

DDoS Detection Systems for Cloud Data Storage

Ting Cao, Jianzhou Mao, Tathagata Bhattacharya, Xiaopu Peng

Wei-Shinn Ku, and Xiao Qin

Department of Computer Science and Software Engineering

Auburn University, Auburn, AL.

{tzc0028, jzm0122, tzb0063, xzp0007, weishinn, xqin}@auburn.edu

Abstract—Distributed Denial-of-Service (DDoS) attack is one of the most common and effective attack types aiming to deny or limit services. To mitigate the problem of DDoS attacks to online cloud services, anomaly detection methods of user access behaviors are widely used to defend against attackers. We pay particular attention to anomaly detection methods after reviewing recent development efforts of application-layer DDoS detection techniques for cloud storage systems. Inspired by a dynamic analysis of access behavior changes in active users, we propose a DDoS anomaly detection model to discover DDoS attack sources by diagnosing users' similarities. The overarching goal of our solution is to pinpoint DDoS by monitoring the similarity of active users around existing users at a low cost. This goal is achieved by our proposed model embracing the following key steps. First, a sample user set is originated. Then, the active users' requests are tracked to gauge similarity measures between each active user and sample users. Finally, if the deviation of similarity exceeds the prescribed thresholds, detected users will be flagged as anomalous ones.

Index Terms—cloud, application layer DDoS attack, anomaly detection, data security, user behavior, similarity, sample selection

I. INTRODUCTION

Distributed Denial-of-Service (DDoS) attacks are one of the most severe threats existing in the Internet accompanied by a growing number of cloud services [1]. DDoS attacks can be implemented in a short period of time via a large group of virtual machines. An attack generally causes an enormous number of requests making crowded internet traffic, thereby inhibiting legitimate users from accessing services [2].

The DDoS detection techniques can be classified in three categories, namely, signature-based, anomaly-based and hybrid schemes [3]. We begin this study by illustrating the concepts of DDoS threats and detection techniques in cloud storage systems. After that, we survey a diversity of anomaly-based detection models. In particular, we propose a novel anomaly-based DDoS detection system for application layer powered by user-similarity comparisons.

On demanding requirement of application DDoS detection models is to deliver accurate and quick performance amid the diagnosis of cloud storage systems. Existing anomaly DDoS detection techniques customized for cloud data storage systems are focused on monitoring online requests at the application layer. It is urgent and prudent to detect DDoS attacks among online requests because DDoS attacks make storage servers unavailable while infecting remote machines through zombie code.

An effective way of preventing attacks from application layer DDoS attack is to compare active users' current behaviors with historical behaviors recorded in a given time period. With the assistance of such behavioral comparisons, a DDoS detector is slated to determine whether or not cloud storage servers are in risks. Another key element that improves the DDoS detection performance is overhead control. Existing anomaly-based DDoS detection techniques that apply user behavioral comparisons lack extra cost analysis [4][5]. It is evident [6] that 10% of data are accessed by 90% of users, and an DDoS attack tends to issue requests with random objects [7]. We propose a novel DDoS detection model for application layer to boost the security of online cloud storage systems by deploying a user similarity measure module.

We devise the DDoS detection model that seamlessly integrates two distinct modules - a sample user generator and a similarity comparison module. The first module elects a group of high fitness users from all the legal users to build a sample user set, which is periodically updated by our system. The second module is in charge of calculating the similarities among active users and sample users. We treat a user as an abnormal user if the user's similarity measures are dramatically changed with a high percentage.

Two reasons motivate us to advocate for the similarity-based DDoS detection strategy. First, online cloud servers are accessed and shared by a large-scale user pool. Users' historical access records offer excellent samples to discover outlier users such as bots. Second, when bots attempt to simulate human users, it is a challenge to find attack sources with a low overhead. If a DDoS detector fails in swiftly pinpointing attackers, any detection delay may pose potential security threats to the entire system and users. Our DDoS detection model ensures that active users' access behaviors pass through a thorough yet lightweight analysis.

The overarching goal of the similarity comparison module is to glean users' current and historical requests records, followed by calculating similarity measures of active users and the sample ones. More specifically, the similarity module illustrates the users' behavioral changes with respect to user requests. This module works in full capacity to dynamically track and monitor the active users' similarity metrics to sense any abnormal behaviors.

The rest of this paper is organized as follows. Section II provides an overview of DDoS attack models, detection techniques in cloud storage system and articulate various application layer DDoS detection techniques in cloud. Section III

outlines our proposed similarity-based DDoS detection model. We conclude this paper in Section IV.

II. DDoS ATTACKS AND DETECTION METHODS

A. DDoS Attacks models

Distributed Denial-of-Service (DDoS) attack - an advanced form of DoS attack - is a coordinated attack against one or multiple targets through a group of computers [8]. DoS attacks are categorized into the network layer attacks, application layer attacks, data flooding attacks, and protocol-feature-based attacks [9]. All the types of DoS attacks aim to exhaust system resources at various layers in computing systems. For example, a network layer attack attempts to crash routers through buffer overrun errors in the password checking routine.

