Bandwidth amplification factor

As introduced in Section 2, the effect of an attack can be measured by the bandwidth amplification factor which is defined as the ratio of the traffic volume suffered by the attacker and the victim, respectively. For each request of the third-party service API, the maximum traffic volume a victim would suffer corresponds the maximum fetching size limit as shown in Table 4. In Table 4, we also show the sent size, received size for the attacker. We compute the bandwidth amplification factor (BAF) using the following formula:

$$\text{BAF} = \frac{\text{Fetching Size}}{Sent_{attacker}}$$

The calculated bandwidth amplification factor is shown in Table 4. Note, the received size was excluded since the attacker can reset the connection after sending the request.

Among all the providers, Google has the smallest request size which is only 225 bytes since login is not required, no cookies and other tracking headers are needed. It has an amplification factor up to 63296 with a 14 MB maximum fetching size. Other providers have the request size between 1KB to 4KB and amplification factor between 3555 to 134682 which are several folds higher than that of traditional amplification attacks (around 318 to 1395) [36].

4.6 Caching

Caching in the third-party services would significantly affect the bandwidth consumption of the victim websites. Hence, we check the cache capability of each service provider and the results are shown in Table 3. We find that 9 out of the 10 third-party services will cache the resources except Fotolog. In particular, with the cache mechanism, when the attacker repeatedly calls the service APIs to fetch the same resource (exactly the same URL) from the victim, the third-party service will fetch only once. However, the attackers can play a trick to append different query strings in the URL to make it appear to be different to the third-party services and thus resulting in cache misses. Based on this fact, we conduct an EDoS attacks in Section 5 and propose a redirection based EDoS defense mechanism in Section 6.1.3.

4.7 Registration cost and anonymity

We further demonstrate the registration requirements and cost in creating the user account in the third-party services. The results are shown in Table 5 from which we can see that there is no burden for the attackers to use these third-party services and they can keep fully anonymous. Thus, EDoS attacks by abusing the public services are likely to be launched in the real world.

	Register Requirements	Cost	Anonymity
Facebook	Email or phone	Free	Yes
Fotolog	Email	Free	Yes
Github	Email	Free	Yes
LinkedIn	Email	Free	Yes
Microsoft	Email or Phone	Free	Yes
Pinterest	Email	Free	Yes
Tagged	Email	Free	Yes
Tencent	Phone	Free	Yes
Tumblr	Email	Free	Yes

Table 5: Account registration requirement and cost. Google is not included as its fetching services can be called without logging in.

5 An EDoS Attack

In this section, we conduct an EDoS attack by abusing the third-party services introduced above. We analyze the attack traffic the victim suffered as well as the economic amplification factor of the attack.

5.1 Setup

To show the power of EDoS attacks by abusing the public third-party services, we conduct the following experiment. The victim website is deployed as what has been introduced in Section 4.1. It hosts multiple resources, and we assume in the experiment, the attacker only targets on requesting one of them. The targeted resource is a JPEG image with file size equals to 9719146 bytes, approximately 9.26MB which is within the maximum fetching size of all the service providers. It is hosted with the URL `http://victim/img.jpg`.

The attacker (a laptop in the lab with a single IP address) simultaneously calls each provider to fetch the targeted resource for 2000 times with a request interval of 10 seconds.

In order to differentiate the service providers in the logs of the victim's website, we append a naive unique query string to the URLs when calling each service providers' service APIs. Besides, in order to avoid hitting the cache of the service providers and thus push them to fetch the resources for each request, we append a different *requestId* for each request. In this way, the service providers will treat each URL as a different one. Above all, the requested URL looks like:

```
http://victim/img.jpg?s={provider}&id={requestId}
```

For example, the URLs the attacker asks Pinterest's service to fetch in the 2000 requests could be like:

```
http://victim/img.jpg?s=pinterest&id=1
http://victim/img.jpg?s=pinterest&id=2
...
http://victim/img.jpg?s=pinterest&id=2000
```

5.2 Traffic consumptions

The total expected network out traffic of the victim server is about 181GB (10 providers $\times$ 2000 requests $\times$ 9.26MB / request). We show the monitored network out traffic of the victim server in Figure 4 which has a total volume of 166.41GB and total elapsed time about 5.5 hours. The first thing worth noting is that none of the requests hits the caches though they are actually requesting the same resource. It can be inferred that the service providers are not capable of distinguishing whether two different URLs are requesting the same resources or not.

