Attribution of Economic Denial of Sustainability Attacks in Public Clouds

Mohammad Karami and Songqing Chen(✉)

Department of Computer Science, George Mason University, Fairfax, VA, USA
{mkarami,sqchen}@gmu.edu

Abstract. The cloud pricing model leaves cloud consumers vulnerable to Economic Denial of Sustainability (EDoS) attacks. In this type of attacks, an adversary first identifies web resources with high levels of cloud resource consumption, and then uses a botnet of compromised hosts to make fraudulent requests to these costly web resources. The attacker's goal is to disrupt the economical sustainability of the victim by inflicting cost through fraudulent consumption of billable cloud resources.

In this paper, we propose two different Markov-based models to profile the behavior of legitimate users in terms of their resource consumption and to detect malicious sources engaged in fraudulent use of cloud resources. Our experimental evaluation results demonstrate the effectiveness of the proposed attribution methodology for identifying malicious sources participating in EDoS attacks.

Keywords: Economic Denial of Sustainability · EDoS detection · Markov chain · Hidden semi Markov model

1 Introduction

As a new paradigm, cloud computing is reshaping the entire information technology industry. Cloud service providers enable their consumers to access shared computing resources in a flexible way without the need for upfront investment on infrastructure, platform, and software. Although the adoption of cloud computing has experienced significant growth in recent years, some concerns regarding the unique features of cloud computing environments have hindered its broader adoption. Security and privacy concerns in particular are frequently ranked as one of the top reasons why some organizations are reluctant to adopt cloud computing [5, 26, 27].

The understanding and mitigation of security and privacy risks of the public cloud computing model has been an active area of research in recent years. The research efforts, however, have been primarily focused on protecting the confidentiality and integrity of sensitive data processed in public cloud environments as well as ensuring the continuous availability of cloud services for their intended users [23]. Very little attention has been paid to security threats targeting the cost model of consumers running their services on the public cloud [11].

Services running on public clouds are vulnerable to fraudulent resource consumption attacks aiming at increasing the financial burden of the victim service.

This is enabled by exploiting the utility-based pricing model of the cloud where consumers are charged for the actual consumption of computing resources such as CPU cycles, RAM, bandwidth, and storage [14].

An adversary can conveniently rent a botnet [24] consisting of thousands of bot machines to incur artificial cost to a victim service. The target of the attack will have to pay for the cost of fraudulent resource consumption resulted from requests made by bot clients. By keeping the rate of fraudulent requests made by individual bots low to mimic the behavior of legitimate users, and intelligently focusing on requests that are most costly in terms of resource consumption, an attacker can sustain the attack over an extended period of time and maximize the effectiveness of the attack.

In practice, any device with an Internet connection is capable of launching an EDoS attack. The attacker can simply instrument the device to send HTTP GET requests to the victim service at the highest rate possible. This is basically the method used in application layer DDoS attacks where the attacker's goal is to render a targeted service unavailable to its intended users by overwhelming victim's resources. However, this will very quickly result in a significant deviation from the request rate of normal users and this artifact can be used for detecting and blocking the offending source [6, 15, 30].

In this paper, we focus on an adversarial scenario in which the attacker's goal is to increase the financial burden of the victim. This attack is also refereed to as Fraudulent Resource Consumption (FRC) attack by some researchers in the literature [11, 14]. In a recent empirical study, Wang et al. [29] show how practical EDoS attacks can be launched by abusing popular third-party services provided by companies such as Google, Facebook, etc.

In this paper, we assume that the attacker is intelligent in the sense that she makes requests that are resource-intensive resulting in higher costs for the victim. To be effective, an EDoS attack needs to be stealthy and remain undetected for an extended period of time (e.g., weeks or months). To this end, not only that malicious requests must not cause any noticeable degradation of service quality, but also the quantity of requests made by malicious sources should not be significantly different from those of legitimate users. Although high-rate DDoS attacks with the intention of overwhelming resources of a victim hosted on a public cloud can increase the resource consumption cost for the victim, in our threat model we assume that targets are properly protected against such attacks and we instead only focus on addressing low-rate and stealth EDoS attacks.

As malicious clients participating in a stealth EDoS attack make requests in a similar rate as legitimate users, this type of attacks can be challenging to detect and mitigate. In this paper, we present a method for detecting stealth EDoS attackers by directly assigning a cost to each user request in proportion to the resources consumed to serve that request.

The proposed methodology is based on statistical anomaly detection. First, we process web server logs to identify the sequence of requests made by each individual user over a predefined period of time. Next, according to the amount of resources consumed to serve each request, a relative cost value is assigned to each request. The result is a dataset consisting of a sequence of request costs for each of the legitimate users in the processed web access logs. The sequence of request costs for each user is considered as a random or stochastic process and this data is used to construct two different Markov-based models to capture the behavior of users in terms of the cost they incur to a service over time. We use sequence of request costs collected for normal users as training data to estimate the model parameters. Once the parameters are estimated, at the detection phase, the abnormality of a newly observed sequence of request costs is tested against the trained model to identify malicious sources participating in an EDoS attack.

We use real-world web access logs of about a month from an academic website to experimentally evaluate the effectiveness of the proposed method. Experimental results are presented for the two Markov-based detection methods that we propose, a simple Markov chain model, and a more complex Hidden semi-Markov Model (HsMM). The experimental results show that our proposed detection methods are very effective in differentiating normal users and malicious users participating in EDoS attacks. While most of previously proposed methods require a malicious source to make significantly more requests than legitimate users to be effective, our proposed attribution methodology can successfully detect malicious sources that try to remain undetected by making only a few resource-intensive requests.

