CONTROLLED ACCESS TO CLOUD RESOURCES FOR MITIGATING ECONOMIC DENIAL OF SUSTAINABILITY (EDOS) ATTACK

BY

FARID SALEM BINBESHR

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

May, 2014

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **FARID SALEM SAEED BINBESHR** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS.**

Dr. Basem Al-Madani
Department Chairman

Dr. Sadiq M. Sait
(Advisor)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Zubair Baig
(Member)

8/7/14
Date

Dr. Marwan Abu-Amara
(Member)

To my parents, whose words of encouragement and push for tenacity ring in my ears. To my brothers and sisters, who have supported me throughout the process. To my wife and daughters, for being with me throughout the entire Master program.

ACKNOWLEDGMENTS

All the praises and thanks be to Allah Almighty, for providing me strength and determination to achieve this work. I am grateful to King Fahd University of Petroleum and Minerals for providing a great environment for education and research. I extend my deepest gratitude to my thesis Committee members Dr. Sadiq Sait, Dr. Zubair Baig, and Dr. Marwan Abu-Umara for their time and valuable comments. I would also like to acknowledge the continuing support for research provided by King Fahd University of Petroleum & Minerals (KFUPM) through project no. 11-INF1609-04. A very special thanks goes out to Hadramout Establishment for Human Development (HEHD) and Hadramout University of Scince & Technology (HUST) for the scholarship which enabled me to undertake an M.Sc. program at King Fahd University of Petroleum and Minerals (KFUPM) in Saudi Arabia. Finally, I would like to thank all my friends and colleagues for their encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Farid Salem Binbeshr

TITLE OF STUDY: Controlled Access to Cloud Resources for Mitigating Economic Denial of Sustainability (EDoS) Attack

MAJOR FIELD: Computer Network

DATE OF DEGREE: MAY, 2014

Service providers of the cloud have witnessed a rapidly growing demand to provide services to end-users in a timely manner. Security vulnerabilities against the cloud infrastructure cannot be overlooked. Through exploitation of such weaknesses, the adversary class may disrupt routine cloud operations, and have a debilitating effect on the reputation of the service provider. One attack type specifically affecting cloud services is the Economic Denial of Sustainability (EDoS) attack. Through such a malicious attack, the ability of the service provider to dynamically stretch and accommodate increasing numbers of requests from end-users, is exploited, to make it economically unviable for the service provider to sustain further demand for service from legitimate end-users. In this work, we propose a novel and reactive approach for controlling user requests for service, implemented at the cloud

providers end, to mitigate the effects of an imminent EDoS attack against critical cloud resources. Through this scheme, a 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). We conduct real-world experiments to evaluate the proposed mitigation technique against EDoS attack. The experimental results prove that the proposed approach is able to detect and prevent such attack with low cost and overhead.

CHAPTER 1

INTRODUCTION

Cloud computing has provided a novel paradigm to address the critical need for affordable and convenient resource availability to meet large-scale computing demands of contemporary applications. The cloud paradigm provides various features that are different from traditional networks such as multi-tenancy, shared resource pooling, on-demand network access, service orientation, dynamic resource assignment, self-organization, and pay for use pricing model.

Cloud computing is based on the pay per use model, wherein, vendors and service providers do not have to procure and maintain hardware and software resources to sustain their respective computing activity. Rather, services (mostly scalable) are purchased on a need basis.

1.1 Background and Terminology

The term cloud computing was inspired from the cloud character which is used to represent the Internet [6]. Cloud computing as defined by NIST [7] is "A model for

on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction." In other words, cloud computing is a new paradigm of computing in which on-demand network access, utility-based pricing (pay-for-use pricing Model), and dynamic resource assignment are provided as a service over the internet.

In cloud computing terminology, the term service provider refers to both infrastructure as well as generic service providers. Infrastructure providers manage the platform and lease the resources according to a utility-based pricing. Service providers rent the resources from Infrastructure providers and provide the services to the end users [1], as shown in Fig 1.1.

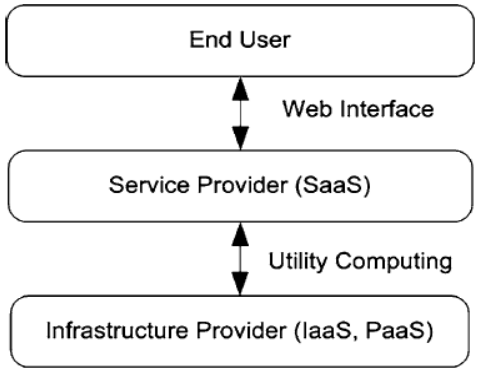

Figure 1.1: Cloud bussiness model [1].

Three different service models are defined for the cloud; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), as shown in Fig 1.2.

IaaS refers to providing storage and compute capabilities as services, usually through spawning of Virtual Machines (VMs) based on cloud service provider de-

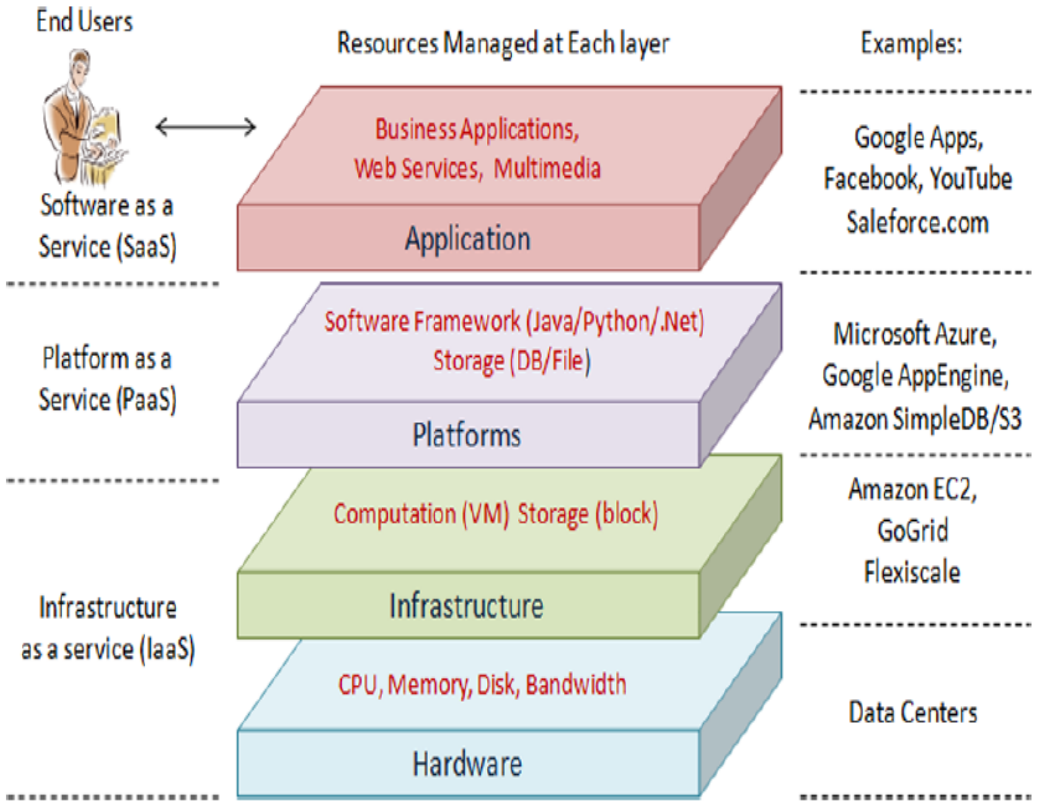

Figure 1.2: Cloud service models [1].

mands. Amazon EC2 [8] is an example suite that is built on the IaaS service model. PaaS provides a layer of software (e.g., Operating System) as a service for leasing out to the service providers, to sustain design and deployment of application-layer services. Google App Engine [9] is an example of a PaaS service model. Software as a Service (SaaS) refers to provisioning of software based on end-user demand, as a service over the Internet. Salesforce.com [10] is an examples of SaaS service model.

Cloud computing can be classified into three types; public, private, and hybrid clouds. In a public cloud, the resources are offered through the Internet for public use. In a private cloud, the resources are limited to one organization. Hybrid cloud is a mix of public and private clouds [1] [11].

1.2 Auto-scaling

Auto-scaling is defined as a characteristic of the cloud that allows the service provider to request for scaling up or down of cloud services (e.g., VMs) automatically, to accommodate variable user demands, according to a predefined condition (e.g., if CPU usage is greater than 80 % for one minute). Three parameters, performance metric, threshold, and duration, are considered with the condition to trigger the auto-scaling. Performance metric parameter is used to expose the state of the cloud services. Threshold parameter defines the limit beyond which if resource scaling is necessary, is achieved through the auto-scaling capability of the cloud. Duration parameter is the period of time during which the auto-

scaling condition is true to trigger the auto-scaling [12]. For example, assume the CPU usage metric has been used to trigger the auto-scaling. The lower and upper thresholds are set to 30% and 80% respectively. And the duration is set to one minute. So, Whenever the CPU usage go beyond 80% for one minute, cloud services will be scaled up. On the other hand, cloud services will be scaled down when the CPU usage goes below 30% for one minute.

1.3 EDoS Attack

Through an EDoS attack, the attacker exploits the auto-scaling feature of the cloud, to scale up the service provider's resources (i.e., VMs). As a consequence, this attack incapacitates the service provider from serving its users, as the cloud providers billing system will charge the attacker operations to the cloud service provider's bill.

EDoS is defined as an attack that targets the service providers economic resources by sending a huge number of requests (e.g, DDoS attacks) that appear to be legitimate, exploiting the auto-scale feature of the cloud infrastructure [13][3]. We illustrate the EDoS attack in Fig 1.3. It is clear that when the attack takes place, cloud service utilization increases, and similarly charges increase.

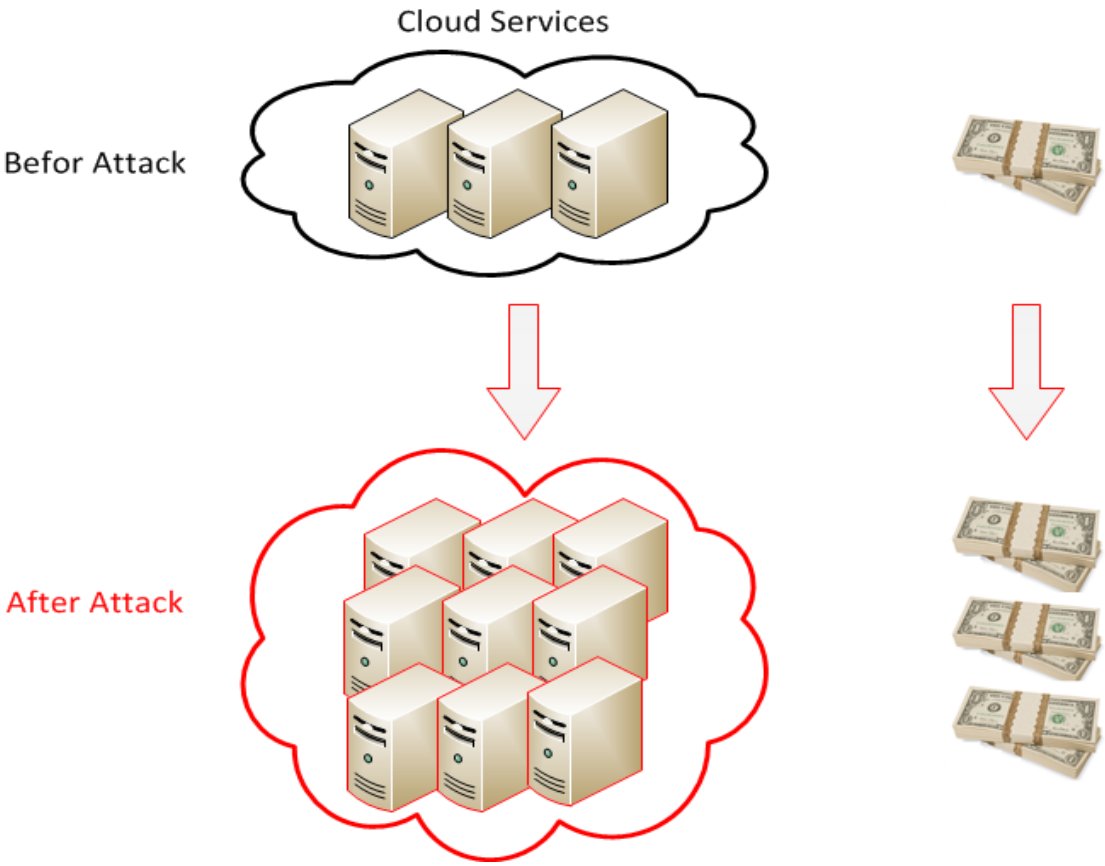

Figure 1.3: Effect of an EDoS attack against the Cloud.

1.4 Problem Statement

Economic Denial of Sustainability (EDoS) attack is considered as one of the security concerns that have hindered the migration of many organizations to the cloud technology. This is because EDoS attacks impact targets the service provider (who rents the resources from the cloud provider) and can cause financial losses. This attack exploits the elasticity feature of the cloud by causing undue increases in resource usage so as to accommodate false attacker demands. As a consequence of pay-per-use model of the cloud, service provider will be charged for the attackers activities. Ultimately, the economic viability of the service provider becomes unsustainable.

A mitigation technique to identify suspicious service requests that targets the service providers end, and mitigates the effects of the EDoS attack through controlled resource usage is proposed in this paper.

1.5 Motivation

According to a survey conducted by International Data Corporation (IDC) [2], security is ranked as the greatest challenge for cloud computing, as shown in Fig 1.4. Particular malicious attacks against the cloud, such as Economic Denial of Sustainability (EDoS), have remained largely unaddressed in the literature.

The disruption of any service provisioned through the cloud can cause a large scale damage to end-users comprising both novice private end-user applications and sophisticated business applications wishing to harness the computing power

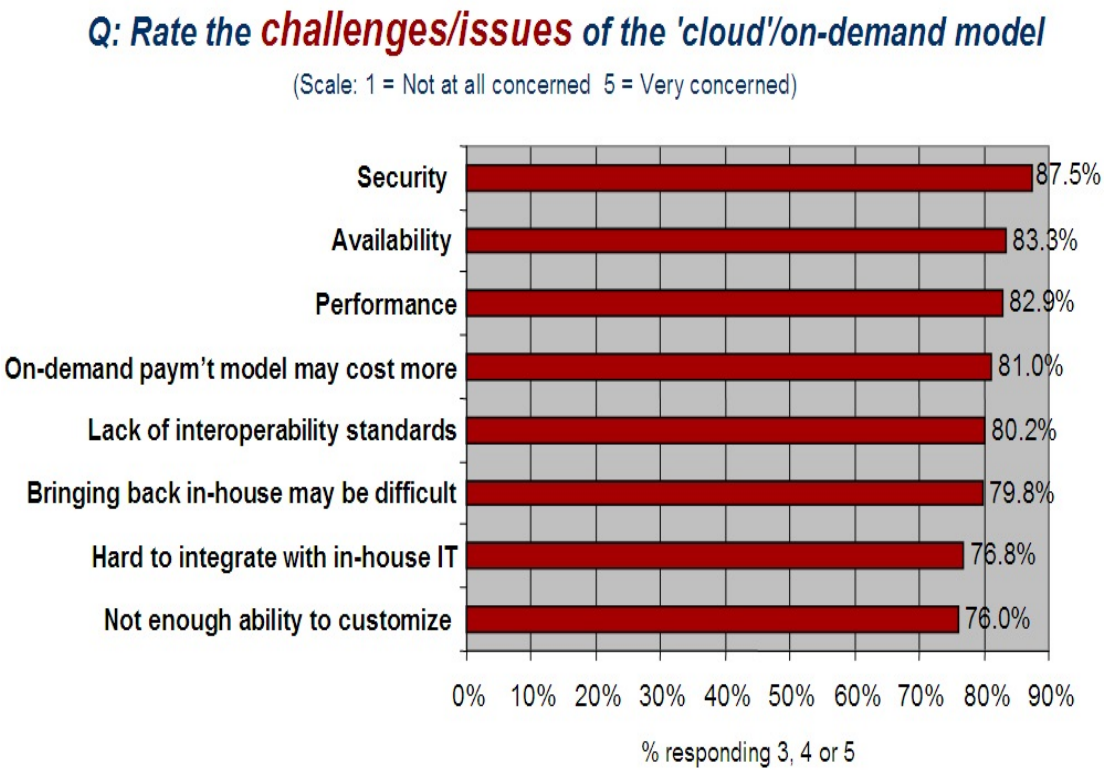

Figure 1.4: Cloud challenges survey [2]

of the clouds elastic resource pool. Users of the cloud pay as they use, based on application needs. Elasticity of resource availability at the service provider is the key driving aspect for the growing popularity of the cloud paradigm. The ability of the service provider to differentiate between legitimate and malicious users/request types, is critical for smooth and affordable resource usage at the service providers end.

However, a comprehensive mechanism for identifying routine cloud usage activity and distinguishing it from malicious activity, is largely unaddressed in the literature.

1.6 Objective

The objective of this research is to propose a mitigation technique against EDoS attack in cloud infrastructure. It should be able to detect and prevent such attack with low overhead cost. For service providers, the technique cost should be less than the expected cost due to the attack.

1.7 Contributions

In this section, we provide a list of contributions:

- Some of the research work found in the literature for addressing EDoS and DDoS attacks are summarized.
- A novel and reactive approach is proposed to detect and mitigate the EDoS

attack in cloud systems with low overhead cost, based on a rate limit technique.

- A scheme for detecting smart attackers is proposed using the RC factor, as explained in Chapter 3.
- The proposed scheme tracks the user behavior and avoids the false positives based on the UTF factor, as explained in Chapter 3.
- The proposed scheme skip checking the legitimacy of new users or legitimate users (with UTF >0.25), as explained in at UTF subsection, who does not break the CRPS or match the RC values.

1.8 Thesis Organization

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

CHAPTER 2

LITERATURE REVIEW

In this chapter, we summarize research work found in the literature for addressing distributed malicious attacks such as EDoS and DDoS against the cloud.