By analyzing the access logs on the victim's server, we notice that some providers (e.g., Tagged, Facebook) will send 2 or more requests (from different IP addresses) to the victim's server for each request from the attacker probably for performance concerns. As a result, it doubles or triples the bandwidth amplification factor. From Figure 4 we can see that at the very beginning, the network out traffic of the victim server grows dramatically up to 1086 MB/min (maximum). About 40 minutes later, Amazon Cloud starts to "protect" the victim website by limiting the network traffic to about 464 MB/min since the networking performance of micro instance type is defined to be "low". We can imagine that the traffic rate will be significantly higher in a more powerful instance with high networking performance.

Even with this simple attack mechanism in which the attacker only has a single IP address and a single user account in accessing the third-party services, most of the service providers did not take any actions to stop the abuse except LinkedIn [14] and Microsoft [15]. LinkedIn blocked all the fetch requests after finishing the 555th requests and Microsoft periodically blocks the requests for 5 minutes.

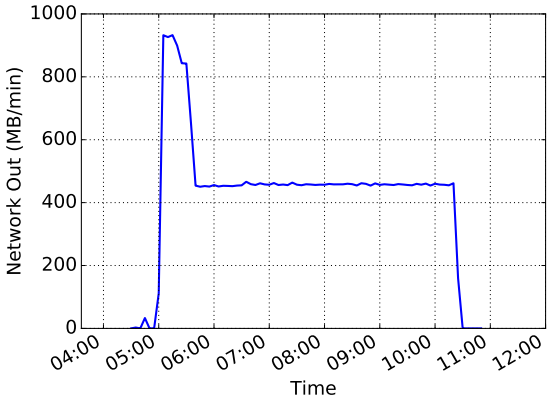


Figure 4: Average network out traffic of the victim server under requests from 10 third-party service providers with a 10-second request interval. Each sample point is the average value within a 5-minute window.

5.3 Economic amplification factor

During the entire attack, the attacker sent out less than 30 MB while the victim sent out 166 GB traffic which turns to a $11.87 bill (bandwidth alone). If the attack lasts overtime with the same traffic rate, the monthly bandwidth bill for the victim will be $1545.07. The average bandwidth amplification factor in this attack is 9617 which is much higher than traditional amplification attacks even under the traffic limit enforced by the Amazon Cloud.

As mentioned in Section 4 that LinekdIn, Pinterest and Github are hosting their resource fetching services in the Amazon Cloud. From the logs in the victim server, we find the requests were served by their servers deployed in US Virginia region. As our victim server is deployed in Amazon Cloud Oregon region, according to the cloud utility model in Section 3, the traffic outgoing from the victim server to LinekdIn, Pinterest and Github is charged at a much lower price ($0.02/GB) compared with the rest ones ($0.09/GB). This fact is verified by the total bill $11.87 we got from the attack.

If the attack is launched inside the Amazon Cloud and lasts for one month with the same traffic rate, the average bandwidth alone economic amplification factor will be 8157.32. As the actions the attackers need to perform are rather simple (i.e., sending HTTP requests), the attackers have more flexibility in choosing low-cost services rather than the cloud. For example, if the attack is launched from a lower cost host, e.g. Zare [27] with the charging rate at $0.0024/GB, the economic amplification factor will be increased to 167859.46. Note that the economic amplification factor can be further increased by carefully planning the attack, e.g. by only abusing those providers that have high economic amplification factors and low blockage rate.

6 Mitigation Strategies

In this section, we propose several ways for cloud consumers to minimize the cost when they are facing potential EDoS attacks. We also show that with the cooperation between the cloud consumers and the public third-party service providers, it will rule out the possibilities for the attackers to abuse the public services in conducting EDoS attacks.