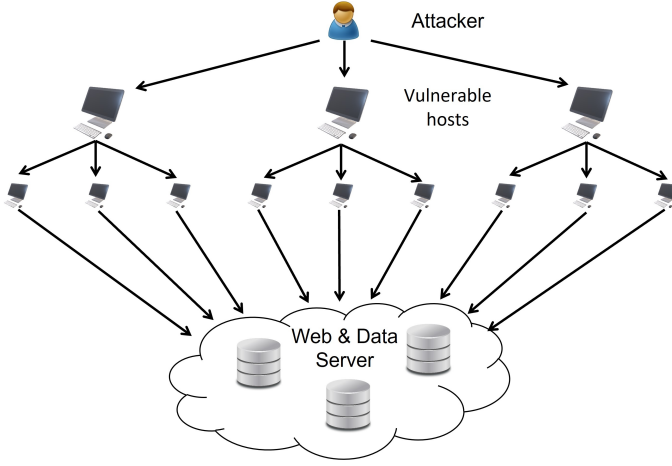


Fig. 1: The high-level DDoS attack procedures

Fig. 1 illustrates a general DDoS attack model. Initially, a attacker scans and explores vulnerability in remote machines. Infected machines are referred to as zombies, which automatically and repeatedly infect more agents as new zombies and send attack packages to target victims. Finally, the attack prevents hosts from performing regular activities and addressing genuine received requests. The attack purposely makes hosts unavailable by flooding the resources of target systems with frequent requests [10].

- *Volume-based attacks.* This kind of attack aims to exhaust the bandwidth of a victim's site. When it comes to applications that involve interactions among multiple users, infection by other users is often necessary to warrant applications' desired functionality. The ability of an attacker to infect such applications is fundamental because these applications, containing shared data, are inherently constructed to be accessed by multiple users including collaborative users [11].
- *Protocol-based attacks.* This type of attack attempts to exhaust a server's versatile resources. For instance, a ping of death attack delivers an oversized ICMP datagram (encapsulated in IP packets) to a victim's computing node. The Ping command fully utilizes an ICMP echo request and echo reply messages, which are commonly applied to check if a remote host is healthy and alive. In

this case, a ping of death attack may cause the remote system to hang, reboot or even crash [12].

- *Application-layer-based attacks.* Because of the improved robustness and bandwidth of servers and networks developed in recent years, attackers are no longer inclined to throttle network bandwidth of victims' servers. Rather, the attackers are focusing on exhausting server resources like CPU cycles, database cycles, main memory or socket connections [13].

Not surprisingly, DDoS is a serious threat to cloud storage systems. Understanding attack strategies is the first line of defense against DDoS threats. In the context of cloud storage systems, DDoS attacks share the following two distinctive features [14].

- *Strategy 1.* Flooding attacks mainly rely on botnets to attack target hosts or networks, forming a many-to-one attack modality to expand attack scope and to worsen the harm of attacks in a cascading manner. Each attack has a focal point where certain services become unavailable in a target network.
- *Strategy 2.* Given an open shared-resource platform lacking source IP address authentication or authentication capability, attackers make use of packet source IP spoofing to strike victims. Because regular traffic at a monitoring point ought to respond to destination and destination-to-source addresses, source IP addresses cannot receive a valid reply from the destination IP address.

We depict the main stream DDoS attacks in Fig. 2. In *application-bug level attacks*, attackers exploit system vulnerabilities and weaknesses to render cloud resources unavailable for users. Among the common attack vectors are protocol vulnerability, system weakness, outdated patches and misconfigurations. For example, vulnerabilities in protocols used by target applications can be explored by hackers, who aim to crash applications by delivering specially crafted packets to overload the applications. [15]

Infrastructure attacks is one of the challenging attacks to detect in the cloud and it gained greater attention with the research community [16]. the attackers only need the IP address of the target without the need to exploit any vulnerability. DDoS flooding attack can be carried out in two different forms, namely a direct attack and a reflector attack.

B. DDoS Detection Models in Cloud

To address the two strategies of DDoS threats in cloud storage system, anomaly-based (see Section II-C) and signature-based models were proposed to facilitate DDoS attack detection. The anomaly-based detection models utilize statistical methods for analyzing behaviors of data packets to sense any anomalies [17]. On the contrary, signature-based models intend to define a set of signatures or predefined knowledge from ground up, followed by identifying a candidate pattern as an intruder [18]. Apart from anomaly- or signature-based approach alone, a host of hybrid techniques were devised by mixing the two fundamental methods [19].

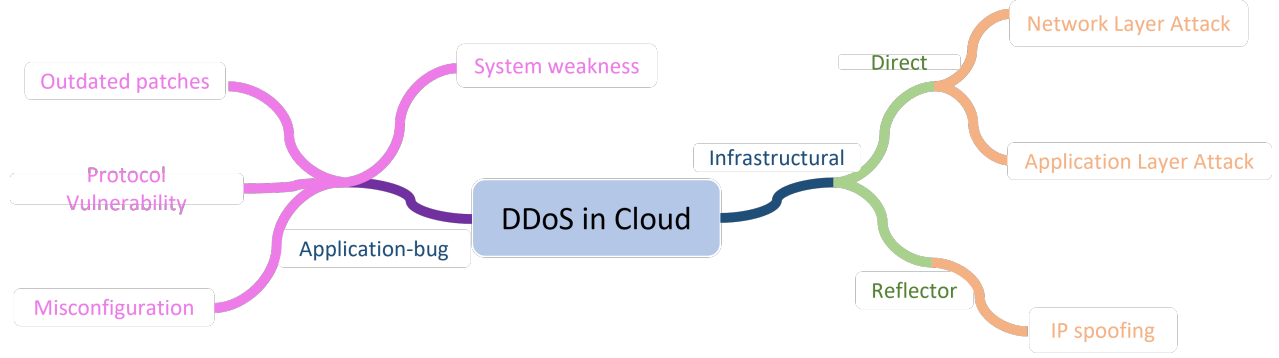


Fig. 2: Main stream DDoS attacks in cloud computing platforms.