The remainder of this paper is structured as follows. We begin with a discussion on the exploitation of the cloud pricing model that motivates this work. Related work is discussed in Sect. 3. Section 4 presents a brief background on Markov chains and HsMM as well as our proposed Markov-based methods for identifying malicious sources participating in EDoS attacks. The details of experiments designed to validate the proposed methodology and their results are presented in Sect. 5. Finally, discussion and conclusion remarks are presented in Sects. 6 and 7, respectively.

2 Exploitation of the Utility-Based Pricing Model

The cloud computing technology provides many attractive benefits such as avoiding the need for upfront spendings on computing infrastructure, improved manageability, security, and elasticity to businesses of various sizes. While the flexibility of the “pay-as-you-go” pricing model adopted by cloud service providers can be beneficial to cloud consumers, it leaves them vulnerable to financial risks imposed by EDoS attacks [11, 14].

To launch an EDoS attack, all an attacker needs to do is to simply send seemingly legitimate requests to a victim service to make it consume cloud resources for which the victim will have to pay for the cost. If the attacker is able to enforce significant fraudulent resource consumption over an extended period of time, the economical sustainability of the victim service could be threatened.

In an EDoS attack, the attack target can be a website or web applications hosted on a third party public cloud and we assume that attack targets predominantly serve public content accessible to all Internet users.

Unlike Distributed Denial of Service (DDoS) attacks, an EDoS attack is not meant to cause availability issues or noticeable degradation of service quality for the users of a target service. To be effective, an EDoS attack needs to be stealthy and remain undetected over an extended period of time (e.g., weeks or months). To remain undetected, a wise attacker will want to keep the rate of fraudulent requests low to blend them into the noise of legitimate requests, while trying to focus on requests resulting in high levels of cloud resource consumption to achieve the objective of the attack.

As documented in recent studies, DDoS-for-hire services can be readily located and rented on underground black markets [4, 16, 17]. These abusive services are often supported by botnets consisting of tens of thousands of compromised hosts and offer both network layer and application layer attacks [28]. With the availability of DDoS-for-hire services, an attacker does not need to be capable of building a supporting attack infrastructure.

The potential impact of an EDoS attack can be best quantified by examining a hypothetical attack on a service hosted on a real public cloud service provider. In the sequel, we consider a hypothetical attack on a victim service hosted on Amazon's Elastic Compute Cloud (EC2) platform. Although cloud consumers are billed for various cloud resources including computing, network, and storage resources, for simplicity, this work only focuses on data transferred from the cloud environment to the Internet to serve received requests. Table 1 shows the cost of outgoing data transfer for Amazon's EC2 platform [3].

Table 1. Amazon EC2 outgoing data transfer pricing as of February 2016.

Traffic volume	Cost
First 1 GB /month	$0.00 Per GB
Up to 10 TB/month	$0.09 Per GB
Next 40 TB/month	$0.085 Per GB
Next 100 TB/month	$0.07 Per GB
Next 350 TB/month	$0.05 Per GB

According to the HTTP Archive [2], which regularly measures the Alexa top 10,000 websites [1], the average page size was 2,225 KB for the homepage of the top 10,000 websites visited in January 2016. However, many websites host a number of much larger web resources such as videos or large compressed files that an attacker can focus on to maximize the cost of resource consumption for a victim operating on a public cloud. For the purpose of our hypothetical EDoS attack, we assume the average size of web resources requested by malicious bots participating in the attack to be 100 MB.

At the rate of only 100 requests per month which is too low to raise any red flags, a single bot would consume about 10 GB of outgoing bandwidth and the monthly bill will increase by 90¢. Sending requests with the same characteristics as the single bot scenario from 1000 bots will approximately cost the victim $900 per month. The inflicted cost grows linearly by increasing the request rate, requesting larger files, or employing more malicious bots. For instance, by locating and requesting files that are 1 GB in size, the fraudulent resource consumption cost of the previous EDoS attack scenario will escalate to about $9000 per month. As seen from the hypothetical attack scenarios, the resource consumption cost accumulated over time can impose an important financial burden to public cloud consumers, especially small businesses.

As individual bots show no trace of excessive request rates, most of existing detection schemes that look for a large number of requests in a short period of time [15, 22] will not succeed at detecting the described hypothetical attack.

It is worth noting that leasing a botnet to carry out an EDoS attack will be a cost factor that an attacker would need to take into consideration. However, due to the fact that only a very small fraction of resources available to a compromised host are actually required to make a few requests at a very low rate, the cost of accessing a botnet can be significantly reduced for an attacker by renting nondedicated botnets shared with other cybercriminals using the bots for various purposes. According to Huang et al. [10], using the pay-per-install marketplace, an attacker can gain access to 1000 compromised machines for as low as $10.

3 Related Work

So far there are only a few studies in the literature directly concerning the issue of EDoS attacks.

Khor and Nakao [18] propose a mitigation mechanism based on cryptographic puzzles to dissuade clients from submitting fraudulent requests. The basic idea of their proposed scheme called self-verifying Proof of Work (sPoW) is to require clients to present a proof of work before a protected service commits its resources to serve clients' requests. When a client first requests a resource, it receives a "crypto-puzzle" from sPoW that mediates all communications between clients and the protected service. The puzzle contains encrypted information necessary to reach the intended service such as the IP address and port number as well as a partial encryption key with *k* bits concealed. The client will have to spend its resources to discover the encryption key by brute forcing the *k* concealed bits so that it can decrypt the information necessary to contact the requested service.

However, sPoW or any other solution based on the "crypto-puzzle" approach [21] are more relevant when malicious sources are sending requests at a high rate to a target service. In an intelligent and stealth EDoS attack, malicious clients can afford to solve the puzzles to submit only a few well-crafted, resource intensive requests and succeed at adding financial burden to a victim service protected by sPoW.

