DDOS ATTACKS IN CLOUD COMPUTING

Ph.D. Thesis

GAURAV SOMANI
ID No. 2013RCP9552

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR
May 2018

DDoS Attacks in Cloud Computing

Submitted in
fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Gaurav Somani
ID: 2013RCP9552

Under the Supervision of
Prof. Manoj Singh Gaur, MNIT Jaipur
Prof. Dheeraj Sanghi, IIT Kanpur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR
May 2018

CERTIFICATE

This is to certify that the thesis entitled "**DDoS Attacks in Cloud Computing**" being submitted by **Gaurav Somani (2013RCP9552)** is a bona-fide research work carried out under my supervision and guidance in fulfillment of the requirement for the award of the degree of Doctor of Philosophy in the Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur, India. The matter embodied in this thesis is original and has not been submitted to any other University or Institute for the award of any other degree.

Place: Jaipur
Date:

(Supervisors)

Prof. Manoj Singh Gaur
Professor
Department of Computer Science and Engineering
MNIT Jaipur

Prof. Dheeraj Sanghi
Professor
Department of Computer Science and Engineering
IIT Kanpur

Declaration

I, **Gaurav Somani**, declare that this thesis titled, "**DDoS Attacks in Cloud Computing**" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Ph.D. degree at Malaviya National Institute of Technology, Jaipur.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Malaviya National Institute of Technology, Jaipur or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date:

Gaurav Somani
(2013RCP9552)

Acknowledgements

First of all, I pay my tribute to Bhagwan Shri Singoli Shyam for the holy blessings that always helps me in my life. I would like to thank my supervisors, Prof. Manoj Singh Gaur and Prof. Dheeraj Sanghi. I really feel fortunate to have such supervisors and thank them for believing in me. Prof. Gaur gave me a lot of freedom and resources to experiment with different ideas. During this period, he asked me some of the most relevant and critical questions seeking sharp details. These detailed discussions are major contributors to the work presented in this thesis. Prof. Sanghi remains the source of the encouragement to pursue my doctoral studies. I am really thankful to him for providing a range of practical ideas that took a final shape in this thesis. Besides having a very busy schedule and responsibilities of deanships at IIT Kanpur and later at IIIT Delhi, he took a real pain by visiting Jaipur almost every other month to see my research progress.

I especially thank Dr.Vijay Laxmi for providing me regular inputs to my research work. I also thank the other committee members, Dr. Girdhari Singh, Dr. ES Pilli and Dr. C. Periasamy for their invaluable feedback to my work.

I have learned a lot while collaborating with Prof. Mauro Conti, University of Padua, Prof. M. Rajarajan, City University of London, and Prof. Rajkumar Buyya, University of Melbourne, who helped me in shaping my work for the top venues in the world. During this thesis, I have also been fortunate to discuss my work with some of the top researchers, who provided me crucial inputs related to my work. Few of them are Prof. Suraj Kothari, Iowa State University, USA, Prof. Herbert Bos, Vrije Universitat, The Netherlands, Prof. Ahmad-Reza Sadeghi, TU Darmstadt, Germany, Prof. N Asokan, Aalto University, Finland, Sitaram Chamarty, TCS, India, Prof. RK Shyamsunder, IIT Bombay India, and Dr. Bruhadeshwar, IIIT Hyderabad.

I also owe a lot to my teachers, Shri Ramesh Agnani, Prof. Hiren Joshi, and Prof. Sanjay Chaudhary for igniting my mind. I thank Prof. MM Salunkhe, Prof. AP Singh, Prof. AK Pujari and Prof. Manish Shrimali for supporting me in my full-time doctoral studies by providing administrative support from Central University of Rajasthan. I would also like to thank all my friends for supporting me during the doctoral studies.

I do not have words to express my tribute to my late grand mother, Smt. Sohan Devi and late grand father, Shri Banshilal Ji. I also owe a lot to my father Shri Satyanarayan Ji for providing abundance of encouragement and my mother, Smt. Savitri Devi for her immeasurable love. I thank my brothers, Abhishek and Parag for their great support and immense bonding. I would like to offer love to my little daughter Anaya and my nephew Aradhya for their affection. Undoubtedly, my real strength remains my wife, Priyanka whom I owe a lot in terms of her sacrifices, patience and unconditional love.

Abstract

In this work, we demonstrate that the traditional DDoS mitigation methods alone are not suitable to combat cloud targeted DDoS attacks. We characterize the effects of DDoS attacks on the cloud services and identify novel collateral damages to the non-targets during the DDoS attacks on cloud services. These damages eventually slow down the overall DDoS mitigation process. These DDoS attack effects are due to the on-demand, multi-tenant and federated nature of cloud environment and resource contention among services. We show that the DDoS attack mitigation can be expedited by proposed novel resource management methods at the victim service-end. Real attack experimentation based evaluations of employing our techniques show significant improvement over attack downtime, mitigation time, and attack cooling-down period.

We make four important contributions in the form of DDoS attack mitigation support solutions in the cloud computing environment. The first contribution is a novel DDoS aware resource allocation (DARAC) technique in which the resource allocation to the victim service is accurately performed by first differentiating between legitimate requests and attack traffic thus refining the real resource requirement with the help of server capacity planning. The remaining three contributions address the problem of internal collateral damage on the DDoS attack victim server which is the main cause of severe downtime and delayed attack mitigation.

In the second contribution, we investigated different aspects of DDoS attacks and find out that the DDoS mitigation service's performance is dependent on two factors. Severity of the "resource-race" formed due to the "extreme DDoS" attacks and the "attack cooling down period". Utilizing these two important factors, we propose a supporting framework for the DDoS mitigation services, by assisting in reducing the attack mitigation time and the overall downtime. This novel framework comprises of a processor affinity based victim-service resizing algorithm to provide performance isolation, and a TCP tuning technique to quickly free the attack connections, hence minimizing the attack cooling down period.

In the third contribution, we show that the DDoS mitigation methods may not provide the expected timely mitigation due to the heavy resource outage created by the attacks. We observe an important Operating System (OS) level "internal collateral damage", in which the other critical services are also affected. We formulate the DDoS mitigation problem as an OS level resource management problem and propose a novel resource containment approach to enforce the victim's resource limits.

Finally, in our last contribution, we address an important problem, whether the resource scaling during attack, always result in rapid DDoS mitigation? We show that the operating system level local resource contention, if contained during attacks, can expedite the overall attack mitigation. The attack mitigation would otherwise not be completely solved by the dynamic scaling alone. To overcome these issues, we propose a new "Scale Inside-out" approach which, during attacks, reduces the "resource utilization factor" to a minimal value for quick absorption of the attack. The proposed approach sacrifices victim service resources and provides those resources towards mitigation in addition to other co-located services to ensure service availability during attack.

Dedications

This thesis is dedicated to my family

Contents

List of Figures

List of Tables

List of Abbreviations

VM	**V**irtual **M**achine
VMM	**V**irtual **M**achine **M**onitor
VCPU	**V**irtual **CPU**
CSP	**C**loud **S**ervice **P**rovider
IaaS	**I**nfrastructure **as a S**ervice
PaaS	**P**latform **as a S**ervice
SaaS	**S**oftware **as a S**ervice
SECaaS	**SEC**urity **as a S**ervice
DMaaS	DDoS **M**itigation **as a S**ervice
DDoS	**D**istributed **D**enial of **S**ervice
EDoS	**E**conomic **D**enial of **S**ustainabiliy
SLA	**S**ervice **L**evel **A**greement
ROI	**R**eturn **O**n **I**nvestment
PoW	**P**roof **o**f **W**ork
SOA	**S**ervice **O**riented **A**rchitecture
TTL	**T**ime **T**o **L**ive
SDN	**S**oftware **D**efined **N**etworking
ISP	**I**nternet **S**ervice **P**rovider
DARAC	**D**DoS **A**ware **R**esource **A**llocation in **C**loud
ACPM	**A**utomated **C**apacity **P**lanning **M**odule
SME	**S**mall and **M**edium **E**nterprises
VSC	**V**ictim **S**ervice **C**ontainment
IDS	**I**ntrusion **D**etection **S**ystems
CDN	**C**ontent **D**elivery **N**etwork
SIO	**S**cale **I**nside-**O**ut

Chapter 1

Introduction

In recent years, Distributed Denial of Service (DDoS) attacks resulted in fatal attack effects to enterprises across the globe. DDoS attacks are also visible owing to the service downtimes faced by important web-services, and remain among the top cyber security threats during last several years. There are survey reports which state that one out of the five enterprises across the globe are affected by DDoS attacks [1]. The growth of DDoS attacks can be visualized by the attacks' progression made from the perspective of the peak attack bandwidth each year. The peak DDoS attack bandwidth touched more than 500 Gbps in 2015 from just 8 Gbps in the year 2000 [2].

Major motivations behind DDoS attacks include business rivalry, political ideology, and cyber war among countries. The most common outcome of DDoS attacks is "unavailability of target service". In addition to the unavailability, there are many short term and long term business and reputation losses, which is actually a set of consequences of the service downtime. Currently, cloud computing is being adopted across the globe to support the major IT needs of organizations in all industry verticals. However, the emergence of cloud computing has also led to the target shift of DDoS attackers towards cloud driven services. As highlighted in [2], more than 33% of the overall reported attacks in year 2015 had cloud services as their attack target.

On the other hand, cloud features such as the profound resources and the pay-as you go accounting model are also becoming attractive to the attackers. Most of the reported attacks usually last between few minutes to hours [2] and some "massive" attacks may last for few days to even weeks. There are many recent DDoS attacks on cloud services among those the attacks on the Amazon EC2 services, the RackSpace and the Linode servers are major attack incidents resulting into heavy service outages[3][4].

In this thesis, we make novel research contributions related to the characterization and mitigation of DDoS attacks which target services hosted on cloud computing infrastructures. Traditional DDoS mitigation methods are not suitable to combat cloud targeted DDoS attacks. We identified novel collateral damages to the non-targets during DDoS attacks on cloud services. These damages eventually slow down the overall DDoS mitigation process. These DDoS attack effects are due to the on-demand, multi-tenant, federated nature of cloud environment, and the resource contention among services.

We show that the DDoS attack effects can be minimized by the novel resource management methods proposed in this thesis. Real attack experimentation based evaluations of our proposed techniques are encouraging and show significant improvement over the attack downtime, the mitigation time, and the attack cooling-down period.

1.0.1 Motivation

We conducted an extensive survey of DDoS attack mitigation solutions for cloud services and found that most of the current solutions are utilizing the techniques which are mere extensions of the traditional DDoS mitigation schemes. On the other hand, there are solutions used by service providers, which mostly work on traffic filtering and quick attack absorption by resource scaling [5]. Additionally, our real attack characterization of DDoS attacks on cloud services shown some of the novel collateral damages.

These adverse effects are mostly due to the features of cloud such as multi-tenancy, auto scaling and migration of services. These “collateral damages” are visible on co-hosted cloud services and network components. We observe resource contention at the level of victim operating system that makes the overall mitigation and detection much slower. We also characterize that cloud features such as auto-scaling may not help in the expected attack mitigation with rapid attack absorption.

This discussion makes it necessary for availability of efficient solutions in the direction of DDoS attack prevention, detection and mitigation with the help of resource management during the presence of attacks.

1.0.2 Objectives

In this thesis, we aim to find answers to the following research questions.

1. Do DDoS mitigation methods applicable to the non-cloud infrastructures help in cloud as well?
2. What are the additional effects of DDoS attacks in cloud computing environment?
3. What factors are responsible for the rise and success of DDoS attacks on cloud services?
4. What techniques are useful in mitigating and minimizing the effects of DDoS attacks in cloud computing environment?

In order to explore and validate the suitable answers to the above research questions, we further divided this work to achieve the following research objectives.

1. To review the state of the art in the area of DDoS attack prevention, detection and mitigation research and prepare a comprehensive solution taxonomy.
2. To design a system model to understand and characterize the DDoS defense requirements in cloud computing environment.
3. To characterize the overall short-term and long-term effects of DDoS attack in cloud computing environment.
4. To design and demonstrate the DDoS aware cloud resource allocation algorithm. This includes an auto scaling strategy which is governed by the VM performance statistics and the incoming legitimate traffic.
5. To design an effective and immediate DDoS mitigation mechanism to lower down the DDoS attack period, associated costs, and DDoS aftereffects such as the collateral damages.

1.0.3 Contributions

We make following contributions in this thesis:

1. We presented a comprehensive literature review with a detailed insight into the characterization, prevention, detection, and mitigation mechanisms to combat DDoS attacks. Additionally, we presented a comprehensive solution

taxonomy providing DDoS attack solution classification. Our survey concluded that there is a strong requirement of solutions which are designed keeping utility computing models in mind. (Chapter 2)

2. In the context of DDoS attacks in the multi-tenant clouds, we argue that instead of just the victim server, multiple other stakeholders are also affected. Some of these important stakeholders are co-hosted virtual servers, physical servers, network resources, and cloud service providers. We show through system analysis, experiments, and simulations that these stakeholders are collaterally affected, even though they are not the real targets of the attacks. The damages/effects to these stakeholders include performance interference, poor web service performance, resource race, indirect EDoS (economic denial of sustainability), service downtime, and business losses. (Chapter 3)

3. To detect and mitigate DDoS attacks in cloud, we argue that the on-demand resource allocation schemes (also known as auto-scaling mechanisms) should also be revisited. We proposed a novel mitigation strategy, DARAC (DDoS aware resource allocation in cloud) that makes accurate auto-scaling decisions by differentiating the legitimate traffic from the attacker traffic. We also argue that most of the solutions in the literature, do not pay much attention to the service quality to legitimate requests during an attack.

   We calculated the share of legitimate clients in the resource addition/buying decision and make subsequent accurate auto-scaling decision. The experimental results shown that our proposed mechanism, DARAC mitigates various DDoS attack sets and takes accurate auto-scaling decisions for various traffic combinations. (Chapter 4)

4. In this contribution, we demonstrated a fundamental difference between a "regular" DDoS attack and an "extreme" DDoS attack. We conduct DDoS attacks on cloud services, where having the same attack features, two different services show completely different consequences, owing to the difference in the resource utilization per request. We studied various aspects of these attacks and found that the DDoS mitigation service's performance is dependent on two factors. One factor is related to the severity of the "resource-race" with the victim web-service. Second factor is the "attack cooling down period" which is the time required to bring the service availability in order after the detection of the attack.

   We utilize these two important factors and propose a supporting framework for the DDoS mitigation services. Our proposed approach assists in reducing the attack mitigation time and the overall service downtime. Proposed novel framework consists of an affinity based victim-service resizing algorithm to provide performance isolation, and a TCP tuning technique to quickly free the attack connections. (Chapter 5)

5. In this contribution, we observed an important operating system (OS) level effect which we term as “internal collateral damage”, in which the other critical services are also affected in the presence of a DDoS attack. We formulated the DDoS mitigation problem as an OS level resource management problem. We argue that providing extra resources to the victim server is only helpful if we can ensure the availability of other important services. To achieve the availability goal, we proposed a novel resource containment approach to enforce the resource limits to victim service. (Chapter 6)

6. In this contribution, we found that the activities such as attack absorption that provide timely attack data input for attack analytics, is adversely compromised by the heavy resource usage generated by DDoS attack. We showed that the operating system level local resource contention, if reduced during attacks, can expedite the overall attack mitigation. The attack mitigation would otherwise not be completely solved by the dynamic resource scaling alone. We conceived a novel relation “Resource Utilization Factor” for each incoming request as the major component in forming the resource contention.

   To overcome these resource contention issues, we propose a novel “Scale Inside-out” approach that during attacks, reduces the “Resource Utilization Factor” to a minimal value for quick absorption of the attack. The proposed novel approach sacrifices victim service resources and provides those freed resources to the mitigation service in addition to other co-located services to ensure service availability during attack. (Chapter 7)

Chapter 2

DDoS attack in Cloud Computing: State of the Art

2.1 Introduction

Cloud computing is a strong contender to the traditional IT implementations as it offers "pay-as-you-go" model based low-cost computing capabilities and services on demand. Governments, as well as industries, migrated their whole or most of the IT infrastructure into the cloud. Infrastructure clouds promise a large number of advantages as compared to the on-premise fixed infrastructure. These advantages include on-demand resource availability, pay-as-you-go billing, better hardware utilization, no in-house depreciation losses, and, no maintenance overhead. On the other hand, there is a large number of questions in cloud adopters mind, which is discussed in literature [6] [7]. Most of these questions are specifically related to the data security and the business logic security [8]. There are many security related attacks, that are well-addressed for the traditional non-cloud IT infrastructures. Their solutions are now being applied to the cloud targeted attacks. As data and business logic is located on a remote cloud server with no transparent control, most security concerns are not similar to their earlier equivalents in non-cloud infrastructures.

One of these attacks, which has been a much visible attack is the Denial of Service (DoS) attack [9]. Traditionally, DoS attackers target the server, which is providing a service to its consumers. Behaving like a legitimate customer, DoS attackers try to flood active server in a manner such that the service becomes unavailable due to a large number of requests pending and overflowing the service queue. A different flavor of DoS is Distributed DoS, or DDoS, where attackers are a group of machines targeting a particular service [10]. There is a high rise in the number of reported incidents of DDoS, which makes it one of the most important and fatal threat amongst many [11].

More than 20% of enterprises in the world saw at least one reported DDoS attack incident on their infrastructure [1]. The authors in [12] show a strong anticipation about the DDoS attackers target shift towards cloud infrastructure and services. Many attacks in last three years support these attack anticipations presented in the report. Amongst many recent attacks, there are few popular attacks which gained a lot of attention in the research community [12]. Lizard Squad attacked cloud-based gaming services of Microsoft and Sony and took down their services on Christmas day in 2015. Cloud service provider, Rackspace, was targeted by a massive DDoS attack on its services. In an another spectacular attack example, Amazon EC2 cloud servers faced another massive DDoS attack. These attack incidents incurred heavy downtime, business losses and many long-term and short-term effects on business processes of victims. A report by Verisign iDefense Security Intelligence Services [3] shows that the most attacked target of DDoS attacks in the last number of quarters is cloud and SaaS (Software as a Service) sector.

More than one-third of all the reported DDoS attack mitigations were on cloud services. One of the most important consequence of DDoS attack in the cloud is "economic losses". Report in [1] estimates the average financial loss due to a DDoS attack to around 444K USD. There are other reports by Neustar [13], which present the economic loss data of Q1, 2015. In this report, the average financial loss is more than 66K USD/hour. DDoS attacks and their characterization become completely different when applied to the context of the cloud. The difference arises mainly due to the consequences of an attack on the victim server. Infrastructure as a Service (IaaS) clouds run client services inside virtual machines (VMs).

Virtualization of servers is the key to the elastic and on-demand capabilities of the cloud, where VMs get more and more resources when needed and return unused resources when idle. Cloud computing's heavy adoption trend is due to the on-demand computing and resource availability capabilities. This capability enables the cloud infrastructure to provide profound resources by scaling, as and when there is a requirement on a VM. Therefore, a VM will not experience a resource outage as ample amount of on-demand resources are available in the cloud. This feature of "elasticity" or "auto-scaling" results into economic losses based DDoS attack that is also known as Economic Denial of Sustainability (EDoS) attack or Fraudulent Resource Consumption (FRC) attack [14].

In this chapter, we aim to provide a survey of DDoS attacks in the cloud environment. We also differentiate these attacks with the traditional DDoS attacks and survey various contributions in this space and classify them. For this purpose, we prepare a detailed taxonomy of these works to provide aid to comprehend this survey.

2.1.1 Need of A Survey on DDoS Attacks in Cloud

There are a number of survey papers available which deal with DDoS attacks, both from the perspective of attacks and mitigation in networks. There are surveys and taxonomies available which also include traditional DDoS mitigations methods including attack traceback, attack filtering and attack prevention [15] [16]. Taxonomies like [17] highlight DDoS in the cloud with the perspective of software defined networks. Surveys such as [18] focus on the solutions which are designed around traffic and behavior change detection. The following are some of the important requirements for this survey:

1. Cloud computing and technologies around it are recent phenomenon. It requires a different treatment regarding the characterization of the attack, detection and prevention. The desirable difference is evident in many recent attack incidents [12].
2. There are quite a good number of recent studies available on DDoS attacks, but there is no specific survey (including surveys on Cloud DDoS attacks) available to consider and gather solutions related to resource management aspects of utility computing.
3. Economic aspects of the DDoS attack (quoted as EDoS) and its consequences on cloud resource allocation is entirely missing from existing surveys; thus, the solutions specific to these issues are required.

2.1.2 Survey Methodology

We performed literature collection by doing an exhaustive systematic search on all the major indexing databases and collecting a huge number of papers related to the area. An initial scan resulted into a subclass of the collection. Another deep scan resulted in the papers we used in our survey and are used in the taxonomy

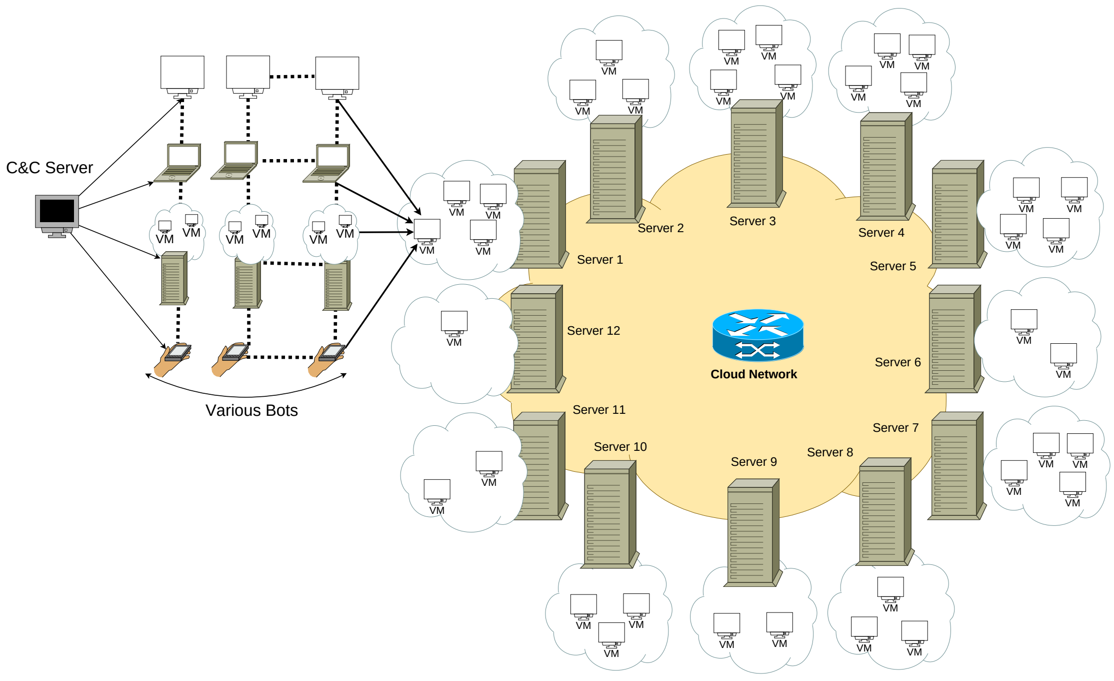


FIGURE 2.1: DDoS Attack Scenario in Infrastructure Cloud

preparation. We believe that the contributions listed in this survey are exhaustive and lists all the important contributions in the emerging area till date.

2.1.3 Contributions

We make following contributions in this chapter:

1. We introduce DDoS attack scenario in infrastructure clouds and identify how various elements of cloud computing are affected by DDoS attacks.
2. We present a detailed survey and taxonomy of solutions of DDoS attacks in cloud computing. Based on the developed taxonomy, we identify weaknesses in the state-of-the-art solution space leading to future research directions.
3. We provide a comprehensive set of performance and evaluation metrics for a uniform comparison and verification among attack solutions.
4. This chapter presents a detailed set of design aspects of effective DDoS mitigation at the end. It includes mitigation strategies at resource allocation level instead of preventive and detection strategies used by existing solutions.
5. This work would help security researchers to deal with the DDoS differently as compared to the treatment given while considering traditional IT infrastructure.

2.1.4 Organization

We discuss cloud computing and its essential features, which are affected by the DDoS attacks in Section 2.2. Section 2.3 details recent attack statistics to help in understanding the need for this survey. Section 2.4 presents the DDoS attack model and the threat model in the cloud. Section 2.5 offers a detailed and comprehensive taxonomy to help the reader to understand the broad solution space for DDoS attacks applicable to cloud computing. This taxonomy has three major branches which we discuss in three different sections. These three sections are attack prevention (Section 2.6), attack detection (Section 2.7) and attack mitigation (Section 2.8). In Section 2.9, we provide the guideline towards solutions to DDoS attack mitigation. We draw conclusions of this work in Section 2.10.

2.2 DDoS Attacks and Cloud Computing

Cloud computing provides an on-demand utility computing model where resources are available on "pay-as-you-go" basis. In particular, the cloud provider is an "Infrastructure as a Service (IaaS)" provider, who provisions VMs on-demand. On the other hand, a service provider is a cloud consumer who places the web service in the form of a VM (say an e-commerce application) in the infrastructure cloud provided by the cloud provider. Figure 2.1 depicts a typical cloud computing environment with a large number of servers running VMs.

2.2.1 DDoS Attack and Cloud Features

DDoS attacks have recently been very successful on cloud computing, where the attackers exploit the "pay-as-you-go" model [12]. There are three important features, which are the major reasons behind the success trends of cloud computing. On the other hand, the same set of features is proven to be very helpful to DDoS attackers in getting success in the attacks (discussed in Section 2.2.2). We now discuss these three features in detail:

2.2.1.1 Auto Scaling

Hardware virtualization provides a feature to shrink-expand resources of a VM while it is running. These properties permit the allocation of additional CPUs, main memory, storage and network bandwidth to a VM when required. Additionally, this can also be used to remove some of the allocated resources when they are idle or not needed. Multiple providers use this resource allocation mechanism with the help of auto scaling [19] web services, which allows cloud consumers to decide the resource need on the basis of resource utilization or similar matrices. The same feature is extended towards adding more VM instances on more physical servers and stopping when there is no need. Machine level scaling (vertical scaling) and data center or cloud level scaling (horizontal scaling) are two crucial features of utility computing.

