2016

Controlled access to cloud resources for mitigating economic denial of sustainability (EDoS) attacks

Zubair A. Baig
Edith Cowan University, z.baig@ecu.edu.au

Sadiq M. Sait

Farid Binbeshr

10.1016/j.comnet.2016.01.002
Originally published as: Baig, Z. A., Sait, S. M., & Binbeshr, F. (2016). Controlled Access to Cloud Resources for Mitigating Economic Denial of Sustainability (EDoS) Attacks. Computer Networks, 97, 31-47. Original article vailable here.

Controlled Access to Cloud Resources for Mitigating Economic Denial of Sustainability (EDoS) Attacks

Zubair A. Baig, Sadiq M. Sait* and Farid Binbeshr*
Security Research Institute & School of Computer and Security Science
Edith Cowan University
Perth, Australia
*Department of Computer Engineering
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
z.baig@ecu.edu.au, sadiq@kfupm.edu.sa, g201001900@kfupm.edu.sa

Abstract

Cloud computing is a paradigm that provides scalable IT resources as a service over the Internet. Vulnerabilities in the cloud infrastructure have been readily exploited by the adversary class. Therefore, providing the desired level of assurance to all stakeholders through safeguarding data (sensitive or otherwise) which is stored in the cloud, is of utmost importance. In addition, protecting the cloud from adversarial attacks of diverse types and intents, cannot be understated. Economic Denial of Sustainability (EDoS) attack is considered as one of the concerns that has stalled many organisations from migrating their operations and/or data to the cloud. This is because an EDoS attack targets the financial component of the service provider. In this work, we propose a novel and reactive approach based on a rate limit technique, with low overhead, to detect and mitigate EDoS attacks against cloud-based services. Through this reactive scheme, a limited access permission for cloud services is granted to each user. Experiments were conducted in a laboratory cloud setup, to evaluate the performance of the proposed mitigation technique. Results obtained show that the proposed approach is able to detect and prevent such an attack with low cost and overhead.

Keywords: Economic Denial of Sustainability Attacks (EDoS), Cloud Computing, Network Security, Rate Control

1. Introduction

Cloud computing is a model to provision on-demand network access to a shared pool of computing resources that can accommodate varying end-user demands, with minimal service provider

intervention (1). It allows for utility-based pricing and dynamic resource assignments for services provisioned to end-users. Cloud providers manage submission of requests to the cloud based on leasing agreements of a utility-based pricing model. Service providers request resources from cloud providers, subsequently paying only for actual resource utilization, and in effect providing services to end users (2).

The cloud computing platform has gained immense popularity in recent times by allowing end-users to lease computing resources, and only pay based on individual usage. Auto-scaling is a fundamental characteristic of the cloud facilitating near-instantaneous scaling up or down of the required cloud-based services. This is based on variable user demands, which are derived from pre-agreed Service Level Agreements (SLAs). Implementation of auto-scaling within the cloud translates to rules for auto-allocation of resources such as the number of CPUs, amount of memory, or the number of networking devices. As a result, overwhelming of cloud resources is avoided. For instance, if the CPU usage for a given set of users belonging to a service provider, exceeds 80% for a period of time amounting to a minute, the cloud provider will dynamically allocate additional CPUs to reduce the overhead on the existing CPUs. On the contrary, reducing demand for resources in the cloud is handled through deallocation of computing resources, in real-time, again based on predefined rules at the provider's end. In general, the most common parameters considered by a cloud provider to facilitate auto-scaling are: *performance metric, threshold, and duration*. The *performance metric* parameter is a tuple defining the percentage of resource utilization within the cloud, in terms of the CPU utilization rate, memory usage and networking resource usage, during a given window of time. The *threshold* parameter defines the aggregate value calculated based on current *performance metric* values, at which point auto-scaling of resources is triggered. The *duration* parameter defines the period of time during which the auto-scaling condition is to be active for (3). Scaling up or down of computing resources does not exceed thresholds of maximum resource allocation to end-users of a given service provider, as stipulated in the SLA. As a simple example, if the upper and lower thresholds on CPU utilization are set to 30% and 80% at time of cloud initialization, and the duration is set to one minute then, whenever the CPU utilization exceeds 80% persistently for a period of one minute, additional CPUs will be allocated (i.e., scaling up). On the other hand, CPU resources will be scaled down when the CPU utilization falls below 30% for a period of one minute.

The cloud computing paradigm in an ideal situation will operate according to the needs and

demands of end-users, thus ensuring quality of end-user experience, through accurate auto-scaling of resources. Cloud security is rated as the greatest issue faced by cloud providers (4). EDoS attack is considered as one of the cloud security concerns, that has remained broadly unaddressed in the literature. In the presence of the adversarial class, routine operations of a cloud provider may be disrupted, through diverse and sophisticated malicious activity, conducted for achieving one of many objectives, such as: disclosure of sensitive cloud data, modification/tampering of data, high volumes of incoming requests for cloud resource allocation, identity theft, etc. Through an EDoS attack, the adversarial entity exploits the auto-scaling feature of the cloud to cause intentional and unwanted scaling up of computing resources at the cloud provider. Resulting cost associated with provisioning of cloud resources is billed to the service provider. After a persistent effort by the adversary over a period of time, the service provider is charged with an exceedingly high amount for unused and unrequested services. Economic viability of the service provider is thus left unsustainable. EDoS attacks may be launched through generation of a large volume of service and/or resource requests that appear to be legitimate. The cloud provider accordingly scales its resources to accommodate all end-user requests, duly abiding by the SLA (5)(6).