Existing signature-based techniques receive a wide adoption in detecting network-layer DDoS attacks. For example, *Short* is one of the most commonly deployed signature-based detection techniques on multiple platforms [20]. Enlightened by the traditional signature-based approaches, Lee *et al.* implemented a data mining framework where inductively learned models is applied to build supervised classifiers [21]. This novel solution enables detection systems to discover anomalies and know intrusions by the virtue of mining consistent and useful patterns.

With regard to deployment positions, the signature- and anomaly-based detection mechanisms can be classified into application-level and network-level mechanisms. In the past decades, a majority of DDoS attacks were concentrated on network bandwidths because this resource type can be easily exhausted. With the advancement in robustness of network services, DDoS attacks against network bandwidth recently gave way to new attack modalities that exhaust server resources like CPU and memory at the application layer.

C. Application Layer DDoS Detection

Table I summarizes an array of DDoS Detection techniques for cloud storage systems. The prior studies were mainly focused on anomaly detection techniques that cope with infrastructural DDoS attacks. Therefore, we propose a novel anomaly detection solution in Section III after a thorough review of the existing anomaly detection schemes. From a technical point of view, DDoS attacks carried out at the application layer share the following distinctive characteristics [31].

- *Legitimate Requests.* An attack request mimic regular requests, making it arduous to discriminate attack requests from legitimate ones. Since the application layer DDoS attacks proceed through legitimate HTTP packets, most of network-level filters and firewalls are unable to sense such anomaly requests from application attacks.
- *Limited Resources.* The bottleneck in application-layer DDoS attacks lies in server resources. With the innumerable number of requests, the Internet host can be exhausted sooner or later.
- *Targeted Attacks.* Application layer DDoS attack may occur on the various targets such as CPU, memory

or socket connections. An attack can only attempt to exhaust one resource without affecting the other types of resources. Nevertheless, the entire system becomes unavailable under such an attack that deteriorates a single resource.

A raft of DDoS detection models are aiming to protect cloud storage systems from imminent threats of application layer DDoS attacks. Recall that attack requests and legitimate requests are alike, it is futile if not impossible to detect such attacks by simply examining individual requests. More often than not, the existing solutions are generally reliant on an analysis of requests stream tracking or requests template matching. For example, Ranjan *et al.* piloted one of the earliest studies focusing on the application layer, where an attack detection mechanism inspects characteristics of HTTP sessions by using statistical and rate-limiting [32]. Xie *et al.* proposed a popularity-aware techniques to detect DDoS for popular website through an access stream analysis [33]. On the flip side, SOAP-based DDoS repeatedly delivers SOAP requests to the URL, aiming to misuse the flexibility of security specifications. Rahaman *et al.* devised a detection model to specify stringent requirements, which should be fulfilled by incoming SOAP requests [34].

An increasing number of application-layer DDoS detection models are anchored on modern machine learning algorithms. For instance, the K-means clustering algorithm is adopted to classify incoming requests [35]. Meng *et al.* built a learning model powered by discrete-time Markov chains to compare users' current behaviors with normal history behaviors [4]. Umarani and Sharmila explored a Naive Bayes and K-nearest neighbor based classifier to identify if a request stream is legitimate through an access matrix from HTTP traces. Om *et al.* [36] integrated k-Means, Naïve Bayes and K-nearest neighbour to construct an anomaly detection approach, where a feature selection model removes irrelevant attributes to implement a clustering method.

The previous techniques were forged to maximize the prediction and detection accuracy of application-layer DDoS attacks. It is evident that detection accuracy and extra resource cost are somehow opposite aspects. Therefore, a handful of techniques were focused on both detection accuracy and overhead control. For example, Sreeram *et al.* assessed the feature metrics to diagnose request stream behaviors and to create