Scalability is achieved by spreading an application over multiple physical servers in the cloud. Scalability is driven by high speed interconnects and high speed as well as ample storage. Virtualization of operating systems plays an important role while considering the scalability of VMs. VM cloning and its subsequent deployment are quite fast. Hence, whenever there is a requirement, cloned VMs can be booted on other servers and used to share the load. Scalability is also strongly supported by the live migration of VMs, where a running virtual server can be migrated to another bigger physical server without almost no downtime offering uninterrupted scalable operation.

2.2.1.2 Pay-as-you-go accounting

On-demand utility model has become very attractive for consumers due to its leaner resource accounting and billing model. "Pay-as-you-go" model allows a cloud consumer to use resources without physically buying them. A VM owner may want to add or remove more resources on-the-fly as and when needed. Other benefits of using cloud platform offer better hardware utilization and no need of arrangements like power, space, cooling and maintenance. Pricing or accounting plays an important role while understanding DDoS attacks in the cloud. Mostly, cloud instances are charged on an hourly basis and thus the minimum accounting period is an hour. Resources can be allotted on fixed basis, pay-as-you-go basis and by auctions. Similarly, storage and network bandwidth are measured using total size and total data (in and out) transfer. It is very clear that these models are "pay-as-you-go" models and are still evolving.

2.2.1.3 Multi-tenancy

Multi-tenancy gives the benefit of running more than one VMs from different VM owners on a single physical server. Multi-tenancy is a way to achieve higher hardware utilization and thus higher ROI (Return on Investment). An individual user may want to have more than one VMs running similar or different applications on a single physical server.

2.2.2 DDoS Attack Scenario in Cloud

A typical attack scenario is as shown in Figure 2.1. An infrastructure cloud has many servers and these servers run VMs in a multi-tenant virtualized environments. In addition of aiming at "Denial of Service", attackers might also aim to attack economic sustainability aspects of cloud consumers. Discussions on this attack started right after the inception of cloud computing [14]. There are few other contributions where this attack has been termed as Fraudulent Resource Consumption (FRC) attacks [20] and bandwidth exhaustion attacks [21].

Attackers thoroughly plant bots and trojans on compromised machines over the Internet and target web services with Distributed Denial of Service attacks. DDoS takes the shape of an EDoS attack when the victim service is hosted in the cloud. Organizations exist (also known as "Booters"), which provide a network of bots to their consumers to plan DDoS attacks on their rival websites [22]. Motives of these attacks range from business competition, political rivalry, extortions to cyber wars among countries.

The cloud paradigm provides enormous opportunities and benefits to consumers and the same set of features are available and may be useful for DDoS attackers. An attacker who plans a DDoS attack would send enough fake requests to achieve "Denial of Service". However, this attack would generate heavy resource utilization on the victim server. The auto scaling [19] takes this "overload" situation as feedback and adds more CPUs (or other resources) to the active pool of resources of this VM. Once a VM gets deployed, it starts as a "Normal load VM". Now, let us assume that the DDoS attack has started and consequently VM gets overloaded ("Overloaded VM"). The overload condition triggers auto-scaling features of cloud resource allocation, and it will choose one of the many strategies available in the literature for VM resource allocation, VM migration, and VM placement [23]. Overloaded VM may be given some more resources or migrated to a higher resource capacity server or may be supported by another instance started on another server. If there is no mitigation system in place, this process will keep adding the resources. This situation may last till service provider can pay or cloud service provider consumes all the resources. Finally, it will lead to "Service Denial". In turn, this leads to on-demand resource billing, and thus economic losses over and above the planned budget may occur. One trivial solution is to run VMs on fixed or static resource profile where the SLA does not have any provision for additional resources on demand. In this case, the DDoS will directly result in "Denial of Service" and all the attractive features of the cloud will be lost.

2.3 Attack Statistics and Impact Characterization

In this section, we provide coverage to various attack statistics and their impact on various victim organizations. We also covered few characterization studies to quantify the effects of DDoS attacks in the cloud.

2.3.1 Attack Statistics

Denial of service attacks are quantified and studied by many security solutions providers in the market [24] [25] [26] [27]. There are a number of other reports which state about the impact and rise of DDoS/EDoS attacks in the cloud. It was also anticipated that there will be a major target shift of the DDoS attackers from traditional servers to cloud-based services [12] and it has even been proven by the Q1 reports of 2015 [3]. As per this report [3], most of the attack targets were cloud services in Q1, 2015. According to the report published in March 2015 by Neustar [13], economic losses per hour at peak times are 470% more than the previous year. Lizard Squad planned attacks on Microsoft and Sony gaming servers, is the first example. Similarly, Amazon EC2 servers and Rackspace servers, which are cloud service providers, were attacked using a large DDoS attack in early 2015. Economic aspects of these attacks are also challenging. Greatfire.org was targeted by a heavy DDoS attack in March 2015, costing it an enormous bill of $30,000 daily on Amazon EC2 cloud [28]. As per report in [1], the average financial damage by a DDoS attack is up to 444,000 USD.

Akamai Technologies, in their report for Q4 2013, presented that almost the whole world's IT Infrastructure (188 countries), got affected by the reported DDoS attacks [29] and there was around 50% rise compared to last year. According to [30], In Q1 of 2014, the attackers started relying on other flavors like reflection and amplification based DDoS instead of bot-net based DDoS. Even the innovative "DDoS as a Service" tools are making it easier for hackers to plan these attacks. As per Q1, 2014 report of [26], the total DDoS attacks within last one year has increased by a significant number (47%). Another paramount figure to ponder is target servers. More than half of these DDoS attacks targeted towards entertainment and media industry which is mostly hosted in the cloud.

There are some other reports by Arbor Networks [27], which state that NTP based reflection and amplification attacks are the new forms of the DDoS. There is an additional attack that is termed very dangerous, started showing its effect parallel to a DDoS attack. This attack is known as "smoke screening", which is an attack to plan information or data breach behind a DDoS. While DDoS distract whole staff in mitigating or preventing from the present situation, the attacker may plan other attacks to harm. As per this report by Neustar [25], around 50% of the organizations have suffered from the "smoke screening" attack while they were only mitigating DDoS.

Repetition of the attack is also a major issue, and most of the targeted companies (90%) have faced repetitive attacks leading to vast business losses. The growth and adoption of cloud [31] and DDoS mitigation solutions in the cloud are two major points complementary to each other. Enterprises took few years to start adopting infrastructure clouds after its inception in 2007, and now many of organizations are entirely or partly transformed their IT infrastructure into cloud [32].

2.3.2 DDoS Attack Impact Studies in the Cloud

After the inception of the term "EDoS attack" by Christofer Hoff in 2008 [14], there are some works related to the characterization of the DDoS attack in the cloud and study its impacts. To see the effect of the DDoS attack, the authors in [33] conducted an important experiment, where they calculated the maximum possible charges on a cloud service. The authors conducted the experiment by sending 1000 requests/second with 1000 Megabits/second data transfer on a web-service hosted on Amazon CloudFront for 30 days. This experiment accumulated an additional cost of $42,000 for these additional requests.

The authors in [20] characterized the effectiveness of the EDoS attack on cloud consumer's bills. The authors in [20] have calculated the additional cost when there is only one attacker that is sending one request per minute for one month.

Even this could gather total 13GB of data transfer assuming a normal web request size of 320KB. A similar experiment was conducted in [34] where a web server cluster running on extra-large instance at Amazon EC2 was targeted with an EDoS attack. The observation showed that bills grew on the basis of the number of requests and deployment of additional resources. The authors in [35] presented the potential of malicious use of browsers of legitimate users to plan an EDoS attack. The authors use social engineering based web-bug enabled spam email to use legitimate browsers for the attack [35]. The authors argued that rented bots are easy to detect by the DDoS mitigation infrastructure and web-bugs in the form of a spam email to plan an EDoS attack can be used easily. They planned an attack on Amazon S3 infrastructure and characterized the attack effects. Other attack characterizations are presented in [36] and [37].

2.4 Attack and Threat Models

We prepared the attack model and the threat model of DDoS attacks in the cloud. We discuss these models in the following:

2.4.1 Attack Models

For last few years, DDoS attacks hold a major contribution among all the cyber attacks. The attack models valid a few years ago are no more suitable for the today's DDoS attacks. Cloud and its profound resources forced attackers to plan more powerful attacks. Attackers may be a group of machines under a single roof, a large number of geographically spread bots forming a botnet or may be an attacker cloud having comparable resources to a botnet. As said by the authors in [38], the DDoS attack scenario is an "arms-race" between the victim and the attackers. This "arms-race" term is a good fit for the cloud-based and cloud originated attacks and reaches a different level and scale. We detail the attack model with various attack features in Table 2.1. The attack can be planned by one or more methods shown in Figure 2.1, which, in addition to having botnet-driven attacks, also includes cloud originated DDoS attacks. These bots are also known as BotClouds [39]. There are other sets of unexplored attackers like mobile phone bots and IoT devices which require attention now [40].

We argue that the solutions to modern day DDoS should be designed keeping the attack model shown in Table 2.1 in mind. Attack packets, frequency, bandwidth,

Features	**Details**
Packets	HTTP GET, POST, TCP SYN, ICMP and amplification by DNS, NTP and TFTP
Frequency	Typical minimum frequency is >500 requests/s, depends upon resources at both the ends [41]
Bandwidth	1 Gbps to 300 Gbps [42]
Attackers	A single source, a network or botnet. Cloud servers (BotClouds) may be used to utilize scaling to win the "arms race".
Methods	Low rate, flood, flash mimic, amplification and massive
Repetition	repetition may be done from different sources after few minutes/hours
Duration	Minutes to hours (average 72 minutes [42]) and some lasts a few days.
Targets	Multimedia, government, e-commerce, cloud services and others
Motives	Competition, rivalry, extortion, smoke-screening and cyber war among countries
Timings	Important days for the enterprises including product launch, sale launch, important days for countries etc.

Table 2.1: DDoS Attack Model in Cloud

attacker profiles, methods to plan the attack, repetition, duration, targets, timing, and attack motives are few important aspects which should be considered while designing novel solutions. We included the most important features of attacks to help the community to understand the modern DDoS attacks and their shape.

2.4.2 Threat Models

DDoS attacks targeted towards cloud platforms are quite similar to the attacks on fixed infrastructures. Attackers aim, and the attack consequences make these attacks different in the cloud. DDoS, in the form of EDoS, may target the service provider on its economic sustainability. It is important to note that the DDoS attack consequences in the cloud are not limited to the victim VM and its resources. Cloud computing platform has multiple other VMs in the multi-tenant environment, other physical servers, network devices and bandwidth in the cloud. Surprisingly, other components are also affected. These possible threats are listed in Table 2.2. These threats are substantiated by contributions made by Xu et al. in [43] citing it as co-residence attacks. Some additional threats to cloud providers are dispute resolution in billing as the significant resource allocation would result in obvious billing, and payment recovery accordingly may become an issue [44] [45]. Other threats are from internal nodes. Cloud infrastructure may be used by an attacker to plan an attack. A large amount of compute capacity of the cloud can be used for these malicious activities [46]. Abuse of cloud services for the purpose of attack planning has recently become an important threat. A

Threats to	**Details**
Victim Server	Economic losses, service downtime, unnecessary resource addition, VM instance creation, migrations, business & reputation losses.
Co-hosted VMs	Performance Interference, resource race, and extra migrations due to resource exhaustion on physical server
Host Server	Extra migrations and it would not be able to fulfil the requirements of co-hosted VMs due to resource consumption by victim VM
Victim Owner	Downtime, economic losses, short term and long term business losses.
Cloud Provider	Extra migrations, performance interference to other VMs, large bandwidth bottleneck, downtime, and higher energy consumption
Other servers	Incoming migrations, VM instance creation and consequent issues due to VM instances under attack
Service end-users	Poor service quality, service downtime and problems to dependent . services

Table 2.2: DDoS Threat Model in Cloud

detailed analysis and survey of various abuse detection methods in the cloud are presented in [47].

1. *Threats to Service Provider:* In addition to economic or sustainability threats, the service provider might face a number of performance issues in their cloud-based service. It is of particular importance when the EDoS attack is happening. The cloud allocates additional resources after diagnosing an "Overload" state of a particular VM.

   During this period, the service may become unavailable, or there may be high response times and even request timeouts. In the worst case, when there is no resource cap (limit) associated with a VM, the cloud may theoretically allocate an infinite amount of resources. Practically, this needs few additional aspects. Once resources of a physical server, are exhausted using allocation, Cloud resource allocation methods may migrate the VM in consideration to another physical server where more resources are available. The cloud provider may also start some additional VMs to scale the service due to higher load. In both the cases, there is a downtime associated, which attributes to both VM migration or booting VM instances on other servers. Other repercussions include business and reputation losses generated due to downtime.

2. *Threats to cloud provider and other tenants:* The most significant threat to the cloud provider is the effect of the DDoS on the underlying infrastructure as well as other co-existing services. As cloud infrastructure is multi-tenant,

multiple VMs co-exist together. Due to continuous resources requirements of the DDoS affected VM, other VMs on the same physical machine would face performance interference [48] and resource contention. Even the genuine resource requirement of other VMs would be affected because the resource allocation mechanism would take a decision on the basis of needs at that moment. The co-hosted tenant may face downtime and migration overhead, because of resource exhaustion on the physical server due to the affected VM. We discuss these issues in much greater details with our contributions in Chapter 3.

2.5 Taxonomy of DDoS Solutions

This section presents the detailed solution taxonomy of DDoS attacks in the cloud. We gathered the final set of contributions using the systematic search methodology discussed in Section 2.1.2. We comprehensively surveyed the works related to the DDoS defense in the cloud and prepared a taxonomy as shown in Figure 2.2. We prepare this taxonomy by keeping a view that this work would serve the purpose of providing a clear, detailed and complete picture of the solutions space, different ideas, and approaches available in the literature. We also provide nomenclature to taxonomy fields to classify different contributions. We segment the taxonomy in three important parts which are attack prevention (P), attack detection (D), and

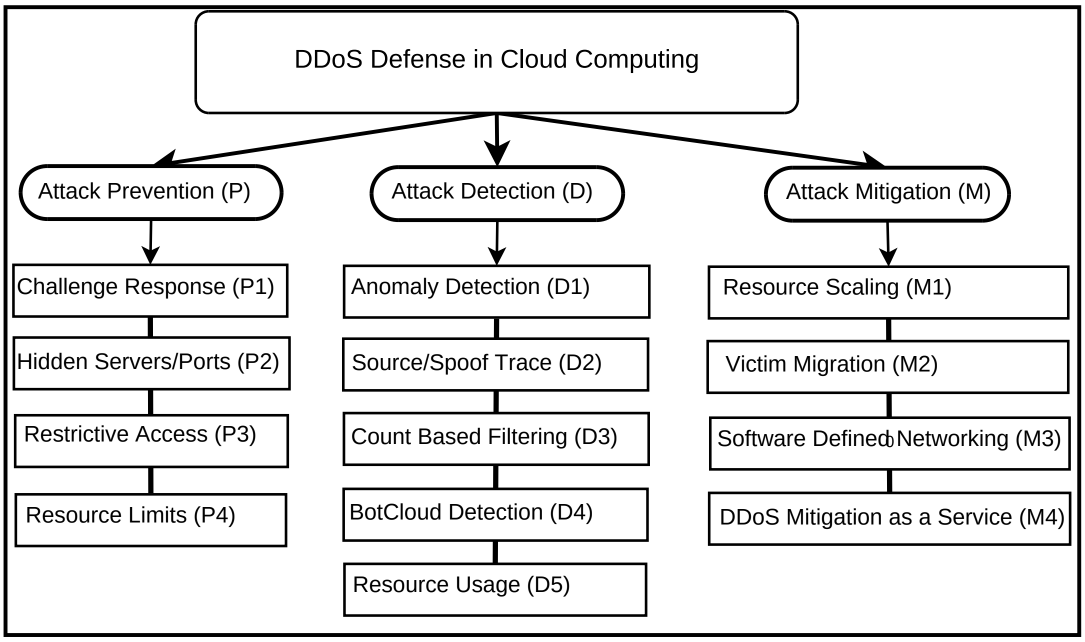


FIGURE 2.2: DDoS attack prevention, detection and mitigation in cloud: a taxonomy

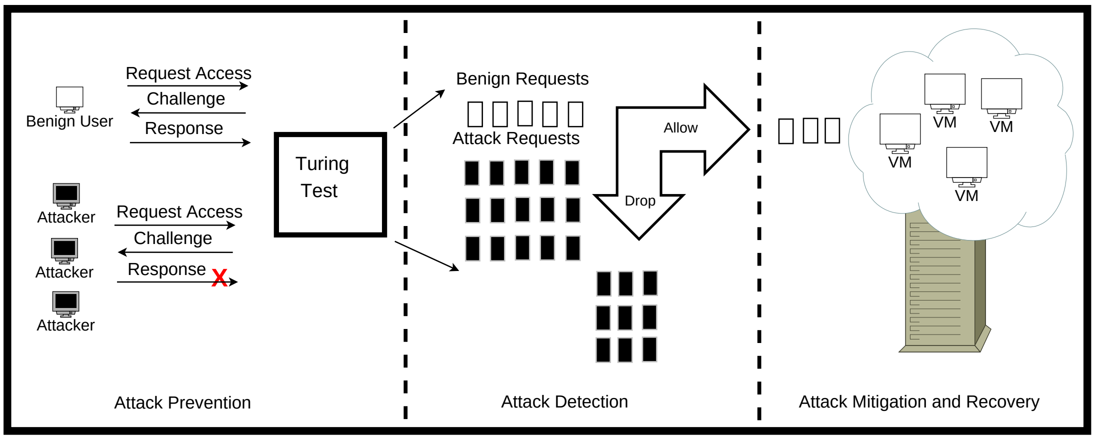


FIGURE 2.3: DDoS Protection in cloud at various levels

attack mitigation and recovery (M). Few of these works contributed in all three or two divisions of this classification, therefore, we discuss these contributions in all the related sections. The typical attack DDoS attsolution space looks like the one shown in Figure 2.3. At the first instance when the requests come, a simple "Turing test" may help in preventing the attack. The next stage is anomaly detection to both prevent and detect the attack. There is a large number of contributions related to attack detection which focus on the area of traffic monitoring and analysis. The third stage is based on the methods which are helpful in mitigation as well as recovery. Cloud computing features and profound resources help at this stage. We highlight the need for more solutions at this stage in section 2.8. There is a large number of contributions available at each stage, and they are listed in the next sections. However, in the Figure 2.3, we could just show a simplified gist, which misses many other solutions at each stage.

Before moving on to the discussion of various DDoS solution categories in the next section, we make an effort to propose important evaluation and performance metrics for various categories of our taxonomy. Table 2.3 shows the metrics related to the all three subclasses and their subcategories. It is important to highlight that in the next sections, we use these metrics in our discussion to compare the suitability of various solutions. There are many solutions which do not use any evaluation or performance metrics. However, we believe that these important metrics can help the community to orchestrate solutions which are verifiable against the important properties we list in the Table 2.3.

	Subcategory	**Important metrics to benchmark the solutions**
Attack Prevention	Challenge Response	Accessibility, usability, puzzle generation, storage, and false alerts
	Hidden Servers/ports	Redirection, puzzle metrics, overhead of replicas, and load balancing
	Restrictive Access	Accessibility, usability, response delay and false alerts in admission control
	Resource Limit	Cost and overhead of management of additional reserved resources
Attack Detection	Anomaly Detection	Overhead cost of training and profiling and false positives, and negatives
	Source and Spoof Trace	TTL data verification, traceback costs, false positives, and negatives
	Count Based Filtering	Suitability to various static and dynamic counts in minimizing the false alerts
	BotCloud Detection	Overhead cost of learning and verifying traffic flows and false alerts
	Resource Usage	Overhead of employing monitors and counters, and threshold suitability
Attack Mitigation	Resource Scaling	Auto-scaling decision and threshold suitability
	Victim Migration	Migration downtime, costs and network overhead for deltas
	SDN	Overhead cost of training, profiling, and false positives, and negatives
	DMaaS	Solution costs, service downtime, and other metrics based on different solutions

Table 2.3: Various performance metrics to benchmark the DDoS attack solutions in cloud computing

2.6 Attack Prevention (P)

DDoS prevention in the cloud is a pro-active measure, where suspected attackers' requests are filtered or dropped before these requests start affecting the server. Prevention methods do not have any "presence of attack" state as such, which is usually available to the attack detection and mitigation methods. Therefore, prevention methods are applied to all users whether legitimate or illegitimate. Most of these methods are tested against their usability, which incurs an overhead for the server as well as legitimate clients. We further classify this direction in four subclasses:

1. Challenge Response.
2. Hidden Servers/ports.
3. Restrictive Access.
4. Resource Limit.

For a quick view, the overall theme of each set of these methods, their strengths, challenges, and weaknesses are listed in Table 2.4. We also prepare a list of important individual contributions in Table 2.5. We enlist a brief theme of each solution to provide an overview about the variety of contributions available in each of the subclass.

2.6.1 Challenge Response (P1)

Challenge-Response Protocols (CRP) are designed to identify the presence of real users. Many times, this concept has been applied in an opposite manner, where

Techniques	**Strengths**	**Challenges**	**Limitations**	**Work**
Challenge Response (P1)	Effective and usable methods using puzzles to differentiate human and bots	Overhead of graphics generation and its storage	Image segmentation, OCR, dictionary and parsing attacks, and puzzle accumulation attacks	[49][50][51] [52][53][54] [55][56]
Hidden Server-s/Ports (P2)	Service is being offered to legitimate users while no direct connection is established with the real server in the first instance	Redundant servers ports and load balancing among them is needed	Overhead of additional security layer and redirections	[54][49][57] [58][59]
Restrictive Access (P3)	Admission control or instead of blocking/-dropping responses are prioritized for different classes of users	Quality of service concerns and overhead of maintaining number of connections for delayed period	Not scalable in case of massive DDoS with spoofing by large number of sources	[60][61][49] [57]
Resource Limits (P4)	Limiting the economic losses by restricting the maximum usable resources by a VM	Determining the resource limits and capacity planning of a server	It does not prevent DDoS and its effects, except limiting the economic losses due to cloud billing	[62][63][5]

Table 2.4: DDoS Attack Prevention Techniques in Cloud

the protocol tries to determine if the user is a bot/attacker machine, especially in the case of crypto puzzles or proof-of-work puzzles. One of the most common prevention technique is a Turing test in the form of a CAPTCHA, which is usually one of the most preferred methods in the category of challenge-response protocols. In addition to the methods related to cloud, some important CRPs from traditional DDoS defenses are also added to this discussion.

Graphical Turing tests are popular CRP implementations available today. Instead of showing plain text challenge and seeking an answer, these tests may present an image and a question related to that image. The image may have a picture, text with various impurities like an arc, distortion, and noise. Graphical CAPTCHA may have moving images in the form of .GIF or set of multiple pictures to choose from. Crypto puzzles are used to assess the computational capability of a client. Crypto puzzles are questions seeking output of a function with given inputs. For example, let us consider a hash function $f(x, y)$ with inputs a and b. The client is expected to compute $f(a, b)$ and return the answer back in some stipulated time.

Now, we discuss few important strategies related to challenge response schemes to prevent DDoS attack in cloud computing. EDoS Shield [52] and Alosaimi et al. [53] used graphical Turing tests to prevent the bot driven attack from occurring.

Solution category	**Contribution**	**Major theme of the contribution**
Challenge Response (P1)	[49]	Crypto puzzles to identify benign traffic
	[50]	Crypto puzzle levels based on the attack rate
	[51]	Crypto puzzles to identify benign traffic
	[52]	Graphical Turing tests
	[53]	Both graphical as well as crypto puzzles
	[54]	Proof-of-work puzzles
Hidden Servers/ Ports (P2)	[55]	Turing tests combined with other techniques
	[54]	Moving target approach to hide the servers
	[49]	Secure ephemeral servers with authentication
	[57]	Limits number of connections on hidden ports
	[58]	Moving targets using server replica shuffling
	[59]	Hidden server only visible to benign users
	[64]	Proxy forwards benign requests to the server
Restrictive Access (P3)	[60]	Admission control based on delayed response
	[61]	Human behavior (rate) detection and access
	[49]	Client reputation based prioritized access
	[57]	Admission control puzzles and hidden ports
Resource Limits (P4)	[5]	Resource Scaling to absorb the attack
	[63]	Resource caps to limit the attack effects
	[62]	Cloud metric monitoring and alarms
	[65]	DDoS Aware scaling and capacity planning

TABLE 2.5: DDoS Attack Prevention Techniques in Cloud

The authors in [53] proposed a DDoS Mitigation System (DDoS-MS), where initial two packets from the client side, form the basis of the attack identification and subsequent mitigation. In their work, they used both graphical Turing tests and crypto puzzles to identify the attacker.