A mitigation technique to identify suspicious service requests that targets the service provider's end, and mitigate the effects of the EDoS attack through controlled resource usage is proposed in this paper. The scheme operates through a regular assessment of incoming requests from end-users, by comparing user activity (i.e., numbers and types of requests for cloud services) at the cloud provider, and by controlling the rate at which cloud service requests can be positively responded to. The proposed techniques operates with low overhead and does accurate classification of incoming requests into legitimate and malicious, based on several criteria.

The rest of the paper is organized as follows: Section 2 provides an extensive analysis of the existing work done to identify and prevent distributed attacks against the cloud. An elaborate explanation of the proposed scheme and its implementation is provided in Section 3. Experimental results and their analysis is presented in Section 4. Finally, we provide conclusions and future directions for work in Section 5.

2. Literature Review

Several techniques to detect EDoS attacks can be found in the literature. Each one comes with its share of benefits and limitations. In this section, we summarize the key contributions available

in recent literature. We begin with the proposal of Khor and Nakao (7) who proposed a scheme to ascertain a client's commitment to cloud resource requests, by having them solve crypto-puzzles. Resource access was granted to genuine clients with intents to pay for services utilized. Clients first define a crypto-puzzle difficulty level, k, and subsequently request for access to cloud resources. Due to resource constraints at the cloud provider, if an initial client request is not successfully entertained during a given frame of time, the client may request for a more difficult puzzle to solve. Upon succeeding in solving a puzzle of a given complexity, the server establishes a secure communication channel for message exchange between the client and itself. The proposed scheme has several shortcomings such as the asymmetric power consumption problem, vulnerability to puzzle accumulation attacks, and the correlation between puzzle difficulties and associated false positives, when an adversary with unknown computing capabilities participates (6). Another scheme developed is by Kumar et al (8), who proposed an EDoS attack mitigation scheme that comprises three modules, namely, packet filtering, proof-of-work technique, and egress filtering. The end-users of the cloud service must prove their commitment and legitimacy again by solving crypto puzzles. Only clients succeeding in solving the crypto-puzzles are granted access to the cloud services. The proposed scheme has some limitations such as vulnerability to puzzle accumulation attacks at the puzzle generation server. Moreover, the method proposed does not improve upon existing and popular client puzzle-based schemes for attack mitigation, as found in the literature.

In-Cloud Scrubber (9) is another mitigation technique against EDoS attacks that is based on solving puzzles. The primary function of the in-cloud scrubber service is to generate a puzzle to check the legitimacy of the user accessing the cloud service. The cloud service is switched between two modes: normal and suspected, based on the server and network bandwidth at the cloud provider. The proposed technique is activated while the cloud service is operating in suspected mode. The incoming requests during the normal mode will be immediately directed to the cloud service, whereas, the incoming requests during the suspected mode will be directed to the in-cloud scrubber service for further verification. Client puzzles are known to provide weak access guarantees to end-users (10), as malicious users with high computational power may circumvent legitimate users from gaining access to cloud services. As a result, legitimate users may be facing a debilitating waiting time before they are actually provided service access.

Sqalli et al (6) proposed a mitigation technique called EDoS-Shield. The scheme differentiates between legitimate and malicious requests through verification of human presence at the end-user

Figure 1: EDoS-Shield architecture (6).

machine. Fig. 1 shows the proposed architecture of the EDoS-Shield mitigation technique. The Virtual Firewall (VF) and Verifier Nodes operate in tandem to perform the EDoS mitigation tasks. The firewall filters incoming requests based on two lists, namely, white and black lists. Whenever the client makes an initial access request, the verifier node verifies it through a Turing test. If the client passes the Turing test, its IP address will be held in the white list and subsequent requests from the same client are forwarded to the cloud scheduler, approving resource allocation. On the contrary, if a user fails the Turing test, its IP address will be held in the black list and subsequent requests from this user will be dropped by the front-end firewall. The proposed approach has a few shortcomings. The first being its vulnerability to IP spoofing. An EDoS attack perpetrated by an attacker using a spoofed IP address belonging to the white list of the verifier node, would remain undetected. A second shortcoming is the high number of false positives identified through blocking of a large number of IP addresses belonging to legitimate users, as the two lists are not updated in a timely and accurate manner.

An enhanced version of the EDoS-Shield is proposed in (11), wherein, a Time-To-Live (TTL) field is appended alongside the IP address of end-users requesting for cloud services, was proposed by Al-Haidari et al. In this approach, the authors attempt to thwart the threat of spoofed IP addresses, as the distinctness in IP addresses when accompanied with a TTL field, will help differentiate malicious clients using spoofed addresses from legitimate ones. A similar scheme proposed in (12) allows for classification of network traffic into legitimate and anomalous based on mean absolute variances of TTL values.

VivinSandar and Shenai (13) proposed an approach for ensuring that HTTP and XML-based Distributed Denial of Service (DDoS) attacks do not trigger the auto-scaling feature of the cloud automatically, thus ensuring that an EDoS attack does not transpire as a consequence of such an attack. Their contribution mainly focuses on studying the ability of a DDoS attack through protocol vulnerability exploitation, to cause an EDoS attack against the cloud.

Masood et el (14) proposed a mitigation technique called EDoS Armor for e-commerce applications. EDoS Armor has a dual defense system, comprising admission control and congestion control. Admission control is used to limit the number of end-users who are accessing cloud resources (i.e., web server) simultaneously. Congestion control assigns priorities to permitted clients

based on a browsing behavior learning mechanism. The learning mechanism does client classification into good and bad, based on previously observed client activities. A challenge server is operational as part of the proposed scheme, to authenticate end-users by sending a challenge to each client at time of first request for cloud resource access. EDoS Armor has some limitations. It limits the elasticity feature of the cloud through admission control. The average response time of legitimate clients was also observed to be very high.