Reference	Detection technique	DDoS attack type	Brief description
Virupakshar <i>et al.</i> [22]	Anomaly	Infrastructural	Detect anomalous by applying Naive Bayes and Deep Neural Network machine learning algorithms
Kown <i>et al.</i> [23]	Anomaly	Not stated	Detect anomalous by measuring difference of the cosine similarity
Alqahtani <i>et al.</i> [24]	Anomaly	Infrastructural	Monitor DDoS attacks by using distance algorithm to measure the deviation between the flow rates of requests at different time intervals
Chen <i>et al.</i> [25]	Anomaly	Application-bug	Use k-means clustering algorithm to fast classify data into different groups based on similarities and detect threats with dynamic thresholds
Wang <i>et al.</i> [26]	Anomaly	Infrastructural	Detect threats with a graphical-based anomaly detection module and virtualize the network to separate the enterprise's network traffic from the main network
Meng <i>et al.</i> [4]	Anomaly	Infrastructural	Compare users' current behaviors with normal history behaviors using discrete-time Markov chains
Bakshi <i>et al.</i> [27]	Signature	Infrastructural	Measure the traffic in-bound and out-bound by installing SNORT on virtual switch
Karnwal <i>et al.</i> [28]	Signature	Application-bug	Detect DDoS by filtering requested message at different stages to match the request client IP with previously stored suspicious IP
Gupta <i>et al.</i> [29]	Hybrid	Infrastructural	A virtual machine profile describes the attack patterns needs to be looked over on specific traffic
Modi <i>et al.</i> [30]	Hybrid	Not stated	Combine Bayesian Classifier and Snort based network intrusion detection system

TABLE I: The existing DDoS detection techniques

a bio-inspired-anomaly-based model to cutback training and testing cost [37]. Jiang *et al.* developed a local measurement method accompanied by a remote counterpart to estimate the status of web servers, thereby achieving good trade-offs between the monitoring quality and overhead [38]. Doshi *et al.* restricted computational overhead by using a limited feature set in the variety of machine learning models [39].

Even though an array of existing techniques attempt to curtail detection overhead, high detection accuracy is normally obtained at the price of extra overhead. In this paper, we propose a similarity-analysis-based detection model catering for application-layer DDoS attacks. The overarching goal of our new model is to strike good balance between detection overhead and high accuracy.

III. A SIMILARITY-BASED DDoS DETECTION SYSTEM

In Section III-A, we start the research road map by presenting a basic idea for the development of a similarity-based DDoS detection system. Section III-B depicts a system architecture where user behaviors are gleaned to furnish similarity comparisons in cloud data storage. Then, we elaborate the key concepts and a model in Section III-C. Finally, in Section III-D we outline the high-level algorithm of the similarity-based DDoS detection system catering to optimize DDoS detection performance at the application layer.

A. Basic Idea

In this study, we pay heed to DDoS attack detection at the application layer centered around cloud storage systems. Since

any security solution customized for cloud data storage is obliged to impose a non-negligible performance overhead, we demonstrate that light-weighted similarity comparison techniques are capable of enhancing the performance of a DDoS detection system deployed in modern cloud storage.

The basic idea of our system solution is to analyze users' behaviors through gauging similarity values orchestrated by a user sampling algorithm, which makes similarity decisions in a way that abnormal users are detected for isolation purpose. One of the vital factors benefiting the probability of successful intrusions is the number of access points [40]. As such, we advocate for gauging the users' abnormal behaviors by comparing with a colossal group of normal users. User are treated as out of ordinary folks because of the following two factors:

- First, there is a dramatic change in similarity values between a given user and normal users compared to the given user's history.
- Second, the number of changes in similarity values between a given user and normal users exceeds a specified threshold.

In addition to similarity comparisons, we forge a user sampling module to periodically originate a relatively small sample user set to conserve computing resources. Importantly, the group of sample users are leveraged to optimize the accuracy of anomalous and to curtail the overhead of similarity comparisons. In case the given user matches the above two metrics, our proposed similarity-based detection will mark the

user as an abnormal user.

B. System Architecture

Fig. 3 depicts the system architecture of our proposed similarity-based DDoS detection system, where a similarity comparison manager collaborates with a user sampling module in clouds. The entire system is slated to detect abnormal users from active users' regular requests. Given online user request lists sent to a server, the detection module takes similarity measures by comparing current requests against sampled users. Similarity results will be compared with a history record to identify potential anomalies.

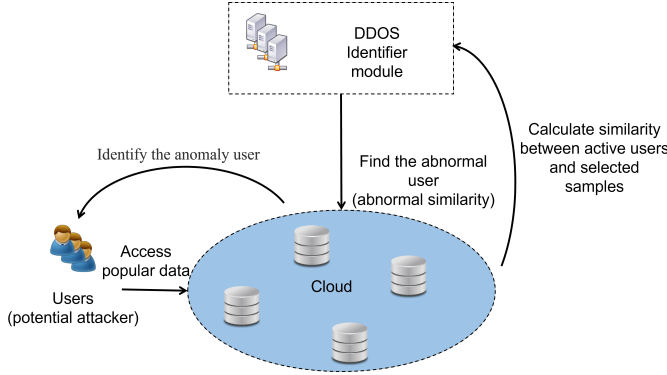


Fig. 3: The architecture of similarity-based DDoS detection model running in clouds. abnormal users are identified through the similarity comparison.

The overarching goal of the system is to alleviate DDoS risks at the application layer by detecting anomalous requests from all the online users requests. To achieve this goal, a management module is engaged to gauge users' similarity measurements, which are maintained as critical moving parts in the metadata of users. Anomalous behaviors are likely to be taken out from users with a growing number of similarity comparisons. It is arguably true that the management module in this architecture is adroit at governing anomaly detection, where similarity comparisons are configured in accordance to tracking users' similarity changes. With the similarity manager in place, outlier users are assigned a red flag to be handled by the security module.