The authors in [52] proposed a solution that filters requests on the basis of graphical Turing tests (CAPTCHAs). In this mode, a Virtual Firewall (VF) shield is designed which distinguishes the incoming requests on the basis of two lists, white and black. These records are updated on the basis of the success and failures of graphical Turing tests. To prove the effectiveness and novelty of their solution, authors conducted simulations to show the effect of their scheme on end-to-end delay, cost, and other performance indicators like throughput and bandwidth. There are a variety of crypto puzzles with different difficulty levels in [49] [50] [51]. The authors in [49] presented sPoW (Self-Verifying Proof of Work) methodology to mitigate EDDoS (Distributed EDoS). They provided a method to mitigate both network-level EDDoS and Application level EDDoS by extending the work proposed in [56]. In [56], instead of accepting all the traffic, the mechanism only accepts the traffic that it is capable of taking. The authors in [49] provided an innovative solution where they use crypto-puzzle to identify legitimate customers. These crypto-puzzles are self-verifying and do not run on the server. Instead of the server, the client computes the solution. On the basis of the time taken to solve the crypto-puzzle servers/nodes in the intermediate path, it will be decided whether the incoming traffic is legitimate traffic or not. The salient feature of this approach is that DDoS attacker may send their traffic even at a higher rate by speedily computing the puzzle, even in this case, sPoW approach does not allow the traffic. On the other hand, if DDoS traffic comes at a normal rate (equivalent to the rate at which legitimate customer sends) then their approach is successful in limiting the traffic.

Challenge Response schemes provide an easy way of implementing the attack prevention methods by addressing the most common automated, bot originated, and rate based attacks. The authors in [66] describes a list of qualities for crypto puzzles. A crypto puzzle should be solvable in a definite time and should not have other possible methods. Additionally, the server should be able to compute answers and verify them with ease.

Proof-of-work approaches are crypto puzzles but may have advanced features to utilize the client computation power and based on the correctness of the solution and computation time, the authentication and prioritized access is granted [49] [54]. This approach has multiple benefits including computation overhead shifting to the client and stopping overwhelming computationally equipped clients.

Accessibility and conversion rates are two important points to assess the challenge-response protocol implementations such as CAPTCHAs [67]. CRP implementations are designed and tested from the perspective of their attack persistence, accessibility, overhead, puzzle generation, and storage requirements. Many of these issues are related to the area of Human Computer Interaction (HCI). One of the important aspects of "Challenge-Response Protocols" is *Accessibility*, which should be considered while designing the question generation module. Designing difficult questions so that bots cannot construct their answer is quite an easy task, but a normal user should also be able to answer the questions with adequate comfort. Solutions based on Turing tests should be examined using a usability and accessibility study. Text puzzles are known to be cracked using dictionary attacks or parsing attacks. There is a number of limitations which are posed by [52], like the puzzle accumulation attack where an attacker sends a large number of requests for getting puzzles but does not solve them. It would result in an extra overhead of generating the puzzles at the server end. These Turing tests require additional overhead to generate graphics and storage space to store images. There are multiple works related to CAPTCHA cracking using image segmentation and optical character recognition (OCR).

2.6.2 Hidden Servers/Ports (P2)

Hidden servers or hidden resources such as ports is an important method to remove a direct communication link between the client and the server. Keeping an intermediate node/proxy to work as a forwarding authority, helps in achieving the objective of hiding the servers. The important jobs of this forwarding authority may include balancing the load among the servers, monitoring the incoming traffic for any vulnerability, and fault-tolerance and recovery of the servers.

Various approaches hiding the server resources include hidden proxy server [54], ephemeral servers [49], and hidden ports [57]. The authors in [54] proposed a moving target method to defend from DDoS attacks. The authors proposed the inclusion of many hidden proxy servers which may be dynamically assigned and changed to save legitimate clients. This approach has some practical issues like scalability, inclusion of large no. of proxy servers, and shuffling. Even different web services may not like to have changing server addresses in between connections. This method uses client puzzles using PoW (Proof-of-Work) to distinguish between attackers and normal traffic. Some of these approaches randomly allocate different hidden servers. Jia et al. [58] used the moving target based mechanism by shuffling the targets to confuse the attackers. This is achieved using the server replicas. This solution requires the overhead of maintaining the replicas and managing the moving target strategies. Additionally, the authors proposed strategies of effective shuffling assignments of clients requests to servers. The authors in [59] proposed a DDoS detection mechanism which is a request rate based detection method. The proposed method black lists incoming client request on the basis of their threshold rate.

By this blacklist, access is granted by special "army nodes" creating a virtual firewall. The authors argued that this way, the server could continue to serve legitimate clients. Similarly, the authors in [64] proposed a solution where a proxy server tests and forwards the benign requests to the server behind the fore-front service. Hidden servers or ports are preventive mechanisms to save the real service to face a DDoS attack. Therefore, requests to these hidden servers or ports are redirected by the authentication/proxy server which is the first server to be encountered by a client. Authentication provides an extra security layer to secure the actual service. Hidden servers can help in stopping the malicious or massive traffic to affect the real server. This extra layer may also support other purposes of redirection and load balancing among servers. The major limitation of this approach include the cost of the intermediate servers, time delay, and the computation overhead of redirection and its management at the intermediate nodes. Additional overhead includes the cost of maintaining the server replicas and their backup management.

2.6.3 Restrictive Access (P3)

Restrictive access techniques are basically admission control methods to take preventive action against the service capacity. Some of these strategies implement the prevention by delaying responses/access to the suspected attackers or even additional clients. In many of the contributions, this delay is introduced by prioritizing the legitimate clients or selecting clients with "good" past behaviors. There are few other techniques such as "Delayed access" and "Selective access" which are mostly similar, except that the strategies to provide the access to the clients are different.

In some cases, reputation is the basis of the admission control mechanism, in which some users are preferred over the others on the basis of reputation [49]. The correctness of crypto puzzle solution within a definite time and past web access behavior helps in calculating the reputation value. The authors in [49] named the reputation value as "capabilities". The authors in [60] gave a solution which did not drop any request based on its behavior, instead, they delayed the access to them. This delayed access prevents the attack to occur and even does not trigger auto-scaling. The proposed method controls the user access requests by their past web access history. In case these claims reach certain thresholds, the request responses are delayed instead of dropping the requests. These thresholds are decided from the request history of users. The effectiveness of delayed responses is questionable in real environments because of user accessibility issues, which requires timely responses. The authors in [61] followed a different approach where if a user does not behave as per typical human behavior, it is blocked for a specific period and then it is again unblocked. The authors proposed a subclass of the

DDoS attack and termed it as index page based attack where the very first page or homepage of the website is targeted for a DDoS attack [61]. The first page of every website should be free and can be fetched without solving any puzzle or authentication. The authors show attacks on this first page, where no Turing test prevention mechanism may work. The authors provided human behavior based identification to mitigate the attack and drop the requests of an attacker.

Instead of queuing all the clients, the authors in [57] proposed an admission control algorithm, where a limited number of clients are served simultaneously. These clients solve the Turing test and are assigned to the hidden ports using a port key. Once the test is passed by legitimate users, the proposed mechanism tries to limit the number of clients at any time by using an admission control algorithm. The server allocates resources on the basis of priority calculated based on the user behavior. The behavior is basically the web behavior on an e-commerce site on the basis of multiple parameters.

Most of the admission control methods, which implement restrictive access to stop the DDoS attacks to occur, are primarily based on delayed access or reputation based access. These methods provide a good way to optimize the server capacity by allowing requests based on the available resources. On one hand, these input control methods are solely dependent on the server capacity and client capability to compute the puzzle responses. At times, this restriction may limit the server to address the accessibility or usability perspective for fresh clients. As discussed in Section 2.6.1, the problems associated with the puzzle based solutions are also applicable here. Additionally, in case of sophisticated or stealthy attacks, the malicious attackers may try to earn the "reputation" before they show their real malicious behavior.

2.6.4 Resource Limits (P4)

As discussed in Section 2.3 on attack characterization, it was visible that the economic bills generated by a DDoS attack can be enormous. Resource limits can help in preventing these economic losses by correct auto-scaling decisions. However, deciding whether the resource surge is due to the DDoS attack or due to the real genuine traffic, is a very difficult task. Another way to prevent these resource losses is to put fixed resource services or "capped" resource limits on each service in the cloud. By doing this, we will miss the advantages of important features of cloud computing such as on demand resource allocation.

There are number of discussions and demands by cloud consumers on providing a track of resource utilization in the form of alerts. Additionally, some of the providers, provide the real-time monitoring services [62]. They also provide resource limits in the form of "Caps" on maximum resources a VM would be able to buy and sustain.

Resource limits can surely restrict the cost penalty on the dynamically scaled resourced but they can also limit the usage of on-demand computing feature of cloud computing. Attack prevention mechanism discussed above present a variety of methods available for the preventive security. However, it is important to note that these prevention mechanisms alone can not help in combating the DDoS attacks in cloud infrastructures. Another line of support from other mechanisms such as detection and mitigation mechanisms may help once the attack is present.

2.7 Attack Detection (D)

Attack detection is a state where attack signs are present on the server in terms of its services and monitored performance metrics. These attack signs are initial signs, where the attack has just started to take the shape, or there may be a situation, where the attack has already deteriorated the performance. These methods may seem to be similar to "attack prevention" at times, and many of contributions provide solutions in the same manner. Various performance metrics, which are monitored and affected due to an attack range from large response times and timeouts to higher memory and CPU usage. We further classify this section into five subcategories:

1. Anomaly Detection.
2. Source and Spoof Trace.
3. Count Based Filtering.
4. BotCloud Detection.
5. Resource Usage.

For a quick view, the overall theme of each set of the classified methods, their strengths, challenges, and weaknesses are listed in Table 2.6. We also prepare a list of important individual contributions in Table 2.7 where we enlist a brief theme of each solution to show the variety of contributions available in each subclass.

Techniques	**Strengths**	**Challenges**	**Limitations**	**Work**
Anomaly Detection (D1)	Machine learning and feature based detection	Feature identification, testing and minimizing false alarms and IP spoofing	Scalability issues and overhead of training, matching and statistical analysis of traffic features	[68][69][70] [20][71][72] [73][74][75] [57][76] [77]
Source and Spoof Trace (D2)	Identifying the source of of web requests to stop spoofing	Filtering at edge routers and suitability of TTL based methods	Cooperative mechanisms require network devices and service support	[78][79][80] [81][82][83] [84][85][86]
Count Based Filtering (D3)	Hop count, number of connections or number of requests based threshold filtering	Requires TTL hop data of real user. Heterogeneous implementations of hop count. Deciding on count threshold is a challenge	IP spoofing issues may defeat the (non-TTL) schemes. Only successful in case of two different TTLs for same source IPs are received. False alarms. Probing is also needed.	[55][51][52] [61][59][87]
BotCloud Detection (D4)	Detecting the attack sources inside the cloud by monitoring the features of VMs and the network	Identifying the activities and their thresholds for various suspicious activities	Very difficult to detect all kinds of attack flows (including zero-day). The detection only works at the edge of attack originating cloud.	[46][88][89] [90][39]
Resource usage (D5)	OS level/hypervisor level detection methods to monitor abnormal usage	Interpreting the high utilization whether it is due to attack or due to the real traffic	Only gives a signal about the possibility of attack and requires supplementary detection mechanisms	[91][46] [92]

Table 2.6: DDoS Attack Detection Techniques in Cloud: D1 Pattern Detection

2.7.1 Anomaly Detection (D1)

Anomaly detection methods identify anomalous patterns from packet traces, established connections, and web access logs or request headers. The specific pattern to identify in the log or the trace is decided by attack traces and other past historic behaviors. Web behavior is modeled using a large number of characteristics and metrics working upon those characteristics. Most of the contributions used the web behavior of normal web traffic as a benchmark pattern. These methods collect the normal web behavior during the period when the attack is not present. On the other hand, few contributions prepare attack behavior profile and than filter-out the attack traffic by learning based detection. Feature selection, dataset preparation, and testing or profiling against these learned rules are the three important set of operations involved in these detection strategies.

Now, we discuss few important strategies related to DDoS attack anomaly detection in cloud computing. Idziorek et al. [68] worked on web access logs and argued that legitimate web access patterns follow "Zipf" distribution. Based on the web

Solution category	**Contribution**	**Major theme of the contribution**
Anomaly Detection (D1)	[68]	Anomaly traffic detection using Zipf's law
	[69]	Co-variance profiling of IP/TCP flags[72]
	[71]	Filtering based on Jensen-Shannon Divergence
	[93]	Co-relation based attack flow analysis
	[74]	IP/TCP flags based confidence filtering
	[75]	"Helinger" distance based multi-stage solution
	[57]	User profiling using walk-through on site pages
	[76]	Filtering using SOAP headers
	[20]	Identification of a genuine web session
	[77]	Profiling based on time spent on the pages
Source and Spoof Trace (D2)	[79]	Back propagation neural networks tracing
	[81]	SOA-Based Trace back to reconstruct the path
	[82]	OS fingerprinting to stop IP spoofing
	[84]	Multi-stage source checking using text puzzles
	[83]	Source authentication using token at each router
	[85]	Source tracing based on location parameters
	[78]	Deterministic packet marking of ingress routers
	[94]	Multiple filters to stop spoofing
	[73]	TTL probing to find genuine TTLs
	[86]	Statistical filtering based spoof detection
Count Based Filtering (D3)	[55]	Hop count and request frequency thresholds
	[51]	TTL matching to detect IP spoofing
	[61]	Request threshold for a human in unit time
	[87]	Threshold on number of connections by a source
	[73]	TTL probing to find genuine TTLs
	[59]	Request count threshold by each source
BotCloud Detection (D4)	[46]	Network/VMM checks to find attack VMs
	[88]	CSP driven attack flow check and source trace
	[89]	Bot detection in VMs using NetFlow
	[90]	Hypervisor led collaborative egress detection
	[39]	Virtual Machine Introspection (VMI)
Resource Usage (D5)	[91]	VM resource utilization threshold for detection
	[46]	Resource counters and traffic thresholds for VMs
	[95]	Resource usage anomalies and introspection
	[65]	DDoS Aware auto-scaling to combat EDoS
	[92]	Resource usage of attack target servers

Table 2.7: DDoS Attack Detection Techniques based on Pattern Detection

access pattern training, they could identify outliers, which do not follow this distribution in pattern [68]. On the other hand, the authors in [69] used the baseline profiling of various IP and TCP flags which entails the network behavior model. The authors proposed the detection of flooding in the cloud using the training of normal and abnormal traffic and used the covariance matrix approach to detect the anomaly. Amongst other approaches, Shamsolmoali et al. [71] proposed statistical filtering based attack detection. Proposed approach calculates divergence between normal traffic and attacker traffic on the basis of Jensen-Shannon Divergence [72]. Initially, they used the traditional TTL based differentiation among the legitimate users and spoofed attackers. After IP spoofing filtering, they applied the Jensen-Shannon divergence to identify the anomalies in the traffic to achieve around 97% accuracy. There are few performance issues with TTL based approach. TTL based filtering is not useful unless we have a large database of actual TTL values of genuine requests using probing [73]. In [74], the authors derived the web behavior using IP and TCP header fields. By this, they could calculate the confidence value in the detection strategy.

The major idea of this work was the claim that IP address and TTL values are related to multiple past contributions; therefore, the same can be extended to other fields in IP and TCP headers and a score for each incoming packet can be calculated. Jeyanthi et al. [75] proposed an approach to detect the DDoS attack on the basis of entropy. The authors use "Helinger" distance which differentiates between the attack and genuine traffic distributions. The authors have used traffic rate and entropy and predict the arrival rates of incoming traffic based on history. The authors in [57] demonstrated an application specific way of differentiating web requests based on their behavior on an e-commerce site. This work created two client profiles, one for good clients and another one for bad clients. Based on user walk-through on pages, purchases, searches these profiles are created and decides the customers' priority. Resource access pattern by clients is the main idea to detect the attackers. In [76], the authors created normal web profile, which include HTTP and XML header features. The number of elements, content length and depth help in creating the normal user profiles. The proposed methods identify the outliers which deviate from these profiles. The authors in [77] argued that an attacker would not spend any time on a page but would request them like a flood. They gathered TSP behavior of users as well as of bots and identified that the attackers TSP is mostly negligible or even if it is not near zero, it is constant or periodic.

The most important strength of these attack detection techniques lies in the machine learning of the past history of benign traffic or the attack traffic. With the advent of the paradigms such as big data analytics and software defined networks these detection methods gained much importance in quick attack detection and monitoring. The authors in [10] present a detailed survey of detection techniques for traditional infrastructures. These techniques are now becoming popular for cloud targeted attacks. The major challenges for detection techniques lie in the behavior identification in terms of features and their training. The most important evaluation criteria for these methods lie in the false alerts (positive and negatives) they generate during the testing of the incoming traffic. Other important challenge lies in stopping the IP spoofing which can defeat many of the detection strategies.

2.7.2 Source/Spoof Trace (D2)

There is a number of trace back algorithms in the literature, which identify and stop the spoof attack by tracing the source. Source traceback schemes are employed to stop/detect the identity spoofing techniques. These techniques are important as most of the detection/prevention methods model the user behavior or profile based on some identity which is mostly an IP address in case of web access. In the attack cases where IP spoofing is employed, the detection mechanisms can be defeated very easily.

Let us have a look at some of the techniques related to this subclass of solutions. In [79], the authors have done the source trace for SOAP requests. The authors used back propagation neural networks to tackle both the popular variants of the DDoS attack, which are HTML DoS and XML DoS. The authors in [81] drops all the spoofed packets at the edge routers using egress filtering.

The authors proposed a method to identify the source of the attack by "Service Oriented Architecture (SOA)" based technique. They proposed a source traceback method by introducing an additional server before the real web server. This additional server is known as SBTA (SOA-Based Trace back Approach), which marks each packet by cloud trace-back tag and also reconstructs the path to know the source. The proposed method uses a database to store and compare each incoming packet, and it requires an additional server to mitigate the attack. Osanaiye et al. in [82] proposed an IP spoofing detection method, which is based on the matching of the OS versions of both attackers and real IP owners. The authors argued that the OS fingerprint of the spoofed attacker can be found out by asking the real OS fingerprint from the owner. The authors in [83] used source authentication approaches where a cryptographic token is verified at each router to authenticate the source. Similarly, authors in [84] used source checking approaches. Source traceback approaches also used hop count or TTL values discussed in Section 2.7.3. Other important contributions in this area, include tracing sources by location [85] and statistical filtering [86]. There are other surveys available in this direction, which discuss contributions related to botnets, their trends, and detection methods [80].

The source traceback and spoof identification methods are very important for all the detection methods. However, being a cooperative detection mechanism, these methods require a support from many other network devices such as edge routers, and services. Additionally, IP address being a “source provided address”, it is extremely difficult to design spoof protection against massive spoofing by large scale botnets.

2.7.3 Count Based Filtering (D3)

This subclass on “Count Based Filtering” also fits into the class of prevention mechanisms as well. Many a times, techniques use thresholds to detect the initialization of attack and later they can use the thresholds to identify the presence of the attack. These techniques use network resources such as hop-count, number of connections or number of requests in a unit time from a single source as count thresholds.

The authors in [55] proposed the detection scheme, where apart from other schemes, a hop count filter helps in identifying the spoofed packets. Similarly, the authors in [51] used TTL values alone for the purpose of DDoS prevention cum detection. As per this work, they store the TTL values corresponding to various IP addresses in the white and black lists. If there is a new request then it is sent to the graphical Turing test and on the basis of verification, it is added to the white list or black list. Those who are in white-list but with a different TTL, are also sent to the Turing test and on success their TTL value is updated. The authors in this paper extended their earlier work of the EDoS Shield [52] and improved it for the case of IP spoofing. Their solution is based on hop-count diversity, where attacker packets are claimed to have same hop count, and thus they can be detected. In this strategy, if a user sends N request in period P, access to this user is only allowed, if his request count is less than threshold T_H.

The authors in [61] used request count on the basis of human behavior and dropped all subsequent requests from the same IP for a finite period. The authors in [55] have proposed a method to mitigate HTML and XML DDoS attacks by multiple level filtering on the basis of client puzzles, hop count and packet frequency. Various filters at server side incur significant overheads and latency for ordinary users. Similarly, the authors in [59], used the request count method to identify attackers and blacklist them. DDoS Deflate [87] is a popular open source DDoS detection tool which is dependent upon the threshold of number of connections established by each source.

The major strength of these solutions lie in their easy deployment and support by the available OS level firewalls such as `iptables` and APF. These methods also give administrators a quick control over the situation. However, these methods may not suite the requirements of all the users as the thresholds for a whole domain behind the NAT may not be similar to the thresholds required for dependent web-services. Additionally, methods such as TTL/hop-count requires a user database having the actual hop-count/TTLs. Other issues arise due to a variety of heterogeneous implementations of hop-count in different systems. On the other hand, the IP spoofing techniques may defeat the (non-TTL) schemes. Overall, the false positives or negative are important performance issues related to these count based filtering approaches.

2.7.4 BotCloud Detection (D4)

Any cloud DDoS attacker may also use cloud infrastructure for its own nefarious purpose. Cloud infrastructure can be used for the purpose of installing botnets. These clouds are known as BotClouds. This subcategory describes the contributions which tries to find or detect the internal attack VMs in the cloud network. Most of these BotCloud related solutions are source based or Cloud Service Provider (CSP) based approaches.

The authors in [46] presented a cloud level detection method to identify if there are attacker bots running inside hosted VMs using network level and VMM level checks. Another contribution in this direction applies Virtual Machine Introspection (VMI) and data mining techniques to separate the infected VMs from other VMs in multi-tenant VMs [39]. The authors prepared a list of typical actions of malware bots infected VMs and used a clustering algorithm to identify the infected VMs based on the training. There are other BotCloud related solutions available in [88] [89] [90]. The authors in [88] provided a solution where the cloud provider checks the traffic flow and perform the anomaly detection using source traceback techniques at the network. The authors in[89] provide a solution based on the SDN approaches using Bot detection with the help of NetFlow protocol. The authors in [90] used hypervisor based checks to detect the vulnerabilities in the guest

VMs using collaborative egress detection technique. Advanced methods such as one in [39] proposed a detection using virtual machine introspection (VMI).

The major strength of these methods lie in their deployment at the CSP end. By this, CSP has a control to monitor at the network edge for any anomaly in the traffic behavior or other performance counters. However, these methods are not capable of detecting all kinds of attack flows such as zero-day or stealthy flows. On the other hand, this kind of detection methods only work at the edge of attack originating cloud. In case, the CSP does not provide support for such detections, these attacks may become massive utilizing the profound resources of cloud computing.

2.7.5 Resource Usage (D5)

Utilization of various resource of the cloud or a physical server by a VM can also provide important information about the presence of the DDoS attack or an anticipation of the upcoming DDoS attack. Cloud environments run infrastructure as a service cloud using virtualized servers where hypervisor can monitor the resource usage of each VM on physical server. Once these VMs start reaching the decided resource utilization thresholds, the possibility of an attack can be suspected.

In [91], the authors provided solutions on the basis of available resources with VMs and their upcoming requirements. Similarly, Latanicki et al. in [46] used performance counters and traffic to identify resource usage of VM and devise possible mitigation of the attack. Resource utilization possesses a very important and indirect metric to identify the possibility of an attack. The authors in [92] used resource limits as the sole method of the DDoS detection and then proposed mitigation methods. The authors in [95] modeled the resource usage anomalies of VMs using virtual machine introspection to detect the possibility of resource surge due the DDoS attack.

DDoS attacks being resource intensive attacks provide a indirect relationship for the success of these resource usage based profiling and detection methods. Auto-scaling mechanisms are triggered on the basis of “overload” and “underload” states of the targeted VMs. This aspect also provide a possible co-relation between the VM resource usage and a DDoS originated resource surge. The limitation of these set of approaches lies in interpretation of the high resource utilization. It is very difficult to conclude whether the resource surge is due to the attack or due to the real traffic. As the resource surge only gives a alert about the possible resource surge, we may require other supplementary detection mechanisms.

After discussing the attack detection solutions at length, it is clear that the traffic filtering based on the attack patterns is a major part of the DDoS attack solutions. Most of the methods are based on machine learning artifacts and provides a way to control the input traffic. However, the detection methods alone may not suffice for the purpose of integral protection from the DDoS attacks. The role of attack prevention solutions for the first hand protection and the role of attack mitigation solutions to ensure the resource availability for effective mitigation, can not be ignored.

2.8 Attack Mitigation (M)

In this section, we group methods which help a victim server to continue serving the incoming requests in the presence of an attack. The service downtime is a major business parameter for websites and an organization may lose a significant number of prospective customers [13]. Attack mitigation and service recovery are complementary to each other and both help the victim server to remain available. These methods are used temporarily and once the attack subside, the server may be brought back to the actual situation. We further classify this section into four subcategories:

1. Resource Scaling.
2. Victim Migration.
3. Software Defined Networking (SDN).
4. DDoS Mitigation as a Service (DMaaS).

For a quick view, the overall theme of each set of the classified methods, their strengths, challenges, and weaknesses are listed in Table 2.8. We also prepare a list of important individual contributions in Table 2.9 where we enlist a brief theme of each solution to show the variety of contributions available in each subclass.

2.8.1 Resource Scaling (M1)

Dynamic auto-scaling of resources is one of the most popular features of the clouds. It is also treated as one of best mitigation methods to counter DDoS attack allowing server availability or continuity with scaled resources. Auto scaling can be

Techniques	**Strengths**	**Challenges**	**Limitations**	**Work**
Resource Scaling (M1)	Provides a quick relief to resource bottlenecks resource bottlenecks	Correctly deciding whether and when extra resources are required	False alarms may lead to EDoS. Co-hosted VMs may also be affected	[19][96][92] [46]
Victim Migration (M2)	Migrating the DDoS victim service to other servers which helps in minimizing losses	Migration candidate selection and migration host selection	Migration costs and overheads. Subsequent migrations/swaps in cloud	[91][46][97]
Software Defined Networking (SDN)(M4)	Abstract and timely view of the network and the incoming traffic using controllers	SDN may itself become an easy target of the DDoS attacks	Mostly useful at network boundaries and ISP level network control	[98][99][17] [100][101][99]
DDoS Mitigation as a Service (DMaaS)(M5)	Cloud based hybrid mitigation using extra resources or remote traffic monitoring and prevention services	Cost overhead issues. Methods are mostly similar to the on-premise solutions but mitigation expertise is an advantage	Solutions may not cater various kinds of applications and attacks. Local issues may not be visualized by DDoS mitigation-as-a-service	[64][102][103] [63][62]

Table 2.8: DDoS Attack Mitigation (M) Techniques in Cloud

done horizontally, where new instances may be started on the same or different physical server to serve incoming requests till the victim server is facing the attack. In vertical scaling, resources like CPU, memory and disk can be scaled in the same VM or the same logical unit. These extra resources can help the victim machine to survive the attack and keep running. One of the major disadvantages of this strategy is that it can become an advantage for the attacker to increase the attack strength to even deplete added resources and generating a requirement of more resources shaping the attack into an EDoS [19].