2.1 EDoS Attack

Khor and Nakao [14] proposed an approach ,sPoW , towards proving client commitment through solving crypto-puzzles. The purpose of the scheme is to thwart the effects of an EDoS attack, by granting access to only those clients willing to pay for service. Clients request for server access at the cloud provider's end by first defining the crypto-puzzle difficulty level, k, and subsequently requesting for access. If an initial connection request is not successfully made 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. sPoW has several shortcomings such as asymmetric consumption power problem, puzzle

accumulation attack, and puzzle's difficulty for false positives [3].

Sqalli et al [3] proposed a mitigation technique called EDoS-Shield to protect the cloud against EDoS attacks. The key factor proposed for differentiating between legitimate and EDoS requests is through verification of human presence to control an end-user machine. Fig. 2.1 shows the proposed architecture of the EDoS-Shield mitigation technique. The Virtual Firewall (VF) and Verifier Node operate in tandem to perform the EDoS mitigation tasks. The firewall filters incoming requests based on two lists, white list and black list. The verifier node verifies the incoming requests using a Turing test, during first client access. If a user passes the Turing test, its IP address will be held in the white list and subsequent requests from the same IP address will be forwarded to the cloud scheduler for providing necessary services. In contrast, if a user fails the Turing test, its IP address will be held in the black list and subsequent requests from this IP address will be dropped by the firewall. However, the proposed approach has shortcomings. One of them is its vulnerability to IP spoofing. This problem might cause an EDoS attack if an attacker spoofs an IP address belonging to the white list of the verifier node. Another disadvantage is the false positives, in which the EDoS-Shield may block an IP address that belongs to thousands of users due to the misbehavior of one. Likewise, the problem of false negatives, in which a white lister may change its behavior to harm the system.

Al-Haidari et al [15] proposed an enhanced version of the EDoS-Shield [3], wherein, a Time-To-Live (TTL) field is appended alongside the IP address of

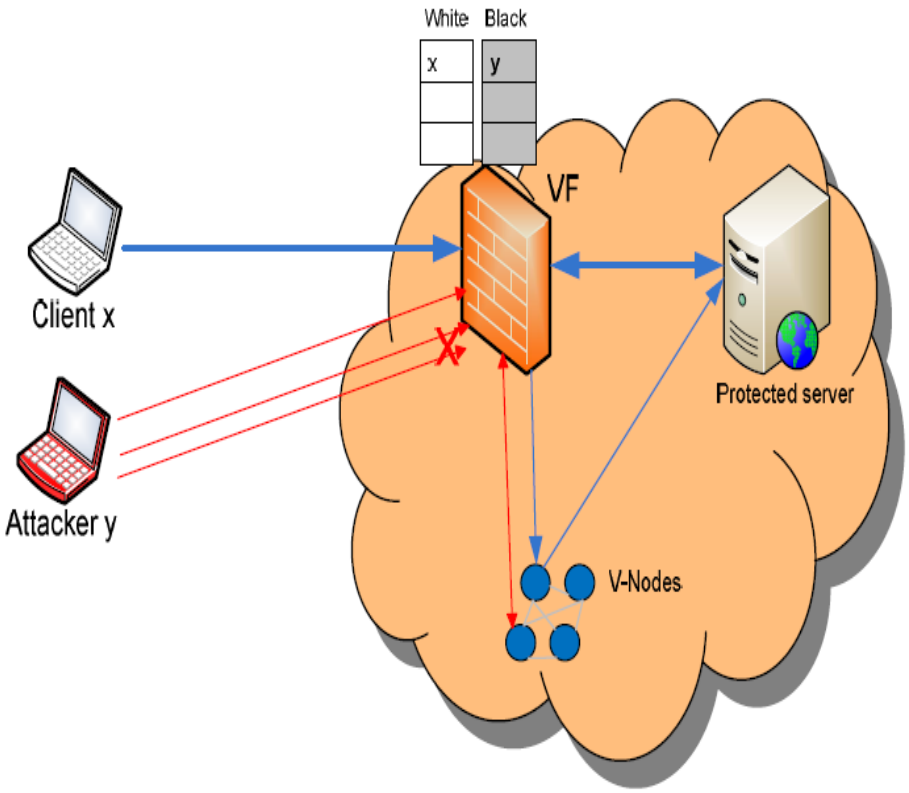

Figure 2.1: EDoS-Shield architecture [3].

cloud service requests. Through such an 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 spoofing clients from legitimate ones. Another scheme proposed to classify network traffic into anomalous based on mean absolute variance of TTL values is provided in [16].

Kumar et al [17] have provided a contrast between traditional DDoS attacks and EDoS attacks. The distinguishing point between the two is stated as follows: while the former tends to deplete available resources to the end users, the latter attempts to inflate the bill of the service provider through exploiting the property of elasticity of cloud resources. The proposed scheme is an in-cloud EDoS mitigation web service comprising three modules, namely, packet filtering, proof-of-work technique, and egress filtering. The clients of the cloud service must prove their commitment for gaining service access 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 puzzle accumulation attack at the puzzle's generation server. Also, the methods (i.e., modules) need more clarification.

In-Cloud Scrubber [18] is a mitigation technique against EDoS attacks in the cloud system. The primary function of In-Cloud Scrubber Service is to generate a puzzle to check the legitimacy of the user for accessing the cloud service. The cloud service is switched between two modes: normal and suspected, based on the server and network bandwidth. In-Cloud Scrubber Service is used while the cloud service

is operating in the 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 verification purpose. Client puzzles are known to provide weak access guarantees to end-users [19], 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 guaranteed service. In-Cloud Scrubber idea is similar to that of In-Cloud eDDoS Mitigation [17].

VivinSandar and Shenai [20] proposed an approach for ensuring that HTTP and XML based DDoS attacks do not trigger the auto-scaling feature of the cloud, thus ensuring that an EDoS does not transpire through such an attack. The 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 [4] proposed a mitigation technique called EDoS Armor against EDoS attack for e-commerce applications in cloud environment. EDoS Armor has dual defense system, the admission control and the congestion control, as shown in Fig 2.2. The admission control is used to limit the number of users client who are accessing the cloud services (i.e., web server) at the same time. The congestion control is used to assign priority to the permitted clients based on a browsing behavior learning mechanism. The learning mechanism is used to classify the clients into good and bad, based on the client activities in the

system. The challenge server is used to authenticate users into the system by sending a challenge for each client at the beginning of the session. EDoS Armor has some limitations. It contrasts the escalation feature of the cloud, in which the simultaneous user requests for cloud service are limited by the admission control. The average response time of the good clients is rather high. Moreover, some users are unwilling to solve such challenge, because it contrasts with the purpose of the web application [21][22].

A framework named DDoS-MS [23] is proposed to mitigate the effect of the EDoS attack in cloud system. The proposed framework enhance the work proposed by Al-Haidari et al [15] in terms of the end-to-end delay. DDoS-MS mitigates the EDoS attack by testing the first two packets of the service requester. Graphical Turing Test (GTT) and Crypto Puzzles are used for the testing process. The former is used to test the packet, whereas, the latter is used to test the packet's source (e.g., the user). The proposed framework made up of a firwall to filter packets based on a white and black lists, a verifier node and puzzle server for verification process, a DNS server, green nodes to hide the location of the protected cloud server, and a filtering router to forward the packets that come from green nodes only. Problems such as false positives and false negatives are still unaddressed using this mitigation system.

A mitigation technique called IPA-Defender is presented by Saini and Somani [24]. IPA-Defender focuses on detecting and mitigating EDoS attack that targets index page of any website, because it is provided freely and without authentication.

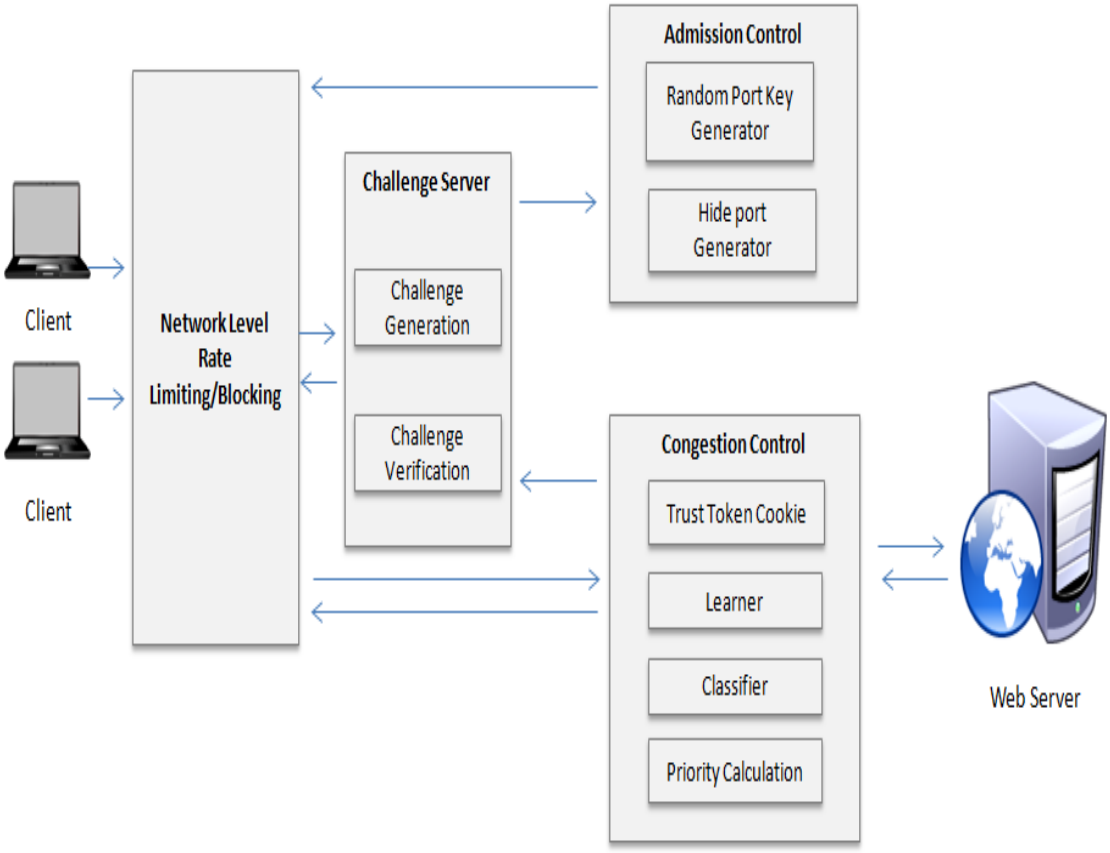

Figure 2.2: EDoS-Armor architecture [4].

The Iptables is used to implement the scheme. The idea of the IPA-Defender is to check each request for the index page. If the page count threshold of the requester is crossed, IPA-Defender drops the request/subsequent requests of that requester, and maintains its IP address in the blacklist for a period of time. The scheme needs more description. Also, IPA-Defender might be feckless to detect fraud requests that rely on the page count threshold. Moreover, the proposed scheme is susceptible to the false positive problem.

Some of the EDoS mitigation techniques discussed earlier are summarized in Table 2.1.

2.2 DDoS Attack

Distributed Denial of Service (DDoS) attacks exploit the asymmetry between the network line rate and server processing rate [25] to overwhelm server resources, and circumvent legitimate clients from timely access to service.

In [26], an adaptive swarm-based scheme is presented. Specific network routes are bound with specific client connections, and dismantled once the cloud-based service is completely provided. Based on variations in the network terrain, the route selected for a given client is modified. Certain connections, such as SSL, are not supported, due to the stateless nature of the scheme. In [27], a traceback mechanism is proposed for identifying perpetrators of DDoS attacks against the cloud. A filtering technique is implemented to ensure that only legitimate packets are seen through by the cloud virtual resources. It is also stated by the authors

Name	Methodology	Implementation	Limitations
sPoW	Packet matching mechanism and crypto-puzzle	No	• Asymmetric consumption power problem • Puzzle accumulation attack • Puzzle's difficulty for false positives
EDoS-Shield	Packet filtering and verification	Yes	• IP spoofing • False positive problem • False negative problem
Enhanced EDoS-Shield	Packet filtering and verification	Yes	• False positive problem • False negative problem
In-Cloud eDDoS Mitigation	packet filtering, proof-of-work technique, and egress filtering	No	• Poor description about the methods • Puzzle accumulation attack • Not implemented
In-cloud Scrubber	Crypto-puzzle	No	• Similar to In-Cloud eDDoS Mitigation • Puzzle accumulation attack • Not implemented
EDoS Armor	Admission control and Congestion control	Yes	• Conflict with cloud's escalation feature • High response time of good clients • Authentication request of web applications
EDoS-MS	Packert filtering and verification	No	• False positive problem • False negative problem • Not implemented
IPA-Defender	Rate Limit	Yes	• A poor described scheme • Lack of detecting fraud requests • False positive problem

Table 2.1: EDoS Mitigation Techniques Summary.

that when the server (or VM) capacity is not exceeded, even rogue packets are served by the proposed scheme.

Other work related to Denial of Service (DoS) attacks against the cloud is summarized as follows. In [28], the authors proposed a mechanism to detect and mitigate the effects of an HX-DoS attack, defined as a combination of HTTP and XML messages, attempting to disrupt cloud activity. The proposed scheme, namely, ENDER, is comprised of two previously proposed schemes, CLASSIE [29], and Added Decision Making and Update (ADMU) [30][31]. Messages arriving at the cloud provider's end are assessed by the CLASSIE analyzer, which is located one hop away from the router or host. The ADMU component of the proposed scheme makes a decision on whether to give access to a cloud resource by computing a likelihood of a message, not previously classified, as constituting an attack.

In [32], a network-level access control mechanism is proposed to prevent DoS attacks against cloud resources. A cloud access manager (CAM) operates within the proposed architecture. The CAM creates access zones. Zone 1 is implicit, implying that if one member of the zone has access to a certain cloud resource, others do as well. Moreover, cloud instances can be part of multiple zones simultaneously. Each zone has certain set of privileges. In addition, policies can be added to the zones, for allowing further security controls.

In [33][**?**], a traffic analysis-based approach for identifying DoS attacks is proposed. Statistical properties are extracted from network traffic based on multivari-

ate correlation analysis. The scheme employs triangle areas to extract correlated feature information from network traffic. In order to accurately identify DoS traffic, a normal network traffic profile is defined through density estimations, aforehand.

A Confidence-Based Filtering (CBF) method is proposed in [34]. CBF method is investigated for filtering DDoS attack packets in cloud environment. Its process needs two periods, non-attack period and attack period.During the non-attack period, the CBF generates a nominal profile for the legitimate packets. The CBF make use of the nominal profile to filter the illegitimate packets at attack period.

The authors of [35] propose a detection framework against DDoS attack in cloud environment. The proposed framework relys on HTTP packet pattern and rule engine in detecting DDoS attack. It made up of three modules, Packet and Log Collection Module (PLCM), Pattern Analysis Module (PAM), and Detection Module (DM). Parsing the packet transmission and web logs is main work of PLCM. PAM is responsible for creating patterns that are used to detect DDoS attack. DM is used to detect DDoS attack based on a normal behavior module.

A defense system against DDoS attack for web services in cloud system is proposed in [36]. The proposed approach focuses o detecting two types of application layer attacks, XML and HTTP DoS attacks. The idea is to extract features from such attacks to build a normal request profile. Thus, this profile is used by the outlier detection to detect suspicious requests. The DDoS defense system is suitable to be deployed in cloud environment to protect cloud providers and cloud

brokers.

CHAPTER 3

THE PROPOSED APPROACH

In this chapter, we present the proposed EDoS attack mitigation technique for the cloud infrastructure. It is a reactive scheme that detects and mitigates the effects of an EDoS attack against the auto-scaling feature of the cloud, where auto-scaling is defined as the characteristic of the cloud that allows the service provider to request for scaling up or down of cloud resources, automatically, to accommodate variable user demands.

The proposed approach make use of *threshold* parameter that accompanies the auto-scaling conditions. Threshold value is a good indicator of the abnormal behaviour as well as its importance to maintain the performance and save the cost. Auto-scaling does not being triggered immediately when the threshold is crossed. The threshold should be crossed for a period of time, as determined by the duration parameter, to trigger the auto-scaling. This period of time enables our scheme to mitigate EDoS attack effects, to ensure that auto-scaling is triggered only for legitimate requests. The scheme is a reactive, because it runs only when

the threshold parameter is crossed.

3.1 The Architecture

The architecture of the proposed mitigation technique against EDoS attack is shown in Fig. 3.1.

The main components of the architecture are the vFirewall and VMInvestigator. vFirewall is used to filter the traffic based on a black list, whereas, VMInvestigator checks the user legitimacy based on a turing test.

When the scheme runs (scale up threshold is crossed), all incoming requests are redirected to the VMInvestigator for further check. Checking process is performed by sending a Turing test to the user. The system tracks the user's behavior. When the user answers the test correctly, its request will be passed to the cloud and its trust will be increased, and vice versa. A detailed inner working of the proposed scheme is provided in Fig. 3.2.

The distinct components of the architecture are elaborated below:

3.1.1 vFirewall:

Is a front-end filtering device responsible for analyzing the incoming requests to the cloud services. Those requests found to be originating from black-listed users, as imposed by predefined firewall rules, are processed accordingly. Typical examples of firewall filtering include: Dropping requests originating from specific IP addresses, Port Numbers, Destined for specific IP addresses and Port Numbers.