C. Concepts and Key Steps

Fig 4 illustrates the main steps undertaken in the DDoS detection system. In a given time slot, the system checks request lists of online users. Next, the system calculates the similarities between the online users and sample users. The similarity comparison is performed in full swing after the similarity calculation is completed. We diagnose any anomalous users through the help of two thresholds: (1) the similarity value change with regard to a single sample user and (2) the amount of similarity changes with respect to multiple sample users (see Eq. 7). The system defines the current online user as an abnormal user when the metrics exceeds these two thresholds.

Popular recommendation algorithms such as the *user-based collaborative filtering* algorithm (UBCF) [41] offers a practical

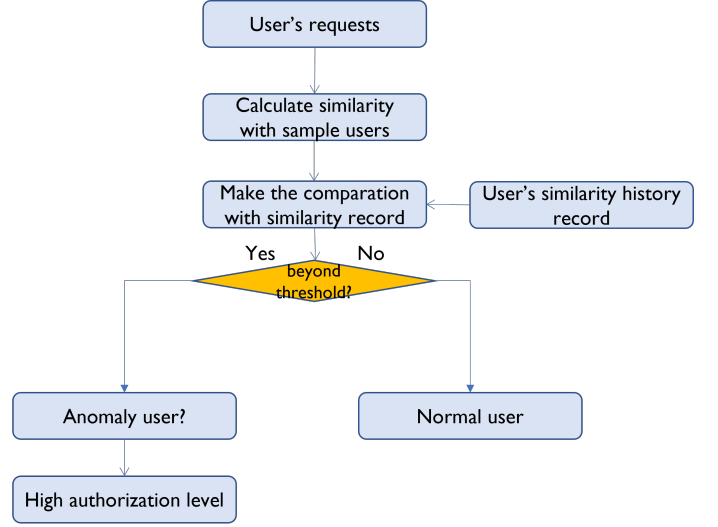


Fig. 4: The main steps to check if the current user is abnormal.

cosine similarity calculation method. Initially, the recommendation module completes a similarity comparison between each user and all the valid users in the system. In this study, we adopt the similarity comparison scheme inspired from UBCF. The DDoS detection system carries out four key steps to consolidate the multiple similarity values into an anomaly detection procedure, where each online user is assessed by the similarity manager amid the routine access process. Given an online user u_x , we denote its requests as list $L_x = \{o_{x_0}, o_{x_1}, \dots, o_{x_i}\}$. Thus, we express the access of data item o_i from user u_x as a step function

$$r_{xi} = \begin{cases} 1, & \text{if } o_i \in L_x \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In our similarity manager, we choose to apply the *pearson correlation coefficient* algorithm to compute similarities between any pair of two users u_x and u_k . Specifically, with respect to user u_x , the similarity is calculated by the accesses comparison defined in (1). Hence, the similarity between users u_x and u_k is formally expressed as

$$\text{sim}(u_x, u_k) = \frac{\sum_{i \in O} (r_{xi} - \bar{r}_x)(r_{ki} - \bar{r}_k)}{\sqrt{\sum_{i \in O} (r_{xi} - \bar{r}_x)^2} \sqrt{\sum_{i \in O} (r_{ki} - \bar{r}_k)^2}} \quad (2)$$

where $\bar{r}_x$ and $\bar{r}_k$ are the average access record numbers for users u_x and u_k . It is worth noting that the average numbers are float numbers in a window between 0 and 1 derived from r_{xi} and r_{ki} lists. O is the entire data item list in the cloud storage system.

The data structure of each online user in the similarity manager is an array of user-similarity pairs, which is defined in (3). Given user u_x and the corresponding similarity value $\text{sim}(u_x, u_k)$ with respect to user u_k , we express the user-similarity pair ϕ_{xk} as

$$\phi_{xk} = \langle u_k, \text{sim}(u_x, u_k) \rangle. \quad (3)$$

Due to high diversity and high overhead on computing similarity among all legitimate users, we propose a user

sampling module at the heart of the DDoS detection system to minimize the overhead of similarity calculation while improving detection accuracy. In the user sampling module, we designate a *fitness* value to select candidate users from the legal user set. Formally, fitness f_k quantifies the degree at which user k is an appropriate sample user. We derive f_k from (4) as:

$$f_k = -\frac{n_O}{n_{O_k}} \sum_{i \in R_k} (r_{ki} - \bar{r}_i)^2. \quad (4)$$

where n_O is the number of available data items in the whole cloud storage and n_{O_k} is the number of valid data items requested in the history record of user k . If a user has a long access history where a wide variety of data items were accessed, then this user tends to have a high fitness. The summation in (4) resembles a variance estimation, where R_k is a set of data objects rated by users k , r_{ki} is a rating record in the history of data item o_i by user k and $\bar{r}_i$ is the average rating by all the legal users. It is evident that a low variance modeled by (4) signifies that a user is an appropriate representative of legal users to be included in the user sample set. Therefore, we set a the fitness to a negative value to acquire a high fitness value in the user sampling module.