We now discuss few important contributions related to attack mitigation and recovery using resource scaling. The authors in [96] proposed a multi-level DDoS detection system for web services. VM owner level (Tenant level), service Level, application level and cloud level detection are placed to have a collaborative DDoS detection system. It is one of those solutions which are utilizing the information from all the stakeholders in mitigating the DDoS attacks. However, there might be large overhead and other security concerns due to information flow among multiple levels.

One of the first and most important contributions in this area, which touches cloud-specific issues is by Shui Yu et al. [92]. The authors in this paper considered the dynamic resource allocation feature of the cloud to help the victim server to get additional resources for DDoS mitigation. In this way, individual cloud customers are saved from DDoS attacks by dynamic resource allocation. Experiments on real website data sets show that their queuing theory based scheme work to mitigate DDoS attack. The authors in [46] presented three different scenarios to stop the

Solution category	**Contribution**	**Major theme of the contribution**
Resource Scaling (M1)	[96]	Multi-level (VM, service, application and cloud)
	[92]	Dynamic resource scaling for quick detection
	[46]	Resource scaling in federated clouds
	[5]	Scaling to absorb the attack
	[65]	Scaling based on capacity planning
	[97]	Scaling over low cost untrusted CDN clouds
Migration (M2)	[91]	Victim VM migration to other physical servers
	[104]	Migrating proxy entry points at overlay
	[46]	Victim VM migration to other physical servers
	[105]	Exploiting VM migrations using DDoS
SDN (M4)	[98]	ISP-level monitoring of traffic and routing
	[99]	Strict authentication and access control
	[100]	Re-configurable network monitoring and control
	[106]	SDN based deep packet inspection
DMaaS (M5)	[64]	Victim cloud-based network service
	[102]	Proof-of-work scheme and ephemeral servers
	[103]	Hybrid (On-premise firewall plus Cloud firewall)
	[63]	Resource caps to limit the attack effects
	[62]	Cloud metric monitoring and alarms

Table 2.9: DDoS Attack Mitigation Techniques in Cloud

DDoS attack in the cloud. These three scenarios include external attacks to internal servers, internal attacks to internal servers and internal attacks to external servers. The authors provided strategies to detect the attack and get recovered using scaling and migrations in a federated cloud environment. Reserved resources are kept in [92] to support the server in attack times. "How much reserved resources should be kept?" is an important question. The cost of additional and idle resources is a drawback. It is one of the flexibility which keeps back up and reserved resources for a rainy day [46]. The authors in [97] provided a mechanism which uses low-cost untrusted cloud servers in the presence of DDoS attacks to scale services frugally. "CDN On Demand" is an open source platform developed to support the mechanism [97]. Industry solutions such as [5] also advocate for quick resource scaling for quick attack absorption.

The resource scaling is an important aspect of cloud computing which is also useful in quick attack mitigation while maintaining the service availability. The resource scaling is a process which is useful for a service to recover by expanding the VM resources or VM instances. However, the resource scaling may also become against the overall idea of cost-savings using the cloud hosting. In case the attacks are stealthy and remains undetected, than the resource scaling may increase the attack costs multi-fold.

2.8.2 Victim Migration (M2)

VM migration has changed the way the entire running server is shifted to another physical server without noticeable downtime. Migration can be used to shift the victim server to a different physical server, which is isolated from the attack and once the DDoS is detected and mitigated, the server can again be shifted back to the actual place.

We discuss few important contributions using victim migration to mitigate the DDoS attack. The authors in [91] proposed a similar strategy by keeping some reserved resources on a server. While the attack is detected, they migrate the victim to those reserved resources and bring it back when attack ceases. The downtime to legitimate customers is one issue which is very important while migration is chosen as a mitigation method. Additionally, if the attack continues for longer duration or repeated, the cost considerations will be high. The authors in [46] also used a similar approach. The authors in [91] proposed a remedial method for the server affected by DDoS to keep it in the running or serving state. The authors proposed to detect the DDoS attack at the level of Virtual Machine Monitor (VMM) instead of any count based or packet filtering. VMM is detecting the possibility of the DDoS attack by continuously monitoring resource utilization levels. Once the resource utilization levels reach a certain threshold, VMM flags a DDoS attack. On signaling, VMM migrates or duplicates the running VM as well the application to a separate isolated environment on the same physical server. This isolated environment is created with the help of reserving some additional resources for backup, where the "victim" is shifted in case of the DDoS attack. Once the attack gets over, the isolated environment again shifts the VM back to its real place. On the other hand, there are characterizations which shows the exploitation of VM migrations using DDoS attacks [105]. Other solutions such as [104] show a different flavor migration using migrating proxy entry points at overlay networks at the victim server-end.

Victim migration to backup resource provides a way to control the attack effects and employ the attack mitigation. Also it may help in scaling the services using migrating to a large sized candidate/host servers where the migratee server can use the additional resources to detect and mitigate the attack.

There are few issues related to the sustainability of these schemes. In particular, wastage of additional resources, which has to be available all the time is a major issue. Detection of the DDoS just by keeping a watch over resource utilization might not be a good idea, as there might be higher utilization because of real traffic during flash events or heavy computation needs. In fact, this behavior might lead to an unnecessary duplication to an isolated environment. Even the overhead of duplicating the system when the attack is evident might not be a wise step to overcome the security of the server and application. If the attack continues, how would server serve its legitimate consumers who are trying to access the service at that point in time? If it does not serve them then "for how long, the service will be down?" is an important factor. Additionally, if it serves them then there is a large overhead of transferring states and keeping data and sessions up to date.

2.8.3 Software Defined Networking (M3)

Software Defined Networking (SDN) is an emerging reconfigurable network paradigm which may change the whole DDoS mitigation space. SDN in its core separates data and control planes of switching to support the network reconfigurability on the fly.

There are few initial and ongoing works related to SDN assisted DDoS mitigation mechanisms. The authors in [98] proposed a SDN-based solution in which ISP-level monitoring of traffic and routing of malicious traffic is done to specially designed secure switches. In this work, the victim is required to request ISP for DDoS mitigation. ISP having an abstract view of incoming traffic applies the traffic labeling using OpenFlow switches. The suspicious traffic is then redirected to security middle-boxes which apply the access policies on the traffic. The authors have left the detection and mitigation part on the customer side. A similar proposal by authors in [99], suggested a prototype implementation of SDN-based detection mechanism. The major idea of this work lies in the strict access control policies for the incoming traffic which requires strict authentication for each incoming request. Advanced deep packet inspection based approaches using SDN are discussed in [106]. A detailed tutorial and guideline of SDN-based solutions are given in [17].

SDN as a paradigm has immense possibilities of support for the attack mitigation for massive as well low-rate DDoS attacks due to its reconfigurability and quick networks view and monitoring.

Mitigation Solutions utilizing SDN capabilities are still evolving and may become very helpful due to their important features. However, studies such as [101] show that even the SDN infrastructure itself can become a victim of DDoS attacks.

2.8.4 DDoS Mitigation as a Service (DMaaS) (M4)

There are multiple cloud based services/third party services which are capable of providing the DDoS protection [26] [24] [27]. Mostly, DDoS protection is done on a server or an intermediate node forwarding packets to the server. There are solutions which are hosted in the cloud and provide DDoS mitigation as a service [64] [102]. Multiple providers in the market offer this facility. However, all these mitigation methods are threshold/count based or human intervention based.

On the other hand, there are not many specific products available to mitigate DDoS targeting a cloud. The authors in [103] proposed a DDoS mitigation service. This service aims to help the physical on-premise firewall to do the mitigation quickly. The proposed solution is termed as a hybrid firewall, which uses both physical firewall and virtual firewall (placed in the cloud). Amazon has started providing resource limits on EC2 instances to provide an initial solution. There were multiple requests from consumers to cloud providers about keeping cap or limit on maximum allowed resources and subsequently there were additions from cloud providers related to resource consumption limit alerts to customers [63]. Additionally, Amazon has created a service, cloudWatch [62], to provide real-time information about various metrics towards a service hosted in Amazon cloud to help the solutions to take necessary mitigation steps.

Third party mitigation services or DDoS mitigation as a Service may become very helpful for attack mitigation and recovery using a on-premise tools and/or cloud based solution. The attack mitigation history and expertise in handling various attacks may become helpful for enterprises seeking specialized help. Also the cloud based service may also utilize the extensive resource support available in the cloud.

The major limitation of these DMaaS approaches include remote mitigation which may not fasten the mitigation process. Additionally, victim service owners may not want to share the control with the third parties due to the privacy issues of their traffic and the business logic. Other important aspects include the cost of the solutions and the sustainability requirements of the victim enterprises. In addition to all the categories of mitigation methods, shutdown is a typical trivial method to stop the DDoS attack on a server. But this method does not provide any solution to the service downtime of the service which is non-negotiable. In some approaches, the victim server is started at another place as a new instance and present instance is shut down. This helps in starting a synced clone at another place. Though there are high chances that the attacker will also attack the new server. A similar idea has been proposed in [54] where attacked proxy servers are shutdown and the traffic is redirected to new proxy servers.

Attack mitigation methods narrated above provide a detailed overview of various attack mitigation and recovery solutions available in cloud computing space. The mitigation methods are usually a supportive layer of protection for the attack prevention and detection solutions.

2.9 Discussion and Future Directions

We referred a large volume of work while preparing this survey. With this rigorous survey, it is clear that most of the contributions in this domain are concentrating on the following five aspects:

1. Characterization or Impact study.
2. Prevention using Turing Tests.
3. Threshold or pattern based filtering.
4. Support to stop IP spoofing.
5. Resource scaling.

Most of the solutions proposed so far are using one or a combination of the above approaches. There are only a few solutions which are including the auto scaling, multi-tenancy and utility model into account. The cloud computing infrastructure may be used to build effective mitigation solutions which ensure the quick attack mitigation and timely recovery to ensure effective service availability.

2.9.1 Solution Considerations

In order to offer an effective solution to DDoS in the cloud, the following features require special treatment. Here, each feature has been discussed with an intention to provide an aid to the ideal solution.

2.9.1.1 Auto-scaling

Auto-scaling in the cloud is usually triggered by monitored metrics of a VM or an application running inside a VM. These are resource usage metrics like CPU, memory and bandwidth and other counters like response time, query processing time etc. Triggering the auto-scaling would either result in an increase or decrease in allocated resources. Controlling Auto-scaling or false triggering of auto-scaling requires specific checks which can verify the real usage. These checks are available at VM level, hypervisor level or even at abstract cloud level.

- Vertical Scaling: This feature deals with the scaling on a physical server where multiple VMs are running with co-hosted isolations. Vertical scaling would deal with adding or removing resources on these VMs. Total resources which are available on the physical server are fixed but each VM may have a different amount of resources at different times. This really depends on the resource allocation policy and the SLA. Any DDoS affected VM would continuously request for more and more resources and available idle resources (with the Cloud Service Provider) should fulfill these requests.
- Horizontal Scaling: This scheme allows adding new instances of the same VM at other physical servers. These instances are created to share the load and maintain the quality of web services. An ideal composite scaling strategy would first rely on vertical scaling followed by horizontal scaling. The decision-making process to start more instances on more servers should look for a true need and cost considerations. Another important point in horizontal scaling is limiting the maximum number of instances of an application. This can be decided by the cloud consumer based on the budget but a restriction on it may lead to losing the business.

2.9.1.2 Multi-tenancy

Multi-tenancy leads to proper hardware utilization of high-capacity servers which would have been underutilized if not implemented as multi-tenant environments. Vertical scaling would have much flexibility in case few VMs are running on a single machine. On the other hand, cloud providers would have ROI (Return on Investment) considerations and would want to host more and more VMs. Other than this, performance isolation and performance interference aspects should also be looked carefully while designing capacity of these servers. DDoS defense mechanism and its design should reflect protecting multi-tenant environments.

2.9.1.3 Pay-as-you-go model

Pay-as-you-go model is advantageous for both consumers and providers. Literature has mostly counted pay-as-you-go models as an advantage for consumers. But this becomes advantageous for a cloud provider when VMs it has hosted in its cloud requires more and more resources on a regular basis. In case, this additional requirement is fulfilled than the consumer needs to pay for additional resources and provider gets benefited. Almost all solutions should keep the accounting and billing model in the perspective while designing cost-aware DDoS defense solutions.

2.9.1.4 Migration

As described in the Section 2.8, VM migration is a very important method to minimize effects of the DDoS in a virtualized cloud. Migrations incur a cost in terms of downtime, configuration changes, and bandwidth usage. If the application does not have the capability to start more instances to share the load, migration is the only way to minimize the downtime and denial of service. As horizontal scaling cannot be done in such cases, the duration for which DDoS attacks lasts would also play a major role. Large attack duration may lead to multiple subsequent migrations here and there, and thus a large number of side-effects to the cloud and other VMs. DDoS defense mechanism should be able to minimize the number of migrations during the attack period by closely working with horizontal scaling.

2.9.1.5 Solution Costs

The most important motivation for the enterprises to shift their service to cloud infrastructure is the cost effectiveness. However, the DDoS attack losses may become multi-fold in the cloud infrastructure as compare to traditional on-premise infrastructure. The major portion of the cloud users include small and medium enterprises which necessitates the sustainability or budget factor as important aspect while designing the solutions.

2.10 Conclusion

This work provides a comprehensive and detailed survey about the DDoS attacks and defense mechanisms available in the cloud computing environment. We have shown through the discussion that EDoS attack is the primary form of DDoS attack in the cloud. DDoS attacks have important characteristics which play an important role while considering utility computing models. This chapter introduces the fundamental aspects of cloud computing which are critical in order to understand the DDoS attack and its impact.

We also presented attack statistics, its impact, and characterization by various contributors. Attack and threat model of DDoS attack in cloud computing is presented in detail to understand various attack features and threats. We prepared a comprehensive taxonomy of works related to DDoS defense mechanisms in cloud computing. We believe that this survey would help to provide a directional guidance towards requirements of DDoS defense mechanisms and a guideline towards a unified and effective solution.

There are a large number of solutions which targeted the DDoS attack from one of the three solution categories of attack prevention, detection, and mitigation. Among these solutions, there are few contributions which are targeting at cloud-specific features like resource allocation, on-demand resources, botcloud detection, and network reconfiguration using SDNs.

This survey plays an important role in providing the basis for the innovative and effective solutions to prevent and deter DDoS attacks in cloud computing. Characterization at the level of a cloud as a whole and multiple clouds would really help in understanding the impact of this attack at a larger level. As discussed in the survey, solutions specifically designed for cloud and its features would surely perform better as compared to traditional DDoS solutions. Cost and attack aware resource allocation algorithms in the cloud would help in mitigating the attack. In the next chapter, we provide details of our results on attack effect characterization of DDoS attacks in the cloud.

Chapter 3

DDoS attacks in Cloud Computing: Collateral Damage to Non-targets

3.1 Introduction

In this chapter, we describe our first contribution to characterize and show the various attack effects on various stakeholders. These effects include both direct and indirect impacts while a DDoS attack is occurring. It is well established that the consequences of the DDoS attack will affect the target server and the services offered. We show that this is not a correct and complete evaluation of attack consequences with servers hosted in infrastructure clouds. An infrastructure cloud will always have multiple "multi-tenant" physical servers hosting a large number of virtual machines (VMs) sharing various resources using different techniques of resource multiplexing and sharing. The control of each VM would belong to its owner, and it should ideally be isolated from other resource-sharing components. However, the multi-tenant and collaborative nature of the cloud makes a large difference in characterizing DDoS attacks as compared to traditional infrastructures. We also make an effort to provide a guideline to distinguish, measure, and minimize these effects, as these effects would change the factors governing cloud pricing, IT chargeback, and dispute resolutions. This chapter provides a comprehensive analysis of the various aspects of DDoS attacks in cloud computing and solutions.

This chapter is organized as follows: Section 3.2 describes DDoS attacks in the cloud. Section 3.3 presents a system model of cloud infrastructure to support the arguments presented in this paper. We list a detailed set of requirements for planned experiments in Section 3.4. Experiments and results are discussed in Section 3.5. Quantification of the results and their applicability contexts are discussed in Section 3.6. Section 3.7 is devoted to identifying the requirements to mitigate DDoS attacks in cloud computing. Finally, Section 3.8 presents the conclusions.

3.2 DDoS Attack in the Cloud

A DDoS attack targets a victim server by sending it a large number of service requests from a group of distributed clients/bots. In an infrastructure cloud, the attack scenario would be similar to the one shown in Figure 3.1. An infrastructure cloud will have multiple high-capacity physical servers hosting VMs to meet the business objectives including return on investment and better hardware utilization. This cloud has a high-speed network to connect these servers to support applications and processes such as live VM migration. Usually, a cloud will have a queue of incoming VMs that will be placed on the servers. The cloud as a resource manager has multiple activities to perform, including VM placement, resource allocation, load balancing, accounting, and billing.

For our discussion, let us concentrate on physical server 3, which hosts four VMs in Figure 3.1. VM1 is being targeted by a DDoS attack. A DDoS attack is usually achieved by targeting one or more of the basic server resources such as the CPU, memory, disk, bandwidth, number of TCP connections, open files, etc. It is important to note that the attack mechanism from an attacker's perspective will mostly remain the same for both on-premise fixed infrastructure and scalable cloud infrastructure.

This makes it easy for an attacker to invest resources using the same attack strategies. However, for mitigation systems, it becomes a different task altogether. Infrastructure clouds are mostly appreciated for their capability to shrink and expand resources by offering on-demand computing facility. Most of these basic resources are being shrunk and expanded by cloud service providers using virtualization and auto-scaling mechanisms [19, 107–109]. This would help the hosted VM avoid reaching a "denial-of-service" (DoS) state by increasing the resources to cope with the increasing demand.

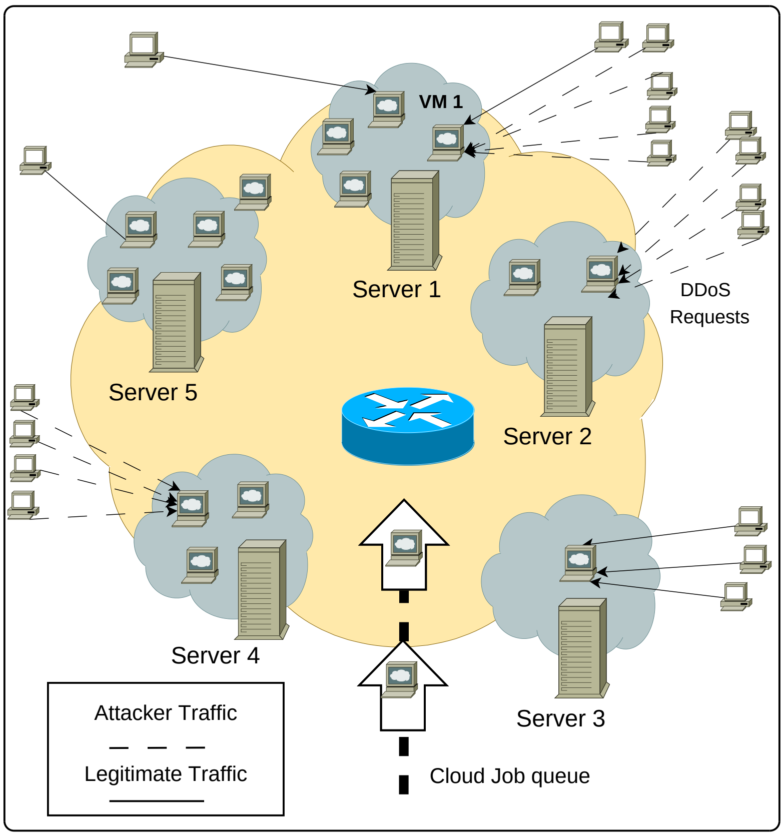


FIGURE 3.1: DDoS Scenario and Various Stakeholders

Even if the resources are exhausted in server 3, the DoS state may not be reached as the service running on VM1 can be scaled up vertically or horizontally on some other server, say, server 1, by creating another VM instance. Theoretically, this may be done for infinite resources and instances owing to the availability of profound resources on the cloud. Obviously, these shrinking and expansion are linked to accounting and "pay-as-you-go" billing, attacking the economic sustainability of the VM owner. Practically, this attack may result in DoS as the maximum allowed resources to a VM owner cannot be infinite.

These intermediate steps of continuous resource acquisition are important reasons behind EDoS attacks, which may or may not converge to a DDoS attack. This can be effectively understood by the resource allocation charts in Figure 3.2. A web server instance (instance 1) has been started on a physical server in a cloud with basic resources of one unit. For simplicity, this one unit of resources represents a set of basic resources needed for one instance of a web server. As the web server is a virtualized server, the total resources are assumed to be 20 units, where four units are always reserved for the hypervisor or host domain and eight units are already occupied by other VMs.

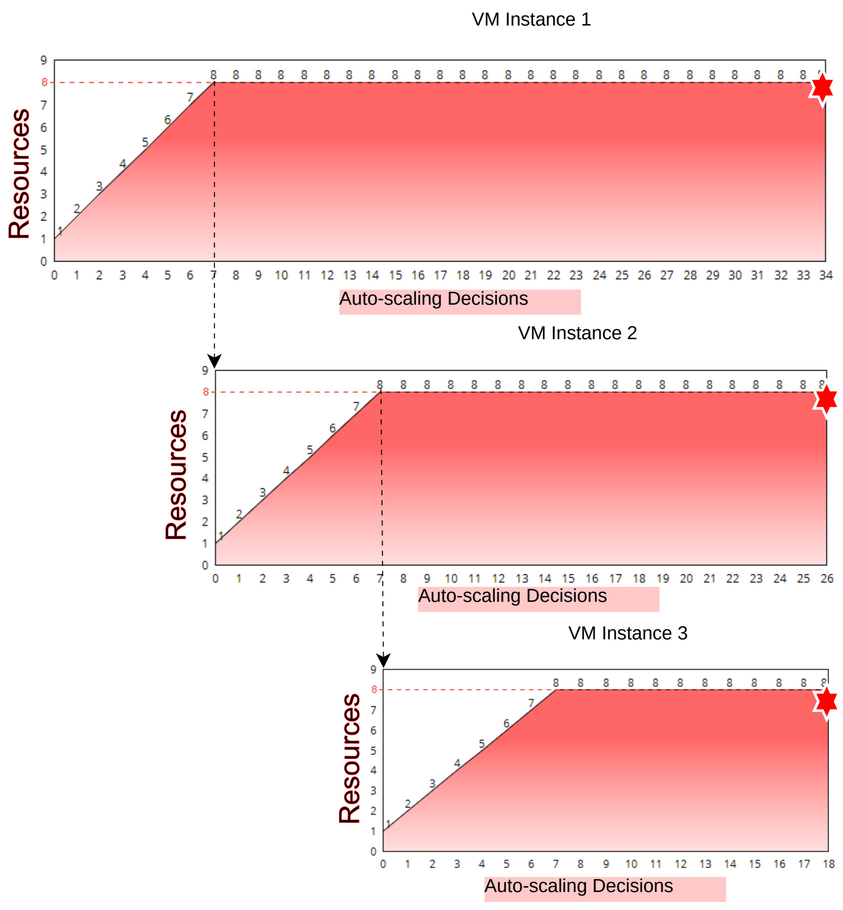


FIGURE 3.2: EDoS and DDoS in the Cloud

Hence, there are only seven units of idle resources left that are available to all VMs to acquire when needed. The auto-scaling decisions represented on the x-axis are taken by an algorithm for keeping track of VM resource utilization and other metrics (similar to the scheme in [110]). While the attack is being applied, each of these decisions would always result in "expansion." As shown, the attack would consistently stress upon one or more of the basic resources and trigger the acquisition of all seven units of idle resources. Once this state is reached, there are no more idle resources available. Now, the auto-scaling algorithm would start another VM instance on the same server or some other server in the cloud. The location of the new instance creation is dependent upon various factors including candidate server identification, VM placement algorithm, future resource requirements, etc. Once another instance is started, there will be two parallel instances that are serving the incoming requests for the same service. Assuming that there is no DDoS mitigation service in place as the attack is being continued, this will also exhaust the resources available for the newly created instance, resulting in the creation of more and more instances.

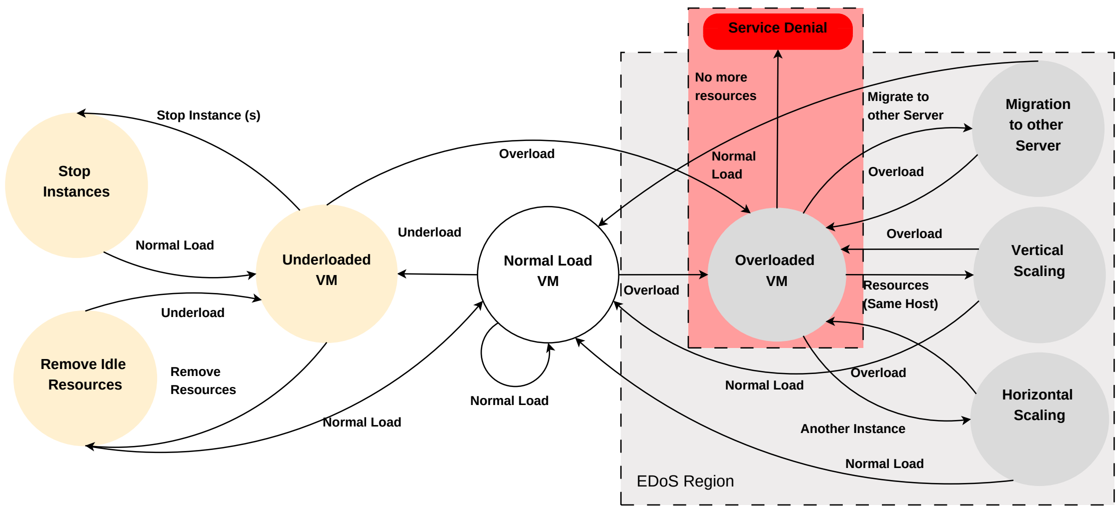


FIGURE 3.3: State Transition in Cloud Auto-scaling

Theoretically, it may eat up all the resources of the cloud or all the resources allocated to the victim VM's owner. Finally, the web server in consideration would reach a state where service denial would happen (represented by a red star in Figure 3.2).