A framework called DDoS-MS (DDoS-Mitigation Scheme) (15) was proposed to mitigate the effect of the EDoS attacks. The proposed framework enhances the work proposed by Al-Haidari et al (11) through improvements on observed end-to-end delay i.e., quality of end-user experience. DDoS-MS mitigates the EDoS attack by analyzing the first two packets received from an end-user. Graphical Turing Test (GTT) and crypto-puzzles are employed for ascertaining end-user commitment to service requests made. The proposed framework comprises a firewall to filter packets based on defined and maintained white and black lists, a verifier node, and a puzzle server for generating and verifying client puzzles, a DNS server, green nodes to hide the location of the protected cloud server, and a filtering router to forward packets that originate from green nodes. False positives and negatives were unreported in the paper.

The IPA-Defender in another EDoS mitigation scheme (16), checks each request for an index page at the cloud provider's end. If the page count threshold for a given end-user i.e., requester, is exceeded, IPA-Defender drops the request/subsequent requests of that requester, and maintains its IP address in the blacklist for a period of time. IPA-Defender might be feckless in detecting fraud requests that rely on the page count threshold alone. Moreover, the proposed scheme is susceptible to false positives.

A summary of EDoS mitigation techniques discussed in this section is tabulated in Table 1.

3. The Proposed Approach

In this section, we present the proposed mitigation technique against EDoS attacks in the cloud infrastructure. The scheme does EDoS detection and subsequent mitigation of the effects of the malicious attack in reactive manner. As stated earlier, auto-scaling based on user demands, facilitates dynamic allocation and deallocation of resources in the cloud. This characteristic of the cloud can be exploited by the adversary class, to cause an unnecessary allocation of resources (based on illegitimate and malicious requests), invoicing the service provider in effect, and causing

Table 1: EDoS Mitigation Techniques Summary.

Name	Methodology	Limitations
sPoW	Packet matching mechanism and crypto-puzzle	• Asymmetric consumption power problem • Puzzle accumulation attack • Puzzle's difficulty for false positives
EDoS-Shield	Packet filtering and verification	• IP spoofing • False positive problem • False negative problem
Enhanced EDoS-Shield	Packet filtering and verification	• False positive problem • False negative problem
In-Cloud EDoS Mitigation	Packet filtering, proof-of-work technique, and egress filtering	• The methods need more clarification • Puzzle accumulation attack • Not implemented
EDoS Armor	Admission control and Congestion control	• Conflict with cloud's escalation feature • High response time of good clients • Authentication request of web applications
EDoS-MS	Packet filtering and verification	• False positive problem • False negative problem • Not implemented

Figure 2: The proposed architecture of EDoS mitigation technique

Figure 3: The EDoS Mitigation Scheme in Action

economic losses. The proposed scheme makes use of the *threshold* parameter, which constitutes a part of the triggering conditions for auto-scaling. The scheme does its analysis of incoming requests only when the request arrival rate from a given end-user, exceeds bounds defined by the *threshold* parameter. Auto-scaling is not triggered solely based on the *threshold* value. Rather, the period of time during which anomalous resource usage is observed, determined by the *duration* parameter. This aids in determining the legitimacy of an auto-scaling request. We identify the need to validate auto-scaling requests through differentiation of end-user requests into two classes, namely, *legitimate* and *malicious*. In subsequent sections we describe the technique in detail.

3.1. The Architecture

The architecture of the proposed mitigation technique against EDoS attacks is illustrated in Fig. 2. The main components of the architecture are the vFirewall and the VMInvestigator. The vFirewall filters incoming traffic based on a maintained and regularly updated black list, whereas, the VMInvestigator checks the commitment and in effect the legitimacy of a user request through implementation of the Turing test.

During the scheme's operation (when thresholds are crossed), all incoming i.e., suspect requests are redirected to the VMInvestigator for further analysis. A Turing test is put forth seeking the end-user's response. In addition, the scheme keeps a track of all user activity. Upon successfully responding to the Turing test, the *suspect* user request is passed through to the cloud resource allocator i.e., load balancer, and the associated trust values for the end-user are incremented based on a predefined rule. On the contrary, if an end-user fails the Turing test, the trust value parameter for the user is decremented again based on a predefined rule. A detailed inner working of the proposed scheme is provided in Fig. 3.

The distinct components of the architecture are elaborated upon as follows:

- **vFirewall**: Is a front-end filtering device responsible for filtering the incoming requests to the cloud services, based on a comparison with a regularly updated black list. Those requests found to be originating from black-listed users, as imposed by predefined firewall rules, are

processed accordingly. vFirewall does not drop any requests from IP addresses found in the black list, rather directs those to the VMInvestigator, for further analysis.

- **Load Balancer**: Does scheduling of jobs that arrive from the vFirewall or from the VMInvestigator. These jobs are distributed evenly across the entire cloud resource base, based on several techniques for job scheduling proposed in the literature, such as those found in (17)(18)(19)(20). In addition, the Load Balancer (e.g., Citrix NetScaler) is responsible for auto-scaling and monitoring its parameters in conjunction with the cloud platform software (e.g., Citrix CloudPlatform). Moreover, it adds a rule to the vFirewall that directs all incoming requests to the VMInvestigator, when the *threshold* parameter is exceeded.