6.1 What can the cloud consumers do?

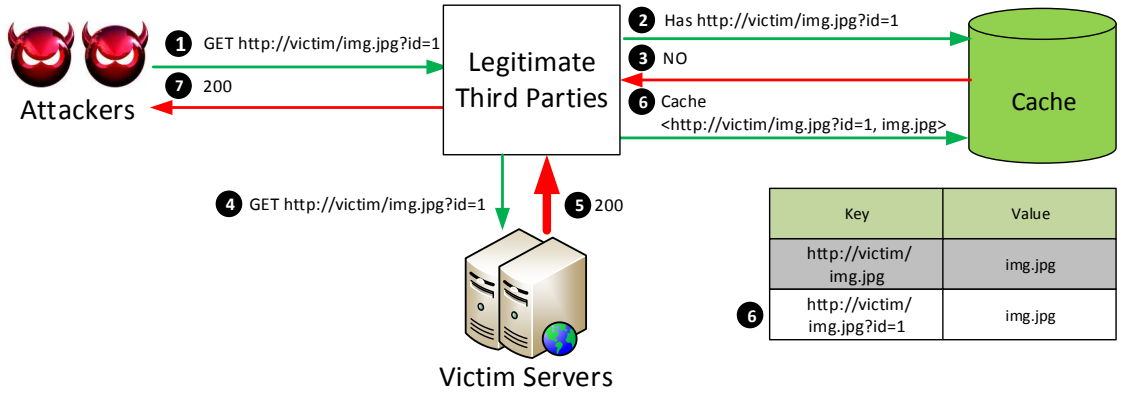
6.1.1 Filtering requests from special agents

As shown in Table 3, most of the third-party URL fetching services use unique user-agent in the request header, thus the cloud consumers can easily setup a whitelist or blacklist to allow certain URL fetching services to access while denying others. For example, an online shopping website may want to deny the request from Github's fetching agent as Github website is mainly used for hosting code. The agent-based filtering approach has the limitations of mis-blocking some legitimate users, e.g., there are very few people who use Github for blogging rather than code hosting, thus these people may have the possibility to include the resource of the shopping websites in their blogs.

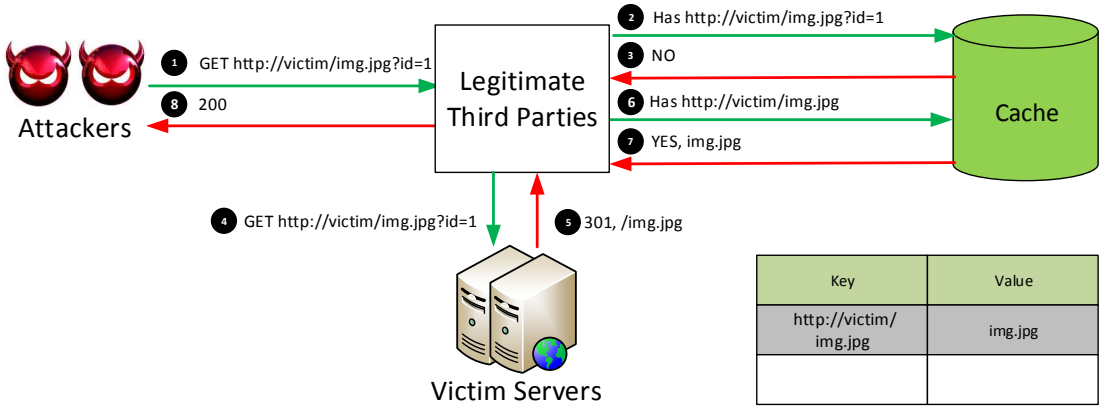
6.1.2 Serving requests with same cloud region servers

As mentioned in Section 3, the price of the traffic outgoing to the same cloud region is only $1/9$ of the price when traffic is outgoing to the Internet (leaving the cloud). If the cloud consumers host their website in multiple clouds or/and in multiple regions within the same cloud, it is beneficial for the cloud consumers to serve the requests using the servers in the same cloud region whenever possible. As a result, it can significantly reduce the cost not only when being attacked but also on a daily basis.

This mitigation scheme has been verified in our experiments in Section 5. In our experiment, our victim server is hosted in the same cloud (Amazon Cloud) but a different region with the third-party services including Github, LinkedIn and Pinterest. Thus the price of the traffic outgoing to these three third-party services is $0.02/GB, while the traffic outgoing to the rest of the third-party services is charged $0.09/GB.



(a) Current deployed model. Victim's server responses to resource requests directly.



(b) Proposed model. Victim's sever drops all the query strings on the static resource and redirects the request to its actual URL. Fetching services cache based on the actual (redirected) URLs.

Figure 5: Proposed EDoS defensive mechanism.

6.1.3 A redirection-based defense mechanism

The root cause of the effective EDoS attacks is because the attackers can generate numerous different URLs even for a single resource by appending different query strings. The attackers can call the third-party services to fetch the resource with the generated URLs. Unfortunately, the third-party service cannot distinguish whether two URLs actually point to the same resource. Therefore, even though most of the third-party services do have caching mechanisms, they will not be able to hit the cache due to the differences in the URLs and thus they will re-fetch the resources from the victim's website on every request made by the attacker.