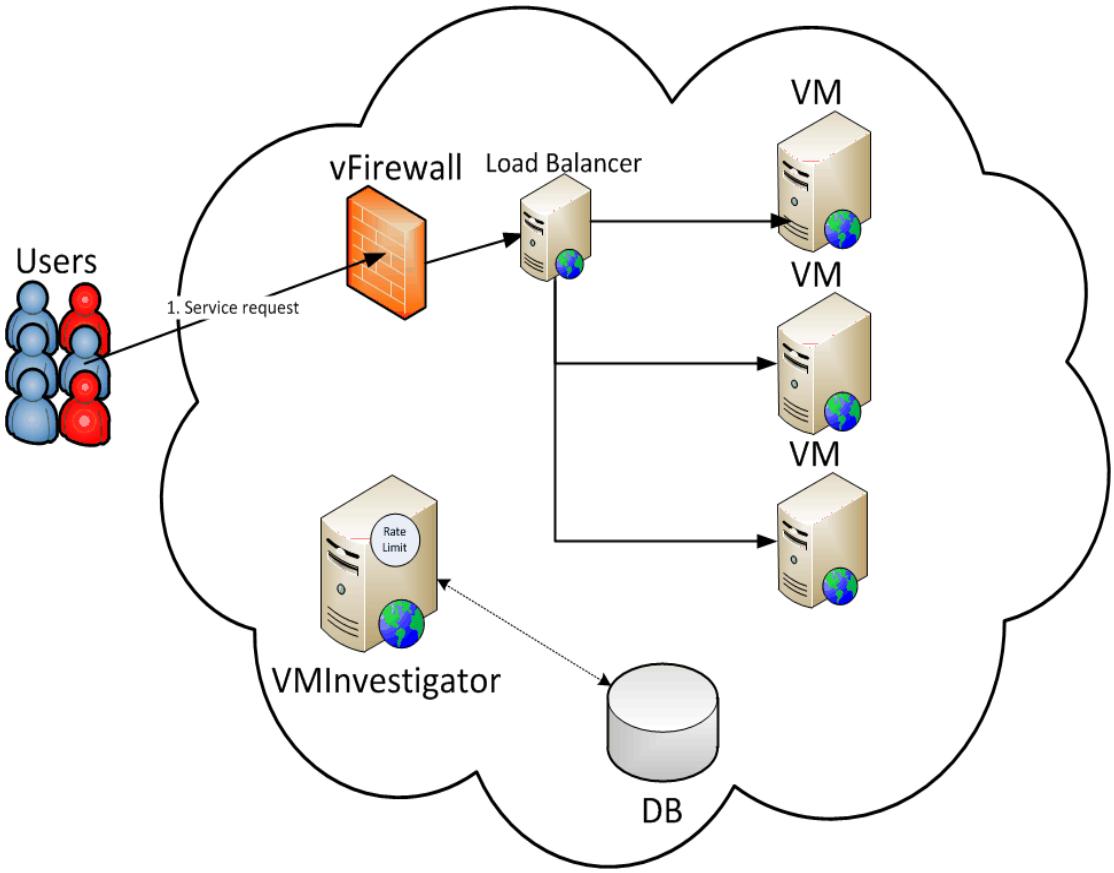

Figure 3.1: The proposed architecture of EDoS mitigation technique

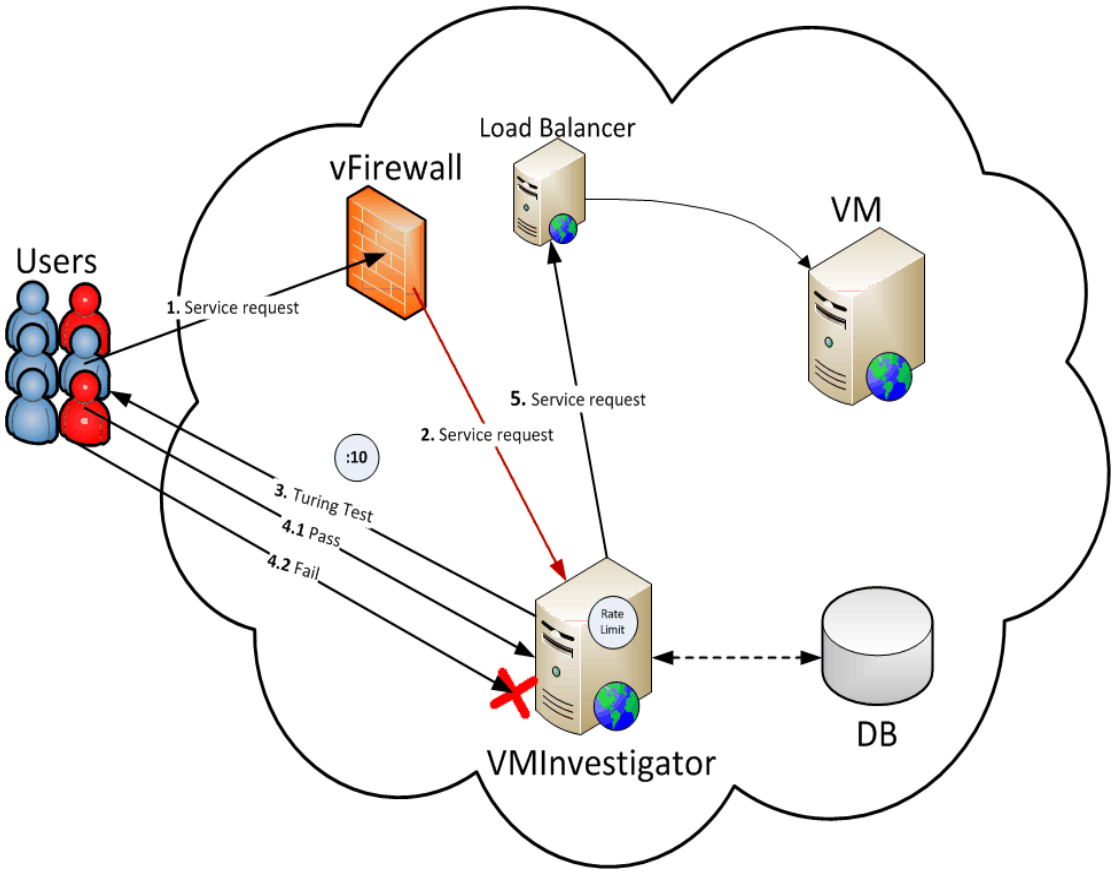

Figure 3.2: EDoS Mitigation Scheme in Action

Advanced firewall versions also perform payload analysis for deep-level understanding of sophisticated attacks that may target the cloud provider resources. However, for the proposed architecture, the vFirewall does not drop any requests from IP addresses found in the black list, rather directs these to the VM Investigator, for further processing.

3.1.2 Load Balancer:

Does scheduling of jobs that arrive from the vFirewall or from VMIvestigator. These jobs are distributed evenly based on several techniques for job scheduling proposed in the literature, such as those found in [37][38][39][40].

Also, 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 vFirwall that directs all incoming requests to the VMInvestigator, when the auto-scaling condition threshold is crossed.

3.1.3 Data Base (DB):

This component of the proposed architecture is mainly used by the rate limit technique which is implemented by the VMInvestigator. It has two tables, Ratelimit and Blacklist.

Rate limit table is used to track the past behaviour of users. This table includes five fields, IP, LastActivity, RequestsCount, UTF, and Count. IP represents the

user identity. LastActivity keeps the last seen of a certain user in the system. RequestsCount keeps the user's requests number per minute. UTF denotes for User Trust Factor, which is explained in the VMInvestigor section. And Count is used to keep the user's requests per second.

Blacklist table stores the IP addresses of the malicious 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 a fast access to these IP addresses for the VMInvestigator, because the DB is located at the same machine as the VMInvestigator.

3.1.4 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 a controlled access to subsequent service requests from end-users, to prevent indiscriminate resource allocation to malicious users that are perpetrating an EDoS attack.

Incoming request is directed to the VMInvestigator in two cases, either when the IP address of this request is held in the black list of the vFirewall, or when the auto-scaling (i.e., scale up) condition threshold is crossed. Fig. 3.3 illustrates how the incoming requests are processed at VMInvestigator.

Rate Limit

Rate Limit technique is used to control the access of the cloud services and reduce the overload of the VMInvestigator. Through this technique, a limited ac-

cess 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).

Concurrent Requests Per Second (CRPS): CRPS means how many requests are permitted from a single IP address during a second. Breaching the CRPS is determined by checking the Count and LastActivity fields of the user. For instance, assume CRPS is set to 5, which mean the system allows 5 requests per second, the system initially checks the Count field of a user. So, when the sixth request (>5) of that user access the system, the time difference between the current time and LastActivity of the user is checked. The user breaches the CRPS, if the time difference is less than a second. If the time difference is more than a second, Count is rest to 1.

Random Check (RC): RC is used to help in detecting smart attackers, who can subvert the system by sending requests based on the CRPS value (i.e., assuming the attacker knows the CRPS value). RC values are taken from the inerval [1,TRPM], and its count (*RC_values_count*) is same as the CRPS value. TRPM stands for Total Requests Per Minute, and its value equals *CRPS*60*.

Smart attackers can subvert the system by sending same requests as allowed by the system or less. VMInvestigator selects the RC values randomly. So, in case of sending less requests than what is allowed by the system, it is expected that the random check may not take place while the RC value(s) are aligned to the end

of the interval (i.e, [1,TRPM]). To avoid such situation, the interval [1,TRPM] is divided equally into sub-intervals (if CRPS > 1), based on the *RC_values_count*. And hence, VMInvestigator picks an RC value from each interval. For instance, if CRPS is set to 3, *RC_values_count* will be 3, $TRPM$ will be 180, and the RC values will be taken 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]. therefore, VMInvestigator picks an RC value 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 request number that matches the RC value(s) access the VMInvestigator, this check takes place. VMInvestigator rests user's requests count (identified by the RequestsCount field) to 0 every minute. Also, RC values are changed by the VMInvestigator periodically, using a Cron job .

User Trust Factor (UTF): UTF is used to track the past behavior of users who visit the system. UTF value is initiated at the VMInvestigator when a request from a new user with a certain IP address passes the VMInvestigator. The default value of UTF is 0.5, where UTF value belongs to the interval [0,1]. UTF is classified into three levels, good, average, and bad UTF [41]. The interval of good UTF, average UTF, and bad UTF are (0.75-1], [0.25-0.75], and [0-0.25) respectively.

UTF value is affected by the Turing test. If a user fails (e.g., given a wrong

answer or is timed out) the test, UTF value is reduced by 0.02. If it passes the test, UTF value is incremented by 0.01. To penalize the failure users, UTF increment should be less than the UTF decrement [42]. This is because test failure is an indicator of the attack.

VMInvestigator checks the UTF value of all users periodically. Users with bad UTF are marked as malicious users, and as a consequence, their IP addresses are held in the black list of the vFirewall. Users with good UTF are removed from the black list (if they are in the black list).

It is noticed from Fig. 3.3 that, VMInvestigator firstly checks if the user violates the CRPS condition. If yes, then its UTF is checked. The purpose of this step is to give a chance for the false positives (e.g., legitimate requests from different users have a shared IP address coming at the same time). If the UTF is less than 0.25 (i.e., bad UTF), then VMInvestigator drops the request. If the UTF is more than 0.25, a chance is granted for the user to get the cloud service when passing the Turing test. UTF value is incremented by 0.01 in case of passing the test, whereas, it is reduced by 0.02 in case of failing the test.

If the user does not break the CRPS, VMInvestigator checks if the request number matches the RC value(s). The purpose of this step is to detect smart attackers. For example, an attacker knows that the system allows only 1 req/sec (CRPS==1) for each user, and it subverts the system from multiple machines (e.g, zombies) by sending the same rate (1 req/sec) or less (2 req/sec) per machine. Assuming each machine has a different identity (IP address) and it is new to the

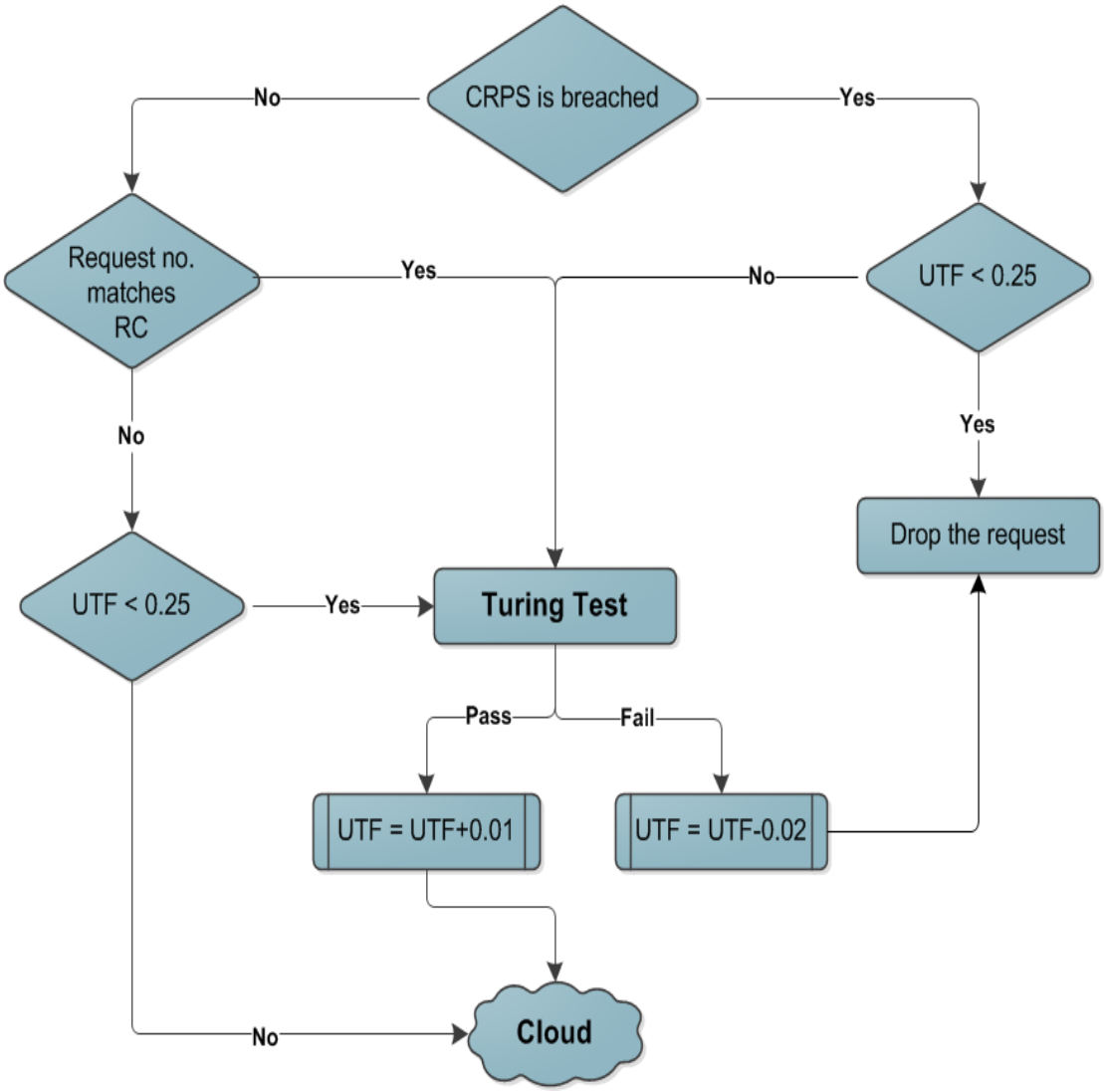

Figure 3.3: VMInvetigator workflow.

system (i.e., UTF=0.5), all requests from these machines will be directed to the cloud services. To solve this problem, Turing Test is sent to each machine while the request number matches the RC value(s). Failing the test, causing a reduction of the UTF value, which in turn will prevent from getting the service directly. By the way, RC is not one value, it depends on the CRPS, if CRPS = 5, RC will be 5 values from the interval [1,(5*60)].

UTF value is checked again when the user does not violate the CRPS condition, to prevent the bad users (e.g., black listers such as botnet) of getting the service directly.

If the user is new visitor to the system, its information is stored in the Rate-Limit table of the DB, and its request is directed to the cloud services.

3.2 Implementation

In this section, we illustrate who is responsible for implementing the proposed EDoS attack mitigation technique and how it is implemented. The proposed mitigation technique needs to check the auto-scaling condition thresholds to start running the scheme. Auto-scaling feature is managed by the cloud provider. Therefore, cloud provider is recommended to implement (i.e., offer) the EDoS mitigation technique as a security feature for service providers.

To build the cloud, we use two servers, management server and compute server. The management server is used to manage the cloud services. Citix CloudPlatform 3.0.5 is the software that is installed on the management server. Citix CloudPlat-

form 3.0.5 is running on CentOS 6.2. Compute server or the hypervisor contains the cloud services in the form of VMs. Citrix XenServer 6.0.2 is the software that represents the hypervisor.

Citrix NetScaler VPX 10.e is used to implement the Load Balancer. Its installed as a VM running on Citrix XenServer 6.0.2 of a machine. Least connection algorithm is used by the load balancer to distribute the traffic flow, as shown in Fig. 3.4. 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 used to implement the vFirewall and VMInvestigator. PHP scripts and Cron jobs are used to implement the rate limit technique of the VMInvestigator. Cron job is a linux command that runs tasks (e.g, commands, scripts) at specific date and time. It is used by the VMInvestigator to periodically execute PHP scripts such as the one that is responsible for reset the RequestCount value of the users, the one that is checks users' UTF, and the one that picks the RC values. Mysql is used to implement the DB.

In our implementation, we add a component called VMObserver to check the auto-scaling condition thresholds of the cloud services. Checking the thresholds work is done by the Citrix NetScaler to trigger the auto-scaling, whereas, it is done by the VMObserevr to run the EDoS attack mitigation technique. So, if Citrix NetScaler checks the thresholds for both works, VMObserver becomes useless and being removed, as we shown in Fig. 3.1. Also, due to inability to control the Citrix NetScaler, VMInvestigator takes care of adding the rule at vFirewall that directs

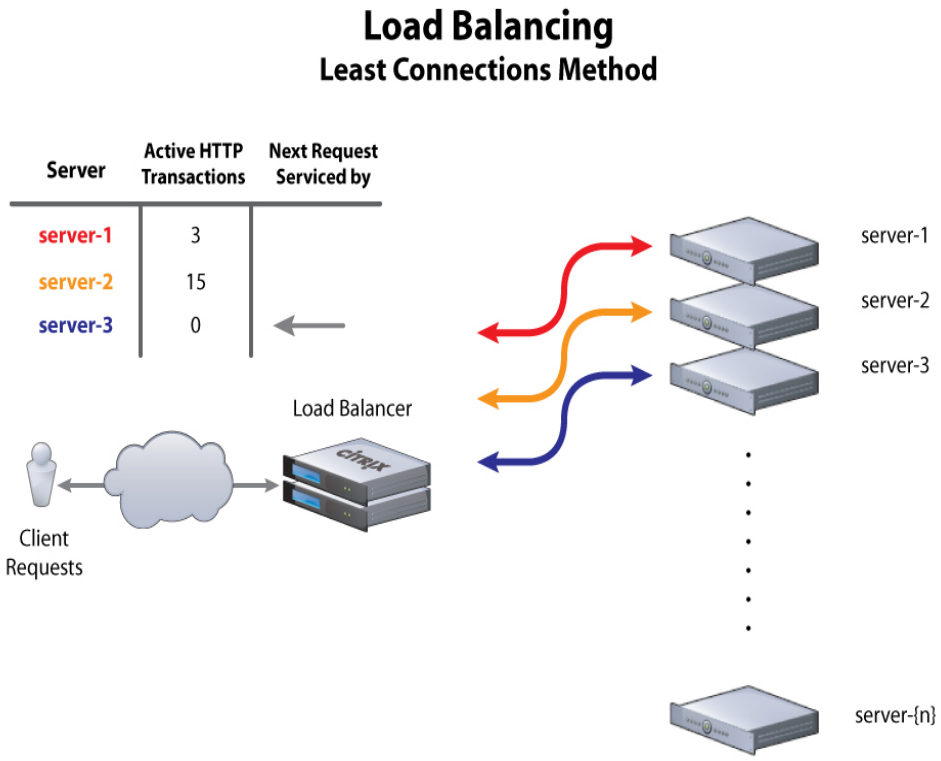

Figure 3.4: Load balancing method [5].

all the requests to the VMInvestigator when auto-scaling threshold is crossed.