- **DataBase (DB)**: This component of the proposed architecture is mainly used by the rate limit technique adopted by the VMInvestigator for further analysis of black-listed end-user requests. It has two tables, namely, Ratelimit and Blacklist. The RateLimit table helps keep track of the past behaviour of end-users. This table includes five fields, namely, IP address, LastActivity Time Stamp, RequestsCount, User Trust Factor (UTF), and the Count. The *IP address* represents the user's source IP address, for identification. The *LastActivity* parameter stores the last seen activity for a given user. *RequestsCount* keeps track of the number of requests made by a user in a single minute. The *UTF* maintains a value based on the success or failure of an end-user in responding to a posed Turing test (elaborated upon in the next subsection), and the *Count* is used to store the number of requests made by an end-user during a single second.

  The Blacklist Table stores the IP addresses of suspected users. These IP addresses are a copy of those held in the black list of the vFirewall. The purpose of this replication is to provide rapid access to these IP addresses for further analysis and/or dropping of requests by the VMInvestigator. It may be noted that in the proposed architecture, the database is located alongside the VMInvestigator.

- **VMInvestigator**: The purpose of the VMInvestigator is to check the user legitimacy based on a Turing test. It implements a rate limit (i.e. rate control) technique that provides controlled access to subsequent service requests from *suspect* end-users, to prevent indiscriminate resource allocation to malicious users involved in an EDoS attack.

Incoming requests are directed by the vFirewall to the VMInvestigator in two cases; either when the IP address of the request matches one found in the black list of the vFirewall, or when the auto-scaling (i.e., scale up) condition variable, namely, the *threshold* is exceeded. Fig. 4 illustrates how the incoming requests are processed at the VMInvestigator.

3.2. Limiting the Rate of Request Arrival

A rate limit technique is proposed to control the rate at which any given end-user may request access to cloud services. Through this technique, a limited access permission for cloud services is granted to each user, based on three factors, namely, Concurrent Requests Per Second (CRPS), Random Check (RC), and User Trust Factor (UTF).

The CRPS is a variable that maintains a value to define the upperbound on the number of requests permitted from a single IP address to arrive during a single second. Breaching the CRPS is determined by checking the values of *Count* and *LastActivity* fields of a given user, obtained from the vFirewall. For instance, if we assume the CRPS value to be 5, implying that the system allows 5 requests to arrive from a single user during any second of the system clock. The scheme will check the *Count* field for a given user to confirm the number of access requests arriving from a single user during one second of activity. If a sixth request (>5) for the same user arrives during the same period of time, the time difference between the current system time and the *LastActivity* parameter of the same user are checked. The user breaches the CRPS, if the time difference is less than a second. If the time difference is more than a second, the *Count* parameter for the user is reset to 1, indicating *legitimate* request for cloud service access.

Random Check (RC): The RC parameter helps identify intelligent attackers who can subvert the system by sending requests to bypass the capability of the CRPS parameter in identifying such attacks, assuming the attacker can accurately guess the CRPS value. RC values are random numbers picked from the interval [1,TRPM], and its count (*RC_values_count*) is equivalent to the CRPS value, where TRPM stands for Total Requests Per Minute, and its value equals *CRPS*60*.

Intelligent attackers can subvert the system by sending requests acceptable by the system, when compared against the defined parameters. VMInvestigator selects the RC values randomly. So, in case the attacker sends fewer requests than the predefined CRPS value, it is expected that the random check may not take place while the RC value(s) are selected from the end of the interval (i.e, [1,TRPM]). To avoid such a situation, the interval [1,TRPM] is divided equally into sub-intervals (if CRPS > 1), based on the *RC_values_count*. And hence, VMInvestigator picks

Figure 4: VMInvetigator workflow.

an RC value from each interval. For instance, if CRPS is set to 3, *RC_values_count* will also be equal to 3, $TRPM$ will be 180, and the RC values will be picked from the interval [1,180]. Since *RC_values_count* is 3, the interval [1,180] will be divided into 3 sub-intervals, [1,60], [61,120], and [121,180]. VMInvestigator will subsequently pick an RC value (three RC values in this case), from each interval randomly.

RC_values_count and $TRPM$ are directly proportional to the CRPS value. For example, if CRPS is set to 1, RC will be one value and $TRPM$ will be 60. VMInvestigator counts the user's requests per minute. When the end-user request number that matches the selected RC value(s) requests for cloud service access through the VMInvestigator, this check takes place. VMInvestigator resets the end-user's request count (identified by the *RequestsCount* field) to zero each minute.

User Trust Factor (UTF): The UTF variable provides a crude (1st level) classification of cloud service requests originating from end-users. The value of UTF is calculated at the VMInvestigator when a request from a new user with a certain IP address arrives at the VMInvestigator. The default value of UTF is 0.5, where UTF $\in$ [0,1]. For a typical trust framework, UTF can be classified into three levels, good, average, and bad (21). For our proposed scheme, we select the following intervals for the UTF: good UTF (0.75-1], average UTF [0.25-0.75], and bad UTF [0-0.25). The UTF value varies based on the outcome of the Turing test. If a user fails (e.g., responds with a wrong answer or if a request times out), the UTF value is decremented by 0.02. If the user passes the test, UTF is incremented by 0.01. To penalize failure users, UTF increments change less rapidly than UTF decrements (22). The VMInvestigator checks the UTF value of all users periodically. Users with bad UTF are marked as malicious, and as a consequence, their IP addresses are held in the black list of the vFirewall. Users with good UTF values are removed from the black list (if they are found in the black list).