The similarity manager calculates similarity values of user u_x with respect to all the users registered in sample user set U . Therefore, given user x , there is a list of $\text{sim}(u_x, u_k)$, where we have $k \in S$. In the DDoS detection procedure, anomalies are identified by tracking a changing trend in similarity value ϕ_{xk} . We assume that ϕ_{xk} and ϕ'_{xk} are the current and previous similarity values between users u_x and u_k . We denote δ_{xk} as the discrepancy between ϕ_{xk} and ϕ'_{xk} . Thus, we have $\delta_{xk} = |\phi_{xk} - \phi'_{xk}|$.

Let f_x be the status of the x th user to be evaluated. $f_x = 0$ signifies that u_x is a legitimate user; $f_x = 1$ indicates that u_x is an anomalous one. If δ_{xk} measuring a shift between the current and previous similarity values exceeds a prescribed threshold α , the detection procedure will mark the user as abnormal. More formally, we have

$$f_x = \begin{cases} 1, & \text{if } \delta_{xk} \geq \alpha \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Similarly, anomalies are pinpointed by monitoring similarity value $\text{sim}(u_x, u_k)$. In case $\text{sim}(u_x, u_k)$ goes beyond a designated threshold β , user u_x is flagged as an anomalous one. Thus, we express this rule as

$$f_x = \begin{cases} 1, & \text{if } \text{sim}(u_x, u_k) \geq \beta \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Merging (5) and (6), the following consolidated rule is adopted to discover anomalies.

$$f_x = \begin{cases} 1, & \text{if } \delta_{xk} \geq \alpha \text{ or } \text{sim}(u_x, u_k) \geq \beta \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

In the nutshell, we summarize the DDoS detection procedure into the following four steps, which entail *sample user selection*, *online user request collection*, *similarity calculation*, *similarity comparison*, respectively.

- *Sample User Selection*: The DDoS detection system groups a set of sample users by the sampling technique. The sample users are applied for making similarity comparisons with online users.
- *Online User Request Collection*: While the cloud server acquires a request list from online users, the detection system gleans online requests as lists for all the users.
- *Similarity Calculation*: The DDoS detection system dynamically and repeatedly calculates the similarity measures for all the users. The similarity result of the k th user is expressed as a list of user-similarity pairs, which contains $(u_{k0}, \text{sim}(u_x, u_{k0})), (u_{k1}, \text{sim}(u_x, u_{k1})), \dots, (u_{kn}, \text{sim}(u_x, u_{kn}))$.
- *Similarity Comparison*: The user-similarity list of each user is compared against its historical record. If δ_{xk} and $\text{sim}(u_x, u_k)$ are moving beyond the two designated thresholds α and β , the DDoS detection model identifies the user as abnormal.

D. Algorithms

Our detection model generates a sample user set prior to kicking off the detection procedure, which periodically keeps tracks of legitimate users' behavioral trends. We illustrate below the user sampling process in Algorithm 1.

Algorithm 1: The sample user selection algorithm.

Input:

U' : All the legitimate users in a cloud storage system;
 O : Data item set contains valid ratings in a history record;

Output:

U : Sample user set

- 1: $\text{avg}(O)$;
 - 2: **for all** $u_{x'} \in U'$ **do**
 - 3: $\text{sampleUserSet.add}(\text{fitness}(u_{x'}, \text{avg}))$;
 - 4: **end for**
 - 5: $\text{sampleSelection}(U, h)$;
 - 6: **return** sample user set U ;
-

In Step 1, we calculate the average ratings of all the data items accessed by all the legal users. The $\text{sampleUserSet}()$ function generates a list of user-fitness pairs governed by Equation (4) in Step 3. In case that the fitness values of all the users are obtained, the system elects the top h user as sample users.

When the cloud storage server is accessed by online users, our detection system assumes that every user may become a potential attacker. As such, each online user is thoroughly measured and speculated by our detection system while handling and processing user requests.

Algorithm 2 depicts the procedure of a similarity-based detection model to guard against application-layer DDoS attacks in cloud storage. Recall that a total of four steps (see also Section III-C) are carried out to detect anomalous users by comparing the similarities between each current online user against all the sample users originated by Algorithm 1.

The input information of Algorithm 2 embraces a given user u_x accompanied by a list of online requests O_x , a sample user set U , the similarity history record $S(u_x)$ of user u_x and the two thresholds α , β .

Algorithm 2: The high-level controller of anomaly user detection.

Input:

A given current user ID u_x ; /* potential attacker */
 The online requests O_x of u_x ;
 Sample user set U ;
 The similarity history record $S(u_x)$ of u_x ;
 The threshold settings α and β ;

Output:

f_x : the status of user u_x ;
 1: $C(u_x) = \text{null}$;
 2: **for all** $u_k \in U$ **do**
 3: $C(u_x).add(\text{SimilarityCalculation}(u_x, u_k))$;
 4: **end for**
 5: **for all** $s(u_x) \in S(u_x)$ **do**
 6: **for all** $c(u_x) \in C(u_x)$ **do**
 7: **if** $s(u_x).getKey() == c(u_x).getKey()$ **then**
 8: $\delta_x = |s(u_x).getValue() - c(u_x).getValue()|$
 9: **if** $\delta_x \geq \alpha$ **then**
 10: $g_x = g_x + 1$;
 11: **end if**
 12: **end if**
 13: **end for**
 14: **end for**
 15: **if** $g_x \geq \beta$ **then**
 16: $f_x = 1$; /* Anomalous User */
 17: **else**
 18: $f_x = 0$; /* Legitimate User */
 19: **end if**
 20: **return** f_x ;

In Algorithm 2, Steps 2-4 repeatedly undertake similarity calculations to construct a list of similarity values between the current user u_x and sample users in set U forged by Algorithm 1. More specifically, the similarity calculation is implemented by function *SimilarityCalculation()* in Step 3.