VMObserver conducts a periodic check of the auto-scaling conditions thresholds, as stipulated at initialization time by the service provider. VMObserver is instantiated within each VM of the cloud services and within the VMInvestigator.

The VMObserver, initiated within the VM of cloud services, performs a self monitoring of the auto-scaling conditions thresholds, and sending the statistics to the DB, as illustrated in Fig. 3.5. The VMObserver, initiated within the VMInvestigator, checks the auto-scaling conditions thresholds from the DB. If they are crossed, the mitigation technique starts running.

We implement the VMObserver using a PHP script. This script is executed periodically using a Cron job. Threshold and polling interval of the VMObserver is set based on the auto-scaling configuration of the service provider.

EDoS attack is implemented using Jmeter. Eight VMs with different IP addresses are used to initiate the attack.

Cloud services are implemented using Apache web server. Apache is initiated automatically during the VM startup.

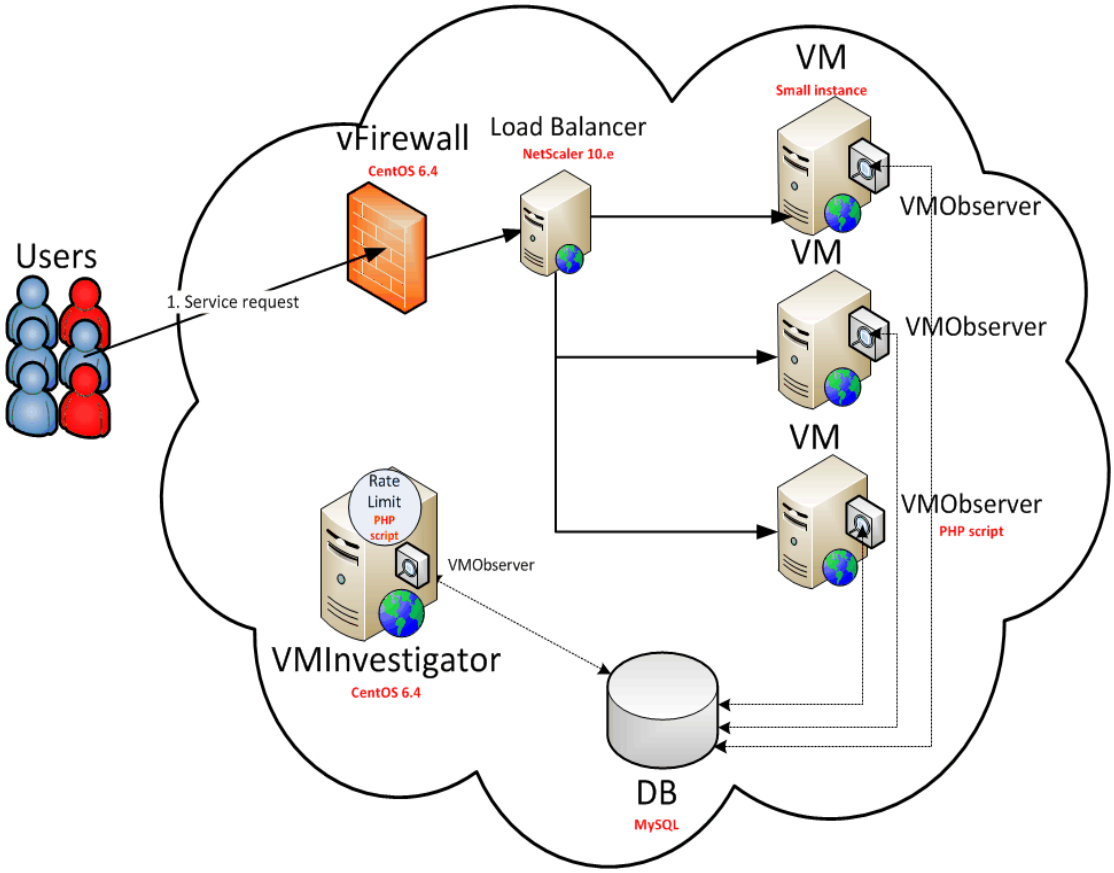

Figure 3.5: The implemented architecture of EDoS mitigation technique.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we illustrate the experimental setup of the proposed mitigation technique against EDoS attack. Also, we present and discuss the experimental results obtained from implementing the EDoS attack against the cloud services with/without the proposed mitigation technique.

4.1 Experiments Setup

Experiments have been conducted to evaluate the proposed mitigation technique and demonstrate the effect of the EDoS attack against cloud services. Table 4.1 shows the parameters that have been used in the experiments.

CPU usage metric has been used to scale up or scale down the cloud services. The upper and lower thresholds of auto-scaling were set to 80% and 30% respectively. The average CPU usage metric is checked every minute according to the polling interval parameter. The duration of the scale up or scale down is set to 1 minute. The aforementioned parameters except the auto-scaling lower threshold

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%

Table 4.1: Experiment parameters.

parameter are set based upon [43]. If the average CPU usage of the VM instances that represents the cloud services is more than 80 percent for 1 minute, one VM instance will be added. In contrast, if the average CPU usage of the VM instances is less than 30 percent for 1 minute, one VM instance will be terminated.

The minimum and maximum number of VM instances are set to 3 and 10 respectively, as it is the default value of Citrix CloudPLatform. The compute offering or VM instance type is set to small. Such a VM instance has 1 CPU core with 500 MHz and 512 MB memory. Its price is assumed to be $0.03, based upon the Wowrack pricing policy [44]. The VM instance Operating System (OS) is CentOS 5.6 (64-bit), since it is the default template of Citrix CloudPlatform. We installed Apache version 2.2.3 in each VM instance to run the web application

(cloud service). Based on the study of Google trace [45], we have generated a fixed load that consumes about 40% of the CPU usage of the cloud services using Jmeter. In this case, Scale down does not add any value to the detector system and is therefore not triggered. This is because the EDoS attack exploits the scale up feature of the cloud.

VMInvestigator updates the RC value(s) every five minutes. Also, it checks the users' UTF every minute.

4.2 Results

In this section, we present the experimental results of four different scenarios to evaluate the proposed mitigation technique against EDoS attacks at service provider's end. The attackers are assumed to be first time visitors to the system. We choose the results once a steady state is achieved during the experiments. Each experiment is repeated ten times.

4.2.1 Without Auto-scaling and Mitigation Technique Scenario

In this scenario, we disable the auto-scaling and mitigation technique. Fig. 4.1 illustrates the effect of the EDoS attack against the CPU usage of the cloud services. It is noted that, the increase in the attack rate increases the CPU usage until it reaches a maximum value at 400 request(s) per second (rps) of the attack rate. Eventually, the cloud services are made unavailable for all the users.

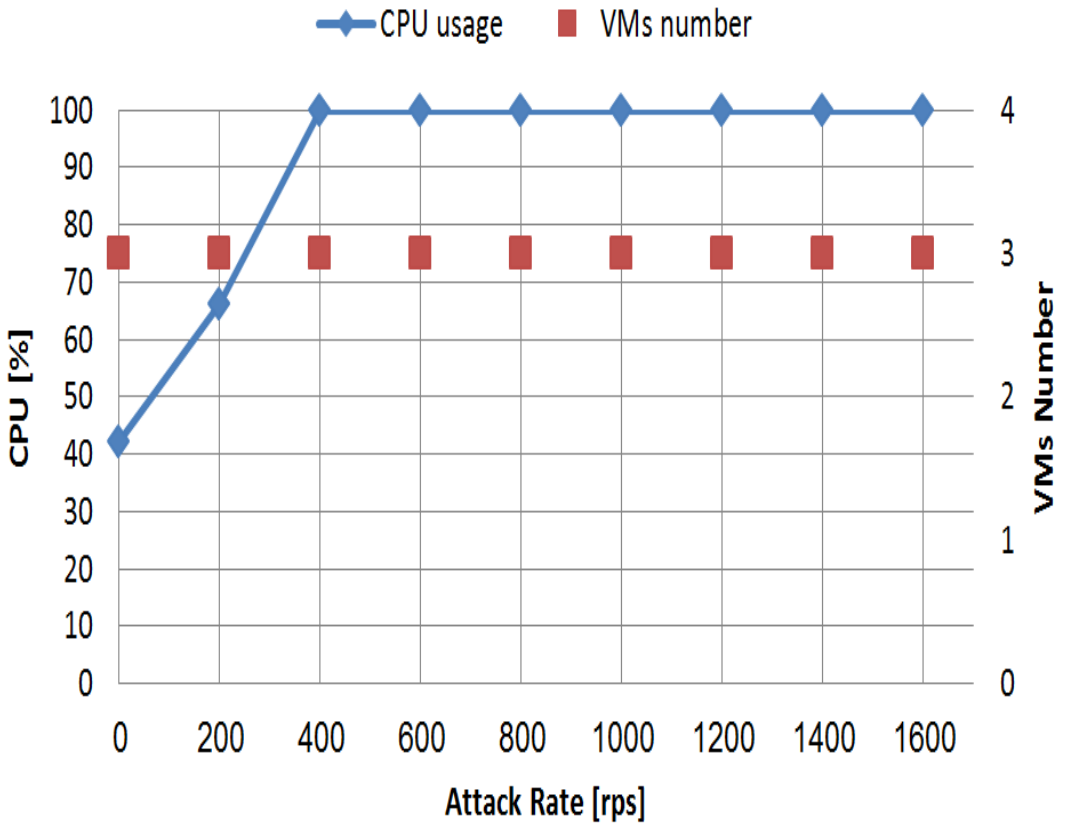

Figure 4.1: EDoS attack effect against CPU usage of the cloud services, without auto-scaling and mitigation technique

Fig. 4.2 shows the effect of the EDoS attack against the memory allocation of the cloud services. The memory allocation is about 130 and 136 MB at 0 and 200 respectively of the attack rate. A significant increase of the memory allocation is noticed at 400 rps of the attack rate, since the CPU usage of the cloud services is fully utilized. After that, increasing the attack rate is accompanied by a minor increase in the memory allocation.

Fig 4.3 illustrates the effect of the EDoS attack on the throughput. NetScaler UI is used to obtain the throughput within the cloud. It is noted that the throughput is exponential to the attack rate. At 400 rps of the attack rate, the system is unable to handle more requests because of the maximum utilization of the cloud resources (i.e., CPU usage).

4.2.2 With Auto-scaling but without Mitigation Technique Scenario

In this scenario, the auto-scaling is enabled but the mitigation technique is disabled. Fig. 4.4 shows the effect of the EDoS attack against the CPU usage of the cloud services. The number of VMs are scaled up from 3 to 4 at 400 rps of the attack rate. It means that the CPU usage goes beyond 80 percent when the attack rate is increased from 200 to 400 rps. Similarly, at 600 rps of the attack rate, the number of VMs are scaled up to 6. The VMs continue to scale up to accommodate the demand until they reach the maximum number (i.e., 10 VMs) at 1200 rps of the attack rate. The CPU usage fluctuates between 60 percent and

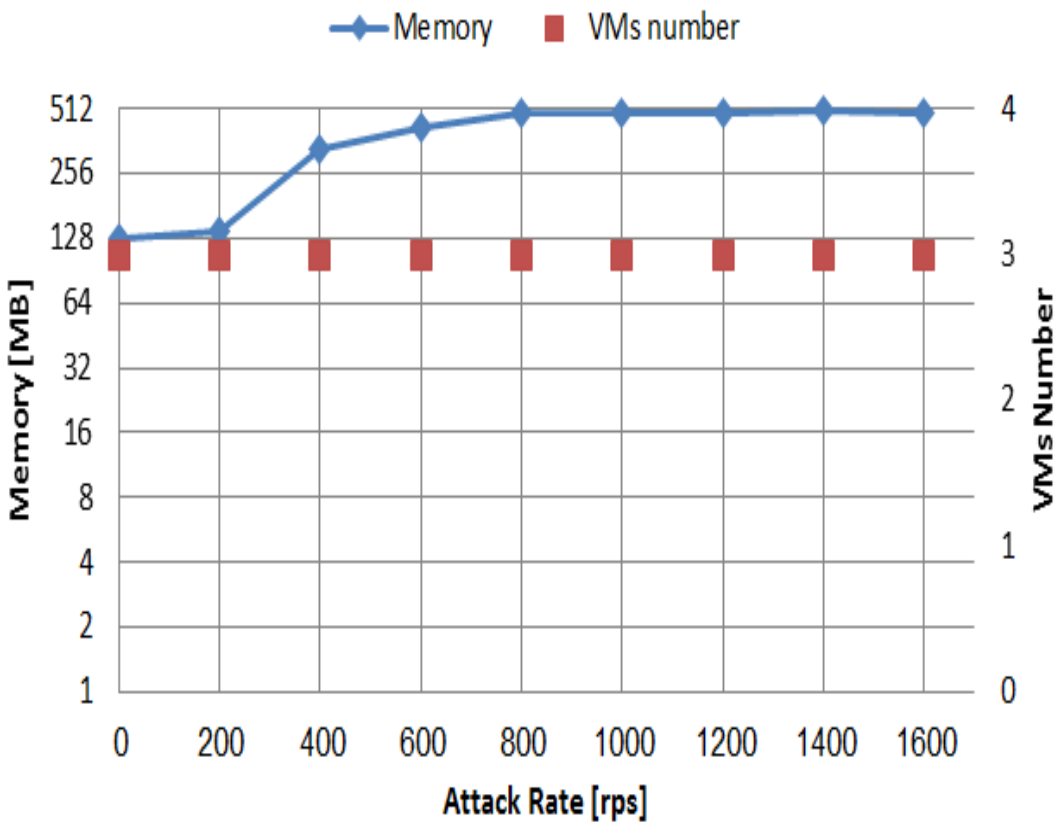

Figure 4.2: EDoS attack effect against memory allocation of the cloud services, without auto-scaling and mitigation technique

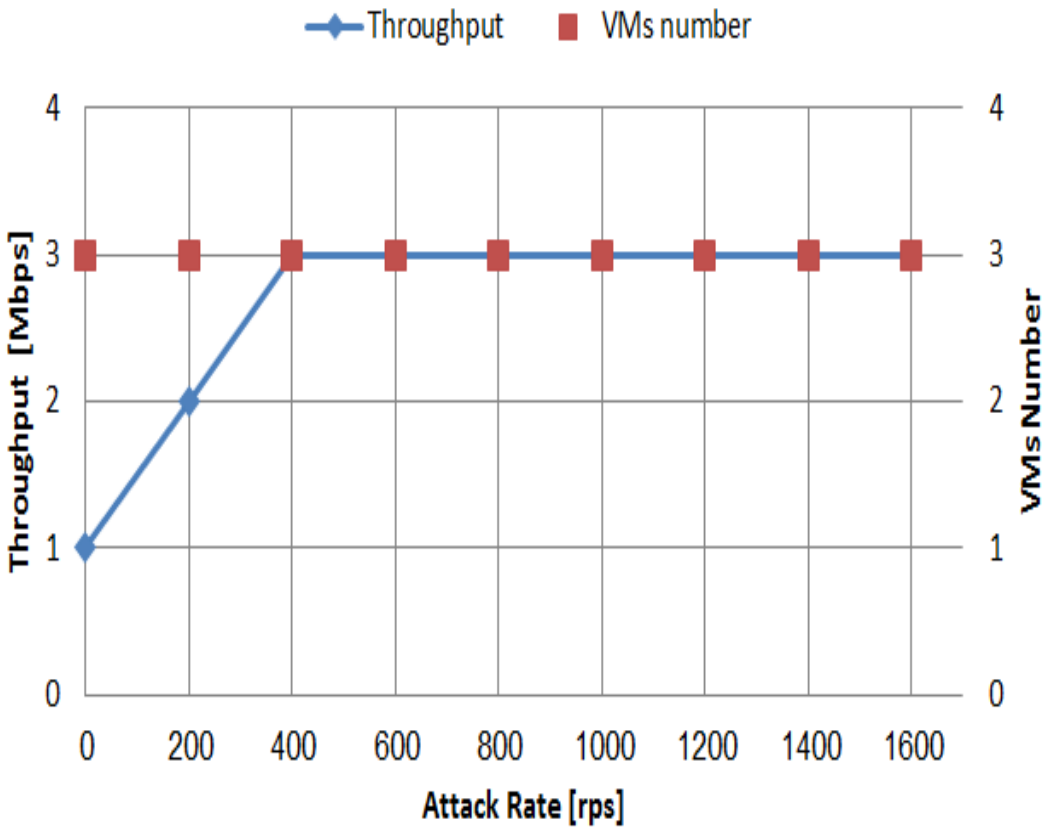

Figure 4.3: EDoS attack effect against throughput of the cloud services, without auto-scaling and mitigation technique

80 percent due to the auto-scaling before it reaches the peak at 1400 rps of the attack rate.

EDoS attack effect on the memory allocation is illustrated in Fig. 4.5. Memory utilization fluctuates around 129 MB until the VM instances reaches its peak at 1200 of the attack rate. EDoS effect on the throughput is clarified in Fig. 4.6. It is noted that throughput grows when the VM instances grow, before the former reaches its peak at 1400 rps of the attack rate.