It may be observed from Fig. 4 that the VMInvestigator first checks if a user exceeds the CRPS value. If so, its UTF is checked. The purpose of this step is to reduce false positives with a reasonable degree of accuracy (e.g., legitimate requests from different users with shared IP addresses arriving at the same time). If the UTF is found to be less than 0.25 (i.e., bad UTF), the VMInvestigator drops the request. If the UTF is found to be more than 0.25, a second chance is

granted to the user to successfully gain cloud service access, upon passing the Turing test. If the user does not exceed the CRPS, the VMInvestigator checks if the corresponding request number matches the RC value(s) generated by the random number generator. The purpose of this step is to detect smart attackers, with a reasonable degree of accuracy. If the user is new to the system, its information is stored in the *RateLimit* table of the DB, and its request is directly forwarded to the load balancer.

4. Experiments and Analysis

In this section, we analyse the results obtained from experiments conducted in a lab environment, to study the performance of the proposed mitigation technique against EDoS attacks. We compare the performance of the scheme to scenarios without any mitigation scheme in place.

To build the cloud, we deployed two servers, namely, a management server and a compute server. The management server is responsible for managing the cloud services. Citrix CloudPlatform 3.0.5 is the software that is installed on the management server, running on CentOS 6.2. The compute server or the hypervisor is the container of cloud services hosted in the form of virtual machines. Citrix XenServer 6.0.2 is the software that represents the cloud hypervisor.

Citrix NetScaler VPX 10.e implements the Load Balancer for the cloud provider. It is installed as a virtual machine running on Citrix XenServer 6.0.2. The *Least Connection Algorithm* is implemented by the load balancer to distribute the incoming traffic. To implement auto-scaling, Citrix NetScaler 10.e is used in conjunction with Citrix CloudPlatform 3.0.5.

Two machines running CentOS 6.4 are deployed to implement the vFirewall and VMInvestigator. PHP scripts and Cron jobs are used to implement the rate limit technique for the VMInvestigator, where a Cron job is a linux command that runs tasks (e.g, commands, scripts) at specified dates and times. It is used by the VMInvestigator to periodically execute PHP scripts such as the one that is responsible for resetting the RequestCount value of the users, required for verifying the UTF values of a given user, as well as to select a random value (RC). The database is implemented using MySQL.

4.1. Experiment Setup in a Lab Environment

We have conducted real experiments that evaluate the proposed mitigation technique and demonstrate the effect of the EDoS attack against cloud services. Table 2 shows the parameters that have been used in the experiments.

Table 2: Experiments Parameters

Prameter	**Value**
Auto-scaling metric	CPU usage
Auto-scaling upper threshold	80%
Auto-scaling lower threshold	30%
polling interval	60 sec
Duration	60 sec
Min VM instances	3
Max VM instances	10
VM instance type	Small
VM instance cost	$0.03
VM instance OS	CentOS 5.6 (64-bit)
Web server	Apache
Legitimate load	40%

The CPU usage metric is used for scaling up or down the cloud services. The upper and lower thresholds of auto-scaling are set to 80% and 30% respectively. The average CPU usage metric is checked every minute as determined by the polling interval parameter. The duration of scale up or scale down is also set to 1 minute. The aforementioned parameters with the exception of the auto-scaling lower threshold parameter are set based on the values provided in (23). Therefore, when the average CPU usage of the VM instances that represent the cloud services exceeds 80% during a single minute, one VM instance will be added to the end-user's resource set. On the other hand, if the average CPU usage of the VM instances is less than 30% for a given minute, one VM instance will be terminated.

The minimum and maximum number of VM instances are set to 3 and 10 respectively, and this is also the default value employed by the Citrix CloudPLatform. The compute offering or VM instance type is small. It has 1 CPU core with 500 MHz and 512 MB memory. Its cost is assumed to be $0.03, based on the Wowrack pricing policy (24). The VM instance OS is CentOS 5.6 (64-bit), since it is the default template of the Citrix CloudPlatform. The Apache version 2.2.3 is installed within each VM instance to run the web application (cloud service). Based on a study of Google traces (25), we generated a fixed load that consumes about 40% of the CPU usage of the cloud services using JMeter. In this case, scale down will not be triggered, as for testing EDoS attack exploits, the scale up feature of the cloud needs to be studied.

Figure 5: EDoS attack effect against CPU usage of the cloud services, without auto-scaling and mitigation technique

Figure 6: EDoS attack effect against CPU usage of the cloud services, with auto-scaling but without mitigation technique

Figure 7: EDoS attack effect against CPU usage of the cloud services, with auto-scaling and mitigation technique

4.2. Performance Analysis at the Service Provider

In this subsection, we present the experimental results to evaluate the proposed mitigation technique against EDoS attacks at a service provider's end. Attackers are assumed to be first-time arrivers at the system. We pick the steady state of the CPU usage during the experiments. Each experiment is repeated ten times and the results are averaged. Fig. 5 illustrates the effect of the EDoS attack against the CPU usage of cloud services, where the auto-scaling and mitigation techniques are inactive. It is noted here that the increase in the attack rate increases the CPU usage until it reaches its maximum value at 400 requests per second. Eventually, cloud services succumb to the attack traffic and becoming unavailable for subsequent users.

Fig. 6 shows the results of the scenario where auto-scaling is enabled but the mitigation technique is disabled. VMs are scaled up from 3 to 4 at an attack rate of 400 rps. This implies that the CPU usage goes beyond 80% when the attack rate increase from 200 rps to 400 rps. Similarly, at 600 rps of attack traffic, VMs are scaled up to 6. The VMs continue to scale up to accommodate the increasing end-user demand until they reach the maximum permissible number of VMs (i.e., 10) at 1200 rps. The CPU usage fluctuates between 60% and 80% due to auto-scaling before it reaches a peak utilization at 1400 rps of attack traffic.