Sqalli et al. propose a mitigation scheme called EDoS-Shield to address the issue of EDoS attacks in cloud environments [25]. The main idea of EDoS-Shield is to detect whether an incoming request is initiated by a legitimate user or by an automated source. EDoS-Shield depends on CAPTCHA tests to verify the source of requests. The proposed architecture consists of virtual firewalls (VF) and verifier nodes (V-Nodes) that are deployed as virtual machines in the cloud. The V-Nodes are responsible for verification of request sources, and VF nodes are implemented to decide if incoming packets should be forwarded or dropped based on the verification results received from the V-nodes. One weakness of the EDoS-Shield mitigation scheme has to do with the cost of additional cloud resources required for deploying the verifier nodes and the virtual firewalls. But, more importantly, this approach requires all users to be verified and research studies suggest that CAPTCHA tests could be annoying for some users and even a certain portion of legitimate users may not be able to solve them [7]. In addition, some existing CAPTCHA tests have been shown to be vulnerable to automated attacks [8], and recently inexpensive CAPTCHA solving services backed by crowd sourced human labor can be used to effectively defeat the protection purpose of CAPTCHA tests [20].

In [12] the authors use a number of statistical self-similarity metrics including Zipf's law, and Spearman's Footrule distance to detect the occurrence of FRC attacks. The proposed detection mechanism only looks at the aggregate pattern of user requests and does not deal with identification of individual malicious sources participating in an attack. In contrast, our proposed method is concerned with identification of malicious sources exhibiting a similar behavior as legitimate users in terms of request rates, but focusing on resource-intensive requests to maximize the cost for the victim service.

Idziorek and Tannian propose a method that attempts to model the behavior of individual users based on the number of requests per session generated by each user over a fixed period of time [11]. A pause of 900 or more seconds between consecutive requests from the same user is used as the criterion to group user requests into web sessions. The premise is that malicious users generating sessions with a random number of requests would be sufficiently different from the profile of normal users, so that an entropy-based detection method could be used to identify malicious sources. This method is based on the assumption of malicious users making more requests/web sessions than legitimate users. However, as mentioned earlier, an intelligent attacker does not necessarily need to make malicious sources to send more requests than legitimate users to succeed. By focusing on web resources that are expensive in terms of resource consumption, malicious sources with similar request rates as legitimate users can be still effective.

In [13], the authors propose a methodology for identifying malicious sources trying to inflate the utility bill of a victim by making fraudulent requests. The proposed methodology combines four different usage metrics including the number of sessions, the number of requests, and the average number of requests per session. For the last usage metric, the overall request frequency distribution of documents hosted on a website is computed, and the requests made by individual users are compared against this distribution. To evaluate a user, a probability score is calculated for each of the four metrics and then an overall average probability is computed. The more deviation observed from the normal usage, the higher would be the probability score and the more likely the user would be a malicious client. Again, this model is heavily influenced by the quantity of requests made by individual users, and it will not be effective for detecting malicious users making a small number of high cost requests. As we will show in Sect. 5, our proposed method is able to detect both malicious sources making an anomalous number of random requests, as well as more subtle malicious sources with a request rate similar to that of legitimate users but focusing on requests that are more costly for the victim.

4 The Proposed Markov-Based Models for Detecting Sources Participating in an EDoS Attack

In this study, our goal is to build an anomaly detection system to identify malicious sources participating in EDoS attacks. In this section we introduce our proposed detection methodology and describe our formulation of detecting malicious sources participating in an EDoS attack using two different Markov-based models.

Most web requests are for HTML documents that are meant to be rendered, and displayed by a user browser. These requests are typically followed shortly by several subsequent HTTP GET requests to fetch objects such as images, scripts, and CSS files embedded in the main requested document. The requests can also be for downloading objects such as binary files over HTTP. Web servers can be configured to log the details of all user requests including the IP address of the requesting host, the requested document, the type of request (GET, POST, etc.), and the size of data transferred to serve the request. Although all the HTTP request types cause resource consumption on the server side, to simplify our experimentations, we only focus on HTTP GET requests in our work.

Proportional to the amount of data transferred to serve a request, a relative cost value can be calculated and assigned to each request. Based on the data size of various requests, one can decide on a small number of buckets to represent different cost values to be associated with user requests. We will see an example of this in Sect. 5 where we use cost values from 1 to 5 for requests in our dataset.

Using collected web server logs, requests made by each individual user during a specific period of time can be identified and mapped to request cost values. The result would be a sequence of request cost values for each user. We assume that individual users (both legitimate and malicious) can be uniquely identified by their IP addresses. Using browser fingerprinting techniques [9] can be a potential solution for cases where some users can not be reliably identified by their IP addresses.

The sequence of request costs from individual users during a specific period of time can be considered as a discrete-time stochastic process and a Markov-based model can be used to describe the behavior of users in terms of the cost they incur to a service over time. A much simpler approach to distinguish between legitimate and malicious users would be to calculate the sum of request costs per user over a predefined period of time and apply a threshold value to identify users exceeding the threshold as malicious. However, as we will show in Sect. 5, such a naive approach will result in high false positive rates where legitimate users are incorrectly identified as malicious.

We use requests made by legitimate users to estimate the parameters of the Markov model and then use the trained model to compute the likelihood of new request cost sequences generated by users. The request cost sequences generated by malicious users would be different from legitimate users and this will result in much smaller likelihood values than those of legitimate users. As we will show in Sect. 5, using the right threshold likelihood value, legitimate users and malicious users can be effectively distinguished.