4.2.3 With Auto-scaling and Mitigation Technique Scenario

In this scenario, both the auto-scaling and mitigation technique are enabled. Fig. 4.7 shows the effect of the EDoS attack against the CPU usage of the cloud services. It may be noted that the CPU usage fluctuated around 40 percent which represents the legitimate usage. Also, the auto-scaling does not take place and the number of VMs remain constant. These results indicate the effectiveness of the proposed approach in mitigating the EDoS attack.

Fig. 4.9 and Fig. 4.8 shows the effect of the EDoS attack on the Memory and throughput respectively. It is noted that the technique succeeds to mitigate the EDoS attack, as both the memory utilization and throughput remain the same.

The cost associated with the cloud services is illustrated in Fig. 4.10. The base case of the VM instances number is three, as stated in Table 4.1. Each VM instance costs $0.03 per hour. The results show that, implementing the mitiga-

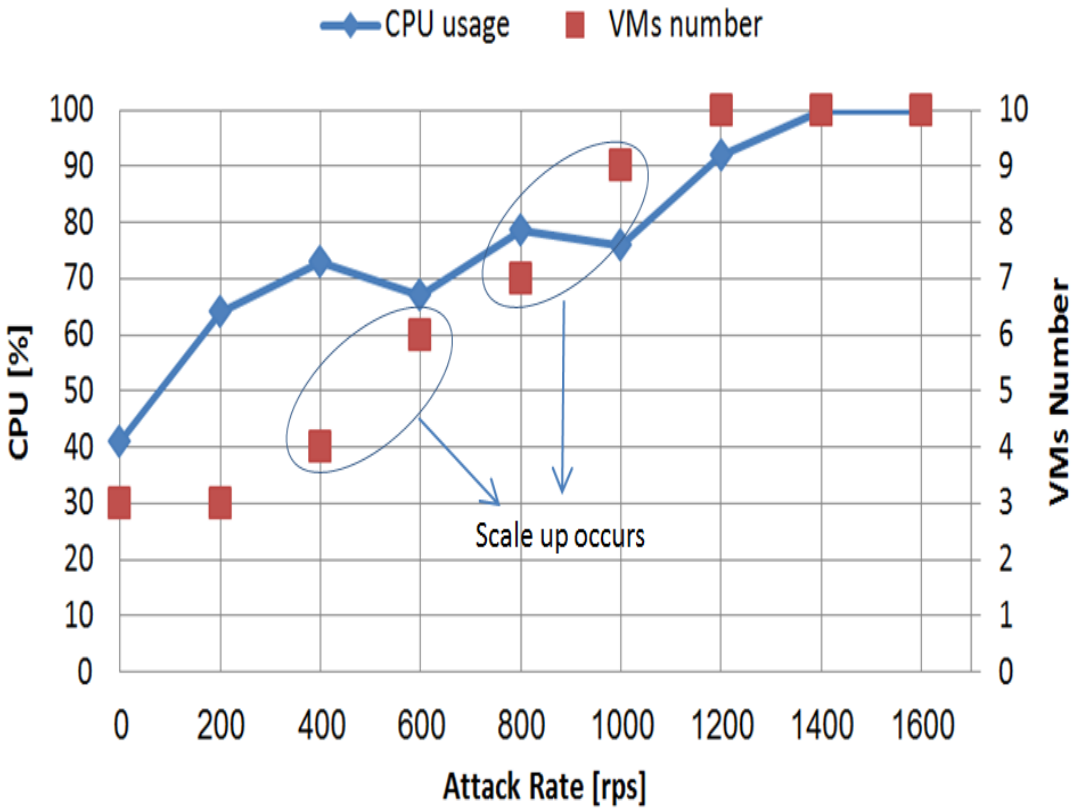

Figure 4.4: EDoS attack effect against CPU usage of the cloud services, with auto-scaling but without mitigation technique

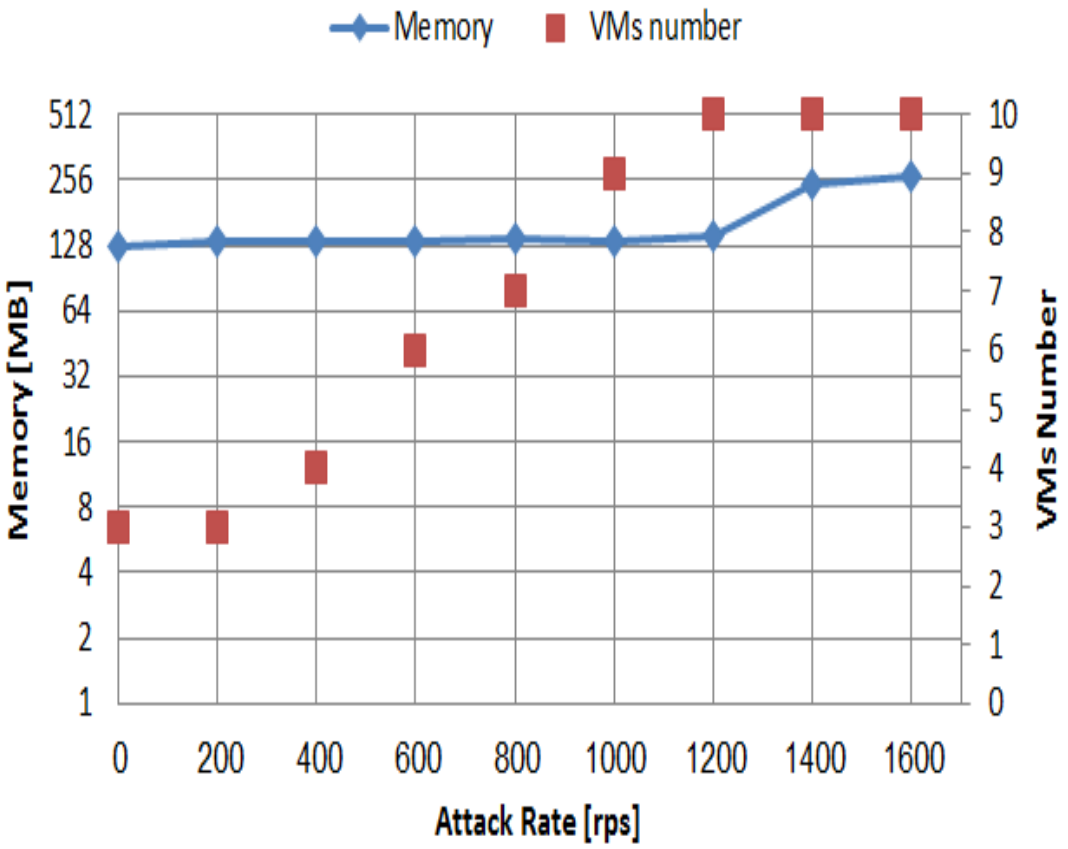

Figure 4.5: EDoS attack effect against CPU usage of the cloud services, with auto-scaling but without mitigation technique

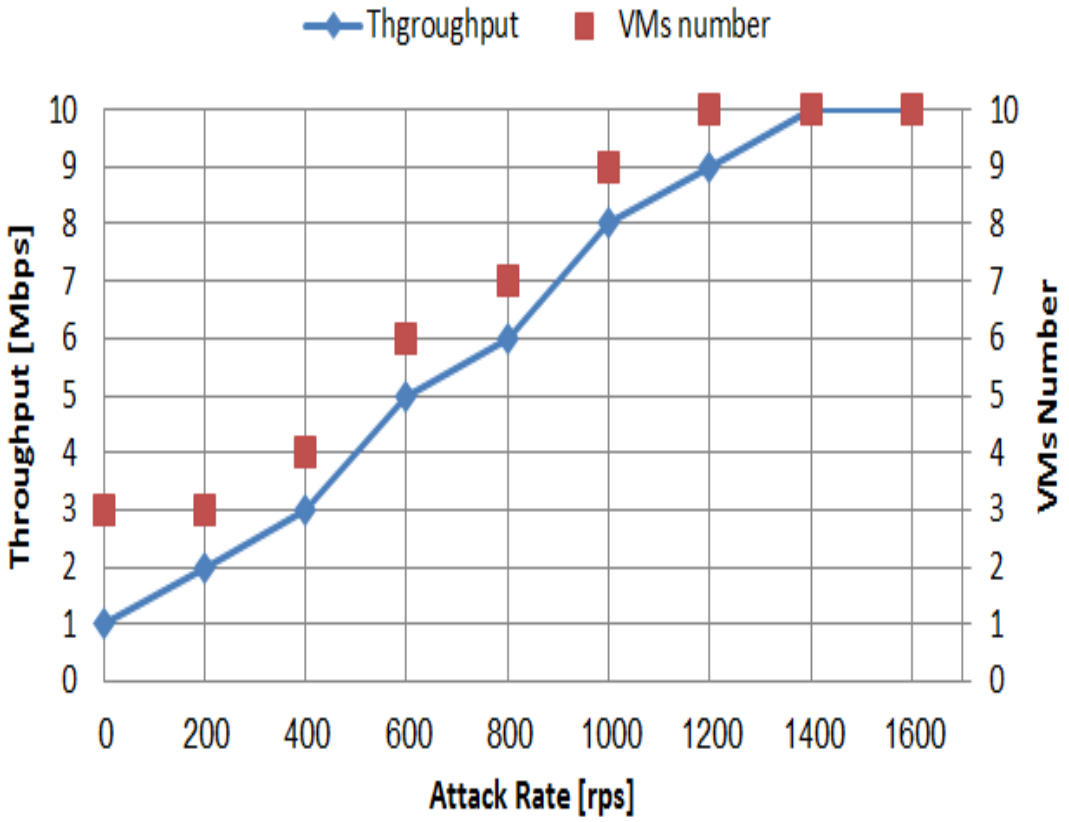

Figure 4.6: EDoS attack effect against CPU usage of the cloud services, with auto-scaling but without mitigation technique

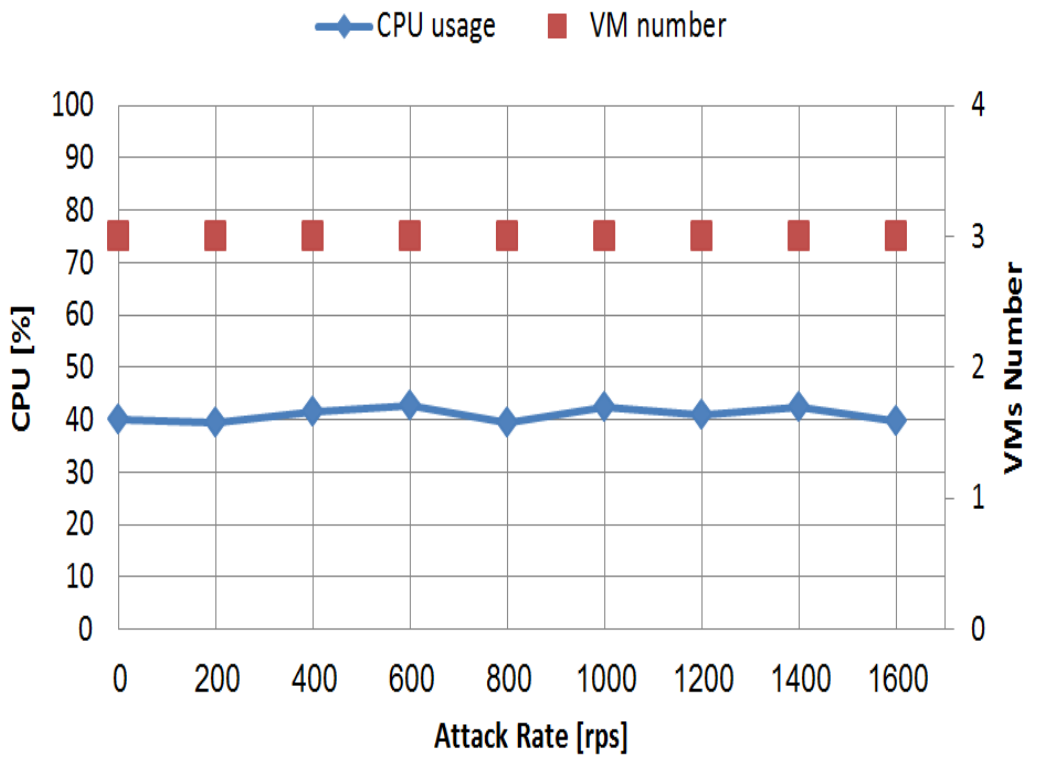

Figure 4.7: EDoS attack effect against CPU usage of the cloud services, with auto-scaling and mitigation technique

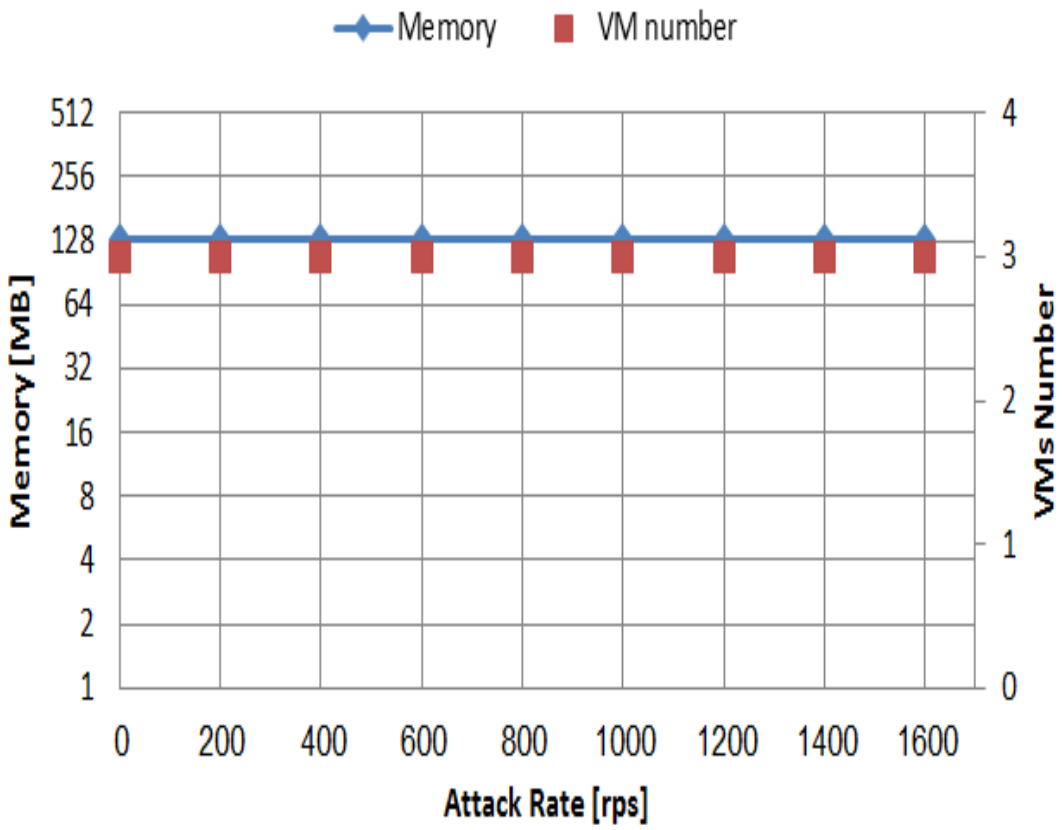

Figure 4.8: EDoS attack effect against CPU usage of the cloud services, with auto-scaling and mitigation technique

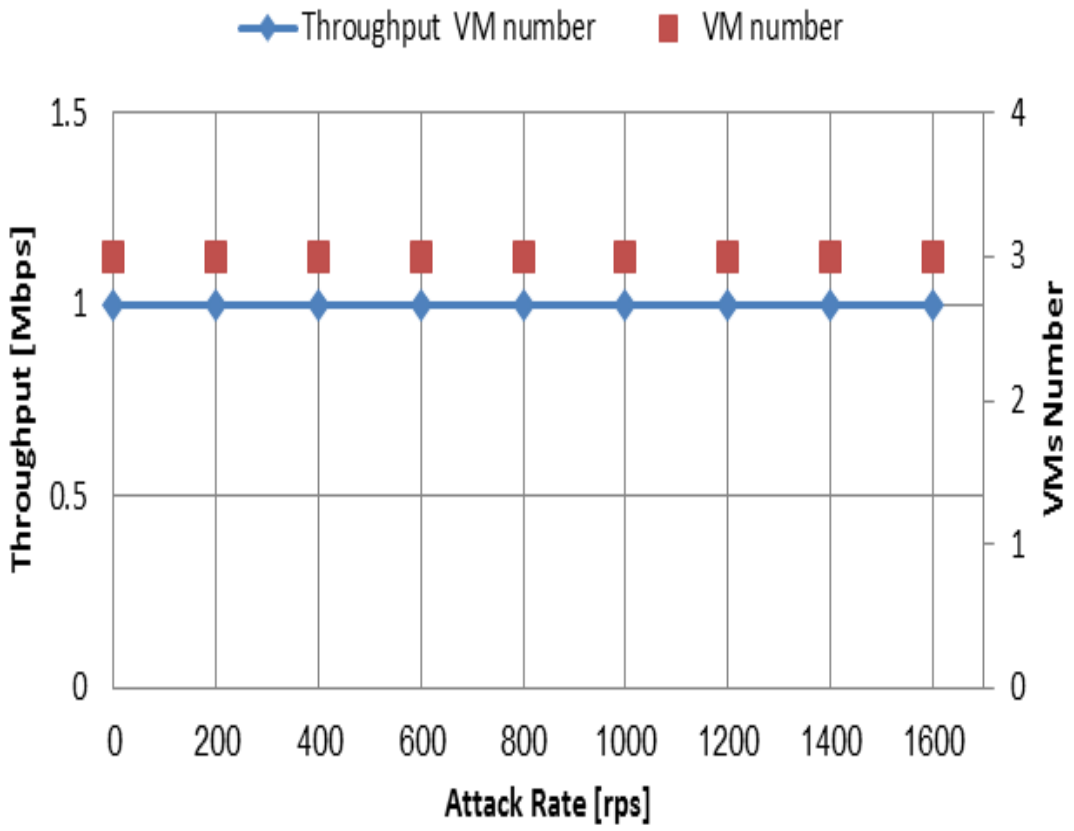

Figure 4.9: EDoS attack effect against CPU usage of the cloud services, with auto-scaling and mitigation technique

tion technique requires an additional cost as compared to the optimal case. The additional cost of the mitigation technique is a result of the VMInvestigator and vFirewall costs (i.e., 2 VM instances cost). However, the cost increases dramatically and then levels out when the VM instances reach the peak (i.e., 10 VM instances) at 1200 rps of the attack rate, in the case of no mitigation technique. It is beyond any doubt that the cost will be continued to increase unless the VM instances that represents the cloud services are not limited by a maximum value.