In order to illustrate the proposed mechanism, we first depict the currently deployed model adopted by most websites and third-party services in Figure 5(a). In this figure, the attacker calls the third-party service to fetch resources (step 1), the third-party service finds it does not have a valid cache for the requested URL even it actually has one in the cache but with a slightly different URL (step 2 and 3), then the third-party service fetches the resources from the victim's website and caches the resources (step 4-6), finally a response of the fetching state is sent back to the attacker (step 7).

We propose an EDoS defense mechanism which requires corporations between the cloud consumer and the third-party services (usually have much more computing resources, e.g., Google, Facebook, Microsoft, etc.) as following.

For a request to a static resource with a query string in the URL from the fetching service, the cloud consumer redirects the request by dropping all the query strings and responses to the requester with a status code 301 (moved

permanently). This action aims to tell the third-party services that the requests generated from the same URL with different query strings actually point to the same resource. It is important for the cloud consumers to do so as the third-party services cannot tell whether two different URLs are pointing to the same resource or not.

Under this mechanism, when the attacker requests the same resource with a different query string, the fetching service sends a request to the victim server as usual but it will get a 301 response with a redirection address (the URL with query string being dropped). Then if the third-party service has a valid cache for the requested resources, it should not re-fetch the resource from the victim's website. If so, the attacker is no longer able to drive the third-party services to make numerous requests to the victim's website.

However, it turns out the single side action from the cloud consumers is not enough to make this defense mechanism work. Though all of the third-party services will follow the redirection except Tencent and most of them have caches (shown in Table 3), only Tagged [20] will recheck whether the resource of the redirected URL is contained in its cache. While for the rest of the third-party services, they will refetch the resources in the redirected link. Thus, to make this defense mechanism work, the cooperation of the third-party services to recheck the cache is needed as shown in the following.

6.2 What can the third parties do?

6.2.1 Cache redirected resources

As mentioned in the previous subsection, the redirection-based defense mechanism requires the third-party fetching services to cooperate with the cloud consumers to follow the redirection, cache the fetched resources, and recheck whether the redirected resources are in the cache in order to make the defense mechanism effective.

The proposed cooperation scheme between the cloud consumers and the third-party services is shown in Figure 5(b). Different from Figure 5(a), the victim servers will drop the query strings in the requested URL and redirect the third-party services to the URLs without query strings (step 4 and 5), the third-party services will recheck whether the redirected URL has a valid cache and fetch resources from the cache when there has a cache hit (step 6 and 7). By comparing Figure 5(b) and Figure 5(a), we see that the victim website experiences much less traffic volume under our proposed scheme, so do the third-party services.

We evaluate the redirection-based EDoS defense mechanism by turning on the redirection option on the victim's server and requesting a single resource 2000 times with different query strings using the third-party services. We find out that one service provider (Tagged [20]) has already employed the proposed cooperation features (i.e., caching, follow redirection, recheck cache) thus the average amplification factor can be reduced from 15464.11 to 0.43. This indicates that the attackers will have no incentive to abuse the third parties' services for launching EDoS attacks under our proposed defense mechanism.

6.2.2 Forward the HEAD request

In our experiment, we notice that even when an attacker requests only the header of the resource, fetching services (Google, Github) still will request the full content from the victim server which is unnecessary. HEAD requests should be forwarded to reduce the amplification factor.

6.2.3 Service limit

Note that the attacker's goal is to drain the financial budget of the victim in a long-term process rather than DDoSing the victim for a short amount of time. Thus, rate/volume limit of those public fetching services becomes an effective approach to this type of EDoS attack.

In the experiment conducted in Section 5, we observe that Microsoft has already employed a rate limiting mechanism by periodically blocking the requests for 5 minutes. We further conduct an experiment to visually illustrate how Microsoft performs the rate limit, and the result is shown in Figure 6. From this figure, we can see that service rate limit can alleviate the hurt for small cloud consumers though it can not fully prevent EDoS attacks.

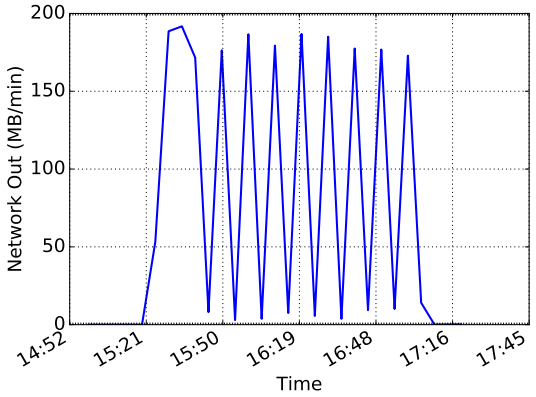


Figure 6: Average out traffic of the victim server when it is being attacked by abusing Microsoft Outlook service alone with a 3-second request interval, each sample point is the average value within a 5-minute window.