In practice, a VM may not choose a model based on this discussion. Instead of creating more instances, it may choose migration to another server with more idle resources or a hybrid strategy combining all these approaches. Even in this case, the convergence from EDoS to DDoS will follow the same path. There are two matrices that are important to extend our discussion. Both these matrices are used in the next section for the development of the system model.
1. Time to reach DoS: The time required to reach a DDoS attack in the cloud will be higher than that in a traditional infrastructure (as there is only EDoS between auto-scaling decisions 1 and 7, and the service may be able to serve the requests).
2. Attacker target: If the attacker's aim is not toward service denial, it may send requests at a lower rate to realize the EDoS attack, which would economically harm but would not converge to DDoS.

3.3 System Model

Considering an infrastructure cloud C that has P_n physical servers (similar to Figure 3.1), we can represent each individual server as

$$P_i, i = 1, 2, ...n. \tag{3.1}$$

Similarly, the set of VMs that will actually run on these machines is V_m and these VMs are represented as

$$V_j, j = 1, 2, ...m. \tag{3.2}$$

Each machine P_i may have p resource types available. Similarly, a VM, V_j, may require q resource types using virtualization. Therefore, a physical server's resources will be

$$P_{ik}, k = 1, 2, ...p \text{ resource types..} \tag{3.3}$$

Similarly, a VM's resources would be represented as

$$V_{jl}, l = 1, 2, ...q \text{ resource types.} \tag{3.4}$$

Usually, resource types are the CPUs (C), memory (M), disk space (D), and bandwidth (B). Additional resources can be added as per need. Therefore, a host P_i will have

$$P_{i1} = C_i, \qquad P_{i2} = M_i, \qquad P_{i3} = D_i, \ and \qquad P_{i4} = B_i. \tag{3.5}$$

Similarly, a VM V_j will have

$$V_{j1} = C_j \qquad V_{j2} = M_j \qquad V_{j3} = D_j, \ and \qquad V_{j4} = B_j. \tag{3.6}$$

The total resource capacity of a physical host would be the total resources available at the host

$$Cap(P_i) = (C_i, M_i, D_i, B_i). \tag{3.7}$$

Similarly, the resource capacity of a VM would be the resources allocated to it

$$Cap(V_j) = (C_j, M_j, D_j, B_j). \tag{3.8}$$

The additional resource requirement of a VM, V_j, would be (positive for expansion and negative for shrinking)

$$Require(V_j) = (C'_j, M'_j, D'_j, B'_j). \tag{3.9}$$

VM placement activity is performed while the VMs arrive in the cloud. Assume that the maximum number of VMs that a hypervisor can support is r. Out of the whole set of VMs V_j, only a few VMs in the subset V_s, s=1, 2,......r, can be placed on a server P_i, if

$$Cap(P_i) \geq \sum_{s=1}^{r} Cap(V_s). \tag{3.10}$$

Moreover, all of the following should also hold:

$$C_i \geq \sum_{s=1}^{r} C_s \tag{3.11}$$

$$M_i \geq \sum_{s=1}^{r} M_s \tag{3.12}$$

$$D_i \geq \sum_{s=1}^{r} D_s \tag{3.13}$$

$$B_i \geq \sum_{s=1}^{r} B_s. \tag{3.14}$$

The requirement of VMs is met by idle resources. If the subset V_s is successfully placed on P_i, then the idle resources on P_i would be

$$Idle(P_i) = Cap(P_i) - \sum_{s=1}^{r} Cap(V_s). \tag{3.15}$$

Moreover, once the resources are allotted to VMs, during their runs, they will be continuously monitored using an auto-scaling algorithm. Additional resource requirement (Equation 3.9) will be considered for the fulfillment (out of idle resources calculated in Equation 3.15) if the following holds true:

$$Idle(P_i) \geq \sum_{s=1}^{r} Require(V_s). \tag{3.16}$$

Similarly, the idle resources of a VM should be removed to help the economic viability of cloud solutions. This removal will add the idle resources to the idle pool of resources of the cloud to further allot them to needy consumers when needed. An "overload" state would arise if one or more equations out of equations 3.11,3.12, 3.13m and 3.14 do not hold true for one or more VMs. Figure 3.3 shows a detailed depiction of resource scaling and shrinking in a cloud infrastructure. Equation 3.17 shows how the overload, underload, and normal load states affect $Cap(V_s)$ using the auto-scaling strategy. U is a utilization metric usually based on CPU usage; however, it may be a different metric based on one or more similar matrices. Identifying "overload" or "underload" conditions is usually decided by the user requirements and by the static and dynamic thresholds such as CPU utilization, response time of the web application, and file upload time [19]. V_{add} and V_{remove} are dependent upon the requirement and supported scaling techniques.

$$Cap(V_s) = \begin{cases} Cap(V_s) + Cap(V_{add}) & \text{if } U >= U_{overload}. \\ Cap(V_s) - Cap(V_{remove}) & \text{if } U <= U_{underload}. \\ Cap(V_s) & \text{if } U = U_{normalload}. \end{cases} \tag{3.17}$$

Auto-scaling would perform one of the following three possibilities or a combination of them.

1. **Vertical scaling:** In this scaling method, resources are added on top of the VM at the same physical server. These resources are added from the $Idle(P_i)$ pool of resources on server P_i. In the presence of an attack, Equation 3.16 may not hold true after regular vertical scaling. In that case, either horizontal scaling or migration is the only available option to respond to the demand.

2. **Horizontal scaling:** In this strategy, usually a cloned instance of the VM is created on a server other than P_i. This scaling strategy is only applicable to multi-instance applications with load balancing scheme in place. Most of the cloud providers use horizontal scaling by quickly cloning the VM instances. The newly created instance will have standard resources from the pool of instances that the cloud provider supports.

Finding out a candidate physical server $P_{candidate}$ to start a cloned VM instance is part of the VM placement problem. A large number of approaches [23] are available to find a server that supports the required resources for a new VM instance by justifying the requirements with the idle resources on the server. A minimum resource requirement of a candidate server has the required idle resources to support the VM instance (eq. 3.18).

$$Idle(P_{candidate}) \geq \sum_{s=1}^{r} Require(V_s). \tag{3.18}$$

3. **Migration to another server:** This method is quite similar to horizontal scaling except that it does not require the creation of an instance. The same running instance is migrated to another server where the required resources are available (Equation 3.18). Therefore, the problem of identifying a candidate server remains the same as in the case of horizontal scaling. There are multiple factors that need to be considered before performing a migration. Some of these factors include the possible future requirements of resources, consequent migrations or swaps, and the migration costs.

3.4 Motivation and Planning of the Characterization

In this section, we present the motivation and planning of the experiments to characterize and quantify the effects. The major aim of these experiments was to see whether the non-target stakeholders of a cloud infrastructure are affected by a DDoS attack. Additionally, the aim of these experiments was to quantify and assess the damage caused. We have identified all the important actors in the DDoS attack scenario in the cloud (also shown in Figure 3.1). These important components are listed in Table 3.1.

The impact characterization experiments were planned in such a manner that the following four categories of effects could be observed in detail: performance issues, additional costs incurred, indirect effects, and effects that are invisible at the moment but have a long-term impact.

1. **Performance**: The performance of a hosted service in a cloud may have multiple factors to measure performance or service quality. It may range from response time to number of concurrent users, timeouts, failures, and number of sessions.

Stakeholders	**Attack target**
Victim server	Direct
Victim server owner	Direct
Users of victim VMs	Direct
Co-hosted VMs	Collateral
Host physical server	Collateral
Other physical server(s) in the cloud	Collateral
VMs on other hosts	Collateral
Network and network devices	Collateral/direct
Users of co-hosted VMs	Collateral
Cloud provider	Collateral/direct
Cloud customers	Collateral

Table 3.1: DDoS in the Cloud: Stakeholders

Accessibility and timely response are two of the most important factors for a web service. These factors are based on other factors including the design of the web page, server performance, and network bandwidth. Conversion rate and web reputation are generally associated with the performance of web service response time [111, 112]. The major aim of our experiments was to determine the performance penalties on all components of a cloud platform while an attack was present. Factors that affect performance include arrival rate of incoming requests, fewer resources, and other performance-deteriorating functions like migration and swaps.

2. **Costs**: Cloud computing infrastructure is one of the most sought-after technology platforms for enterprises today. This is mostly due to the cost benefits it provides to VM owners and the resultant return on investment (ROI) [113]. In "pay-as-you-go" pricing models, costs are directly associated with resources used. CPU, memory, storage, and bandwidth are four important resources that are used in the cost calculation. This is an important factor to look for when considering the economic aspects of EDoS.

3. **Collateral/indirect effects**: These are the effects that are to be quantified on the non-targets, which are components of cloud architectures except the victim VM. The effects of the interest include both performance and cost-based effects.

4. **Invisible effects**: These are effects that are not visible in the first instance when the attack has appeared. End-user satisfaction, quality of service, downtime impact on business and long-term impact on reputation, penalties,

and disputes are a few of the important impacts that are not visible directly while the attack is occurring.

In most of the instances, these effects are not measurable as their real impact is dependent upon multiple factors including business agreements and time.

3.5 Experimental Design and Results

Effect quantification is dependent upon multiple factors such as the size of the cloud, applications, resource allocation strategies, type of attack, and attack strength and its duration. For an effective understanding and classification, the following two experiment sets were planned. The first experiment was planned on a single physical server hosting multiple VMs to quantify server level local effects. The second experiment set was a cloud-scale experiment that was conducted to see the effects of DDoS/EDoS on cloud infrastructure as a whole. This would allow us to see the attack effects from both a local and an abstract perspective.

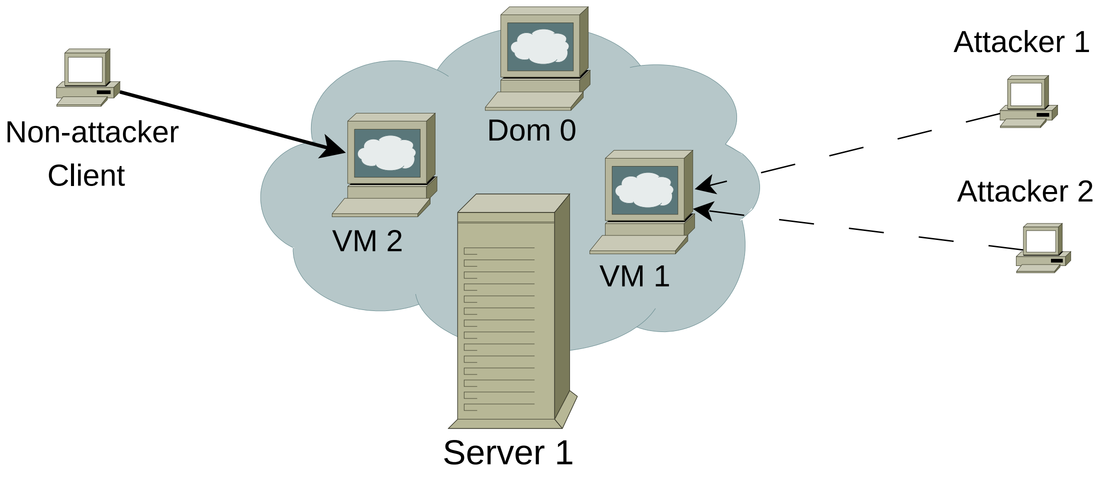


Figure 3.4: Experiment Setup 1: Single Physical Server

3.5.1 Experiment Set 1: Single Physical Server

The main aim of this experiment was to study the attack impact on a DDoSed victim server, co-hosted VMs, and the physical server hosting these VMs. This experiment was conducted using the setup shown in Figure 3.4 and the configuration given in Table 3.2.

Item	**Configuration**
Physical server	HP EliteDesk i7 3 GHz
Total CPUs	(4 Cores, 8 VCPus)
Total memory	4 GB
Hypervisor	XenServer 6.2
Host/Guest/Attacker OS	Ubuntu 14.04 Server
Host CPUs	4 VCPUs (any CPU)
Host memory	732 MB
Guest configuration	As specified in Figure 3.5
Guest application	Apache2
Attackers	Dual Core (2 GB)
Attacker application	ApacheBench2
Request concurrency	1/10/50/250 concurrent
Total requests	1000 in each set
Request size	2 MB
Network	100 Mbps
CPU Affinity	Pinned as specified in Figure 3.5

Table 3.2: Experiment Setup 1: Single Physical Server

For this, a web application's performance was compared when it was hosted alone (Experiment 1A), hosted with another VM with a similar load (Experiment 1B), and hosted with another VM that was under attack (Experiment 1C). These three experiments would enable us to understand the performance variations after the application's co-hosted VM started suffering from a DDoS attack. A test would comprise 1000 requests to be sent by clients to the web server. These requests would be sent to the web server with different concurrent requests (1/10/50 requests/s representing low/moderate/high load, respectively).

The attack consisted of 2000 requests (1000 requests each from two attackers). Each attacker was sending these requests with a concurrency of 250 requests, each of which generated a very heavy load for the given resources. The attacker traffic was planned using the guidelines given in the popular literature [41].

3.5.2 Results: Single Physical Server

The completion time of each test is shown in Figure 3.5 and Table 3.3 for various resource combinations across the three tests. For a web server serving static pages, the basic resources are the bandwidth and disk reads/transfers. Therefore, there is not much impact of increasing CPUs and memory. In the case of a dynamic web server, the basic resources may be extended to CPU cores and memory. Results showed that there was a significant increase in the completion time of the test due

Configuration R: Requests V: vCPUs M: Memory (MB)			**Experiment 1A**	**Experiment 1B: Two VMs**		**Experiment 1C Two VMs &** VM1 (attacked) **F: Request failures**		
R	V	M	T_{VM1}	T_{VM1}	T_{VM2}	T_{VM1}	F	T_{VM2}
1	1	512	181.9s	364.3s	351.2s	491.0s	0	587.7s
1	2	512	181.9s	357.8s	355.8s	432.5s	90	587.0s
1	3	512	181.8s	356.5s	356.7s	457.8s	0	587.3s
1	4	512	181.8s	357.0s	357.6s	427.4s	357	586.8s
1	1	1024	181.7s	356.6s	356.9s	491.7s	0	587.5s
1	2	1024	181.6s	356.6s	356.7s	451.9s	0	587.0s
1	3	1024	181.8s	357.9s	355.9s	501.9s	0	588.4s
1	4	1024	181.8s	356.9s	357.3s	508.5s	2	588.3s
10	1	512	178.7s	343.9s	349.8s	589.5s	468	357.3s
10	2	512	178.8s	355.3s	356.2s	657.4s	327	357.1s
10	3	512	178.8s	357.3s	357.3s	252.7s	0	357.3s
10	4	512	178.8s	343.3s	357.5s	200.5s	96	357.3s
10	1	1024	178.8s	354.1s	357.4s	254.0s	0	357.4s
10	2	1024	178.7s	357.3s	357.4s	578.1s	864	357.1s
10	3	1024	178.7s	355.5s	356.2s	584.0s	825	357.2s
10	4	1024	178.7s	327.7s	357.5s	225.5s	54	357.6s
50	1	512	178.9s	357.7s	357.5s	252.6s	381	357.3s
50	2	512	178.8s	351.8s	351.8s	250.4s	396	357.0s
50	3	512	178.9s	353.4s	353.4s	178.8s	0	356.3s
50	4	512	178.9s	331.3s	317.4s	179.0s	18	309.7s
50	1	1024	179.0s	331.8s	328.6s	177.9s	75	351.2s
50	2	1024	178.7s	291.8s	291.6s	221.9s	387	357.3s
50	3	1024	178.8s	343.7s	343.6s	242.4s	387	357.0s
50	4	1024	178.8s	343.2s	313.4s	178.9s	15	344.0s

Table 3.3: Experiment 1: Comparison of Test Completion Times

to contention. The resource contention race became tough from Experiment 1A to Experiment 1B and even tougher in Experiment 1C. Similarly, a significant rise can be seen in Table 3.4 for data transfer rate, connection times, and requests/second. At a later stage, the test completion time decreased owing to request failures. Therefore, victim VM1 could not respond to many of the requests (failures listed in Figure 3.3). This shows the visible effects on the victim server and co-hosted VMs. Other than performance interference by a DDoSed VM, the behavior would trigger auto-scaling (based on the increased response time and requests), resulting in economic losses to the co-hosted VMs even though they were not directly DDoSed.

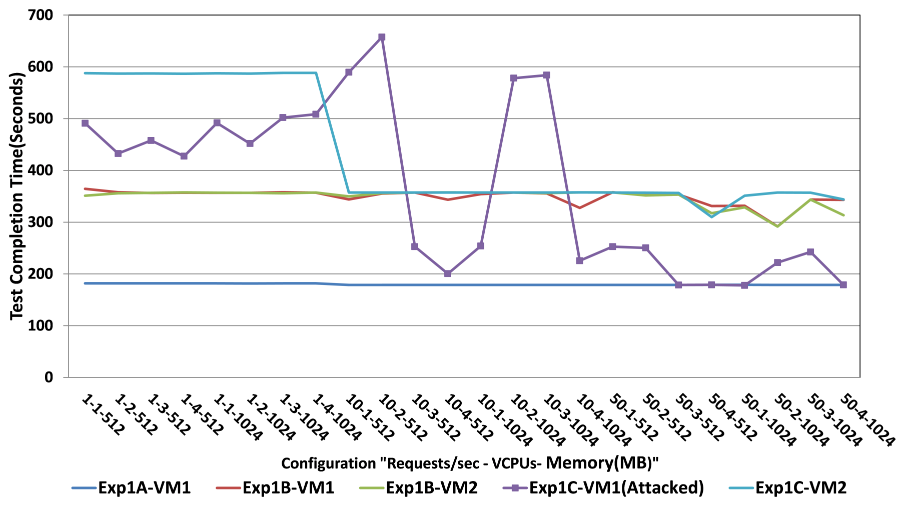


Figure 3.5: Experiment 1: Comparison of Test Completion Times

Experiment, VM	Transfer Rate (Kbytes/s)	Served Requests/s	Connection Times (ms)				
			Min	Mean	+/-SD	Median	Max
1A, VM1	11,260.32	5.5	180	182	2.4	181	223
1B, VM1	5832.58	2.85	181	351	34.6	357	431
1B, VM2	5621.92	2.74	181	364	42.7	357	602
1C, VM1	4171.77	2.04	10,811	107,510	38,643.3	107,256	389,555
1C, VM2	3485.17	1.7	564	588	13.1	585	726

Table 3.4: Comparison of Performance Metrics in Experiment 1

3.5.3 Experiment 2: Cloud Scale

To characterize the overall impact of DDoS attacks in an infrastructure cloud, we conducted a comprehensive cloud-scale experiment. The setup is as shown in Table 3.5 and is similar to that shown in Figure 3.1. The major aim of this experiment was to study the effects on the cloud after some or more DDoSed VMs were introduced in the cloud. The authors of CloudSim [23] developed multiple schemes related to host overloading detection, VM selection for migration, host underloading detection, and VM placement, and tested them with PlanetLab and random traces. As shown in Table 3.5, there are five overload detection algorithms and four VM selection algorithms for migration implemented by Beloglazov and Buyya [23], making 20 combinations in all by choosing these algorithms (5*4).

These combinations are governed by a few constants and input parameters, which were attached to their names by Beloglazov and Buyya [23], like iqr-mc-1.5. While aiming to effectively show the effects, we realized that this would be an ideal setup to use as it can help evaluate the effects of DDoS/EDoS on all these strategies to get a comprehensive insight.

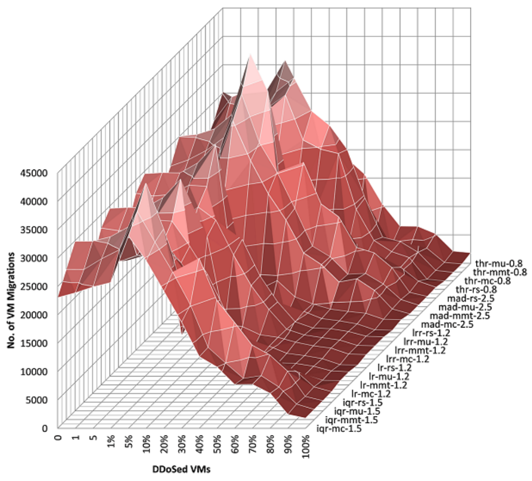
Item	**Configuration**
Simulation environment	CloudSim 3.0.3
No. of servers	800
VM Traces	"PlanetLab" traces of 3 days
No. of VMs in the cloud	Set 1: 1052 VMs (03-03-2011) Set 2: 1516 VMs (22-03-2011) Set 3: 898 VMs (06-03-2011)
Server configurations	1. HP ProLiant ML110 G4 Intel Xeon 3040, 2 cores 1860 MHz, 4 GB (400 servers) 2. HP ProLiant ML110 G5 Intel Xeon 3075, 2 cores 2660 MHz, 4 GB (400 servers)
Overload detection algorithms	THR: Threshold MAD: Median absolute deviation IQR: Interquartile range LR: Local regression LRR: Robust local regression
VM Selection algorithms	MMT: Minimum migration time MC: Minimum correlation RS: Random selection MU: Minimum utilization
Other algorithms	DVFS: Dynamic voltage frequency scaling NPA: Non-power aware

Table 3.5: Experiment Setup 2: Cloud Scale

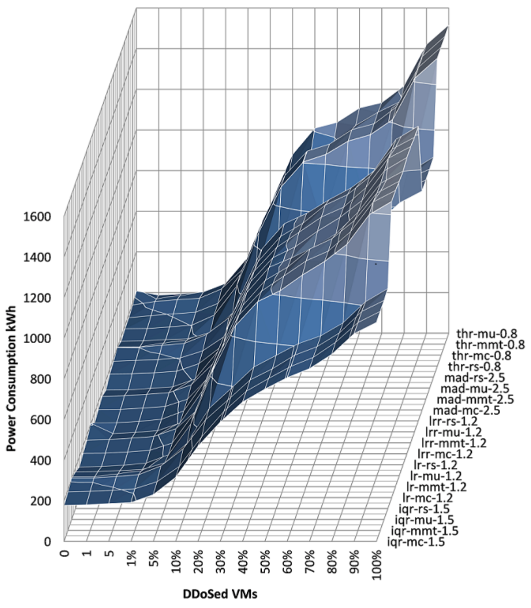
To conduct real quantification and evaluation, we used PlanetLab traces comprising the CPU utilization of VMs with an interval of 5 min. Three sets (three different days) were chosen in these experiments, indicating low (898 VMs), moderate (1052 VMs), and high (1516 VMs) load test cases. The average utilization in all of the PlanetLab traces was below 50%. The DDoSed VMs were introduced by inserting VMs with heavy utilization (100%). It is well established that CPU utilization of this order is achieved during DDoS attacks on computational resources [114]. Two most important metrics for evaluating the performance of a cloud, energy consumption and VM migrations, were evaluated by introducing a few or more DDoSed VMs.

3.5.4 Results: Cloud Scale

The results of three sets of VM traces are shown in Figure 3.6, 3.7 and 3.8. A few result items for set 1 (1052 VMs) are given in Table 3.6 for reference to the charts.



(A) VM Migrations



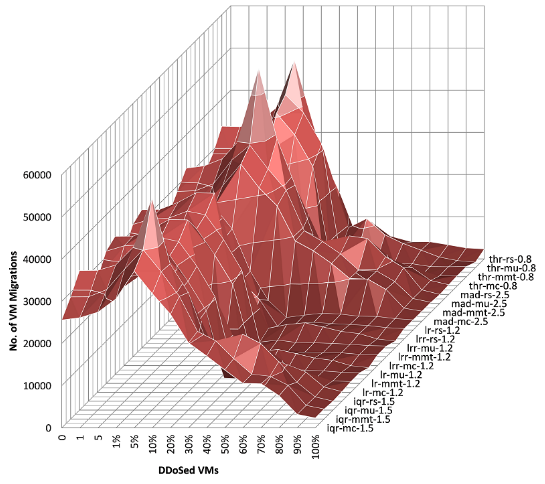
(B) Energy Consumption

FIGURE 3.6: Results of Experiment 2: Set 1 (1052 VMs)

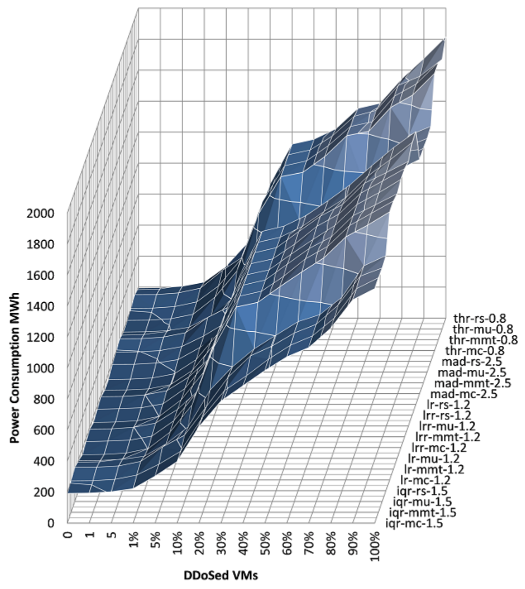
Both energy consumption and number of VM migrations were plotted with different numbers/shares of DDoSed VMs among the normal VMs and 20 sets of overload detection and VM selection algorithms. The non-power-aware scheduler showed 2410 kWh of consumption for all combinations, and DVFS ranged between 600 and 1200 kWh from 0 to 100% of DDoSed VMs (not plotted in the charts). The energy consumption charts show an almost linear increase with the increase in the DDoSed VMs in all the strategies.