Fig. 4.11 shows the results of the response time for services running in the cloud. We used an integrated tool with Firefox named Firebug [46], to measure the response time of the request. The request, that measures the response time, accesses the Load balancer directly in the optimal case. In the other cases, it goes through the vFirewall then to the load balancer. It may be noted that the response time with mitigation technique in action roughly stays the same, and its value is close to the optimal value. However, the response time without mitigation technique is stable especially when the scale up of the VM instances still occurs. After that, the response time increases considerably, because the VM instances that represents the cloud services reach their peak. The response time is about 93 ms when the mitigation technique is in action.

4.2.4 Smart Attacker Scenario

In this scenario, we assume that the attacker is new to the system, and knows the CRPS value and subverts the system based on it. The idea of this scenario is

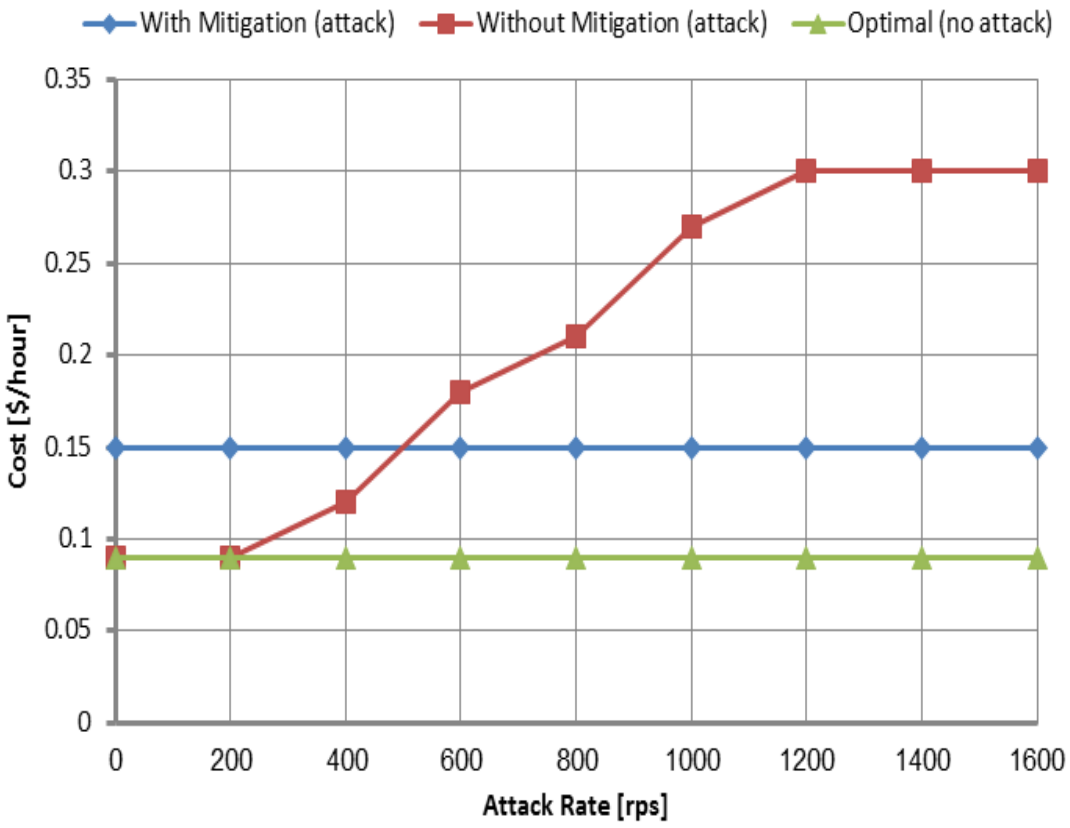

Figure 4.10: EDoS attack effect on the cost

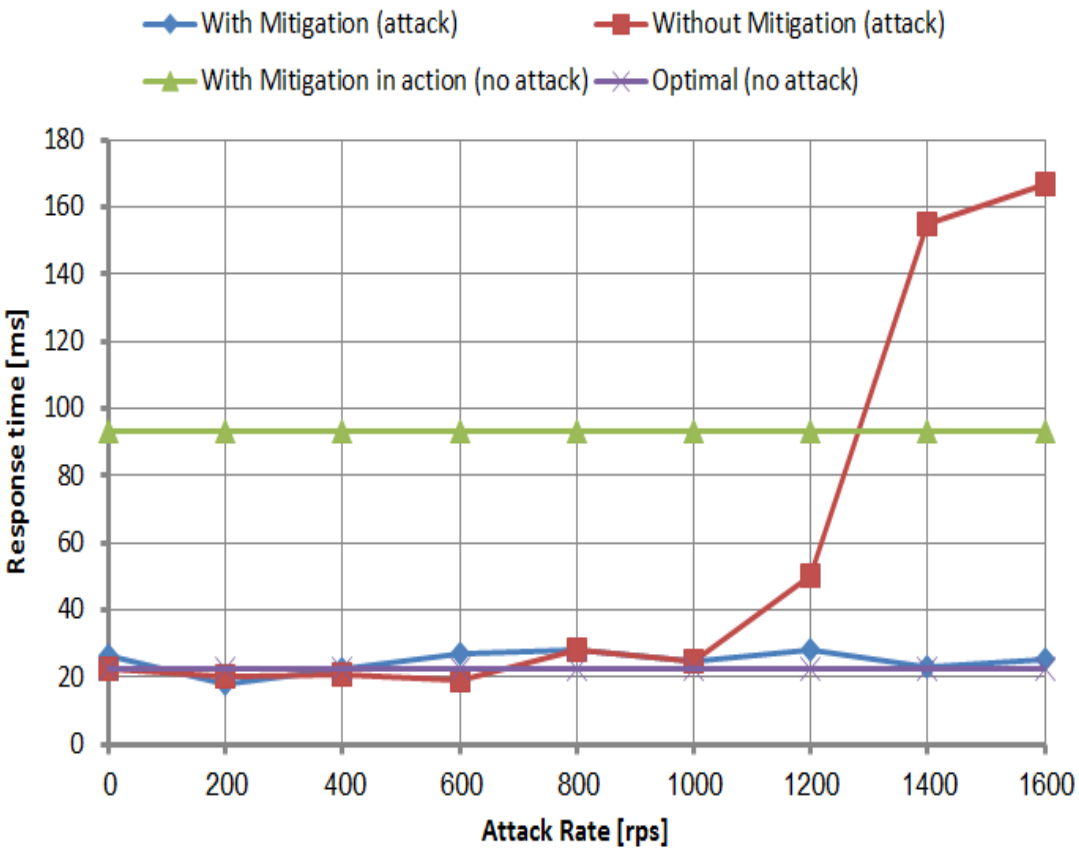

Figure 4.11: EDoS attack effect on the response time

similar to the Fraudulent Resource Consumption (FRC) attack [47]. For example, an attacker is sending 1 rps or less when the CRPS is set to 1, which mean the system allows only one request per second. The configuration of this scenario is set based on Table 4.1.

In Fig. 4.12, 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 VMInvestigator). The results show how many malicious requests accessing the cloud services per minute, and when the proposed mitigation technique succeeds to stop them. It is noted that, the illegitimate requests (false negatives) for the UTF decrement of 0.1 for minute 1 is around 36, and 0 at minute 2. So, for minute 1, the technique was able to stop about 24 requests out of a total of 60 requests from getting through. Given the setup values of that experiment, there could be at most 1 request out of 60 requests that matched 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 is > 0.25 (at 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, about 58 and 38 of the malicious requests passed to the cloud at minute 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 not answering the Turing Test. For

the UTF decrement of 0.02, the illegitimate requests fluctuates 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 matching the selected RC value. The other 2 requests are stopped due to the Turing Test that is sent because of breaching the CRPS value where some requests are delayed and come at the time frame of the next requests. The mitigation technique succeeds in stopping these requests at minute 6.

Fig. 4.13 shows the corresponding UTF value of the requester for previous scenario (CRPS=1). It is noted that the requester's UTF falls substantially for UTF decrement 0.1 and 0.05. It goes from 0.5 to 0 at minute 2, for the UTF decrement of 0.1, whereas, it goes from 0.5 to 0.38 and then from 0.38 to 0 at minute 2 and 3 respectively. The rquester's UTF falls gradually of the UTF decrement of 0.02. About 0.06 is deducted from the user's UTF During the minute, until it reaches 0 at minute 6. As we mentioned early, The UTF deduction is a result of not answering the Turing test that is send due to the Random Check, or due to breaching the CRPS value.

Fig. 4.14 shows the result of the false negatives when CRPS is set to 5, and the attacker sends 5 rps. The RC values are 31, 62, 149, 225, and 263. It is noted that, about 292 and 148 malicious requests (false negatives) access the cloud services at minute 1 and 2 respectively, for UTF decrement of 0.02. At minute 1, the technique succeeds to stop 8 requests out of the 300 that are getting through the system. 5 of them are stopped due to the Turing Test that is send as a result of

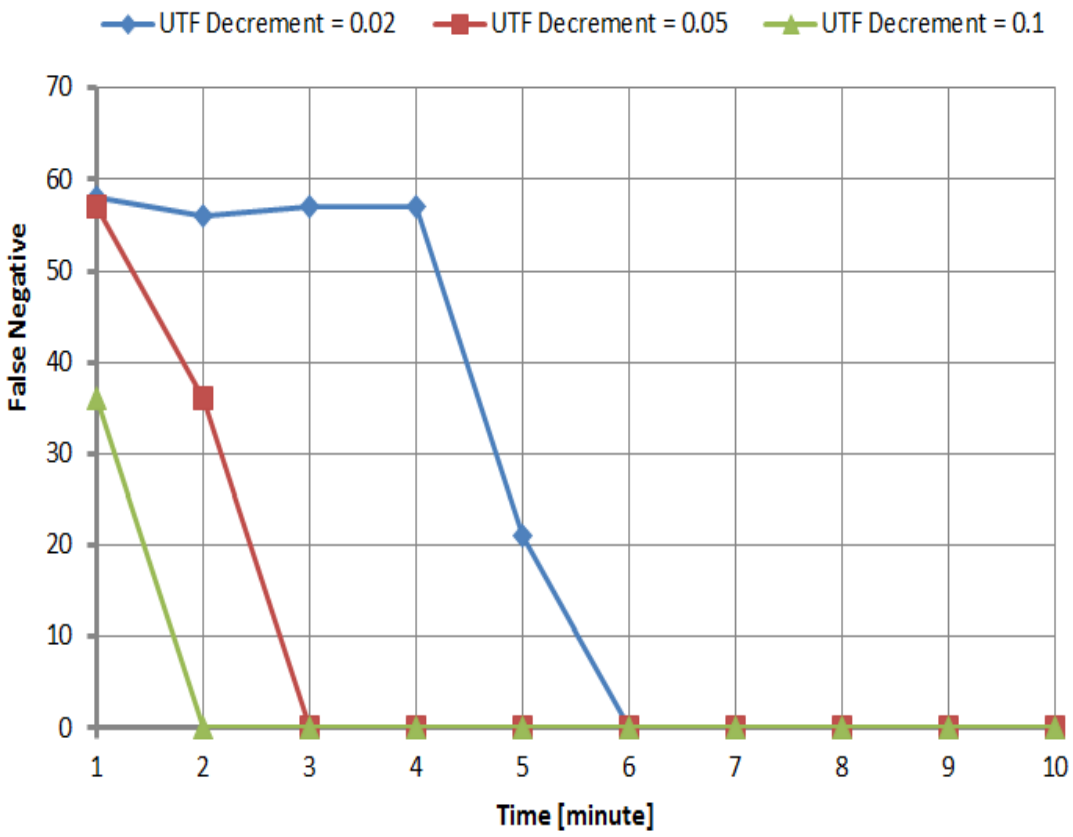

Figure 4.12: Plot shown the experimental results of the false negatives with time for CRPS = 1.

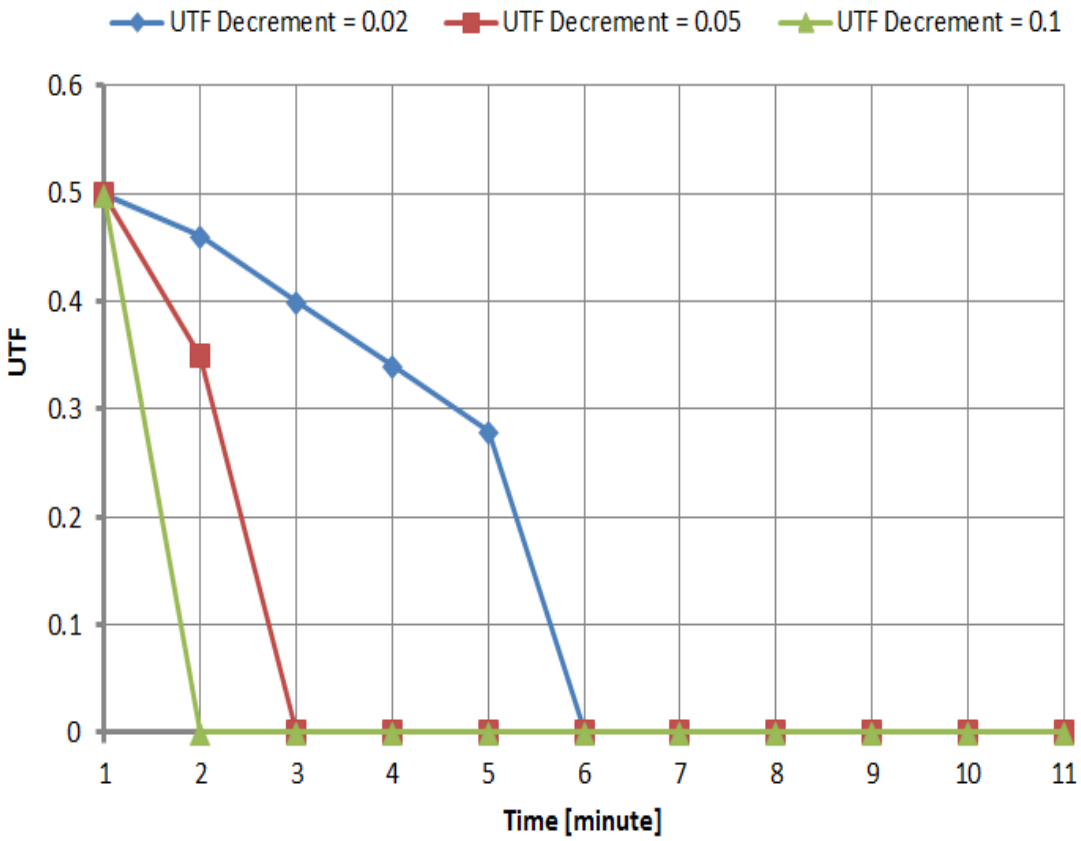

Figure 4.13: Plot shown the experimental results of the UTF value with time for CRPS = 1.

matching the selected RC values, whereas, the other are stopped due to Turing test that is sent as a result of violating the CRPS factor. At minute 2, 152 out of 300 are blocked by the technique. The degradation of the UTF value under 0.025 During this minute, results in detecting these large number of requests, as shown in Fig 4.15. The technique succeeds to stop these requests at minute 3. At minute one, there are 219 and 149 malicious requests accessing the cloud for UTF decrement of 0.05 and 0.1 respectively. At minute two, the proposed mitigation technique get rid of these malicious requests.

Its expected that the attacker can send less number of requests than what are permitted to subvert the system. Fig 4.16 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 decrement of 0.02, 236 illegitimate requests out of 240 access the cloud at the minutes 1, 2, 3, and 4. It is clear that the four requests are detected because of the Turing Test that sent to the user as a result of matching the selected 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 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 does not provide any benefit in such scenario. The illegitimate requests for minute 5 are about 30, and 0 for minute 6. For UTF decrement of 0.05, there are 236 and 61 of the malicious requests accessing the cloud at minutes 1 and 2 respectively. The system stops these requests at minute 3. The illegitimate requests for the UTF decrement of

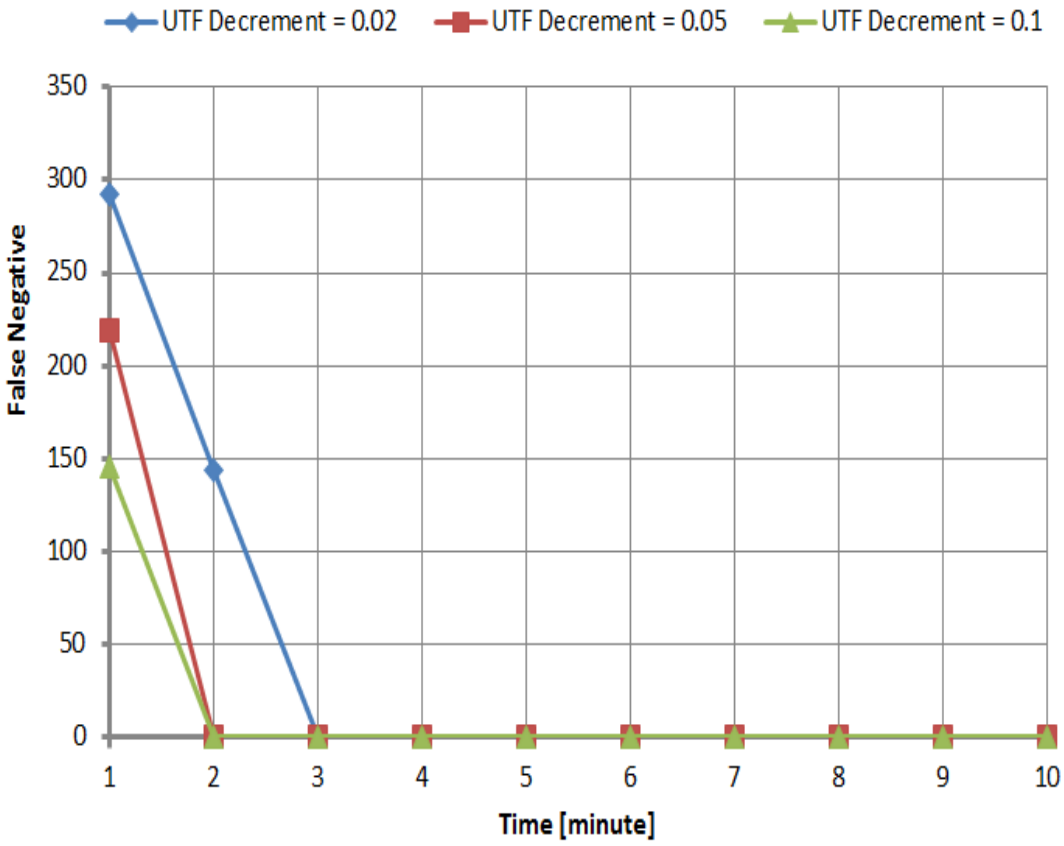

Figure 4.14: Plot shown the experimental results of the false negatives with time for CRPS = 5.