7 Related Work

In this section, we first introduce the studies that have been proposed to conduct EDoS attacks to cloud consumers. Then we review the EDoS defense mechanisms in the literature and their limitations in defending the attacks proposed in this paper. Finally, we discuss the latest protection mechanism offered by the cloud service providers.

On the EDoS attack strategies, Vlajic et al. [39] proposed an attack mechanism by sending spam emails which contain web bugs. A web-bug is an invisible HTML reference to a large image hosted in the victim's website [35]. Thus when the email receivers open the email, the web-bug code will be activated and result in automatically downloading the images. Therefore, it would raise high bandwidth consumption of the victims. However, as the advancement of the spam email detection techniques, this attack mechanism becomes less effective [37].

Different from DDoS attacks, the objective of the attackers in EDoS attack is not to make the system completely unavailable temporarily, but to drain the budget of the victims in a long term process. Thus, existing DDoS defense mechanisms [40, 31] are not effective in defending EDoS attacks.

On the EDoS defense mechanisms, Sqalli et al. [38] propose a filtering-based approach which creates a virtual firewall to filter traffic based on whitelist and blacklist. The users are added to whitelist if they pass the CAPTCHA (Graphic Turing test) verification. Masood et al. [34] design an EDoS mitigation technique based on admission and congestion control. It works by limiting the number of valid users to connect to the servers to avoid burdening. It verifies whether a user is a human or a bot by sending her a challenge to solve, and a user who solves the challenge correctly is classified as a valid user. Similar to [38, 34], Kumar et al. [32] develop a defense mechanism based on puzzle verification in which each client accessing the website will have to solve a given puzzle. A client will be granted access only if she can solve the puzzle.

Most of the EDoS defense mechanisms [38, 34, 32] rely on requesting a client to solve a challenge/puzzle in order to differentiate valid users and bots. These defense strategies would disturb the legitimate clients and decrease their satisfaction. Besides, these mechanisms will not allow the accessing of the web resources through third-party agents, which however is a popular approach based on the web 2.0 architecture [29]. For example, for a website about traveling, it might call the APIs of a website about hotel booking [29] to recommend hotels, and it might receive requests from social website agents when people mention it on social websites. Therefore, a defense mechanism that can also distinguish legitimate third-party requests is highly desired.

In regard to the EDoS defense, most of the cloud service providers offer defense mechanisms by providing monitoring services for cloud resources [12, 2, 17]. For example, Amazon provides a service called CloudWatch [2] to enable the real-time monitoring of the resource usage such as CPU utilizations, request counts and network traffic volume. It also allows cloud users to set alarms on customized metrics such that when the value exceeds the pre-defined threshold, an alarm will be sent to the users. For example, for a cloud consumer who hosts a public website in the cloud, she can set the maximum budget for using the cloud resources. When the spending exceeds the budget, she will get an alarm. However, in that case, the user can only choose either to discontinue her services and thus results in denial of services to legitimate clients; or to pay more money. Therefore, CloudWatch cannot truely save the cloud consumers from EDoS attacks.

8 Conclusion

The "pay-as-you-go" cloud utility model makes the cloud consumers vulnerable from Economic Denial of Sustainability (EDoS) attacks. In this paper, we propose a new EDoS attack mechanism based on abusing the free public third-party services. Through studying the characteristics of various pubic third-party services, we find out that the proposed attack strategy can achieve a bandwidth amplification factor of more than 134682. If the attacker is inside a cloud, then she will be able to achieve an economic amplification factor about 8157. The third-party services allow the attackers to conduct EDoS attacks at a negligible cost while keeping their identity anonymous. Our experiments demonstrate that even for a weak attacker who only has one laptop and a single IP address, she can siphon thousands of dollars per month from a cloud consumer with a small scale attack. As a complement, we propose several mitigation strategies and highlight that a cooperation between the cloud consumers and the third-party service providers is needed to effectively defend the proposed EDoS attacks.

Acknowledgements

We appreciate constructive comments from anonymous referees. This work is partially supported by an ARO grant W911NF-15-1-0262 and a NSF grant CNS-1524462.

References

[1] Alexa. http://www.alexa.com/.