Usually, energy consumption is a function of CPU utilization and that is evident in the charts. Similarly, service pricing/costs based on energy may also be calculated. The number of VM migrations was increasing till a certain point (5-10% of DDoSed VMs) and then it started decreasing and reached a minimum number of migrations. The reason for this increase was the introduction of DDoSed VMs at multiple servers in the cloud, followed by the overload situation of VMs given in Equation 3.11,3.12, 3.13, 3.14, and 3.15 requiring migrations (creation of new instances was not implemented in the simulation environment).

After having 10-20% of attacked VMs, the number of overloaded VMs increased but the places or other candidate servers for hosting these VMs decreased. Mostly, the reason is that the other servers were also getting stressed and were not in a position to support the requirements of incoming migratory VMs (Equation 3.16). With an insertion of 5-10% of attacked VMs, the corresponding increase in the number of VM migrations was near 50% (23K to 34K).

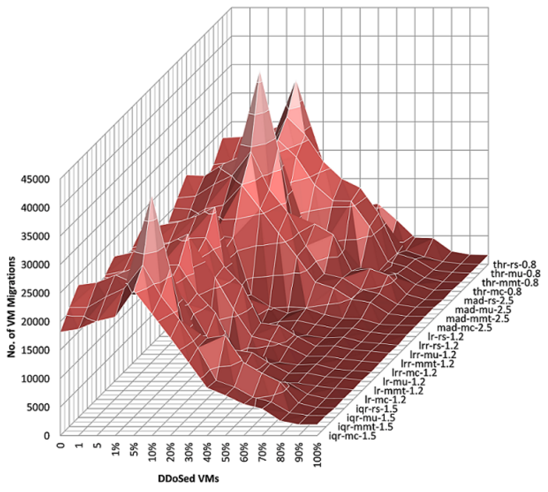


(A) VM Migrations

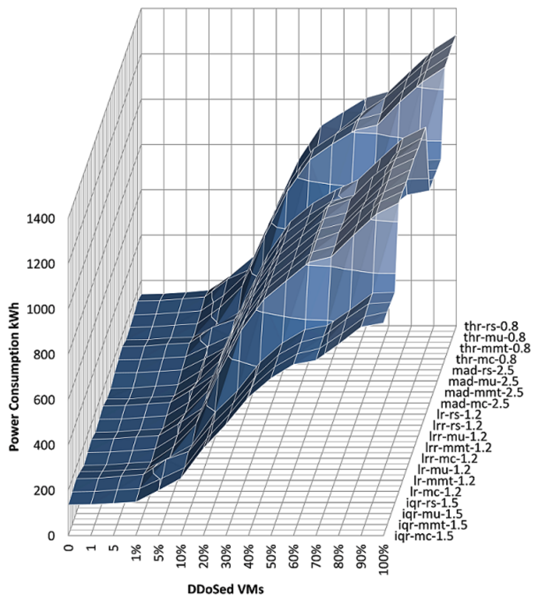


(B) Energy Consumption

FIGURE 3.7: Results of Experiment 2: Set 2 (1516 VMs)



(A) VM Migrations



(B) Energy Consumption

FIGURE 3.8: Results of Experiment 2: Set 3 (898 VMs)

Cloud-scale DDoS attacks and their possibility cannot be predicted, at least for public clouds. There are multiple recent incidents related to Amazon, Greatfire.org, Linode, and Rackspace, which are discussed in Section 3.1. Additionally, multiple works such as [115–117] have strongly reported and evaluated the consequences of cloud-scale attacks.

Energy Consumption (kWh)								
Algorithms	**0**	**1**	**5%**	**10%**	**20%**	**30%**	**50%**	**100%**
DVFS	803.91	788.92	794.81	870.54	906.79	1000.49	1067.91	1205.67
iqr-mc-1.5	177.1	179.9	229.69	314.8	467.58	586.54	749.15	1077.42
mad-mc-2.5	176.13	177.45	227.01	311.05	461.81	583.35	747.78	1077.42
thr-mu-0.8	206.73	186.33	243.88	373.27	631.33	878.07	1037.31	1515.85
No. of VM Migrations								
Algorithms	**0**	**1**	**5%**	**10%**	**20%**	**30%**	**50%**	**100%**
DVFS	0	0	0	0	0	0	0	0
iqr-mc-1.5	23,035	23,881	33,924	29,542	24,142	18,717	10,796	1784
mad-mc-2.5	23,691	24,452	35,828	28,801	22,985	18,343	12,534	1784
thr-mmt-0.8	26,634	30,825	37,323	28,113	4119	5133	5267	1392

Table 3.6: Experiment 2: DDoS Effects on Set 1 of 1052 VMs

3.6 Collateral Damages and Contributors: A Discussion

The experiments in Section 3.5 have shown that the indirect effects on other components of the cloud must not be neglected. The traditional DDoS mitigation mechanisms would not help here as the cloud context would require control over resource provisioning and auto-scaling. The following is a summary of the effects that appeared during the experiments, which should be kept in mind while developing services and mitigation solutions in the cloud.

1. *Victim VM*: Economic losses that may ultimately reach service denial; unnecessary resource buying; performance issues such as low throughput, high response time, and request failures; service downtime; and short-term and long-term business and reputation losses.

2. *Co-hosted VMs*: Indirect EDoS, unnecessary resource buying and performance interference, unnecessary migration/VM instance creation, and performance issues such as low throughput, high response time, and request failures.

3. *Host physical server*: Resource overload situation and higher power consumption.

4. *Other physical server(s) in the cloud*: Indirectly affected by migrated VMs and VM instance creation, which may result in overload.

5. *VMs on other hosts*: Indirectly affected by migrated and new VM instances that became co-hosts to these VMs.

6. *Cloud as a whole*: Heavy energy consumption, running costs, cooling costs, VM migrations, service-level agreement (SLA) violations, and business losses.
7. *Network and network devices*: Heavy bandwidth consumption, network losses, and poor quality of service.
8. *Users of victim VMs*: High response time, poor quality of service, service downtime, and related business/reputation impacts on dependent services.
9. *Users of co-hosted VMs*: High response time, poor quality of service, service downtime, and related business/reputation impacts on dependent services.
10. *Cloud customers*: Business losses, SLA violations, service health, and business difficulties; the cloud could have accommodated and run more VMs; requests to create additional VM instances for existing customers might not be fulfilled.

In the following section, we identify the specific contributors to the characterized collateral damages. These observed scenarios/effects are the major reasons for these effects.

3.6.1 Performance Interference

Performance isolation is a property provided by virtualization [48]. Resource contention among VMs for the basic resources and the resulting performance contention are also important effects to be considered. Performance contention is an outcome of resource sharing. In the presence of a DDoSed VM, which may be mainly stressing a specific resource, say, a disk data transfer, co-hosted VMs would also experience difficulty regarding turnaround time for disk transfers. Web services are usually considered mixed load applications, and, hence, this contention may be visible for multiple resources such as the CPU, disk, and bandwidth.

3.6.2 Resource Race Among VMs

Let us take the case of server 1 in Figure 3.1. This server is running four web server VMs that belong to four different organizations or owners taking advantage of the multi-tenant cloud. When VM1 is DDoSed by attackers, it will ask for more resources and these will be added using auto-scaling.

If the attack continues, it will ask for more resources again and again, thus acquiring the maximum that is available on server 2 for a VM. Finally, VM1 will be flagged for either additional instance creation or migration at other servers (Figure 3.3). At the same time, there may be a stage where other co-hosted VMs—VM2, VM3, and VM4—have some real requirement for resources because of real web requests and will ask for the same. Obviously, this requirement cannot be met by the same physical server and these VMs may also be flagged for either migration or additional instance creation on some other servers.

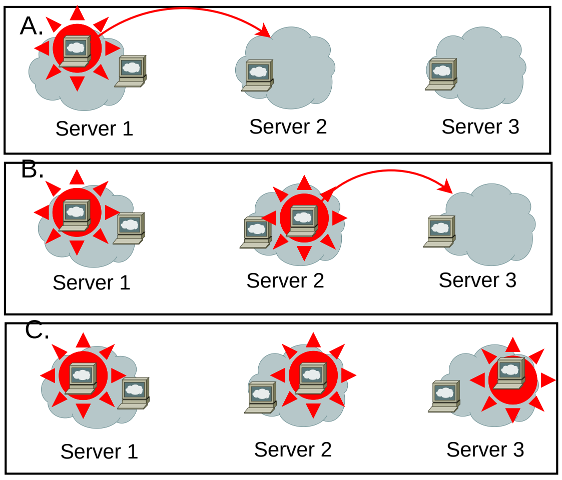
Creating another instance or migrating to another server is an advanced memory transfer/copy process and incurs a downtime. Additionally, they incur resource usage, overhead costs, and downtime of the services. Although resource race is part of a multi-user, multi-tenant system, this fake race, however, appeared only because of one DDoSed VM on the server. The co-hosted VMs that are not a victim of the attack may also need to get migrated or start another instance because of this situation.

3.6.3 VM Placement and Load Balancing

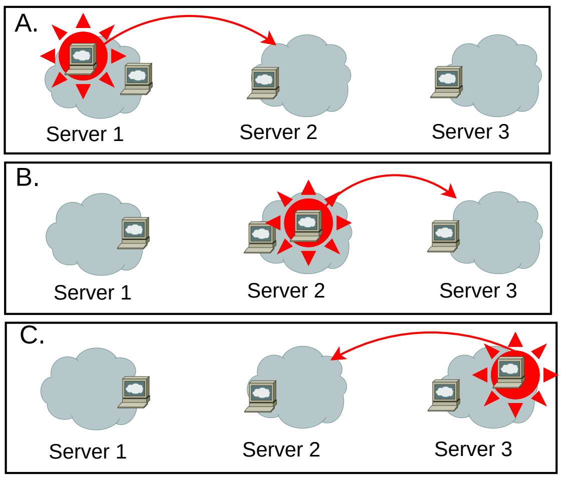
Most of the VM placement algorithms place VMs in a cloud using an optimization method such as dynamic bin packing [118]. Live VM migration is also used as one of the activities of auto-scaling (Figure 3.3). As discussed in Section 3.6.2, DDoSed VMs may result in unnecessary migrations of one or more VMs. There are two possibilities: first, the DDoSed VM may be migrated as a direct effect of DDoS; second, other VMs may be migrated owing to the decision of the migration decision-making algorithms such as minimum migration time (MMT) and minimum utilization (MU) [23]. In the worst case, there might be a swap of VMs or multiple migrations/swaps (reshuffles) among physical servers for an effective load management and resource allocation. To summarize, the following detailed consequences may occur:

1. **New instance/clone creation**: In case the auto-scaling algorithm chooses to create a new instance for a DDoSed VM, the effect will be epidemic and will adversely affect many VMs and the network bandwidth. This can be seen in Figure 3.9a, where the VM under attack (on server 1) is dynamically scaled by creating VM instances on server 2 and server 3. In case a new instance is created for other VMs, obviously, they will be charged for it, resulting in an “indirect DDoS/EDoS.”.

2. **Effect spread and migrations**: A DDoS VM, if migrated owing to the "overload" situation, and if the attack continues, will be migrated again soon to some other server and will continue getting migrated. This will spread the additional effects on other co-hosted VMs and servers epidemically. This scenario is shown in Figure 3.9b, where the VM under target is getting migrated from Server 1 to Server 2 and again to Server 3.



(A) Effect Spread and Instance Creation



(B) Effect Spread and Migrations

Figure 3.9: Contributors to Collateral Damages (starred VM is the victim VM)

3. Migrant selection: It may happen that VMs, except the victim VM, may be selected for migration to other servers. This will directly affect the services offered by them owing to migration cost, downtime, and related overheads. Moreover, if they are migrated because of fake alarms such as a higher response time for a web server, these VMs will suffer from economic losses indirectly, as they are buying extra resources. This extra buying is due to the shared resource being stressed by the co-hosted DDoSed VM.

4. Migrations, swaps, and shuffles: If migrations are converted into VM swaps, it will be a fatal and exhaustive effect for multiple physical servers and VMs. VM swaps are a reality of cloud environments and are needed if a migration is not able to cope with the requirement. A load balancing heuristic is needed to balance the load between two or more physical servers by migrating VMs in between [119]. Large performance penalties will appear on the network owing to the migrations/swaps.

3.7 Solution Space

In this section, we give a detailed guideline and direction for solution design. There are a few recent contributions that meet one or a few of the defined requirements. We refer to them in each of the requirement in which they have made a contribution. However, there is an immense need for solution development that takes all these aims into consideration. We draw lessons from the effect characterization studies given in this work to design this solution space. The following are the major requirements of an efficient DDoS mitigation solution in the cloud, which also aims at minimizing the indirect effects of DDoS attacks on other stakeholders.

1. **Strong isolation**: Performance isolation and resource isolation are two areas that require a thorough relook and design consideration to achieve a strong isolation. To achieve this, we need fair resource sharing and accounting. Pinned or dedicated resources are also one important alternative to provide isolation; however, it is a costly solution that is limited to a few resources such as CPUs and bandwidth [48]. Similar requirements for a strong isolation are provided in [43].

2. **Victim separation**: Cloud incident management systems should handle both incoming DDoS to a VM and any malicious internal VM sending DDoS traffic to an outside network. We need mechanisms at multiple levels to identify the victim and separate it from the other tenants to minimize the effects. Reserved resources, shutdown, and backup servers are a few available solutions in this direction. The solutions provided by the authors in [46, 54, 92] are some of the important contributions that advocate additional resources for mitigation. The authors in [91] provide a victim separation method that migrates the server to backup resources and isolates the service. Additionally, the server is migrated back to its original resources once the attack is over. This approach is similar to shutting the server down, where no efforts are spent to stop the attack. The authors in [97] also provide a backup resource based on a low-cost solution to use resources over the untrusted cloud.

3. **DDoS-aware auto-scaling**: We can see in both the experiments and the system model that the major cause of EDoS attack is incorrect decisions made by the auto-scaling algorithm. Auto-scaling algorithms must be designed by keeping DDoS utilization surges in mind. It has been quite a difficult problem to differentiate DDoS traffic from legitimate traffic. However, it is

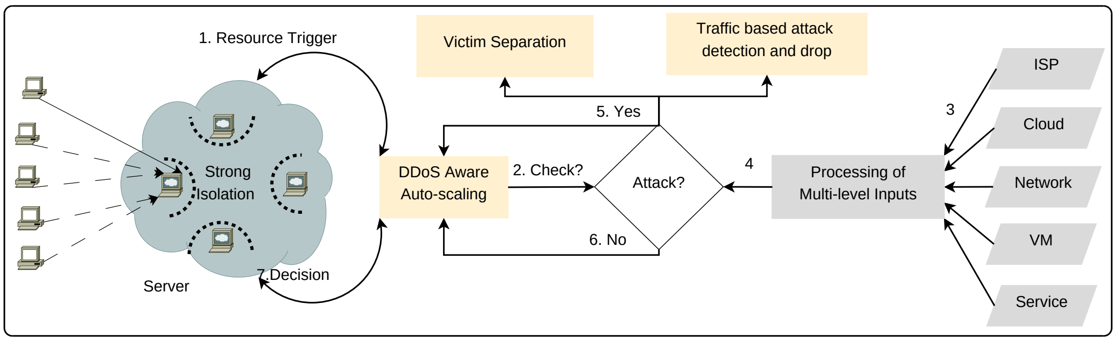


FIGURE 3.10: Solution Space: DDoS Attack Mitigation and Minimization of Collateral Damages

comparably easier to decide the presence of an attack. This fact may help in devising EDoS-aware auto-scaling algorithms [65]. The authors in [65] have provided a solution that uses the presence of the DDoS attack and subsequently decides whether the requirement is due to legitimate traffic or to attack traffic. This helps in wisely taking auto-scaling decisions, while at the same time serving legitimate customers.

4. **Collaborative solutions**: Internet service provider, cloud/hypervisor, network, VM, and application are the five layers of the solution space that, if done collaboratively, can detect and mitigate the attack effects with high assurance. Multi-level solutions with one or more of these levels have been tried by the solutions in [46, 92, 120]; however, efforts are needed to minimize the additional effects proposed in this work.

5. **Verifiable and fine-grained accounting**: Issues related to loss sharing and dispute resolution relate to proper accounting. Most of the cloud players in the market have fixed pricing models that are based on hourly metering. There is an immense need to have fine-grained and verifiable (at the VM end) accounting methods that show the real resource usage. This is especially needed for the resources that are shared. This will help in implementing the pricing models that follow the “pay-as-you-go” billing models. A few important solutions in these directions are given in [121–123].

We have provided an example solution that has the required ingredients to address these requirements in Figure 3.10. We have made an effort to incorporate all the requirements given in 3.7 in this design.

One important observation is that the past contributions in the literature mostly target traffic differentiation to identify the attackers and block them. On the other hand, we argue that, to overcome the collateral damages, we need to have other

important features in addition to traffic differentiation. The primary requirement of our solution is a strong isolation among the co-hosted VMs. This requires no resource contention and performance interference.
1. A resource increase trigger will invoke the auto-scaling algorithm while the attack is in place.
2. Now the auto-scaling algorithm, which is DDoS aware, would ask the attack decision module to check the presence of the attack.
3. The attack decision module has a multilevel information and mitigation system in place, which may help in providing solutions at all the ends.
4. This information is processed and sent to the attack decision module to take a decision.
5. Once it has been detected that there is an attack, the decision module may want to trigger a few important processes.
(A) There is no resource increase to serve the attack traffic. However, there might be a resource increase required to serve the new benign traffic or fasten the mitigation process.
(B) The victim should be separated with a dedicated resource plan, in which resources are not shared with any other VMs.
(C) Traditional traffic-based filtering can be applied to do low-level mitigation.
6. If there is no attack, then the auto-scaling scheme may take decisions according to its traditional auto-scaling practices to maintain service availability.
As our study is a characterization and effect orchestration study, we leave the evaluation and analysis of the solution space open for future research.

3.8 Conclusion

This work provides an insight into the effects of DDoS attacks in cloud computing. In addition to the obvious targets, which are either a victim server or a network, we have argued and shown that almost all the components and stakeholders of a cloud architecture are affected by a DDoS attack. Attack quantification depends on many factors, including the strength of the DDoS attack, victim application, and resources. We have developed a system model of cloud computing resource allocation to help in understanding the role of auto-scaling algorithms in a DDoS attack and its success. This model has also detailed the resource "overload" state of a VM under a DDoS attack and its possible spread using vertical scaling, horizontal scaling, and migrations. Furthermore, features such as auto-scaling, migration, multi-tenancy, resource race, performance interference, and isolation have been identified. These features multiply the impact of DDoS in virtualized infrastructure clouds. It has been shown that multiple unrelated and non-targeted VMs, servers, and users are also affected by a DDoS or EDoS attack in the cloud.

An effort has also been made to differentiate and correlate DDoS and its economic version, EDoS, with the help of factors such as the time to reach DoS and attacker targets. Attack effect spread, migrant selection and overhead of cloning, and migration and swaps were discussed to quantify these effects on experiments and their planning. We kept performance issues, costs, overhead, and invisible effects, and their assessment as major objectives of the experiments. To understand the effects from a microscopic view, we conducted a single-server experiment to determine the effects on the victim server and the co-hosted VMs. Additionally, to have a detailed abstract view, we performed cloud-scale simulations to determine the consequences of DDoS attacks using real workload traces of PlanetLab.

Various algorithms of VM placement, VM migration, and resource allocation were configured with various shares of DDoSed VMs in the workload. These experiments have given an insight into the multiple indirect impacts of DDoS attacks on non-targets such as co-hosted VMs, host physical server, neighboring physical servers, cloud as a whole, network and network devices, and end users of all the services. Performance, cost, power, and various overheads were the major effects that were determined from these experiments. This study provides a strong motivation toward re-examining the design of cloud resource allocation algorithms and strong performance isolation. Additionally, systematic and specific efforts are needed in the direction of mitigating EDoS and DDoS attacks in cloud platforms.

In the end, we provide a solution space and guidelines by creating a line of requirements for an efficient solution. We kept collateral damages in mind while designing these requirements. We assure the suitability of the solution space with the help of individual contributions made in the past. We keep the real evaluation of this design open to future contributions from the security research community. Smokescreening, malware spread, dispute resolution, IT chargeback, SLA designs, and loss sharing are some of the most important aspects that are completely open while looking at DDoS attacks in cloud computing.

In the next chapter, we describe our "DDoS Aware Resource Allocation in Cloud (DARAC)" approach which aims to correct the decisions of traditional auto-scaling algorithms in a manner such that the victim VM does not lead to a DDoS attack. The issue of collateral damage due the DDoS attacks among VMs, have been extended to the level of victim operating system in Chapter 4, 5, and 6.

Chapter 4

DDoS Aware Auto-scaling in Cloud

4.1 Introduction

One of the important characteristic of cloud is elasticity of resources, which enables cloud based services to be scaled to a large magnitude. We see in Chapter 3 that DDoS attacks are successful on cloud services as they dynamically scale their servers in magnitude. DDoS attacks have a different behavior when targeted to cloud. They may not disrupt the target services immediately but may stress consumer's monetary strength. This has been attributed as fraudulent resource consumption in [124]. Cloud consumer, in anticipation that the resource utilization trigger as genuine need, may scale the server and would be trapped in this catch. These types of attacks were first coined by Christopher Hoff in 2008 as Economic Denial of Sustainability (EDoS) attacks. Subsequently, DDoS attacks on cloud was picked up in [124] where the authors explains the fraudulent resource consumption as a threat to cloud consumers.

In this chapter, we argue that DDoS attacks on the cloud should be treated differently. Cloud DDoS attacks cannot be detected and mitigated as if they are traditionally addressed in a fixed and dedicated server infrastructure. We propose that cost and performance aspects are essential parameters in detecting and mitigating such types of attacks. We propose a novel mitigation scheme, which takes auto-scaling decisions on the basis of a real requirement of legitimate traffic by "falsifying" the attacker traffic. This is achieved by identifying legitimate requests and their share in auto-scaling decisions. Our novel attack mitigation strategy, DARAC (DDoS Aware Resource Allocation in the Cloud), which is tuned to take correct auto-scaling decisions by ascertaining the legitimate traffic load supported by server capacity planning. After filtering the attack traffic, we calculate the share of legitimate clients in resource addition/buying and make subsequent accurate auto-scaling decisions. Our testbed evaluation results on real situations demonstrate considerable improvement in the average response time of the victim web-service under attack.

We organize the rest of the chapter as follows: Section 4.2 discusses DDoS attacks in cloud and initial experiments to show the effects of DDoS attacks on cloud computing. Section 4.3 details various requirements of an effective solution to DDoS attacks in cloud computing. Section 4.4 enumerate our proposed strategy towards DDoS mitigation. We show detailed evaluation of our proposed approach and related experimental results in Section 4.5. In Section 4.6, we detail various salient features and shortcomings of our proposed approach. Finally, in Section 4.7, we conclude the chapter.

4.2 Auto-scaling during DDoS Attacks in the Cloud

Services in the cloud can be scaled up or scaled down using an auto-scaling utility. This utility may become a target of the DDoS attackers, who may send a huge number of requests that may result in scaling resources up and cause massive losses to the cloud consumers. An on-demand cloud service will scale virtual servers based on the resource demand. However, these fake resource claims will force the cloud consumers to pay for the fraudulent traffic. Initially, when the servers face the attack, the usage based bill starts rising. When the service level agreement (SLA) gets saturated for the maximum allowed resources, the EDoS attack transforms into the DDoS attack. We performed real attack experiments which illustrate this effect in detail. We show the experimental attack scenario in Figure 4.1. We list various configurations and details of the resources used in this experiment in Table 4.1.

Virtual Machines in a cloud are usually bound by an SLA between a cloud service provider and a cloud consumer. An auto scaling policy [19][107][109] is one of the most important aspect of an SLA. There is a range of auto-scaling policies detailed in the literature and used in the production environments [125]. Few providers use customized policies where users can specify the underload and overload conditions. Auto-scaling methods add more resources or removes the idle resources from the

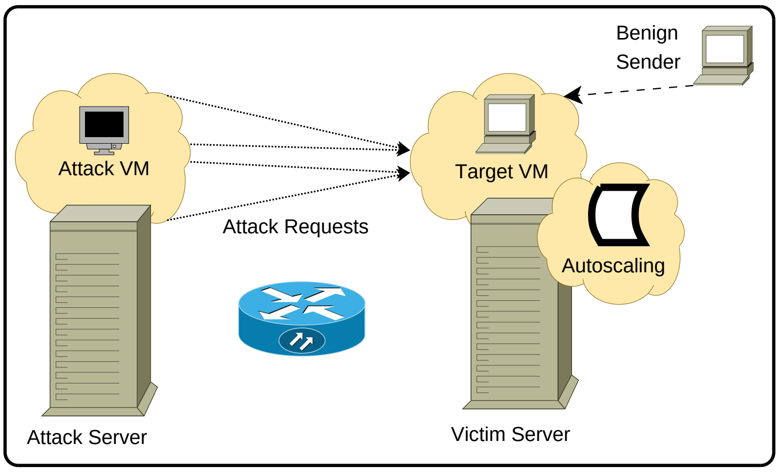


FIGURE 4.1: Experimental DDoS Scenario

VM based on the the underload and the overload conditions. We are using one such simple policy which sets these utilization thresholds to 70% ("overload") and 30% ("underload"). This auto-scaling algorithm runs in the background on the hypervisor and monitors the resource utilization of hosted VMs. If the average of CPU utilization in last one minute exceeds the threshold of "overload" state by attaining more than 70% of utilization, one additional VCPU is hot plugged to support the load. Similarly, if the memory utilization exceeds the threshold of 70% ("overload" state) usage for last 1 minute then an additional chunk of 1GB memory is added to the VM. This resource addition (VM expansion) continues as long as the SLA (costs and resource caps) supports the expansion. The same is true for the idle resource removal, where utilization threshold of "underload" state is 30%. If the utilization goes below this level, auto-scaling mechanism removes a fixed amount of memory say 1GB and/or one vCPU. Auto-scaling will account and bill only for the resources which are used to follow the principles of "pay-as-you-go" accounting. It is important to note that the auto-scaling policy may not be optimal in terms of resource usage and cost considerations. However, for our work, we wanted to have a basic auto-scaling policy that can help us in analyzing the impact of DDoS. We describe the auto-scaling policy used in our experiments in Table 4.1.