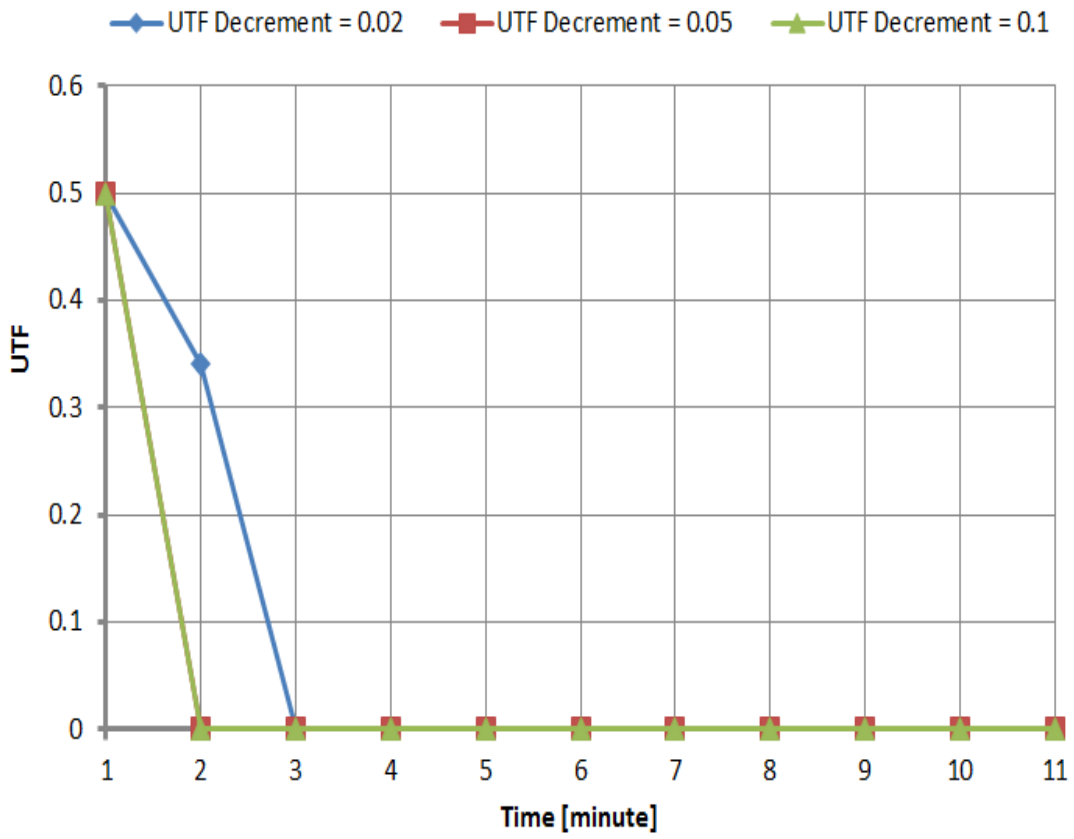

Figure 4.15: Plot shown the experimental results of the UTF value with time for CRPS = 5.

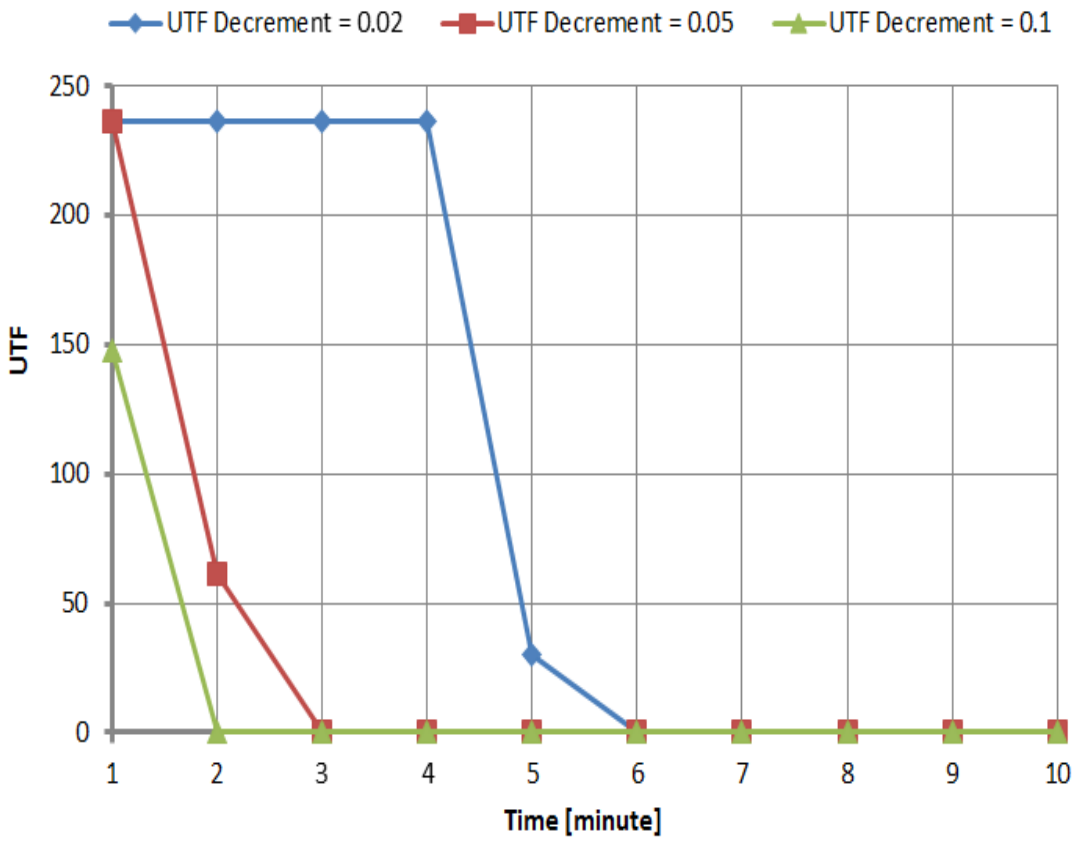

Figure 4.16: Plot shown the experimental results of the false negatives with time, when the attacker sends less requests than allowed by the system for CRPS = 5.

0.1, for minute 1 is 149, and 0 for minute 2.

Fig 4.17 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 deducted by 0.08 during each of the first three minutes, for the UTF decrement of 0.02. During the 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, 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.

Fig 4.18 shows the result of false negatives, when the attacker sends more requests than allowed by the system. In this scenarion, we set the CRPS to 5, and the attacker sends 6 rps. It is noted that, the mitigation technique succeeds to stop the malicious requests during the first minute, for all values of the UTF decrement. About 53, 30, and 15 malicious requests accessing the cloud for UTF decrement of 0.02, 0.05, and 0.1 respectively. Breaching the CRPS is undoubtedly the main reason of detecting these requests more quickly. The corresponding UTF of this scenario is demonstrated in Fig 4.19. At minute 2, the requester's UTF is 0 for each value of the UTF decrement.

Fig 4.20 shows the result of the malicious requests when CRPS is set to 10, and the attacker sends 10 rps. RC values are 18, 81, 140, 214, 292, 357, 415, 446, 528, and 592. We can notice about 580, 291, and 139 malicious requests accessing the cloud during the first minute, for the UTF decrement of 0.02, 0.05, and 0.1

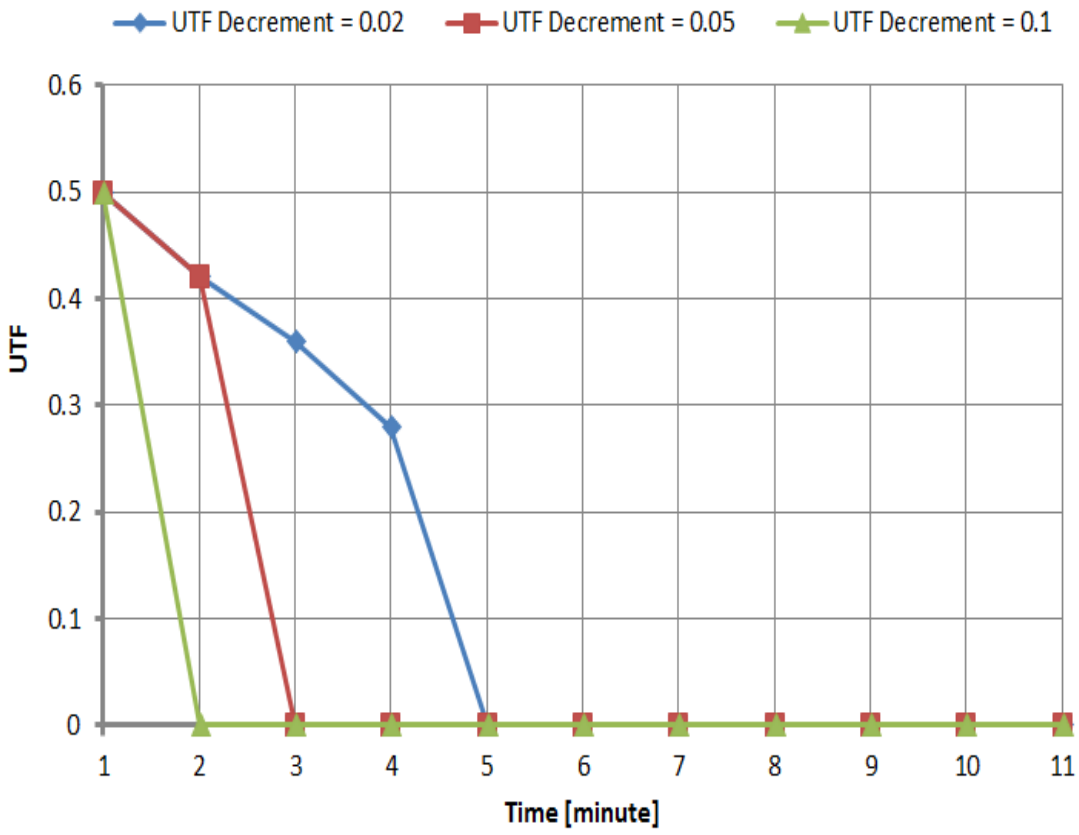

Figure 4.17: Plot shown the experimental results of the UTF value with time, when the attacker sends less requests than allowed by the system for CRPS = 5.

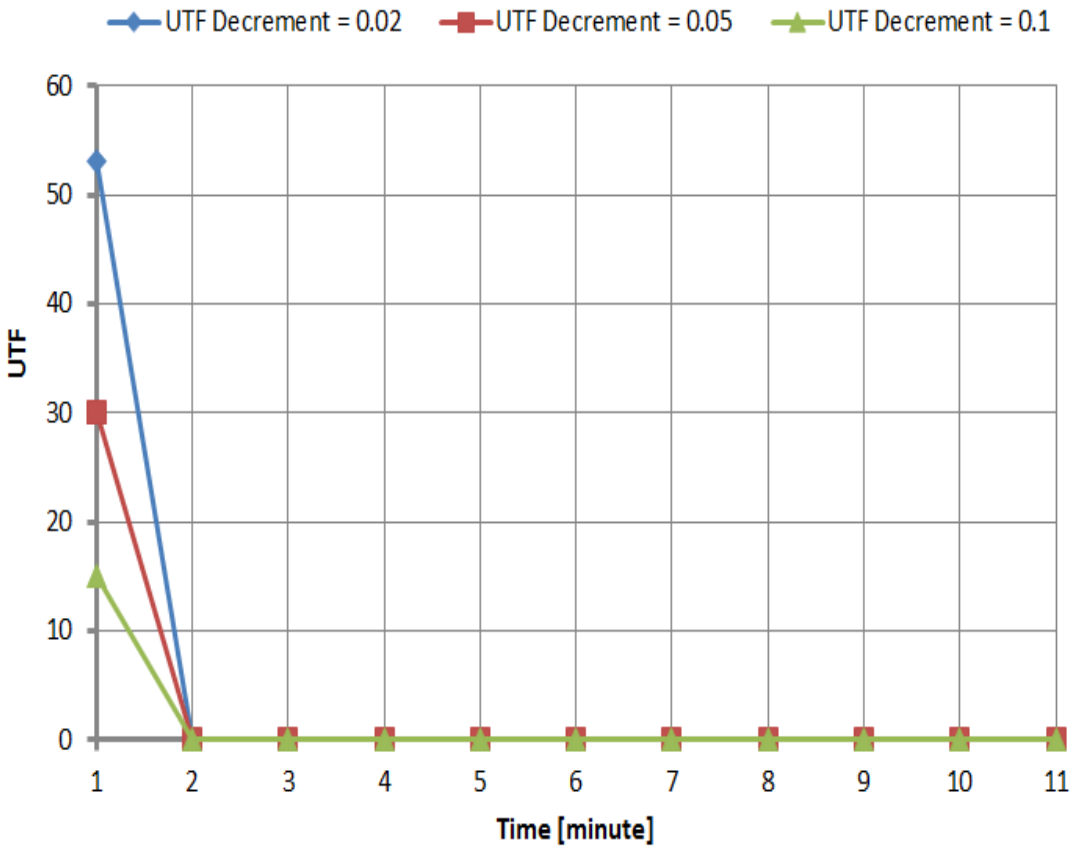

Figure 4.18: Plot shown the experimental results of the false negatives with time, when the attacker sends more requests than allowed by the system for CRPS = 5.

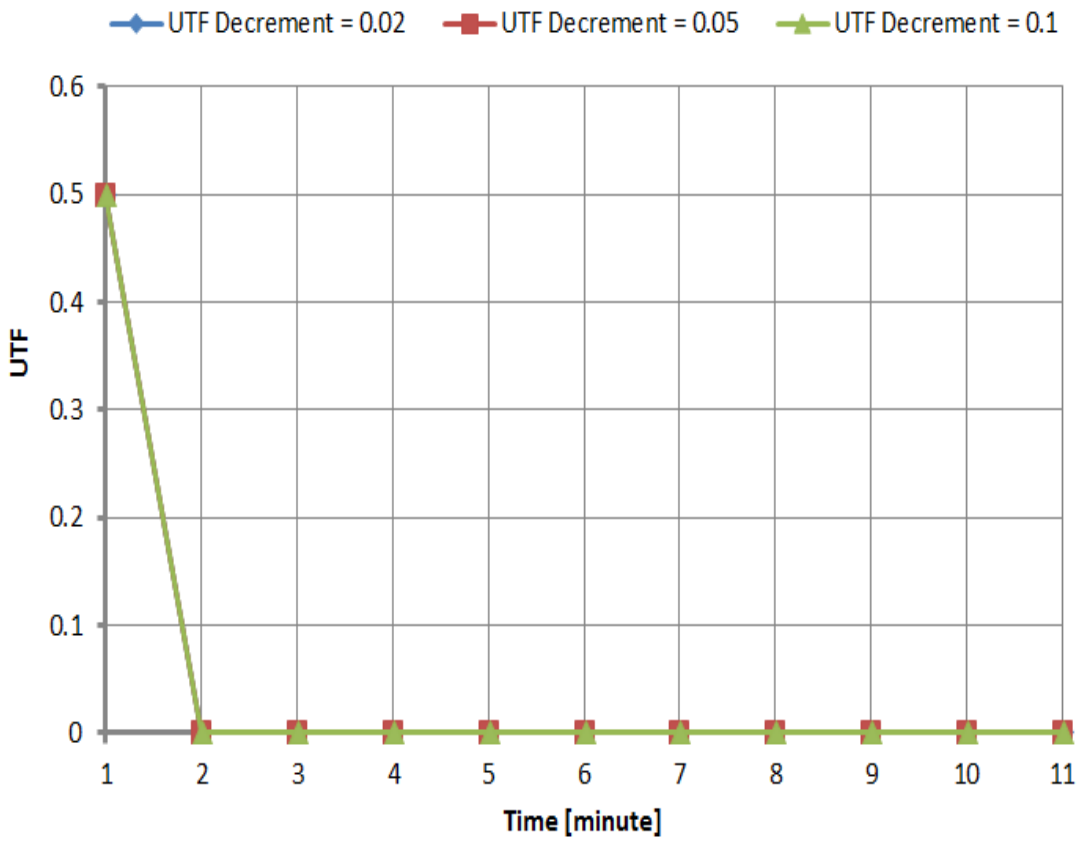

Figure 4.19: Plot shown the experimental results of the UTF value with time, when the attacker sends less requests than allowed by the system, CRPS = 5.

respectively. The corresponding UTF is illustrated in Fig 4.21. As we mentioned earlier, the requester's UTF value falls substantially due to the failure of answering the Turing Test that is sent due to matching the RC value or breaching the CRPS value.

In Fig 4.22, we scale the malicious users to 10, 50, and 100 based on the results obtained from Fig 4.12 for the UTF decrement of 0.02. During each of the first three minutes, about 570, 2850, and 5700 malicious requests accessing the cloud for malicious users of 10, 50, and 100 respectively. At minute 4, these malicious requests go down until they are stopped at minute 6. Moreover, we study the effect of these requests on the CPU usage of the cloud services, as shown in Fig. 4.23. We can notice that, the CPU usage of the cloud services fluctuates around 70, 60, and 45 percent for malicious users of 100, 50, and 10 respectively.