In this paper, we propose and evaluate the detection performance for two different Markov-based models. The first one is a simple Markov chain model in which the observed request costs are considered as the states of the Markov chain. We also evaluate a HsMM which has more complexity and computational cost but can outperform the simple Markov chain model for detecting low rate attacks focusing on high cost requests. In our HsMM, the request cost values are the observable outputs and the hidden states represent different levels of resource consumption by users. The models that we propose and evaluate are both discrete-time and the discrete points in time correspond to user requests as recorded in the web access logs. In the subsequent subsections, we give a brief background on the theory of Markov chains and HsMM and briefly discuss the process of learning model parameters using training data.

4.1 Markov Chain Model

A Markov chain models the state of a system with a random variable taking states from a finite state space as the time passes. Given the current state of a Markov chain denoted as *st*, the state of the chain at time $t + 1$ will only depend on s_t. In a stationary Markov chain, the state transition probabilities are assumed to be constant and independent of time. Consider a Markov chain model with M states denoted as $S = \{s_1, s_2, ..., s_M\}$. A Markov chain model can then be specified by its parameters as $\lambda = (\{\pi_m\}, \{a_{mn}\})$ where:

- $\pi_m \equiv \Pr[s_1 = m]$ is the initial state probability distribution. s_t denotes the state taken by the model at time t and $m \in S$. The sum of initial state probabilities adds up to 1 $(\Sigma_m \pi_m = 1)$.
- $a_{mn} \equiv \Pr[s_t = n | s_{t-1} = m]$ is the state transition probability for $m, n \in S$, satisfying $\Sigma_n a_{mn} = 1$.

Given the model parameters, the probability of a particular sequence of states $s_1, s_2, ..., s_T$ to be taken by the model is computed as follows:

$$P(s_1, s_2, \ldots s_T) = \pi s_1 \sum_{t=2}^{T} a_{s_{t-1} s_t}$$

The initial state probability distribution and the state transition probability matrix can be readily learned from historical observations of the system states. These two model parameters can be learned using the formulas below [19]:

$$a_{mn} = \frac{N_{mn}}{N_m}$$

$$\pi_m = \frac{N_m}{N}$$

where:

- N_{mn} is the number of observed direct transitions from state m to state n.
- N_m is the number of observations where the Markov chain is in state m.
- N is the total number of observations.

In our context, we process web access logs to compute the sequence of request costs for normal users and then use this data to learn the parameters of a Markov chain model representing the resource consumption behavior of normal users. At the detection phase, newly observed sequences of request costs are analyzed to compute the likelihood of those sequences being supported by the trained Markov chain model. The resource consumption behavior of users participating in EDoS attacks would be different from that of normal users and the requests from these users are therefore expected to receive low likelihood of support when analyzed by the trained model.

4.2 Hidden Semi-Markov Model

A hidden Markov model (HMM) is a Markov model in which the states of the system being modeled are not directly observable (hidden). HsMM extends the traditional HMM by allowing states to have variable durations [31]. The duration of a state represents the number of observations made while in that state. Consider a HsMM with M states denoted as $S = \{s_1, s_2, ..., s_M\}$. A HsMM can be specified by its parameters as $\lambda = (\{\pi_m\}, \{a_{mn}\}, \{b_m(k)\}, \{p_m(d)\})$ where:

- $\pi_m \equiv \Pr[s_1 = m]$ is the initial state probability distribution. s_t denotes the state taken by the model at time t and $m \in S$. The sum of initial state probabilities adds up to 1 $(\Sigma_m \pi_m = 1)$.
- $a_{mn} \equiv \Pr[s_t = n | s_{t-1} = m]$ is the state transition probability for $m, n \in S$, satisfying $\Sigma_n a_{mn} = 1$.
- $b_m(k) \equiv \Pr[o_t = k | s_t = m]$, for $m \in S$, $k \in \{1, ..., K\}$ is the state output distribution. The observable output at t is denoted by o_t and k is the index into the observable output set with cardinality K. The output distribution satisfies $\Sigma_k b_m(k) = 1$.
- $p_m(d) \equiv \Pr[\tau_t = d | s_t = m]$ is the state residual time distribution, for $m \in S$, $d \in 1, ..., D$. D represents the maximum interval between any consecutive state transitions and the residual time distribution satisfies $\Sigma_d\, p_m(d) = 1$.

Then, if at time t, the pair process (s_t, τ_t) takes on the value (m, d), where $d >= 1$, the semi-markov chain will remain in state m until time $t + d - 1$ and will transit to the next state at time $t + d$. The states themselves are not directly observable. The observables are a sequence of observations $O = (o_1, ..., o_T)$. The notation o_a^b represents the observation sequence from time a to time b and conditional independence of observed outputs is assumed so that $b_m(o_a^b) = \Pi_{t=a}^b b_m(o_t)$ The model parameters are initially estimated and are then updated as new observations o_t are collected. This process is known as parameter reestimation and it can be done by following the forward and backward algorithm proposed by Yu and Kobayashi [32]. The forward and backward variables are defined as follows:

$$\alpha_t(m, d) \equiv \Pr\left[o_1^t, (s_t, \tau_t) = (m, d) | \lambda\right]$$
$$\beta_t(m, d) \equiv \Pr\left[o_{t+1}^T, (s_t, \tau_t) = (m, d) | \lambda\right]$$

which can be recursively computed by forward and backward algorithms. Next, the three following joint probabilities are defined that can be expressed and computed in terms of the model parameters and the forward and backward variables defined above. These probabilities are used to readily derive the reestimation formulas to update the model parameters after collecting new observation sequences.