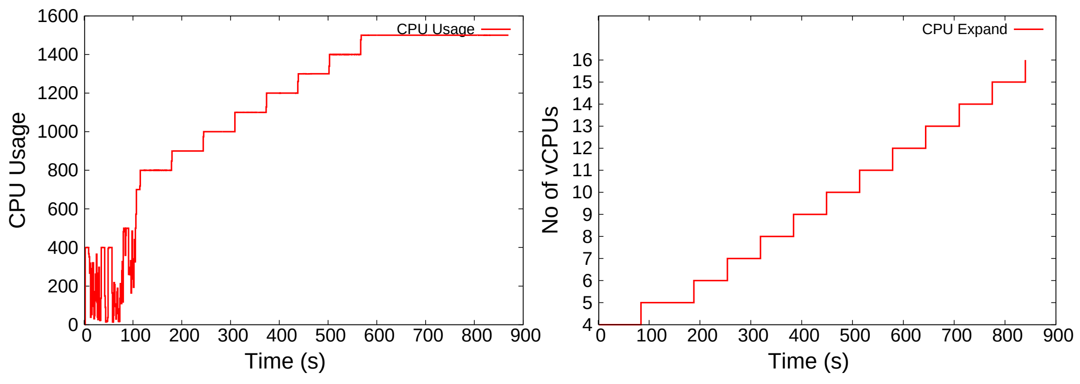
The attack VM and the target VM are run on two different Dell PowerEdge physical servers which are connected through a 1 Gbps network switch. The attack VM is running Apache Benchmark tool-set (`ab2`) which sends the attack traffic to the target web-service on target VM. The attack concurrency is 500 requests for a total of 10000 requests. The attack frequency is set in conformance to the similar popular real attack traces in [41]. The target VM is hosting a web-server

	Item/Resource	**Configuration Settings/Values**
Attacker Settings	Physical Server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 4 processors (12 cores each) memory 96 GB
	Attack service	ApacheBench2
	Configuration	4 vCPUs and 4 GB
	Traffic rate	500 concurrent requests (Total 10000 requests)
Victim Settings	Physical server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 4 processors (12 cores each) memory 96 GB
	Victim service	Dynamic image conversion service
	Configuration	1 vCPUs and 1 GB
Auto-scaling Policy	Initial Resources	vCPUs=4 and Memory=4GB
	Min to Max Memory	4 GB to 16 GB
	Min to Max vCPUs	4 vCPU to 16 vCPUs
	Monitoring Period	1 minute
	Condition for Overload	70% Utilization
	Condition for Underload	30% Utilization
	Increase-Decrease Factor	1 GB and 1 vCPU
Other Settings	Benign Service	4 vCPUs and 4 GB
	Traffic rate	1 concurrent web request (Total 100 requests)
	Operating System	Ubuntu 14.04
	Network Speed	1 Gbps
	Hypervisor	XenServer 6.5

Table 4.1: Experimental Configuration for the Impact Analysis of DDoS Attacks on Auto-scaling

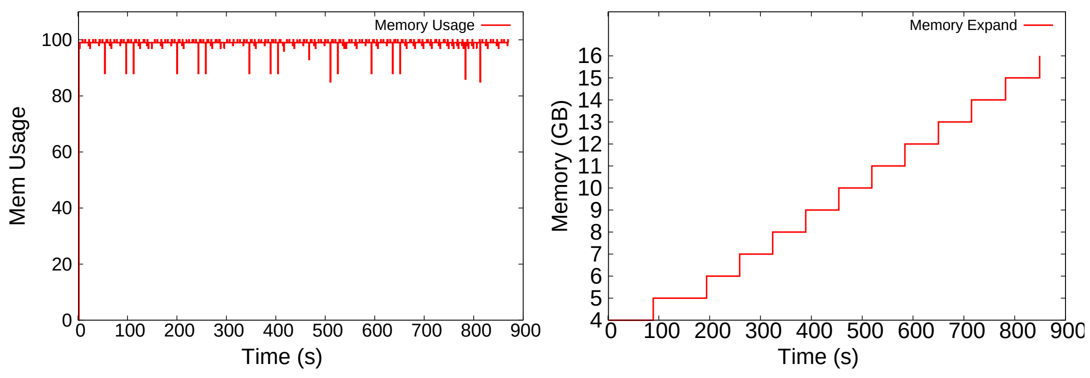
which is running a dynamic `php` website which converts an image of size 1MB from JPEG to PNG format. The website under attack is a generic representation of most common dynamic websites these days. We also have a benign sender in the experimental setup which is an additional VM running on a different physical server. A benign sender represents the benign traffic of 1 concurrent request for a total of 100 requests. The benign sender is configured in such a manner that it waits for the response to come for a very long period (1000s). This setting helps us in knowing the attack state and the service response behavior during an attack. We synchronized the attack experiments in such as manner that the attack requests and the benign requests are sent at the same time.

We show the behavior of the auto-scaling algorithm by observing various important parameters during attack in figures 4.2a, 4.2b, 4.3a, and 4.3b. Once the attack starts, the CPU utilization reaches 100% (400% for 4 vCPUs). The usage remained 100% for the whole duration of the attack thereafter. We can see that despite providing extra resources in the form of vCPUs, the usage didn't fall. In



(A) vCPU Usage (B) vCPU Expansion

FIGURE 4.2: Behavior of Auto-scaling during a DDoS Attack: vCPU Expansion



(A) Memory Usage (B) Memory Expansion

FIGURE 4.3: Behavior of Auto-scaling during a DDoS Attack: Memory Expansion

Figure 4.2b and Figure 4.3b, we see that the target VM is hot plugged with an additional vCPU and memory after every minute of observation till the maximum available resources or the cap is reached. In our experiments, we set the resource cap (maximum allowed resources to a VM) as 16 vCPUs and 16 GB of main memory. The benign sender used in our experiments sends one concurrent request for a total of 100 requests. We run the above experiment for 1000s and we see that the initial first requests was failed due to the time-out of 1000s. We did not observe the effects further from the experiment after the experiment period was over. Similarly, due to the attack, the memory usage exceeds the threshold of 70% utilization (∼85%-100%) and the auto-scaling algorithm keeps adding more memory every minute. We observe a main memory increment by a factor of 1GB after every overload trigger. We also show the behavior of the auto-scaling algorithm in the form of resource increase triggers in Table 4.2. In this table, after 60s of heavy CPU utilization and memory utilization, the auto-scaling algorithm takes the first decision to add resources. The auto-scaling algorithm adds an additional vCPU at 84s and an additional chunk of main memory at 89s. We also see that a similar

Auto-scaling Trigger	**Time (s)**	**Expanded vCPUs**	**Time (s)**	**Expanded Memory**
No Trigger	0 (start)	4vCPUs	0 (start)	4GB
1	84	5vCPUs	89	5GB
2	188	6vCPUs	194	6GB
3	254	7vCPUs	259	7GB
4	319	8vCPUs	324	8GB
5	384	9vCPUs	389	9GB
6	449	10vCPUs	454	10GB
7	514	11vCPUs	519	11GB
8	579	12vCPUs	584	12GB
9	644	13vCPUs	650	13GB
10	710	14vCPUs	715	14GB
11	775	15vCPUs	782	15GB
12	840	16vCPUs	849	16GB

Table 4.2: Auto-scaling Triggers and Resource Changes

trend is continued till the scaled resources reach the resource limit. We see that the DDoS attack forced the cloud consumer to buy more and more resources. By doing this incremental resource buy-in and the fraudulent resource consumption, the cloud consumer also faces the economic sustainability issues. In the attack experiments presented above, we focus on vertical scaling. On the other hand, if the SLA has “unlimited” addition of resources with horizontal scaling [125], then instead of adding more resources on the same server, the cloud may start more VM clones/instances on other servers or even migrate the server to a “resource spacious” server. This will be economically disastrous as the cloud consumer will face heavy bills for these extra instances generating no revenue. We discussed the aspects related to the vertical and horizontal scaling in Chapter 3.

4.3 Need of Effective Mitigation

We argue that the real problem behind the success of DDoS in the cloud is due to fake resource allocation and consumption. If “auto-scaling” can differentiate between the malicious traffic and benign traffic and estimate their individual shares in triggering the resource allocation, we believe that such attack can be stopped. Auto-scaling in the cloud is an attempt to keep the cloud resources (for example, VM instances) aligned with the increase or decrease in the requirements of the running application, so that the application is available and can scale as per the demand. It also ensures that the application owner has to pay excess charges only for the hours of high usage. The following equation gives a formal representation of an auto-scaling policy.

$$C(\texttt{VM}) = \begin{cases} +C(\texttt{VM}_{add}) & \text{if } \texttt{VM}_{state} = over - load. \\ -C(\texttt{VM}_{remove}) & \text{if } \texttt{VM}_{state} = under - load. \\ \texttt{No Change} & \text{if } \texttt{VM}_{state} = normal - load. \end{cases} \tag{4.1}$$

Resource capacity of a service is represented by the amount of resources available on the VM instance hosting the service. We represent it by the function *C()*. In the case of an *over-load* state of the VM under consideration, a dynamic auto-scaling strategy would go for immediate addition in the form of VM_{add}. These can be in the form of resources on top of a VM or a separate VM instance on a physical server. Similarly, in case of *under-load* state, VM_{remove} resources are removed. Auto scaling policies provided by the leading cloud providers [19, 110, 125, 126] can be categorized into following three categories:

1. **Manual auto-scaling:** Manual auto scaling is the most elementary auto scaling policy where the application owner can manually spawn or destroy the capacity up to a maximum or minimum threshold respectively. This requires manual monitoring of the application metrics and a scale up or down as per the need [127][128].

2. **Scheduled auto-scaling:** Cloud resources can be increased or decreased based on a schedule. This helps in scaling for already known changes in the traffic pattern of the application. A use case for such a scaling policy can be a scheduled e-commerce sale, the duration of which is known in advance. If huge traffic is anticipated during the sale then the application owners can schedule an auto scaling out before the expected peak and auto scaling in post peak [129][128][130].

3. **Dynamic auto-scaling:** Auto scaling methods are triggered to scale when a threshold of a resource utilization metric is crossed by a VM. An example metric is the average CPU utilization for a time period. If the CPU utilization is more than 80%, it will result in the spawning of an additional VM instance. Similarly, if the CPU utilization is less than some threshold (30%) for a certain duration, it shrinks the capacity by removing one VM instance [131][132][128][130].

While all the categories of auto scaling policies discussed above try to address the common concern of scaling an application against the load or traffic variation, none of them can evaluate how genuine is the increase in demand. Auto scaling to serve malicious traffic can seriously inflate the bills for the application owners and can threaten their financial existence. Solutions based on Turing tests [52][133], traditional rate based [74][134], threshold based [68] and anomaly detection methods [69][124], may not be enough to mitigate these attacks. In addition to these methods, we argue that a few vital features are needed in an ideal solution for the cloud. Based on the above discussion and the experimental observations, we propose the following requirements for an effective DDoS attack solution in cloud environments.

- **R1. Segregation Accuracy:** Most of the DDoS detection mechanisms are employed at the network gateways or at the application gateways that perform load balancing among multiple VM instances. Usually, these detection mechanisms rely on traffic or request segregation methods to identify malicious traffic to drop and genuine traffic to allow/accept. A segregation function should be able to perform this task with high accuracy which essentially means that it should minimize the false identification of genuine requests as attack requests and vice versa.
- **R2. Identify Actual Resource Requirement:** The overall framework of resource allocation should be able to determine the exact amount of resources needed to support the service to meet the quality of service or SLA requirements. It is quite difficult however to calculate and provide the actual set of resources in the needed to meet requirements. However, the cost implications are directly proportional to the amount of the resources added so it is an important requirement.
- **R3. Availability to genuine users:** The immediate effect of most DDoS attacks is towards "Service downtime" which essentially means that the service will be unavailable to all the users. We understand that during the attack and post-attack detection, the service to genuine users should be provided promptly as if there was no attack.
- **R4. Genuine auto-scaling triggers:** We see that most of the auto-scaling methods are primarily based on an indirect resource utilization metric such as CPU or memory or bandwidth utilization. We also saw in the attack experiments that these indirect auto-scaling triggers do not give correct information about the real resource need.

Requirement R1 is usually a key solution ingredient among many of the DDoS detection solutions presented in the past [68][57][74]. R2 is specifically needed for the utility computing models to tackle the consequences of EDoS. Requirement R3 is an aspect which has not been addressed by the state of the art solutions. As the server under attack, gets busy in attack mitigation, it does not really find resources and time to take care of legitimate customers. Mitigation solutions should also work to maintain service to legitimate customers, but, there is no specific reported work towards this problem in the literature. Requirement R4 is related to the requirement R2 and may help in minimizing the effects of fraudulent resource consumption. In the next section, we describe how the requirements R1-R4 helps us in framing our proposed solution.

4.4 DARAC: DDoS Aware Resource Allocation in Cloud

We propose a novel technique which helps in mitigating DDoS attacks in cloud computing. We term our solution as DARAC (DDoS aware resource allocation in cloud). This solution considers all the identified requirements (R1, R2, R3, and R4) into account in the following manner:

1. **R1:** For the purpose of traffic evaluation and attack traffic blocking, we used a popular and generic connection based attack detection mechanism, `ddos-deflate` [87]. We use this detection mechanism with its default settings where it detects and blocks a source IP address if it creates more than 150 connections. As the novelty of our idea contributes towards requirements R2, R3 and R4 for an effective solution, we do not provide our own traffic segregation method for this purpose. We believe that any other existing method can be used to perform the traffic segregation activity.

2. **R2:** DARAC provides a mechanism by which the "auto-scaling" mechanism of the cloud will always ask DARAC, whether to add resources or not. This is supported by the attack detection mechanism mentioned in point 1 above. In particular, once the attacker traffic is dropped, the features of legitimate traffic recorded in the last 1 minute helps in deciding whether the additional resources are needed. This is intelligent auto-scaling mechanism which is supported by our novel capacity planner (automated capacity planning module (ACPM))".

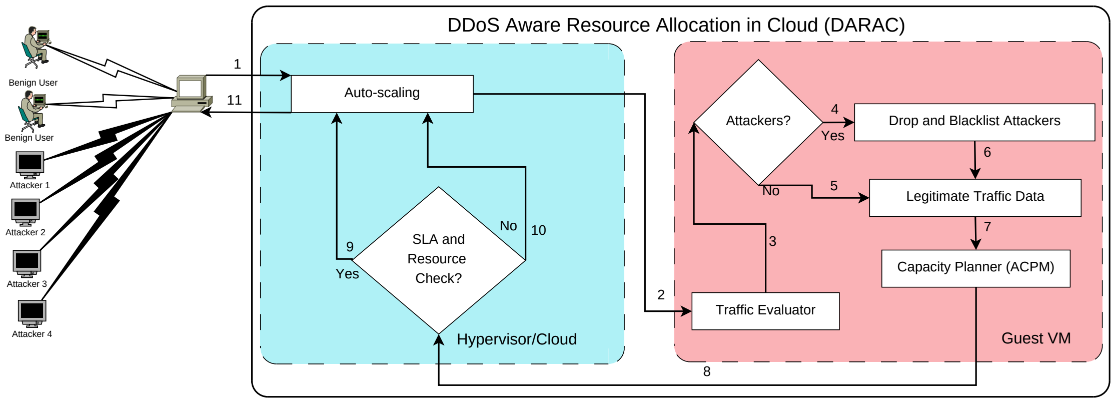


FIGURE 4.4: DARAC: Mitigation Mechanism

3. **R3:** The automated capacity planning module keeps track of both the required quality of service to users and the desirable resources. This is very important in order to serve legitimate customers well. Required QoS and corresponding resources (in terms of CPU, memory, disk and bandwidth) can be stored using a dry stress run or can be learned using machine learning techniques [135].

4. **R4:** In DARAC, we do not directly address the requirement R4 for identifying genuine triggers to auto-scaling. However, our solution supplements this requirement by not relying entirely on a direct metric such as CPU utilization. We first check the genuine load using our capacity planning module which then gives us a trigger about the requirement for auto-scaling.

4.4.1 DARAC Approach

Figure 4.4 shows step-by-step details of DARAC. The steps are as follows:

1. An auto-scaling trigger, which is discussed in section 4.2, is an event in which auto-scaling method increases or decreases the resources of the VM under consideration. The resource increment may be due to a DDoS attack or may be due to the real resource requirements.

2. Auto-scaling requests the DARAC module to evaluate the situation. Specifically, auto-scaling triggers the "Traffic Evaluator" module in DARAC to assess the traffic using segregation.

3. Traffic Evaluator checks if there is an attack. This is done on the basis of segregation based on `ddos-deflate` and blacklisted IPs are collected and passed to step 4.

4. If there are attacker IPs available in the traffic analysis, they are dropped and added to the blacklist.

5. If there is no attack, DARAC directly considers the overall traffic as legitimate traffic and passes the legitimate traffic statistics to the "capacity planner" module.

6. Remaining traffic which is actually legitimate traffic is used to calculate the frequency at which the real traffic is coming in.

7. The calculated legitimate traffic is then sent to the "capacity planner" module.

8. The capacity planner (ACPM) module sees the traffic and consults its capacity plan to get the resources required to support the service quality. The resulting resource capacity is then sent to the auto-scaling algorithm in hypervisor/cloud manager.

9. The auto-scaling algorithm now checks the SLA compatibility or resource caps and decides whether the required resource change can be made. Additionally, resource availability is also checked to see if the required amount of resources are available on the same server.

10. The required resource change is made by auto-scaling.

11. The required resource change becomes effective.

DARAC performs steps 2 to 8 in the VM and steps 1, 9 and 10 using the hypervisor. This is a solution which takes both levels of control into account. This is also important from the perspective of the VM owner's right to decide the resource requirement and privacy of traffic statistics. The VM owner may want to have different QoS levels, resource needs and cost constraints while planning capacity specific to their application requirements. Based on these reasons, the separation has been made. The two most important modules of DARAC are further elaborated as follows.

4.4.2 Traffic Evaluator Module

Traffic evaluator module segregates the legitimate traffic from the attacker traffic. After identifying the attacker sources, these identified sources are added to the blacklist in the `iptables` firewall. After this, the traffic evaluator module calculates the frequency of legitimate traffic that has arrived in the time frame under consideration. For capturing the traffic, we use the web-server logs to see the legitimate traffic. In our proposed approach the incoming traffic is always recorded consistently but the evaluation and drop is not a continuous process. It is purely a trigger-based approach where auto-scaling flags the traffic evaluator module.

4.4.3 Automated Capacity Planning Module (ACPM)

Capacity planning in the cloud is a process of determining the requisite amount of cloud resources to serve the varying demand of an application. The outcome of the capacity planning process is a capacity planning table having resource requirements for different set of incoming requests. Therefore, the capacity planning table is dependent on the nature of the tasks executed by the VM. For instance, capacity planning for a static web server may require more disk I/Os, memory, and network bandwidth with low CPU requirements. On the other hand, a high performance computing (HPC) scientific application may have bigger memory and CPU requirements with low network requirements. An auto-scaling algorithm which works on indirect indication of the load (in terms of resource utilization) on the web-service may unnecessarily increase or decrease the resources. Our proposed capacity planning module helps in correcting the resource scaling methods to have a direct and visible indication of the load in terms of the concurrent requests (requests coming at the same time). To achieve this goal, we propose an automated capacity planning module (ACPM) preparation algorithm (shown in Algorithm 1).

The proposed ACPM module can be more beneficial for unskilled small and medium enterprise (SME) users by helping them to prepare an automated server capacity planning table based on the recent incoming traffic load. The proposed algorithm takes few essential key inputs from a user planning the server capacity. A user has to specify minimum and maximum (possible) number of concurrent requests (C_{min} and C_{max}) it is expecting, and minimum and maximum resources (resource cap) it can acquire for the VM under consideration (R_{min} and R_{max}). In addition a user should also need to specify the quality of service expectation from the service. Page load time or request response time is one of the most useful factor in determining the service quality. We take the expected response time ($S_{required}$) from the user as the quality determination parameter in the capacity planning algorithm. Page load time or the response time is an important criteria

Algorithm 1: Automated Capacity Planning Module's Table Preparation Algorithm

```
ACPM;
Data: INPUT:
$R_{min}$= minimum available resources,
$R_{max}$= maximum available resources (resource cap),
$C_{min}$= number of minimum concurrent requests,
$C_{max}$= number of maximum concurrent requests,
$S_{required}$= required response time, and
$W$ = web-service;
OTHER VARIABLES:
$R_{now}$= current resource assignment to the VM,
$C_{now}$= number of concurrent requests to be sent,
$S_{recorded}$= recorded response time,
$R_{step}$= resource increase step size, and
$C_{step}$= number of concurrent requests increase step size;
Result: ACPM Table
Start;
Set $C_{now} = C_{min}$;
Set $R_{now} = R_{min}$;
while $C_{now} <= C_{max}$ do
    while $R_{now} <= R_{max}$ do
        Send $C_{now}$ requests to the service $W$ and record $S^i_{recorded}$, where $i$=1,2,...$C_{now}$;
        ▷ Comment: $S^i_{recorded}$ is recorded for individual $C_{now}$ requests. Optionally, to save time the internal
        loop can be broken if the $S^i_{recorded}$ is not recorded within a threshold ;
        if average($S^i_{recorded}$, $C_{now}$) <= $S_{required}$ then
                    ▷ Comment: Based on the need, we can also replace the above statement with "if
            (all_below($S^i_{recorded}$, $S_{required}$, $C_{now}$) == TRUE )" ;
            Record $R_{now}$ to the ACPM Table for $C_{now}$;
            else
                Set $R_{now} = R_{now} + R_{step}$;
            end
        end
        if ACPM Table has an entry for $C_{now}$ then
            Set $C_{now} = C_{now} + C_{step}$;
        end
        else
            Show "Resource cap has reached";
            Show ACPM Table with resource entries from $C_{min}$ to $C_{now}$;
        end
    end
end
```

which affects a range of important factors such as customer attention and the business outcomes in the form of transactions. The permissible range of request response time is between 1s to 10s. Reports such as [136] state few important aspects related to page response times based on a study conducted in 2011. The authors observed that if the response time of website is less than 5 seconds than

Algorithm 2: Function average() computes the numerical average of all the recorded response time values ($S^i_{recorded}$) for C_{now} requests

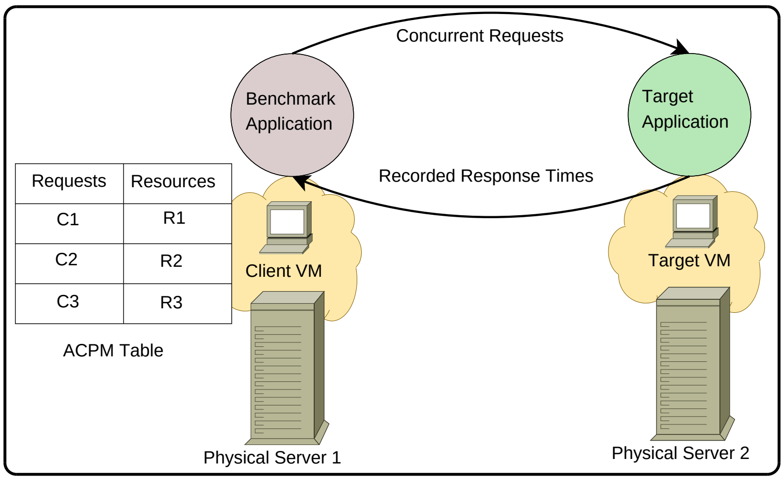
```
FUNCTION average();
Data: $S^i_{recorded}$, where i=1,2....$C_{now}$= recorded response time,
$C_{now}$= connection requests;
Result: Average of $S^i_{recorded}$ values
Start;
$Sum = 0$;
for each i=1 to $C_{now}$ do
    $Sum = Sum + S^i_{recorded}$
end
Show $\frac{Sum}{C_{now}}$
```

Algorithm 3: Function *all_below*() checks whether every individual request i among the concurrent requests C_{now} has a recorded response time ($S^i_{recorded}$) below the required response time ($S_{required}$)

```
FUNCTION all_below( );
Data: $S^i_{recorded}$, where i=1,2....$C_{now}$= recorded response time,
$S_{required}$ = required response time,
$C_{now}$= connection requests;
Result: TRUE/FALSE
Start;
for i=1 to $C_{now}$ do
    if $S^i_{recorded} <= S_{required}$ for all i then
        Show TRUE;
        else
            Show FALSE;
        end
    end
end
```

the web-site is faster than the 25% of the websites across the globe. Similarly, if the response time is below 2.9 seconds, than it is ahead of 50% of the websites across the Internet. On the other hand, report in [137] states additional interesting relations about the response times of the page and the business opportunity. The author says that every extra second of the response times may deter the web-user to return to the website. This aspect is termed as page abandonment. Additionally, based on a survey, the authors state that more than three quarters of the web-users across the globe expect a web response time less than 4 seconds.