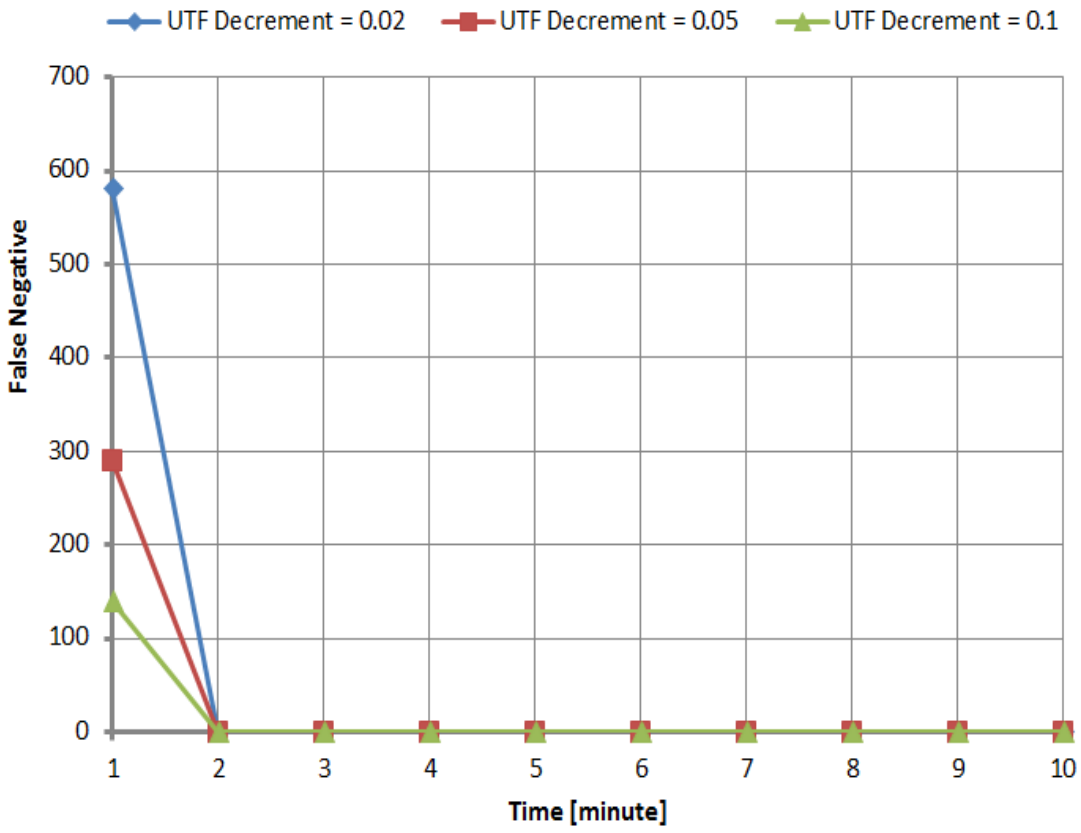

Figure 4.20: Plot shown the experimental results of the false negatives with time for CRPS = 10.

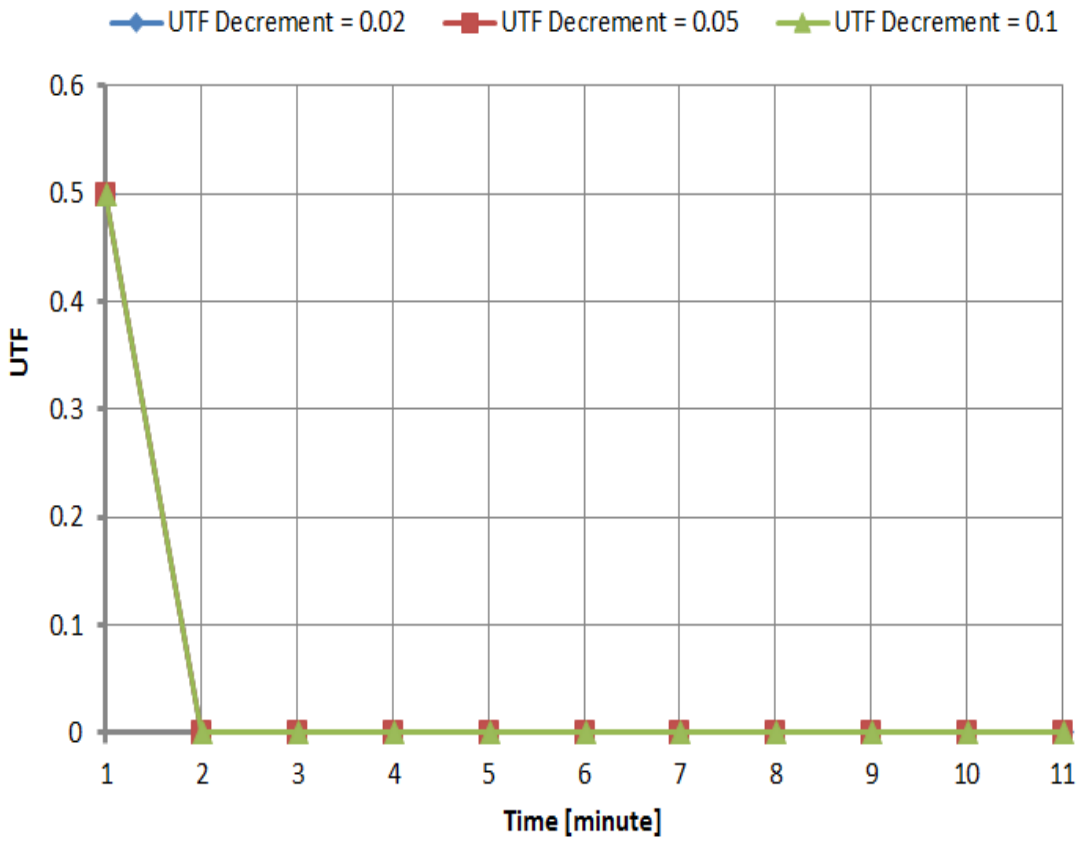

Figure 4.21: Plot shown the experimental results of the UTF value with time for CRPS = 10.

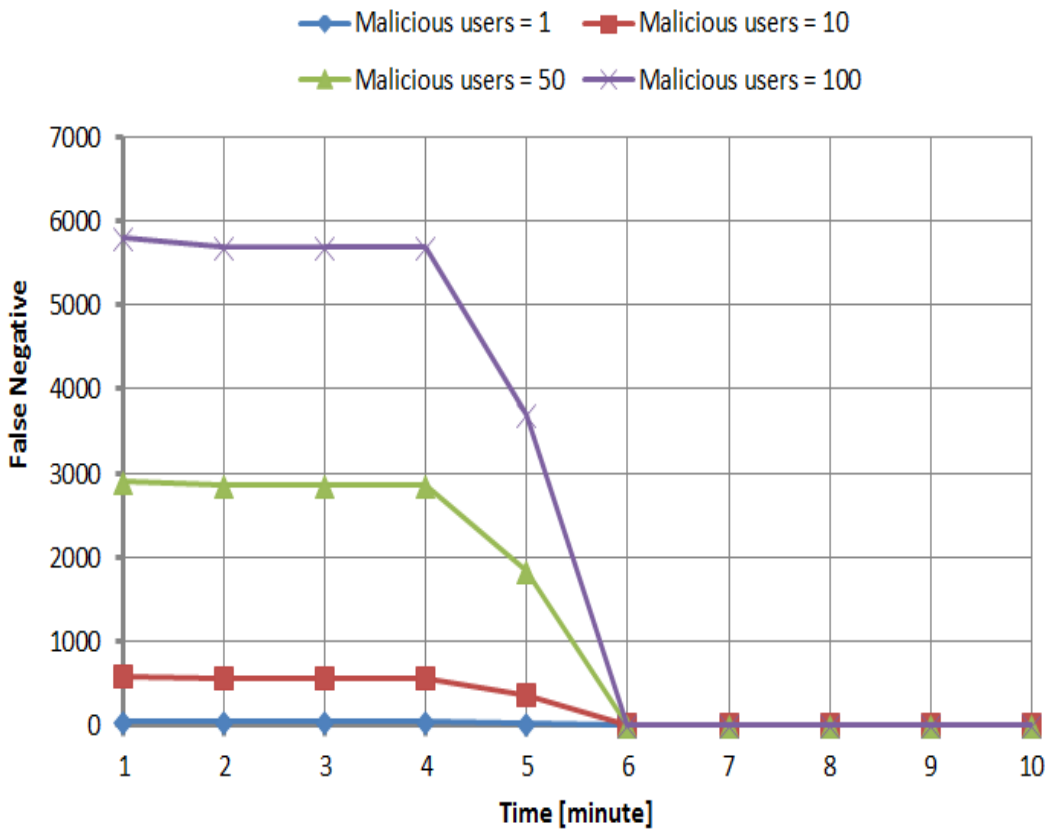

Figure 4.22: False negatives result of multiple users.

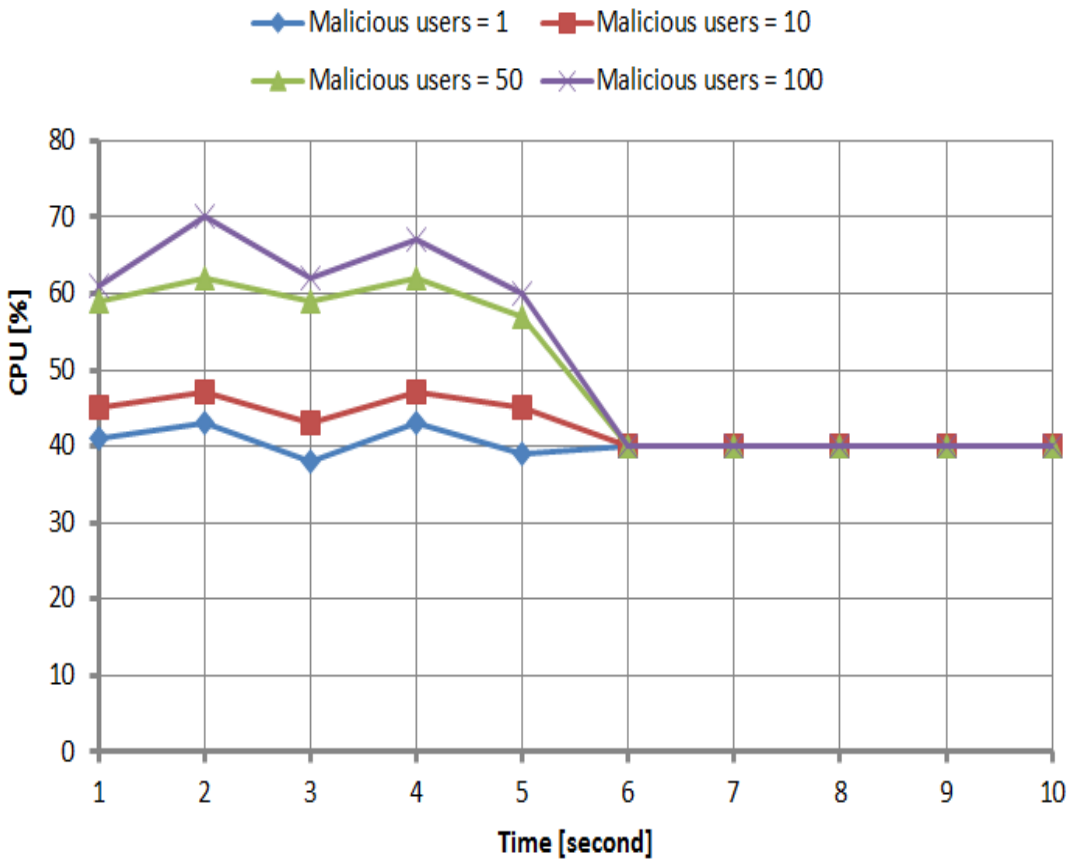

Figure 4.23: The effect of false negatives from multiple users on CPU usage.

CHAPTER 5

CONCLUSION AND FUTURE WORK

A novel reactive scheme has been proposed to provide fair control for cloud resource access by end-users, in the event of an EDoS attack. Through this scheme, a 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 scheme is comprised of several components, namely, vFirewall, Load balancer, DB, and VM Investigator, operating hand in hand, to control access to cloud services, for mitigating the effects of an EDoS attack. None of the requests that surpass the defined thresholds of these factors, are dropped, but rather, the users are provided with a Turing test to get an opportunity for accessing the cloud services. By analyzing the results obtained from the real-world experiment of the proposed scheme, it is evident that the effect of the EDoS attack mitigated when the access to the cloud resources is controlled

over a stretch of time. Moreover, the user perceived delays imposed through the scheme 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 network-level parameters are varied. In addition, we intend to implement diverse application scenarios, and study the schemes performance for each.

REFERENCES

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[2] I. D. Corporation, "New idc it cloud services survey: Top benefits and challenges," http://blogs.idc.com/ie/?p=730, 2009. [Online]. Available: http://blogs.idc.com/ie/?p=730

[3] M. Sqalli, F. Al-Haidari, and 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.

[4] M. Masood, Z. Anwar, S. Raza, and 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*, Dec 2013, pp. 37–42.

[5] Citrix, "Load balancing least connections," http://blogs.citrix.com/2010/09/02/load-balancing-least-connections/, 2010.

[6] SearchCloudComputing, "Cloud computing definition," http://searchcloudcomputing.techtarget.com/definition/cloud-computing, 2010.

[7] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[8] Amazon, "Amazon elastic compute cloud (amazon ec2)," http://aws.amazon.com/ec2/.

[9] Google, "Google app engine," https://developers.google.com/appengine/, 2009.

[10] Salesforce, "Salesforce.com," http://www.salesforce.com/, 2012.

[11] M. H. Sqalli, M. Al-Saeedi, F. Binbeshr, and M. Siddiqui, "Ucloud: A simulated hybrid cloud for a university environment," in *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on.* IEEE, 2012, pp. 170–172.

[12] R. P. Jessica Tomechak, "Citrix cloudplatform 3.0.5 (powered by apache cloudstack) administrator's guide," 2012.

[13] 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.

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

[15] F. Al-Haidari, M. H. Sqalli, and 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.

[16] S. Chapade, K. Pandey, and 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.

[17] P. S. M. K. M. Kumar, R. Korra, "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.

[18] M. Naresh Kumar, P. Sujatha, V. Kalva, R. Nagori, A. Katukojwala, and M. Kumar, "Mitigating economic denial of sustainability (edos) in cloud com-

puting using in-cloud scrubber service," in *In Proc. of the Fourth Intl' Conf. on Computational Intelligence and Communication Networks (CICN)*, 2012.

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

[20] S. VivinSandar and S. Shenai, "Economic denial of sustainability (edos) in cloud services using http and xml based ddos attacks," *International Journal of Computer Applications*, vol. 41, no. 20, pp. 11–16, 2012.

[21] G. Oikonomou and J. Mirkovic, "Modeling human behavior for defense against flash-crowd attacks," in *Communications, 2009. ICC'09. IEEE International Conference on.* IEEE, 2009, pp. 1–6.

[22] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "Ddos-shield: Ddos-resilient scheduling to counter application layer attacks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 1, pp. 26–39, 2009.

[23] W. Alosaimi and 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.

[24] B. Saini and 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.

[25] Z. Baig and K. Salah, "Multiagent pattern recognition for detecting ddos attacks," *IET Journal of Information Security*, vol. 4, no. 4, pp. 333–343, 2010.

[26] R. Lua and K. C. Yow, "Mitigating ddos attacks with transparent and intelligent fast-flux swarm network," *Network, IEEE*, vol. 25, no. 4, pp. 28–33, 2011.

[27] L. Yang, T. Zhang, J. Song, J. Wang, and P. Chen, "Defense of ddos attack for cloud computing," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, vol. 2, 2012, pp. 626–629.

[28] A. Chonka and J. Abawajy, "Detecting and mitigating hx-dos attacks against cloud web services," in *In Proc. of the 15th International Conference on Network-Based Information Systems*, 2012.

[29] Y. X. A. Chonka, W. Zhou and J. Singh, "Detecting and tracing ddos attacks by intelligent decision prototype (idp)," in *In Proc. of the IEEE Workshop on Web and Pervasive Security*, 2008.

[30] C. Yu and H. Kai, "Collaborative detection and filtering of shrew ddos attacks using spectral analysis," *Journal of Parallel and Distributed Systems*, vol. 66, no. 9, pp. 1137–1151, 2006.

[31] Y. Chen, Y. Kwok, and K. Hwang, "ollaborative defense against periodic shrew ddos attacks in frequency domain," *ACM Transactions on Information and System Security (TISSEC)*, 2005.

[32] K. Beaty, A. Kundu, V. Naik, and A. Acharya, "Network-level access control management for the cloud," in *In Proc. of the IEEE International Conference on Cloud Engineering*, 2013.

[33] Z. Tan., A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "Triangle-area-based multivariate correlation analysis for effective denial-of-service attack detection," in *In Proc. of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

[34] Q. Chen, W. Lin, W. Dou, and S. Yu, "Cbf: A packet filtering method for ddos attack defense in cloud environment," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, Dec 2011, pp. 427–434.

[35] J. Choi, C. Choi, B. Ko, and P. Kim, "A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment," *Soft Computing*, pp. 1–7, 2014.

[36] T. Vissers, T. S. Somasundaram, L. Pieters, K. Govindarajan, and P. Hellinckx, "Ddos defense system for web services in a cloud environment," *Future Generation Computer Systems*, vol. 37, pp. 37–45, 2014.

[37] K. Dutta, R. Guin, S. Chakrabarti, S. Banerjee, and 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.

[38] M. Assuncao, M. Netto, F. Koch, and 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.

[39] S. Chen, T. He, H. Wong, K.-W. Lee, and 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.

[40] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li, "Job scheduling model for cloud computing based on multi-objective genetic algorithm," *International Journal of Computer Science*, vol. 10, no. 1, pp. 134–139, 2013.

[41] Z. Zhou, Y. Luo, L. Guo, and L. Sun, "Assessment of p2p trust model based on fuzzy comprehensive evaluation." *Journal of Software (1796217X)*, vol. 8, no. 11, 2013.

[42] Y. Wang, D. S. Wong, K.-J. Lin, and 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.

[43] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pfänder, C. Serowy, I. Astrova, S. G. Grivas, and M. Schaaf, "Scaling in cloud environments," *Recent Researches in Computer Science*, 2011.

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

[45] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, 2012.

[46] Firebug, "What is firebug?" https://getfirebug.com/whatisfirebug.

[47] J. Idziorek, M. Tannian, and 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.

Vitae

- Name: Farid Salem Binbeshr
- Nationality: Yemen
- Date of Birth: September 9, 1986
- Email: *farid.binbeshr@gmail.com*
- Permenant Address: Foah-Mukalla-Yemen