$$\begin{aligned} \varsigma_t(m,n) &\equiv \Pr[o_1^T, s_{t-1}=m, s_t=n|\lambda] \\ \eta_t(m,n) &\equiv \Pr[o_1^T, s_{t-1}\neq m, s_t=m, \tau_t=d|\lambda] \\ \gamma_t(m) &\equiv \Pr[o_1^T, s_t=m, |\lambda] \end{aligned}$$

Now, using the joint probabilities defined above, the model parameters can be reestimated by the following formulas:

$$\begin{aligned} \hat{\pi}_m &= \gamma_1(m) / \sum_{m=1}^{M} \gamma_1(m) \\ \hat{a}_{mn} &= \sum_{t=1}^{T} \varsigma_t(m,n) / \sum_{t=1}^{T} \sum_{n=1}^{M} \varsigma_t(m,n) \\ \hat{b}_m(k) &= \sum_{t:o_t=k} \gamma_t(m) / \sum_{k} \sum_{t:o_t=k} \gamma_t(m) \\ \hat{p}_m(d) &= \sum_{t=1}^{T} \eta_t(m,d) / \sum_{t=1}^{T} \sum_{d=1}^{D} \eta_t(m,d) \end{aligned}$$

The model parameters are reestimated for each observation and after processing all observation sequences, the trained model can be used to compute the likelihood of a new observation sequence by the following formula:

$$\begin{aligned} \Pr[o_1^T|\lambda] &= \sum_m \sum_d \Pr[o_1^T, (s_T, \tau_T) = (m,d)|\lambda] \\ &= \sum_m \sum_d \alpha_T(m,d) \end{aligned}$$

For our HsMM, the request cost values are the observable outputs and the hidden states represent different levels of resource consumption by users. In our implemented model, we use 5 hidden states where the model is always initialized in the first state. Also, in our model a transition can only happen from a lower state to a higher state.

Similar to the Markov chain model, we use requests made by legitimate users to estimate the parameters of the HsMM and then use the trained model to compute the likelihood of new request cost sequences generated by users. The request cost sequences generated by malicious users would be different from legitimate users and this will result in much lower likelihood values than those of legitimate users. As we will show in the next section, using the right threshold likelihood value, legitimate users and malicious users can be effectively distinguished.

5 Experimental Evaluation

We conduct experiments to evaluate the effectiveness of the proposed method for detecting malicious sources engaged in fraudulent use of cloud resources. This section provides a description of our experiments and presents the obtained results.

5.1 Dataset Description

Our experiments are based on request logs from a university department's public web server collected over 32 days from Nov 8, 2015 to Dec 9, 2015. We use the rules below to filter out requests that are irrelevant for our purpose:

- Requests that are not HTTP GET.
- HTTP GET requests with a response code other than 200 (OK).
- Requests with a user agent string indicating access from a non-user entity (e.g., Googlebot, wget, etc.).
- Request sources making requests using 10 or more different user agent strings. This is to remove aggregate request sources such as NAT boxes or web proxies making requests on behalf of their clients. About 90% of all request sources in the dataset only use a single user agent string.
- The access logs are split into two 16-day periods. We only include requests from users making at least 3 requests in one of the 16-day periods.

A request for an HTML document and the subsequent requests for fetching objects embedded in the same HTML document are combined and treated as a single request. Table 2 presents a summary of the normal dataset used for training and testing the proposed methodology.

Table 2. Summary of the normal experimental dataset.

Metric	Train dataset	Normal test dataset
Number of days	16	16
Total number of unique users	4,933	5,252
Total number of requests	36,466	36,474
Avg number of requests per user	7.39	6.94

To generate the normal training dataset, requests in the first half of the logs are grouped based on the request source, and then, proportional to the amount of data transferred to serve the requests, they are mapped to relative cost values. The same process is applied to the requests in the second period of the logs to generate the test dataset representing users with normal resource usage behavior. Based on our observation of the user requests in the dataset, we choose to use the values from 1 to 5 to represent the relative cost of user requests. Thus, the final dataset is a sequence of request costs ranging from 1 to 5 in value for each user. Table 3 summarizes the mapping from the request size to relative cost values and the distribution of requests in terms of their cost values in our dataset.

Although in our experiments we use an observation window of 16 days to profile the behavior of normal users and the same observation period is used for detection of malicious users, the proposed methodology is only sensitive to the resource usage pattern of users and is not restricted to a specific observation period.

Table 3. Mapping of request sizes to relative cost values.

Request size	Relative cost	Percentage of requests
< 500 KB	1	86.7
$\geq$ 500 KB and < 5 MB	2	11.4
$\geq$ 5 MB and < 50 MB	3	1.9
$\geq$ 50 MB and < 500 MB	4	0.1
$\geq$ 500 MB	5	5.4e−03

5.2 Attack Scenarios

To conduct an EDoS attack, an attacker needs to specify the behavior of individual bots by defining the request rate and the requested resources in term of their resource consumption cost. By varying these two parameters, various attack strategies that an attacker is likely to adopt can be constructed and the effectiveness of the proposed detection methodology can be evaluated for those attack strategies. We first consider attack strategies where the attacker focuses on making requests that result in high levels of resource consumption. For these attack strategies we assume that the attacker has a prior knowledge about the rate of requests made by legitimate users, and uses this knowledge to avoid suspicions by making requests with similar rates as legitimate users.

In sequel, we briefly describe a number of various attack scenarios that we use to generate synthetic malicious request sequences to evaluate the performance of the proposed attribution methodology. These attack scenarios are ordered in a decreasing order of attack effectiveness from the attacker's perspective. For the attack scenarios listed below the number of requests made by individual malicious sources is normally distributed with parameters $\mu = 7$ and $\sigma = 2$ which means the number of requests made by malicious users are not significantly different from those of legitimate users. The lengths of malicious request sequences drawn for this normal distribution is shared by all of the attack scenarios.