Steps 5-16 are responsible for comparing between the current similarities calculated in Step 3 and those registered in history record $S(u_x)$, followed by checking if the user is an anomalous user. Set $S(u_x)$ represents a history similarity record between current user u_x and those elected in sample user set U . In Step 7-8, we gauge similarity discrepancy between the similarity values extracted from $S(u_x)$ and $C(u_x)$. The *getKey()* and *getValue()* functions acquire user IDs and similarity values from the user-similarity pairs (see also Section III-C). The similarity comparison is implemented in Steps 10 and 16. While a single similarity value changes beyond the threshold α , the abnormal value increased by 1 to record the similarity anomaly change. While the number of anomaly changes beyond the point β , the detection system identifies the current user u_x as an abnormal user and return the user Id at Step 18.

In our design, the user sampling module ought be performed in Algorithm 1 prior to the DDoS detection procedure. This order of execution is expected because sample user generations are judiciously maintained by cloud storage regardless of the DDoS detection procedure.

E. Overhead Analysis

We reckon that the similarity-based DDoS detection system imposes a relatively low computation overhead. Compared to general machine-learning- algorithm-based detection models, our proposed system has a leading edge in terms of minimizing the cost of data training and analysis. In the worst scenario, the process of electing sample users consumes computing resources in the case of a massive number of legal users. To cut down the overhead of selecting sample users, we implement a light-weight user sampling module by incorporating the random selection algorithm, which randomly picks a relatively small group of legitimate users as the candidates without hefty cost.

IV. CONCLUSIONS

We started this paper with an overview of DDoS attacks as well as detection techniques in cloud storage systems. After presenting a broad list of techniques for DDoS detection, we stay focused on the DDoS threats and detection techniques at the application layer. The focal point of our study is centered around anomaly detection techniques as a safeguard against application-level DDoS threats. We delved into multiple innovative and promising ways of applying machine learning solutions to efficiently identify DDoS zombies in cloud computing environments. Next, we shed bright light on the challenges in DDoS detection, followed by proposing a similarity-based DDoS detection system. The overarching goal of the designed system is to trace similarity changes among users to diagnose the behaviors of users who are out of ordinary. Such users will be marked by our system as potential DDoS attackers at the application level.

At the heart of our proposed DDoS detection system, we devised a user sampling module and a similarity comparison module. The sampling module attempts to elect the most appropriate user set from existing legal users, whose history access records are thoroughly inspected. The implementation of the user sampling module is crucial and indispensable to the detection system because this module is forged to (1) minimize the overhead of DDoS attack detection and (2) to curtail the effects of reasonable changes from a small group of users (e.g., users' interests updates). Along this line, the similarity comparison module cuts down the calculation cost spent in comparing among users behaviors. We expect that highly accurate DDoS threats detection can be successfully implemented in a light-weighted manner by the virtue of our proposed users' behavior analysis.

ACKNOWLEDGMENT

This work is supported in part by the U.S. National Science Foundation under Grants IIS-1618669, OAC-1642133, and CCF-0845257.