In Fig. 7, we illustrate the evaluation of the proposed mitigation technique against EDoS attacks when the auto-scaling feature is enabled. It may be noted that the CPU usage fluctuated around 40% which represents legitimate usage. Also, auto-scaling does not take place as is obvious from the number of spawned VMs. It can therefore be seen that the the proposed approach is successful in mitigating EDoS attacks.

The cost associated with the cloud services is illustrated in Fig. 8. As stated in Table 2, the base case of the VM instance number is three. Each VM instance costs $0.03 per hour. The results show that implementing the mitigation technique incurs an additional cost when compared with the optimal case. The additional cost of the mitigation technique is the collective cost of operation of the VMInvestigator and the vFirewall (i.e., 2 VM instances cost). However, the

Figure 8: Experimental results of the cost

Figure 9: Experimental results of the response time

cost grows dramatically and then levels out when the VM instances reach the peak value (i.e., 10 VM instances) at 1200 rps of the attack rate, when the mitigation technique is inactive. The cost appertaining to the mitigation scheme will continue to increase, if the VM instances that represent the cloud services are not limited by a maximum value, unlike the proposed scheme.

Fig. 9 shows the results of the response time for services running in the cloud. We use the Firebug integrated tool (26), to measure the response time for cloud service requests. The requests, that help calculate the response time, access the Load balancer directly. However, the same requests are diverted through the vFirewall through to the load balancer, for other cases. It may be noted here that the response time with mitigation technique active roughly stays the same, and its value is closer to the optimal value. However, the response time without the mitigation technique in place increases exponentially with increasing number of VMs. The response time was found to be a constant 93 ms when the mitigation technique is in action.

4.2.1. Smart Attacker Scenario

In this scenario, we assume that the attacker is new to the system, and can guess the CRPS value accurately and subvert the mitigation scheme. The idea of this scenario is similar to the Fraudulent Resource Consumption (FRC) attack (27) found in the literature. The configuration of this scenario is set based on values enlisted in Table 2.

In Fig. 10, we set the CRPS to 1, and only one malicious user subverts the system by sending 1 rps (e.g., 60 requests per minute). The *RC_values_count* is one, since CRPS is one, and the RC value is 39 (selected randomly by the VMInvestigator). The results show the number of malicious requests attempting to access the cloud services per minute, and at what point in time the proposed mitigation technique succeeds in blocking these. It is noted that the illegitimate requests (false negatives) for the UTF decrement value of 0.1, for minute 1 is around 36, and 0 at minute 2. So, for minute 1, the technique was able to stop nearly 24 malicious requests out of a total of 60 requests from getting through. Given the setup values of the experiment, there could be at most 1 request out of 60 requests that matches the selected RC value and challenged with the Turing Test, whereas the remaining 23 requests that were blocked must have violated the CRPS condition and were either challenged with the Turing Test if the UTF was > 0.25 (at

Figure 10: Experimental result of the false negatives for one user, CRPS = 1.

Figure 11: Experimental result of the UTF value, CRPS = 1.

most 3 requests out the remaining 23 requests as these 3 requests will reduce the UTF to below 0.25), or dropped right away since their UTF is < 0.25. For the UTF decrement of 0.05 scenario, about 58 and 38 malicious requests passed through to the cloud at minutes 1 and 2, respectively. At minute 3, the mitigation technique succeeds in stopping these requests. At most 6 requests will reduce the UTF to below 0.25, in case of failing the Turing Test. For the UTF decrement of 0.02 scenario, the illegitimate requests fluctuate around 57 until minute 4. Its clear that the technique succeeds in stopping 3 requests out of 60. One of the 3 requests is stopped due to the Turing Test that is sent to the user because of a matching RC value selected by the system. The other 2 requests are stopped because of the Turing Test that is sent to the end-user because of breach of the CRPS values, where some requests are delayed and arrive during the time frame of subsequent requests. The mitigation technique succeeds in stopping these requests at minute 6.

Fig. 11 shows the corresponding UTF value of the requester for the previous scenario (CRPS=1). It is noted that the requester's UTF falls substantially for UTF decrements of 0.1 and 0.05. The UTF value drops from 0.5 to 0 at minute 2, for the UTF decrement of 0.1 scenario, whereas, it goes from 0.5 to 0.38 and then from 0.38 to 0 at minutes 2 and 3 respectively. The requester's UTF falls gradually for the UTF decrement of 0.02 scenario. About 0.06 is deducted from the user's UTF during the minute, until it reaches 0 at minute 6. As was mentioned earlier, the UTF deduction is a result of not answering the Turing test correctly, and when the Random Check value does not match the end-user number.

Fig. 12 shows the result of the false negatives when CRPS is set to 5, and the attacker sends 5 rps. The RC values chosen by the VMInvestigator are 31, 62, 149, 225, and 263. It is noted that, about 292 and 148 malicious requests (false negatives) access the cloud services at minutes 1 and 2 respectively, for UTF decrement of 0.02 scenario. At minute 1, the technique succeeds to stop 8 requests out of the 300 that attempted to access the system. 5 of these were stopped owing to the Turing Test that is sent to those end-users who match the selected RC values, whereas, the others are blocked as a result of violating the CRPS factor. At minute 2, 152 out of 300 suspect requests were blocked by the technique. Decrementing UTF values during this minute helps detect these large volume of suspect requests, as shown in Fig 13. The technique succeeds to stop these

Figure 12: Experimental result of the false negatives, CRPS = 5.