- **Scenario 1 (S1)**: All malicious requests have a cost of 5.
- **Scenario 2 (S2)**: All malicious requests have a cost of 4 or 5.
- **Scenario 3 (S3)**: The request cost is 5 for 75% of malicious requests. The cost for the remaining 25% of requests is uniformly distributed between 1 and 4.
- **Scenario 4 (S4)**: 75% of malicious requests have a cost of 4 or 5. The cost for the remaining 25% of requests is uniformly distributed between 1 and 3.
- **Scenario 5 (S5)**: The request cost is 5 for 50% of malicious requests. The cost for the remaining 50% of requests is uniformly distributed between 1 and 4.
- **Scenario 6 (S6)**: 50% of malicious requests have a cost of 4 or 5. The cost for the remaining 50% of requests is uniformly distributed between 1 and 3.

5.3 Experimental Results

For each attack scenario, we generate a dataset of malicious requests according to the description of that attack scenario. Each test dataset consists of generated malicious request sequences, combined with the request sequences from the normal test dataset. The normal test dataset is the same for all attack scenarios. Also, in our experiments, each test dataset contains the same number of normal and malicious sources (5,252). False Positive Rate (FPR), and False Negative Rate (FNR) are the metrics used for performance evaluation of the proposed attribution methodology under various attack scenarios. These metrics are briefly described in the following:

– **FPR**: The percentage of request sequences generated by legitimate users classified as malicious. Keeping the FPR under a low threshold is very important. Otherwise, legitimate users will be denied access to the protected service.
– **FNR**: The percentage of malicious request sequences generated by sources participating in an EDoS attack scenario not detected by the proposed method. Unlike an Intrusion Detection System (IDS) where it is very crucial not to miss any intrusions, because a single missed intrusion can result in system compromise, in our context, missed malicious sequences would only cause some billable fraudulent resource consumption.

For each attack scenario, the trained model is used for computing the log likelihood for all request sequences in the test dataset of that attack scenario. In general, the request sequences from legitimate users which are similar to the data used for training the model are expected to receive higher log likelihood values. On the other hand, malicious request sequences representing a resource consumption behavior dissimilar to that of legitimate users are expected to be assigned lower log likelihood values. Once the log likelihoods are computed for all request sequences, the detection performance can be evaluated for different threshold values.

Table 4 shows experimental detection results for the attack scenarios based on the strategy of focusing on high cost requests using the simple Markov chain model. Table 5 shows the results for the same attack scenarios using the HsMM. The results are presented for several different threshold values to demonstrate the trade-off between higher false positive rates and lower false negative rates and vice versa. For each threshold value, the resulting FPR and the FNR for each of the six attack scenarios are presented. As shown, both models can achieve low FPRs for reasonable FNRs. The HsMM however consistently outperforms the simple Markov chain model for the same FPR value.

For instance for a FPR of 0.67%, the FNR is 0.00% for the first attack scenario when the HsMM is used. This means that for a small FPR, all sources generating malicious requests according to the description of the first attack scenario (S1) are successfully detected. In comparison, the simple Markov chain Model produces a FNR of 7.24% for the attack scenario and the same FPR. In our experiments, the last attack scenario (S6) is the most challenging to detect as it is more similar to requests from legitimate users. But even for this attack scenario, for a FPR of 0.51%, still close to 70% of malicious request sequences are successfully detected using the HsMM.

Table 4. Experimental results for the Markov chain model for the attack strategy of focusing on high cost requests.

Threshold	FPR (%)	Attack scenario FNR (%)					
		S1	S2	S3	S4	S5	S6
−20	2.32	0.00	2.55	0.97	5.79	5.14	13.58
−25	1.35	0.00	6.32	2.30	10.68	8.87	22.51
−30	0.91	0.00	7.81	3.77	14.58	13.40	31.51
−35	0.67	7.24	16.85	11.23	25.08	22.37	43.01
−40	0.46	7.24	21.63	13.08	31.70	28.64	52.36
−45	0.29	16.28	28.66	22.43	42.25	37.93	63.06

Table 5. Experimental results for the HsMM for the attack strategy of focusing on high cost requests.

Threshold	FPR (%)	Attack scenario FNR (%)					
		S1	S2	S3	S4	S5	S6
−15	2.32	0.00	0.00	0.44	1.64	2.76	7.52
−20	1.26	0.00	2.57	2.02	5.62	6.84	15.69
−25	0.67	0.00	2.57	3.56	8.95	10.68	23.67
−30	0.51	7.24	9.04	10.95	16.70	20.13	33.80
−35	0.36	7.24	11.18	15.19	23.38	28.87	45.35
−40	0.21	16.28	18.55	23.59	33.45	38.42	57.60

It should be noted that the undetected malicious sequences are usually comprised of fewer requests compared to the successfully detected malicious sequences. For instance, when applying a threshold value of –25 for the attack scenario S5 using the HsMM, about 10% of malicious users are not identified as malicious. The average number of requests for these undetected malicious users is 3.98 versus 7.23 for the detected malicious users. This implies that the undetected malicious request sequences are less effective in terms of fraudulent resource consumption and the more effective malicious request sequences that are more aggressive in nature run higher risk of detection.

In our experiments, the FPRs are resulted by legitimate users in the test dataset generating significantly more requests than the other legitimate users. For instance, when applying a threshold value of −25 for the HsMM, 35 legitimate users out of 5252 users are incorrectly identified as malicious. On average, each of these 35 users generates 72 requests. In comparison, the overall average number of requests for all legitimate users is only 7. Longer request sequences result in lower log likelihood values and therefore legitimate users generating a large number of requests contribute to some undesirable false positives. Sometimes a given website may have legitimate users that use the website in unusual ways. If unusual request patterns are expected from specific users, false positives can be avoided by ignoring requests from these known users. In our experiments, the legitimate users with an abnormally large number of requests are incorrectly identified as malicious.