[2] Amazon CloudWatch. https://aws.amazon.com/cloudwatch/.

[3] Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/.

[4] Amazon EC2 Virtual Server Hosting. https://aws.amazon.com/ec2/.

[5] Chrome DevTools. https://developer.chrome.com/devtools.

[6] Cloud Computing Security: From DDoS(Distributed Denial Of Service) to EDoS (Economic Denial of Sustainability). http://rationalsecurity.typepad.com/blog/2008/11/cloud-computing-security-from-ddos-distributed-denial-of-service-to-edos-economic-denial-of-sustaina.html.

[7] Facebook. https://www.facebook.com/.

[8] Fotolog. http://www.fotolog.com/.

[9] Github. https://github.com.

[10] Google. https://www.google.com/.

[11] Google App Engine. https://cloud.google.com/appengine/docs.

[12] Google Cloud Monitoring. https://cloud.google.com/monitoring/.

[13] Google Compute Engine Pricing. https://cloud.google.com/compute/pricing.

[14] Linkedin. https://www.linkedin.com/.

[15] Microsoft. https://www.microsoft.com/.

[16] Microsoft Azure. https://azure.microsoft.com/en-us/.

[17] Microsoft Azure Application Insights. https://azure.microsoft.com/en-us/services/application-insights/.

[18] Microsoft Azure Pricing. https://azure.microsoft.com/en-us/pricing/calculator/.

[19] Pinterest. https://www.pinterest.com/.

[20] Tagged. http://www.tagged.com/.

[21] Target. http://www.target.com/.

[22] Tencent. http://t.qq.com/.

[23] The New York Times. http://www.nytimes.com/.

[24] Tumblr. https://www.tumblr.com/.

[25] Walmart. http://www.walmart.com/.

[26] We are under attack. https://en.greatfire.org/blog/2015/mar/we-are-under-attack.

[27] Zear. https://zare.co.uk.

[28] DNS Amplification Attacks. https://www.us-cert.gov/ncas/alerts/TA13-088A, 2013.

[29] AL-HAIDARI, F., SQALLI, M., AND SALAH, K. Evaluation of the impact of edos attacks against cloud computing services. *Arabian Journal for Science and Engineering 40*, 3 (2014), 773–785.

[30] IDZIOREK, J., AND TANNIAN, M. Exploiting cloud utility models for profit and ruin. In *Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD)* (2011), pp. 33–40.

[31] JIA, Q., WANG, H., FLECK, D., LI, F., STAVROU, A., AND POWELL, W. Catch me if you can: A cloud-enabled DDoS defense. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2014), pp. 264–275.

[32] KUMAR, M. N., SUJATHA, P., KALVA, V., NAGORI, R., KATUKOJWALA, A. K., AND KUMAR, M. Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service. In *Proceedings of the 4th International Conference on Computational Intelligence and Communication Networks (CICN)* (2012), pp. 535–539.

[33] KUMAR, S. Smurf-based distributed denial of service (ddos) attack amplification in internet. In *Proceedings of the 2nd International Conference on Internet Monitoring and Protection* (2007), pp. 25–25.

[34] MASOOD, M., ANWAR, Z., RAZA, S. A., AND HUR, M. A. Edos armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments. In *Proceedings of the 16th International Multi Topic Conference (INMIC)* (2013), pp. 37–42.

[35] MAYER, J. R., AND MITCHELL, J. C. Third-party web tracking: Policy and technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (2012), pp. 413–427.

[36] ROSSOW, C. Amplification hell: Revisiting network protocols for DDoS abuse. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)* (2014).

[37] SHAO, Y., TROVATI, M., SHI, Q., ANGELOPOULOU, O., ASIMAKOPOULOU, E., AND BESSIS, N. A hybrid spam detection method based on unstructured datasets. *Soft Computing* (2015), 1–11.

[38] SQALLI, M. H., AL-HAIDARI, F., AND SALAH, K. Edos-shield - a two-steps mitigation technique against edos attacks in cloud computing. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC)* (2011), pp. 49–56.

[39] VLAJIC, N., AND SLOPEK, A. Web bugs in the cloud: Feasibility study of a new form of edos attack. In *Proceedings of the 2014 IEEE Globecom Workshop)* (2014), pp. 64–69.

[40] WANG, H., JIA, Q., FLECK, D., POWELL, W., LI, F., AND STAVROU, A. A moving target DDoS defense mechanism. *Computer Communications 46* (2014), 10 – 21.