Figure 13: Experimental results of the UTF value, CRPS = 5.

Figure 14: Experimental results of the false negatives when the attacker sends less requests than allowed by the system, CRPS = 5.

Figure 15: Experimental results of the UTF value when the attacker sends less requests than allowed by the system, CRPS = 5.

requests at minute 3. At minute one, there are 219 and 149 malicious requests attempting to access the cloud services for UTF decrements of 0.05 and 0.1 respectively. At minute two, the proposed mitigation technique blocks all these malicious requests.

It is expected that the attacker can send fewer numbers of requests than what are permitted, to subvert the mitigation scheme. Fig 14 shows the result of the malicious requests that access the cloud services, when the attacker sends 4 rps, and CRPS is set to 5. For UTF decrements of 0.02, 236 illegitimate requests out of 240 access the cloud at minutes 1, 2, 3, and 4. it is clear that the four requests are detected because of the Turing Test that was sent to the user as a result of matching RC values. The benefit of dividing the main interval [1,TRPM=300] and picking each RC value from a different interval is manifested in such a scenario, since its expected that the random check may not take place in case of selecting values from the end of the interval [1,300]. The CRPS is ignored in such a scenario. The illegitimate requests for minute 5 are about 30, and 0 for minute 6. For UTF decrements of 0.05, 236 and 61 out of the total malicious requests arriving gain access to the cloud at minutes 1 and 2 respectively. The system blocks these requests at minute 3. The illegitimate requests for the UTF decrement of 0.1, for minute 1 is 149, and 0 for minute 2.

Fig 15 shows the corresponding UTF of the requester when it sends less requests (4 rps) than allowed by the system (5 rps). It is noted that the user's UTF is decremented by 0.08 during each of the first three minutes, for the UTF decrement of 0.02 scenario. During minute 4, the user's UTF falls substantially (because UTF<0.25) until it reaches 0 at minute 5. The UTF value of the requester for the UTF decrement of 0.05 scenario, decreases from 0.5 to 0.42 and then from 0.42 to 0, during the minutes 1 and 2 respectively. It goes from 0.5 to 0 during the first minute for the UTF decrement of 0.1 scenario.

Fig 16 shows the result of false negatives, when the attacker sends more requests than allowed

Figure 16: Experimental results of the false negatives when the attacker sends more requests than allowed by the system, CRPS = 5.

Figure 17: Experimental results of the UTF value when the attacker sends less requests than allowed by the system, CRPS = 5.

Figure 18: Experimental results of the false negatives, CRPS = 10.

Figure 19: Experimental results of the UTF value, CRPS = 10.

Figure 20: False negatives result of multiple users.

Figure 21: The effect of false negatives from multiple users on CPU usage.

by the system. In this scenario, we set the CRPS to 5, and the attacker sends 6 rps. It is noted that the mitigation technique succeeds in blocking malicious requests during the first minute, for all values of the UTF decrement. About 53, 30, and 15 malicious requests access the cloud for UTF decrements of 0.02, 0.05, and 0.1 respectively. Breaching the CRPS is undoubtedly the main reason for triggering the blocking action of the VMInvestigator. The corresponding UTF variation for this scenario is illustrated in Fig 17. At minute 2, the requester's UTF is found to be 0 for each value of the UTF decrement.

Fig 18 shows the result of the malicious requests when CRPS is set to 10, and the attacker sends 10 rps. RC values chosen by the scheme are 18, 81, 140, 214, 292, 357, 415, 446, 528, and 592. We can notice about 580, 291, and 139 malicious requests attempting to access the cloud service during the first minute, for the UTF decrement values of 0.02, 0.05, and 0.1 respectively. The corresponding UTF variation is illustrated in Fig 19. As was mentioned earlier, the requester's UTF value falls substantially due to the failure to respond to the Turing Tests upon finding a matching RC value or while breaching the CRPS value.

In Fig 20, we scale the malicious users to 10, 50, and 100 based on the results obtained from Fig 10 for the UTF decrement of 0.02. During each of the first three minutes, about 570, 2850, and 5700 malicious requests access the cloud for malicious user numbers of 10, 50, and 100 respectively. At minute 4, these malicious requests are dropped until they are completely stopped at minute 6. Moreover, we study the effect of these requests on the CPU usage of the cloud services, as shown in Fig. 21. We can notice that the CPU usage of the cloud services fluctuates around 70, 60, and 45 percent for malicious users 100, 50, and 10 respectively.

5. Conclusion and Future Work

We proposed a novel reactive scheme to detect and mitigate EDoS attacks against cloud-based services. The scheme operates with low overhead and is based on a rate limit technique for preventing suspect requests from being granted service before passing further investigative tests. The scheme comprises several components, namely, the vFirewall, VM Investigator, Load balancer, and the DataBase, all operating hand in hand, to control access to cloud services, for mitigating the effects of an EDoS attack. Limited access permission for cloud services is granted to each user, based on three factors, Concurrent Requests Per Second (CRPS), Random Check (RC), and User Trust Factor (UTF). The proposed scheme is able to detect smart attackers (e.g., fraud attackers) using the CRPS and RC factors. The CRPS factor considers the time rather than the count, which proves to be effective in detecting EDoS attacks rapidly. It tracks the user behavior and avoids the false positives when considered alongside the UTF factor. Our scheme does not request further verification of new users or legitimate users with UTF >0.25, who do not breach the CRPS value. By analyzing the results obtained from experiments conducted, it is evident that the proposed mitigation technique is able to detect and mitigate the EDoS attack effectively. Moreover, the cost and user-perceived delays imposed through the scheme on the underlying cloud communications infrastructure, were found to be minimal. As part of our future work, we intend to further analyze our proposed scheme, and its performance when service provider and network-level parameters are varied. In addition, we intend to implement diverse application scenarios, and study the scheme's performance.