REFERENCES

- [1] J. Xu, F. Zhou, Z. Jiang, and R. Xue, "Dynamic proofs of retrievability with square-root oblivious ram," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, no. 5, pp. 611–621, 2016.
- [2] D. Dittrich, "The dos project's 'trinoo' distributed denial of service attack tool," 1999.
- [3] M. A. Alarqan, Z. F. Zaaba, and A. Almomani, "Detection mechanisms of ddos attack in cloud computing environment: A survey," in *International Conference on Advances in Cyber Security*. Springer, 2019, pp. 138–152.
- [4] B. Meng, W. Andi, X. Jian, and Z. Fucui, "Ddos attack detection system based on analysis of users' behaviors for application layer," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1. IEEE, 2017, pp. 596–599.
- [5] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp, "User behavior anomaly detection for application layer ddos attacks," in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2017, pp. 154–161.
- [6] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.
- [7] W. Yen and M.-F. Lee, "Defending application ddos with constraint random request attacks," in *2005 Asia-Pacific Conference on Communications*. IEEE, 2005, pp. 620–624.
- [8] L. Stein, "The world wide web security faq, version 3.1. 2," <http://www.w3.org/Security/Faq/>, 2002.
- [9] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer networks*, vol. 44, no. 5, pp. 643–666, 2004.
- [10] V. L. Thing, M. Sloman, and N. Dulay, "Adaptive response system for distributed denial-of-service attacks," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2009, pp. 809–814.
- [11] R. Poddar, S. Wang, J. Lu, and R. A. Popa, "Practical volume-based attacks on encrypted databases," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 354–369.
- [12] F. Yihunie, E. Abdelfattah, and A. Odeh, "Analysis of ping of death dos and ddos attacks," in *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, 2018, pp. 1–4.
- [13] K. S. Bhosale, M. Nenova, and G. Iliev, "The distributed denial of service attacks (ddos) prevention mechanisms on application layer," in *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*. IEEE, 2017, pp. 136–139.
- [14] J. Cheng, M. Li, X. Tang, V. S. Sheng, Y. Liu, and W. Guo, "Flow correlation degree optimization driven random forest for detecting ddos attacks in cloud computing," *Security and Communication Networks*, vol. 2018, 2018.
- [15] K. J. Singh and T. De, "Mlp-ga based algorithm to detect application layer ddos attack," *Journal of information security and applications*, vol. 36, pp. 145–153, 2017.
- [16] A. Dhanapal and P. Nithyanandam, "An openstack based cloud testbed framework for evaluating http flooding attacks," *Wireless Networks*, pp. 1–11, 2019.
- [17] P. Kamboj, M. C. Trivedi, V. K. Yadav, and V. K. Singh, "Detection techniques of ddos attacks: A survey," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*. IEEE, 2017, pp. 675–679.
- [18] X. Ma and Y. Chen, "Ddos detection method based on chaos analysis of network traffic entropy," *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, 2013.
- [19] Ö. Cepheli, S. Büyükcörok, and G. Karabulut Kurt, "Hybrid intrusion detection system for ddos attacks," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.
- [20] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks," in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.
- [21] W. Lee, S. J. Stolfo, and K. W. Mok, "Adaptive intrusion detection: A data mining approach," *Artificial Intelligence Review*, vol. 14, no. 6, pp. 533–567, 2000.
- [22] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. Narayan, "Distributed denial of service (ddos) attacks detection system for openstack-based private cloud," *Procedia Computer Science*, vol. 167, pp. 2297–2307, 2020.
- [23] H. Kwon, T. Kim, S. J. Yu, and H. K. Kim, "Self-similarity based lightweight intrusion detection method for cloud computing," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2011, pp. 353–362.
- [24] S. Alqahtani and R. F. Gamble, "Ddos attacks in service clouds," in *2015 48th Hawaii International Conference on System Sciences*. IEEE, 2015, pp. 5331–5340.
- [25] Z. Chen, G. Xu, V. Mahalingam, L. Ge, J. Nguyen, W. Yu, and C. Lu, "A cloud computing based network monitoring and threat detection system for critical infrastructures," *Big Data Research*, vol. 3, pp. 10–23, 2016.
- [26] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "Ddos attack protection in the era of cloud computing and software-defined networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.
- [27] A. Bakshi and Y. B. Dujodwala, "Securing cloud from ddos attacks using intrusion detection system in virtual machine," in *2010 Second International Conference on Communication Software and Networks*. IEEE, 2010, pp. 260–264.
- [28] T. Karnwal, T. Sivakumar, and G. Aghila, "A combor approach to protect cloud computing against xml ddos and http ddos attack," in *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*. IEEE, 2012, pp. 1–5.
- [29] S. Gupta, P. Kumar, and A. Abraham, "A profile based network intrusion detection and prevention system for securing cloud environment," *International Journal of Distributed Sensor Networks*, vol. 9, no. 3, p. 364575, 2013.
- [30] C. N. Modi, D. R. Patel, A. Patel, and R. Muttukrishnan, "Bayesian classifier and snort based network intrusion detection system in cloud computing," in *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*. IEEE, 2012, pp. 1–7.
- [31] A. Praseed and P. S. Thilagam, "Ddos attacks at the application layer: Challenges and research perspectives for safeguarding web applications," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661–685, 2018.
- [32] S. Ranjan, R. Swaminathan, M. Uysal, and E. W. Knightly, "Ddos-resilient scheduling to counter application layer attacks under imperfect detection," in *INFOCOM*. Citeseer, 2006, pp. 1–14.
- [33] Y. Xie and S.-Z. Yu, "Monitoring the application-layer ddos attacks for popular websites," *IEEE/ACM Transactions on networking*, vol. 17, no. 1, pp. 15–25, 2008.
- [34] M. A. Rahaman, A. Schaad, and M. Rits, "Towards secure soap message exchange in a soa," in *Proceedings of the 3rd ACM workshop on Secure web services*, 2006, pp. 77–84.
- [35] C. Ye, K. Zheng, and C. She, "Application layer ddos detection using clustering analysis," in *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*. IEEE, 2012, pp. 1038–1041.
- [36] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 313–316.
- [37] I. Sreeram and V. P. K. Vuppala, "Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied computing and informatics*, vol. 15, no. 1, pp. 59–66, 2019.
- [38] M. Jiang, C. Wang, X. Luo, M. Miu, and T. Chen, "Characterizing the impacts of application layer ddos attacks," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 500–507.
- [39] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.
- [40] K. Chatterjee, V. Padmini, and S. Khaparde, "Review of cyber attacks on power system operations," in *2017 IEEE Region 10 Symposium (TENSYP)*. IEEE, 2017, pp. 1–6.
- [41] T. Cao, X. Peng, C. Zhang, T. K. Al Tekreeti, J. Mao, X. Qin, and J. Huang, "A popularity-aware reconstruction technique in erasure-coded storage systems," *Journal of Parallel and Distributed Computing*, vol. 146, pp. 122–138, 2020.