Comparison with the naive detection approach: as mentioned previously, a naive detection approach based solely on the cumulative sum of request costs will suffer high positive rates. As a concrete example, to achieve a FNR of 0.00% for the S1 attack scenario, the naive approach will need to identify all users with a cumulative sum of request costs of 15 or more as malicious. This however will result in a very high FPR of 9.96% making this approach inapplicable in practice. In contrast, as shown in Tables 4 and 5, using the proposed attribution methodology, for the S1 attack scenario all malicious users can be successfully detected with FPRs less than 1%.

Table 6. Experimental results for the Markov chain model for the attack strategy of focusing on high number of requests.

Threshold	FPR (%)	Number of requests per source FNR (%)					
		50	60	70	80	90	100
−20	2.32	3.77	0.72	0.00	0.00	0.00	0.00
−25	1.35	17.02	3.41	0.32	0.00	0.00	0.00
−30	0.91	42.19	14.38	3.24	0.32	0.02	0.00
−35	0.67	67.82	34.22	12.20	2.65	0.30	0.02
−40	0.46	86.27	59.12	29.86	10.45	2.21	0.49
−45	0.29	95.00	79.38	51.71	25.46	8.85	2.70

Table 7. Experimental results for the HsMM for the attack strategy of focusing on high number of requests.

Threshold	FPR (%)	Number of requests per source FNR (%)					
		50	60	70	80	90	100
−15	2.32	6.51	2.11	0.76	0.15	0.02	0.00
−20	1.26	23.42	10.66	3.27	0.74	0.23	0.06
−25	0.67	49.58	20.91	9.69	4.23	1.52	0.74
−30	0.51	82.77	51.79	25.32	10.62	4.09	2.04
−35	0.36	94.94	79.59	54.72	24.68	12.32	6.78
−40	0.21	98.19	92.90	79.99	53.98	27.88	13.23

As evidenced by the obtained experimental results, attacks based on the strategy of focusing on requests with high resource consumption costs can not go undetected. The alternative for an attacker would be to attempt making requests with a similar distribution of request costs as legitimate users, but in larger quantities to increase the amount of fraudulently consumed resources.

Tables 6 and 7 show experimental detection results for the attack strategy where the attacker focuses on making larger numbers of requests that have the same distribution of request costs as legitimate users. FNRs are reported for various number of requests per source and the same threshold and FPRs as the previous experiments. As expected, malicious users are more likely to be detected when making higher number of requests. For instance, when malicious clients are making 70 requests, for a FPR of 0.67%, more than 90% of malicious sources are successfully detected by the HsMM. For the experiments involving long sequences of malicious requests, the performance of the two models are comparable and none of them consistently outperforms the other model for the same FPR value. However, when the number of requests per source is 80 or more, the simple Markov chain model seems to produce lower FNRs.

It should be noted that from the standpoint of an EDoS attacker, regular requests not causing high levels of resource consumption are not very helpful and this attack strategy only makes sense when malicious sources are able to make a significant number of regular requests and manage to remain undetected.

6 Discussion

The proposed EDoS attribution methodology directly considers the cost of user requests as the metric to model the behavior of individual users. This makes it very challenging for malicious users involved in an EDoS attack to be effective in terms of fraudulent consumption of billable cloud resources and at the same time managing to remain undetected. As demonstrated experimentally, malicious users exhibiting anomalous resource consumption behavior can be quickly identified and prevented after making only a small number of suspicious requests. An attacker can attempt to optimize the requests made by individual participating bots by learning and mimicking the request pattern of top legitimate users in terms of higher usage footprint. However, it is unlikely for an attacker to be able to access historical data on requests of legitimate users or intercept communications to collect such data to optimize the behavior of participating bots. Even assuming the lack of such restrictions, applying such optimized request patterns can still significantly limit the effectiveness of the participating bots. An attacker can try to compensate for the limited utility of individual bots by employing a much larger botnet. However, larger botnets could be very difficult to locate, rent and operate and may not be practical in practice.

The proposed EDoS attribution methodology only relies on resource usage footprint of users for detecting malicious sources. For future work, we plan to incorporate other aspects of user behavior to further improve detection of malicious sources participating in EDoS attacks. For instance, the popularity of requested documents can be computed for each request cost bucket, and this can be considered when computing the likelihood of observed request sequences. In general, attackers are not expected to know the distribution of document popularity on victim websites. Focusing on requests involving documents with the highest resource consumption can result in deviation from the normal document popularity distribution and this additional metric can help to improve the detection performance.

7 Conclusion

The consumers of public cloud services are charged for computing resources that they use. This pricing model exposes the cloud consumers to EDoS attacks where the adversary seeks to increase the financial burden of the victim service by making fraudulent requests that result in high consumption of billable resources.

We have presented a Markov-based anomaly detection scheme to profile the behavior of legitimate users in terms of their resource consumption. To detect users participating in an EDoS attack, the likelihood of request sequences generated by individual clients during a specific period of time is computed by the trained model. Users with likelihood values smaller than a threshold are identified as malicious. The effectiveness of the proposed attribution methodology for identifying malicious sources engaged in fraudulent use of cloud resources has been demonstrated using experimental evaluations for various attack scenarios. While most of previously proposed methods are only effective when malicious sources make significantly more requests than legitimate users, our proposed method is able to detect both malicious sources making an anomalous number of random requests, as well as more subtle malicious sources with a request rate similar to that of legitimate users but focusing on requests that are more costly for the victim.