6. Acknowldegements

The authors would like to acknowledge the support provided by King Abdulaziz City for Science and Technology (KACST) through the Science & Technology Unit at King Fahd University of Petroleum & Minerals (KFUPM) for funding this work through project No. 11-INF1609-04 as part of the National Science, Technology and Innovation Plan.

7. References

[1] P. Mell, T. Grance, The nist definition of cloud computing, National Institute of Standards and Technology 53 (6) (2009) 50.

[2] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1 (1) (2010) 7–18.

[3] R. P. Jessica Tomechak, Citrix cloudplatform 3.0.5 (powered by apache cloudstack) administrator's guide, Citrix Inc.

[4] I. D. Corporation, New idc it cloud services survey: Top benefits and challenges, http://blogs.idc.com/ie/?p=730 (2009).
URL `http://blogs.idc.com/ie/?p=730`

[5] C. Hoff, Cloud computing security: From ddos (distributed denial of service) to edos (economic denial of sustainability), http://rationalsecurity.typepad.com/blog/2008/11/cloud-computing-security-from-ddos-distributed-denial-of-service-to-edos-economic-denial-of-sustaina.html (2008).

[6] M. Sqalli, F. Al-Haidari, K. Salah, Edos-shield - a two-steps mitigation technique against edos attacks in cloud computing, in: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, 2011, pp. 49–56.

[7] S. H. Khor, A. Nakao, spow: On-demand cloud-based eddos mitigation mechanism, in: In Proc. of the Fifth Workshop on Hot Topics in System Dependability, 2009.

[8] M. K. et al., Mitigation of economic distributed denial of sustainability (eddos) in cloud computing, in: In Proc. of the Intl' Conf. on Advances in Engineering and Technology, 2011.

[9] N. K. et. al., Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service, in: In Proc. of the Fourth Intl' Conf. on Computational Intelligence and Communication Networks (CICN), 2012.

[10] V. D. Gligor, Guaranteeing access in spite of service-flooding attacks, in: In Proc. of Intl' Workshop on Security Protocols, 2003, pp. 80–96.

[11] F. Al-Haidari, M. H. Sqalli, K. Salah, Enhanced edos-shield for mitigating edos attacks originating from spoofed ip addresses, in: Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on, IEEE, 2012, pp. 1167–1174.

[12] S. Chapade, K. Pandey, D. Bhade, Securing cloud servers against flooding based ddos attacks, in: Communication Systems and Network Technologies (CSNT), 2013 International Conference on, 2013, pp. 524–528.

[13] S. VivinSandar, S. Shenai, Economic denial of sustainability (edos) in cloud services using http and xml based ddos attacks, International Journal of Computer Applications 41 (20) (2012) 11–16.

[14] M. Masood, Z. Anwar, S. Raza, M. Hur, Edos armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments, in: Multi Topic Conference (INMIC), 2013 16th International, 2013, pp. 37–42.

[15] W. Alosaimi, K. Al-Begain, A new method to mitigate the impacts of the economical denial of sustainability attacks against the cloud, in: Proceedings of the 14th Annual Post Graduates Symposium on the convergence of Telecommunication, Networking and Broadcasting (PGNet), 2013, pp. 116–121.

[16] B. Saini, G. Somani, Index page based edos attacks in infrastructure cloud, in: Recent Trends in Computer Networks and Distributed Systems Security, Springer, 2014, pp. 382–395.

[17] K. Dutta, R. Guin, S. Chakrabarti, S. Banerjee, U. Biswas, A smart job scheduling system for cloud computing service providers and users: Modeling and simulation, in: In Proc. of the 1st Intl' Conf. on Recent Advances in Information Technology (RAIT), 2012, pp. 346–351.

[18] M. Assuncao, M. Netto, F. Koch, S. Bianchi, Context-aware job scheduling for cloud computing environments, in: Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on, 2012, pp. 255–262.

[19] S. Chen, T. He, H. Wong, K.-W. Lee, L. Tong, Secondary job scheduling in the cloud with deadlines, in: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 2011, pp. 1009–1016.

[20] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, B.-N. Li, Job scheduling model for cloud computing based on multi-objective genetic algorithm, International Journal of Computer Science 10 (1) (2013) 134–139.

[21] Z. Zhou, Y. Luo, L. Guo, L. Sun, Assessment of p2p trust model based on fuzzy comprehensive evaluation., Journal of Software (1796217X) 8 (11).

[22] Y. Wang, D. S. Wong, K.-J. Lin, V. Varadharajan, The design of a rule-based and event-driven trust management framework, in: e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, IEEE, 2007, pp. 97–104.

[23] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pfänder, C. Serowy, I. Astrova, S. G. Grivas, M. Schaaf, Scaling in cloud environments, Recent Researches in Computer Science.

[24] WOWRACK, Wowrack cloud pricing, http://www.wowrack.com/cloud-pub-pricing.php.

[25] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, M. A. Kozuch, Towards understanding heterogeneous clouds at scale: Google trace analysis, Intel Science and Technology Center for Cloud Computing, Tech. Rep.

[26] Firebug, https://getfirebug.com/.

[27] J. Idziorek, M. Tannian, D. Jacobson, Detecting fraudulent use of cloud resources, in: Proceedings of the 3rd ACM workshop on Cloud computing security workshop, ACM, 2011, pp. 61–72.