Acknowledgements. We would like to thank anonymous reviewers for their comments. This work was supported in part by an ARO grant W911NF-15-1-0262 and a NSF grant CNS-1524462.

References

1. Alexa. http://www.alexa.com/
2. Http archive. http://httparchive.org/interesting.php?a=All&l=Jan%2015%202016
3. Amazon ec2 pricing (2016). https://aws.amazon.com/ec2/pricing/
4. Alomari, E., Manickam, S., Gupta, B., Karuppayah, S., Alfaris, R.: Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art. *arXiv preprint* arXiv:1208.0403 (2012)
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
6. Beitollahi, H., Deconinck, G.: Tackling application-layer DDoS attacks. Procedia Comput. Sci. **10**, 432–441 (2012)
7. Bursztein, E., Bethard, S., Fabry, C., Mitchell, J.C., Jurafsky, D.: How good are humans at solving captchas? A large scale evaluation. In: 2010 IEEE Symposium on Security and Privacy (SP), pp. 399–413. IEEE (2010)
8. Bursztein, E., Martin, M., Mitchell, J.: Text-based captcha strengths and weaknesses. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 125–138. ACM (2011)
9. Eckersley, P.: How unique is your web browser? In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 1–18. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14527-8_1

10. Thomas, K., Huang, D., Wang, D., Bursztein, E., Grier, C., Holt, T.J., Kruegel, C., McCoy, D., Savage, S., Vigna, G.: Framing dependencies introduced by underground commoditization. In: Proceedings of the 14th Annual Workshop on the Economics of Information Security (2015), Netherlands, June 22–23 (2015)
11. Idziorek, J., Tannian, M.: Exploiting cloud utility models for profit and ruin. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 33–40. IEEE (2011)
12. Idziorek, J., Tannian, M., Jacobson, D.: Detecting fraudulent use of cloud resources. In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp. 61–72. ACM (2011)
13. Idziorek, J., Tannian, M., Jacobson, D.: Attribution of fraudulent resource consumption in the cloud. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 99–106. IEEE (2012)
14. Idziorek, J., Tannian, M.F., Jacobson, D.: The insecurity of cloud utility models. IT Prof. **2**, 22–27 (2013)
15. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In: Proceedings of the 11th International Conference on World Wide Web, pp. 293–304. ACM (2002)
16. Karami, M., McCoy, D.: Understanding the emerging threat of DDoS-as-a-service. In: Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (2013)
17. Karami, M., Park, Y., McCoy, D.: Stress testing the booters: understanding and undermining the business of DDoS services. In: Proceedings of the World Wide Web Conference (WWW) (2016)
18. Khor, S.H., Nakao, A.: sPoW: on-demand cloud-based eDDOS mitigation mechanism. In: HotDep (Fifth Workshop on Hot Topics in System Dependability) (2009)
19. Mitchell, T.M.: Machine Learning, vol. 45, p. 995. McGraw-Hill, Burr Ridge (1997)
20. Motoyama, M., Levchenko, K., Kanich, C., McCoy, D., Voelker, G.M., Savage, S.: Re: Captchas-understanding captcha-solving services in an economic context. In: USENIX Security Symposium, vol. 10, p. 3 (2010)
21. Naresh Kumar, M., Sujatha, P., Kalva, V., Nagori, R., Katukojwala, K., Kumar, M.: Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service. In: 2012 Fourth International Conference on Computational Intelligence and Communication Networks (CICN), pp. 535–539. IEEE (2012)
22. Oikonomou, G., Mirkovic, J.: Modeling human behavior for defense against flash-crowd attacks. In: IEEE International Conference on Communications, ICC 2009, pp. 1–6. IEEE (2009)
23. Ryan, M.D.: Cloud computing security: the scientific challenge, and a survey of solutions. J. Syst. Softw. **86**(9), 2263–2268 (2013)
24. Sood, A.K., Enbody, R.J.: Crimeware-as-a-service—a survey of commoditized crimeware in the underground market. Int. J. Crit. Infrastruct. Prot. **6**(1), 28–38 (2013)
25. Sqalli, M.H., Al-Haidari, F., Salah, K.: Edos-shield-a two-steps mitigation technique against edos attacks in cloud computing. In: 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC), pp. 49–56. IEEE (2011)
26. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011)
27. Takabi, H., Joshi, J.B., Ahn, G.-J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. **6**, 24–31 (2010)

28. Thing, V.L., Sloman, M., Dulay, N.: A Survey of Bots Used for Distributed Denial of Service Attacks. In: Venter, H., Eloff, M., Labuschagne, L., Eloff, J., Solms, R. (eds.) SEC 2007. IFIP, vol. 232, pp. 229–240. Springer, Boston, MA (2007). doi:10.1007/978-0-387-72367-9_20
29. Wang, H., Xi, Z., Li, F., Chen, S.: Abusing public third-party services for EDoS attacks. In: 10th USENIX Workshop on Offensive Technologies (WOOT 2016) (2016)
30. Wen, S., Jia, W., Zhou, W., Zhou, W., Xu, C.: Cald: surviving various application-layer DDoS attacks that mimic flash crowd. In: 2010 4th International Conference on Network and System Security (NSS), pp. 247–254. IEEE (2010)
31. Yu, S.-Z.: Hidden semi-Markov models. Artif. Intell. **174**(2), 215–243 (2010)
32. Yu, S.-Z., Kobayashi, H.: An efficient forward-backward algorithm for an explicit-duration hidden Markov model. IEEE Sig. Process. Lett. **10**(1), 11–14 (2003)