Based on the user inputs, the Algorithm 1 runs and prepares a capacity planning table which has entries for various set of concurrent requests and required resources. Our proposed algorithm automatically prepares the desired capacity planning table with minimum user intervention. On the other hand, the same algorithmic procedure allows a lot of flexibility in terms of reconfiguration to suit various needs. The algorithm can run on a client machine connected to the web-service under consideration. We show the execution scenario and the capacity planning table preparation on a benchmarking client in Figure 4.5.



Requests	Resources
C1	R1
C2	R2
C3	R3

FIGURE 4.5: Automated Capacity Planning Table: Preparation Scenario

Now, let us understand the ACPM table preparation algorithm in detail. Algorithm 1 starts with the minimum resources ($R_{now} = R_{min}$) which are applied to the VM hosting target web-service W. Now, we use a benchmarking application such as Apachebench (`ab2`) to send C_{now} requests to the target web-service W. Initially, we set C_{now} to minimum number of concurrent requests (C_{min}) which is specified by the user. We also configure our benchmarking application to record the response time for each one of the C_{now} requests individually ($S^i_{recorded}$, where i=1,2,...C_{now}). Optionally, for optimized benchmarking, if we do not get the $S^i_{recorded}$ responses within a target period such as (K * ($S_{required}$)), we may stop the benchmarking for the current resource assignment. This means that the current resource assignment could not match the required service quality by serving the requests within period K * ($S_{required}$). A small value of K (e.g. K=2) will quickly break the loop and also quickly prepare the capacity planning table by saving time and moving to next set of resources. A higher value of K will help if the user is interested in a average service quality. Once we receive the recorded response time for each individual request ($S^i_{recorded}$, where i=1,2....C_{now}), we check an important condition which we can deal in one of the following two ways.

- **Condition 1: average($S^i_{recorded}$, C_{now}):** In this case, we compute the average of the received response times using the function presented in Algorithm 2. This condition allows us to choose resources R_{now} where the C_{now}

concurrent requests are served within the $S_{required}$ time on the average. However, this does not necessarily mean that all the requests will be served within the $S_{required}$ time. This condition results into a capacity planning table where a set of few or more concurrent requests out of C_{now} requests may get served with a response time higher than the $S_{required}$ time .

- **Condition 2: all_below($S^i_{recorded}$, $S_{required}$, C_{now}):** In this condition, we follow the function presented in Algorithm 3. This function makes sure that each individual request out of the C_{now} requests will be served within $S_{required}$ time. This is a strict condition which will result into a capacity planning table assuring about the equal service quality to all the concurrent requests.

If we find that the chosen condition (*average*() vs. *all_below*()) is true for the current resource assignment R_{now} (for requests C_{now}), we place the resource entry in the ACPM table. In the next step, we run the benchmark for C_{now} + C_{step} concurrent requests with the same resources (R_{now}). This suggests that we may find the current resource assignment as an appropriate one, even for the increased concurrent requests. Otherwise, we increase the resources by one step (R_{step}) and repeat the above process once again with the new resources (R_{now}=R_{now}+R_{step}). The process of increasing step wise resources may continue till we get the required response time values based on the selected condition (either *average*() or *all_below*()). The algorithm will complete its run, if we reach the resource cap (R_{max}), or there is a recorded entry for each set of concurrent requests from C_{min} to C_{max} with a step increase (C_{step}) in the prepared table. Now, let us come to an important aspect of the ACPM module preparation algorithm which is about deciding the R_{step} and C_{step}. The selection of appropriate values for both the variables is performed as follows:

- **Number of concurrent requests increase step size, C_{step}:** This variable decides the number of entries we would like to have in the capacity planning table. For example if we take C_{step} as 10 with C_{min} = 10 and C_{max} = 300. We will have 30 resource entries for different concurrent requests such as 10, 20, 30, 40, 50, 60...up to 300. It is obvious that having very small C_{step} size will also increase the size of the capacity planning table and the table preparation time. On the other hand, a very large C_{step} may not help in getting the effective use of capacity planning exercise. We also see that a variable C_{step} value may provide a balanced solution for cases such as a capacity planning table with C_{min}=10 and C_{max}= 500. In this case, for entries from C_{now}, 10 to 100, C_{step} may chosen as 10. On the other hand, for higher concurrent request entries such as 100 to 500, C_{step} may be chosen as 50 or 100. While preparing the capacity planning tables for our experiments, we used a variable C_{step} utilizing this example.

- **Resource increase step size,** R_{step}**:** Virtual machine monitors help in increasing and decreasing the resource vertically (on top of the same VM) with the help of resource hot-plugging features of virtualized resources such as vCPUs, memory, network cards and disks. A step can be configured as an increase/change in all the resources in a single go. For example, a step increase in all the resources can be addition of 1 vCPU, 1 GB memory, 1 Gbps additional network bandwidth, and 10GB expansion in the storage. On the other hand, a step can also be configured to increase/change in the individual resource such as vCPU addition by 2 vCPUs to memory addition by 1 GB. This single resource step may be beneficial to see the real need of the application. For example, a static web application may not need vCPU increase but the increase in the memory and disk. While preparing the capacity planning tables for our experiments, we implemented the resource increase step using increase in a single resource at a time.

While increasing the resources, we observed an important factor related to the resource scaling. Once we increase the resources of a VM, it requires some time before the new resources are available for use by the VM. In other words, VMs take a little time before the resource are actually adapted into the system. The observation is also applicable to the resource reduction operation. We observe that this time is also affected by the kind of activities the guest operating systems is performing at the time of resource change. In the case of heavy memory usage, decreasing memory will take greater time as compared to increasing memory owing to the immediate operations to shrink the memory (page-out to the swap space). In the case of vCPUs, the task migration and load balancing operations will occur. We term this time period as “resource realization time (RRT)” as this is the time required before a resource change is realized in the system. In order to show the efficacy of our argument, we conduct a set of experiments with two basic resource vCPUs and main memory. We perform various sizes of resource increase and decrease operations on these two resources and observe the time when these resource changes are realized in the guest operating system running on top of the VM. We show the outcome of this experiment in Table 4.3.

vCPUs (from)	vCPUs (to)	RRT	Memory (from)	Memory (to)	RRT
1 vCPUs	2 vCPUs	0.611s	1 GB	2 GB	8.141s
1 vCPUs	4 vCPUs	0.823s	1 GB	4 GB	7.214s
4 vCPUs	8 vCPUs	0.389s	4 GB	8 GB	9.211s
8 vCPUs	12 vCPUs	0.366s	8 GB	12 GB	13.213s
12 vCPUs	16 vCPUs	0.253s	12 GB	16 GB	9.192
1 vCPUs	8 vCPUs	0.479s	1 GB	8 GB	10.206s
1 vCPUs	16 vCPUs	0.817s	1 GB	16 GB	15.253
2 vCPUs	1 vCPUs	3.453s	2 GB	1 GB	10.563s
4 vCPUs	1 vCPUs	2.623s	4 GB	1 GB	10.204
8 vCPUs	4 vCPUs	3.756s	8 GB	4 GB	15.218s
12 vCPUs	8 vCPUs	3.753s	12 GB	8 GB	5.190s
16 vCPUs	12 vCPUs	3.758s	16 GB	12 GB	5.237s
8 vCPUs	1 vCPUs	7.480s	8 GB	1 GB	12.324s
16 vCPUs	1 vCPUs	17.204s	16 GB	1 GB	15.334s

Table 4.3: Experimental Results Showing Resource Realization Time (RRT) in Various Resource Change Cases

We see that the resource increase and decrease changes require a considerable time and any new decision of resource change should also wait for the resources (from the last change) to be really applicable. The RRT time ranges between less than a second to as high as 15 seconds. In case of multiple resource changes in a single step it may be even more. In the capacity planning preparation algorithm, while changing the resources, we wait for around 30 seconds before sending any new requests to use the realized resources. We believe that the proposed algorithm can also be used where horizontal scaling (resource increase in the form of additional VM instances). Additionally, the proposed algorithm may also be useful for a user/provider who is not sure about the resource needs and is interested in getting the resource requirements without having any resource limitations (R_{max}).
Now, in the next section, we detail the preparation of capacity planning tables for few important web-service cases.

4.4.4 Capacity Planning: Use cases

We use the Algorithm 1 to prepare ACPM capacity planning tables for following cases.

1. Dynamic image conversion website with the image size 500KB
2. Dynamic image conversion website with the image size 1MB
3. Static image download website with image size 1MB

4. Static image download website with image size 2MB

We use the scenario shown in Figure 4.5 to prepare the capacity planning tables for the above four cases. We list different set of variables and their possible values used in the preparation of capacity planning in Table 4.4. We see that concurrent

Resource	No. of Combinations	Details of Combinations
Request/s	23	5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 300, 400, 500 (with a variable step of 5, 10, 20 and 100)
vCPUs	16	1 vCPUs to 16 vCPUs (with a step 1 vCPUs)
Memory	16	1GB to 16 GB (with a step of 1 GB)
Hard disk	1	100 GB (fixed size)
Network	2	100Mb/s and 1000 Mb/s

Table 4.4: List of Resources and their Possible Values

requests have 23 sets with variable step size (C_{step}). For vCPUs and main memory of a VM, we have a total of 16 sets if we try all possible combinations with a step size (R_{step}) of 1GB. The infrastructure available with us could only allow use to change the values of network bandwidth between two values (100Mb/s and 1000 Mb/s). However, with the features such as Link aggregation [138], the network bandwidth can also be expanded to more higher values. We did not change the disk size because of hardware limitations where the on-the-go disk scaling was not possible. On the other hand, the four cases above does not have much impact of disk resource as the image conversion site and image download site mostly perform all the work in the main memory and does not store any outcome in the disk except the experimental results. We take a fixed size disk of 100GB. In the worst case, a run of Algorithm 1 will execute a complete set of 11776 steps (23 concurrent request steps * 16 vCPU steps * 16 memory steps * 2 network steps). It will take a huge time of more than 32 hours even if we break the internal loop in 10 seconds (K* $S_{required}$ with K=2 and $S_{required}$ =5 seconds). To improve this, we propose following practical enhancements in Algorithm 1 while executing it for the above cases.

1. We should not always test and find suitable resources for the C_{now} requests starting from the minimum resources. If an resource entry ($R1$) for C_{now} is found then the resource entry for $C_{now} + C_{step}$ will be found in the resources greater than or equal to $R1$. We already support this enhancement in the Algorithm 1.

Request/s	vCPUs	Memory	Network
5	2 vCPUs	4 GB	100Mb/s
10	2 vCPUs	4 GB	1000Mb/s
15	3 vCPUs	5 GB	1000Mb/s
20	4 vCPUs	6 GB	1000Mb/s
25	5 vCPUs	6 GB	1000Mb/s
30	6 vCPUs	6 GB	1000Mb/s
35	7 vCPUs	7 GB	1000Mb/s
40	7 vCPUs	8 GB	1000Mb/s
45	9 vCPUs	10 GB	1000Mb/s
50	10 vCPUs	10 GB	1000Mb/s
55	11 vCPUs	12 GB	1000Mb/s
60	12 vCPUs	12 GB	1000Mb/s
65	14 vCPUs	13 GB	1000Mb/s
70	15 vCPUs	15 GB	1000Mb/s
>70	16 vCPUs	16 GB	1000Mb/s

TABLE 4.5: Capacity Planning Table for Dynamic Site: Image Size 500KB

2. While assigning resources to the target application, the main memory should always be allotted by giving at least 1GB memory per vCPU. For example, if we allot 8 vCPUs to the target application than the minimum memory to start the benchmarking would be 8 GB.

3. For few initial requests such as 5 concurrent requests to 10 concurrent request the network bandwidth of 100 Mb/s gives the required results. For higher values of concurrent requests, the outcome of using 100Mb/s does not give desired results ($S_{required}$). By just using only one network bandwidth step of 1000Mb/s, the worst case execution of the Algorithm 1 reduces to the half. However, in the case of hardware support for the network scaling, this optimization is not useful.

4. In the cloud accounting and metering models, the cost of resources usually follows a generic relation where vCPUs and memory resources are costlier than all the other resources such as storage and bandwidth. This relationship can also help in minimizing the step sizes and subsequently narrow down the focus to computational resources or memory resources.

We show the prepared capacity planning tables in Table 4.5 for dynamic site (500KB), Table 4.6 for dynamic site (1MB), Table 4.7 for static site (1MB), and Table 4.8 for static site (2MB). Similarly, we also show the response time behavior of different target services in Figure 4.6 (500KB dynamic site), Figure 4.7 (1MB dynamic site), Figure 4.8 (1MB static site), and Figure 4.9 (2MB static site).

Request/s	vCPUs	Memory	Network
5	4 vCPUs	8 GB	100Mb/s
10	4 vCPUs	8 GB	1000Mb/s
15	5 vCPUs	8 GB	1000Mb/s
20	7 vCPUs	8 GB	1000Mb/s
25	8 vCPUs	8 GB	1000Mb/s
30	10 vCPUs	10 GB	1000Mb/s
35	11 vCPUs	12 GB	1000Mb/s
40	12 vCPUs	12 GB	1000Mb/s
45	14 vCPUs	16 GB	1000Mb/s
>45	16 vCPUs	16 GB	1000Mb/s

Table 4.6: Capacity Planning Table for Dynamic Site: Image Size 1MB

Request/s	vCPUs	Memory	Network
5	1 vCPUs	1 GB	100Mb/s
10	1 vCPUs	1 GB	1000Mb/s
20	1 vCPUs	1 GB	1000Mb/s
30	1 vCPUs	1 GB	1000Mb/s
40	1 vCPUs	1 GB	1000Mb/s
50	1 vCPUs	2 GB	1000Mb/s
60	2 vCPUs	2 GB	1000Mb/s
70	2 vCPUs	2 GB	1000Mb/s
80	2 vCPUs	2 GB	1000Mb/s
90	2 vCPUs	2 GB	1000Mb/s
100	2 vCPUs	2 GB	1000Mb/s
120	3 vCPUs	3 GB	1000Mb/s
140	3 vCPUs	3 GB	1000Mb/s
160	3 vCPUs	6 GB	1000Mb/s
180	3 vCPUs	12 GB	1000Mb/s
200	4 vCPUs	12 GB	1000Mb/s
300	6 vCPUs	16 GB	1000Mb/s
400	10 vCPUs	16 GB	1000Mb/s
500	14 vCPUs	16 GB	1000Mb/s
>500	16 vCPUs	16 GB	1000Mb/s

Table 4.7: Capacity Planning Table for Static Site: Image Size 1MB

Amongst the above four cases, we prepare the capacity planning table for the case of dynamic site with image size of 1MB using the *average*() function in the Algorithm 1 and Algorithm 2. Similarly, we use the same condition of using *average*() function in the case of static site having 1MB as the image size. This means that the capacity tables for these cases fulfill the requirement of of having the average response time less than the required response time ($S_{required}$). The other two cases are prepared using the function *all_below*() described in Algorithm 1 and Algorithm 3. We show the response time behavior of all the four cases in different graphs. We show the response time behavior of dynamic website with image

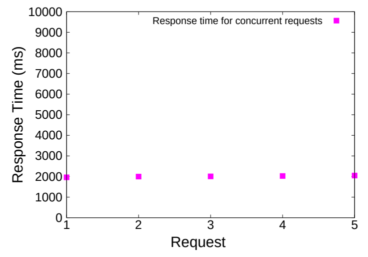
Request/s	vCPUs	Memory	Network
5	1 vCPUs	3 GB	100Mb/s
10	1 vCPUs	4 GB	1000Mb/s
20	1 vCPUs	4 GB	1000Mb/s
30	1 vCPUs	4 GB	1000Mb/s
40	1 vCPUs	4 GB	1000Mb/s
50	1 vCPUs	4 GB	1000Mb/s
60	2 vCPUs	6 GB	1000Mb/s
70	2 vCPUs	6 GB	1000Mb/s
80	3 vCPUs	6 GB	1000Mb/s
90	4 vCPUs	6 GB	1000Mb/s
100	6 vCPUs	6 GB	1000Mb/s
120	6 vCPUs	12 GB	1000Mb/s
140	6 vCPUs	12 GB	1000Mb/s
160	8 vCPUs	12 GB	1000Mb/s
180	10 vCPUs	12 GB	1000Mb/s
200	12 vCPUs	12 GB	1000Mb/s
>200	16 vCPUs	16 GB	1000Mb/s

Table 4.8: Capacity Planning Table for Static Site: Image Size 2MB

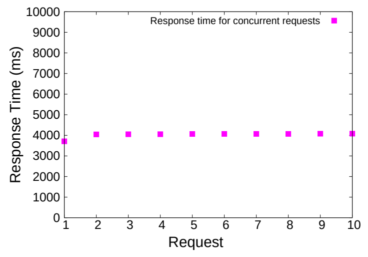
size of 1MB in Figure 4.7. In these graphs, we show the response time behavior with different C_{now} entries available in the capacity planning table. In addition, we also show the response time behavior of the entries which forces the ACPM module preparation algorithm to complete the execution. We show the terminal conditions of the Algorithm 1 in last section. These conditions include reaching C_{max} value, R_{max} value, or reaching a C_{now} value where the $S_{required}$ is not found by any of the resource assignments up to R_{max}. Now, we discuss the details of the prepared capacity planning tables in the following:

Case I: Dynamic image conversion website with the image size 500KB: We show the capacity planning table for this case in Table 4.5, and the response time behavior of the site for various concurrent request sets in Figure 4.6. In this case, we specified 5 seconds as the value of required response time ($S_{required}$). We set the benchmarking test to expect a resource assignment for the inclusion in capacity planning table if the recorded response for each request out of all the concurrent requests have a recorded response time less than or equal to 5 seconds. We achieve this requirement by using function all_below() in Algorithm 1. The resultant capacity planning table has entries for concurrent requests between 5 requests to 70 requests. The benchmarking test for 75 concurrent requests did not fulfill the condition of all_below() with the maximum available resources. We show the response time behavior for this case (C_{now} = 75) in Figure 4.6o.

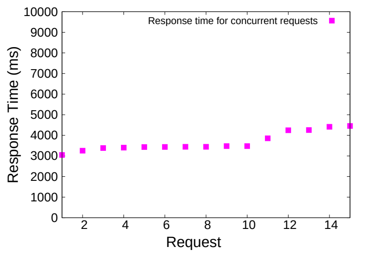
Case II: Dynamic image conversion website with the image size 1MB:



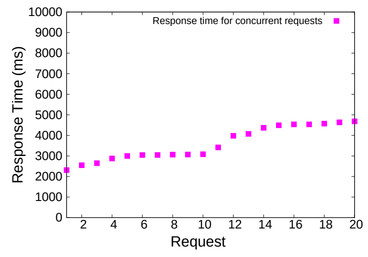
(A) Dynamic Server (500KB): 5 Concurrent Requests



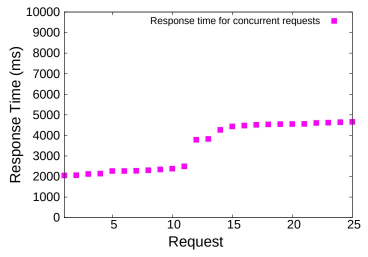
(B) Dynamic Server (500KB): 10 Concurrent Requests



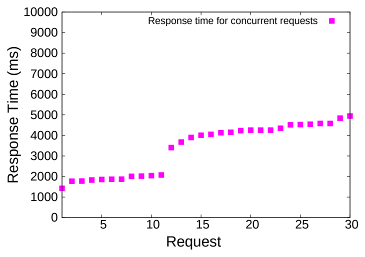
(C) Dynamic Server (500KB): 15 Concurrent Requests



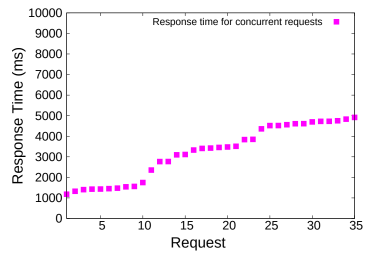
(D) Dynamic Server (500KB): 20 Concurrent Requests



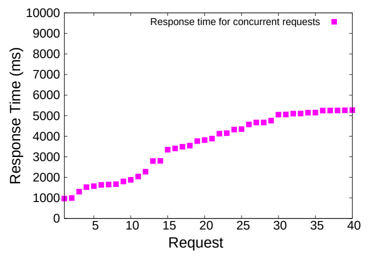
(E) Dynamic Server (500KB): 25 Concurrent Requests



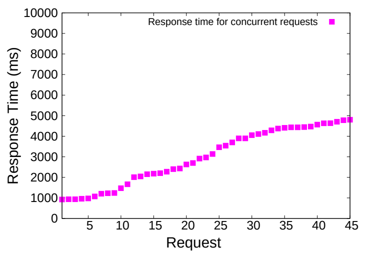
(F) Dynamic Server (500KB): 30 Concurrent Requests



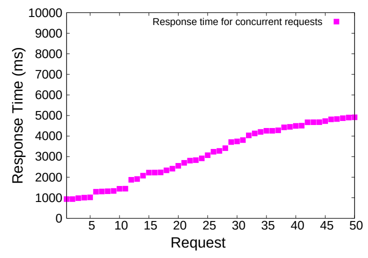
(G) Dynamic Server (500KB): 35 Concurrent Requests



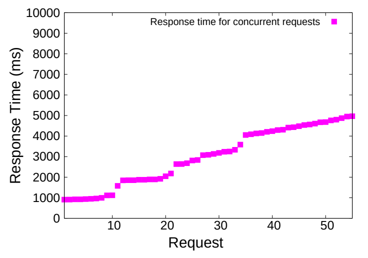
(H) Dynamic Server (500KB): 40 Concurrent Requests



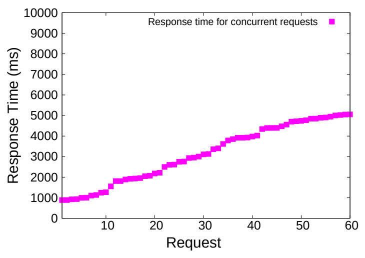
(I) Dynamic Server (500KB): 45 Concurrent Requests



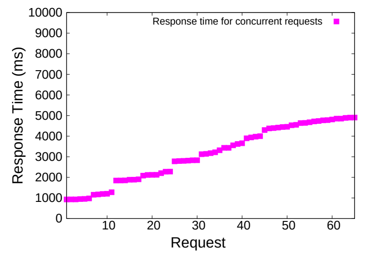
(J) Dynamic Server (500KB): 50 Concurrent Requests



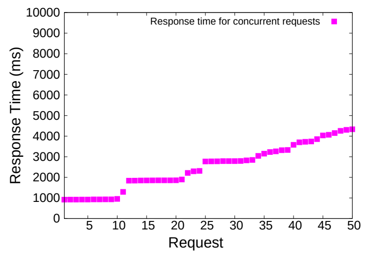
(K) Dynamic Server (500KB): 55 Concurrent Requests



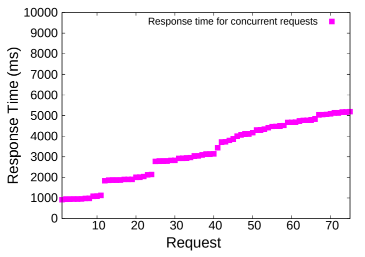
(L) Dynamic Server (500KB): 60 Concurrent Requests



(M) Dynamic Server (500KB): 65 Concurrent Requests



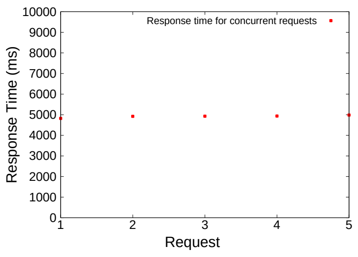
(N) Dynamic Server (500KB): 70 Concurrent Requests



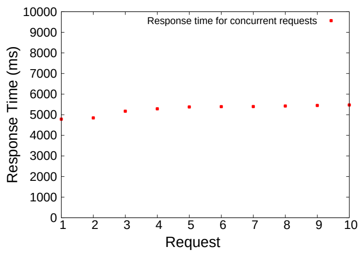
(O) Dynamic Server (500KB): 75 Concurrent Requests

FIGURE 4.6: Recorded Response Time Behavior of Dynamic Server: Image Size 500KB

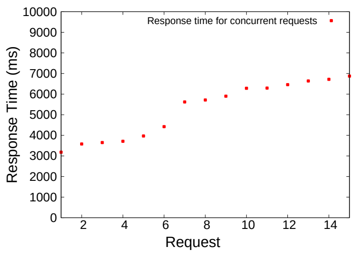
We show the capacity planning table for this case in Table 4.6 and the response time behavior of the site for various concurrent requests in Figure 4.7. The target application in this case is much resource intensive as compared to the Case I and converts an image of 1MB from one format to the other. In this case, we specified the required response time as 7 seconds as average response time of the



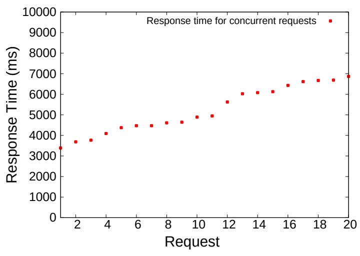
(A) Dynamic Server (1MB): 5 Concurrent Requests



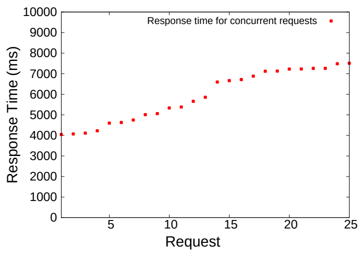
(B) Dynamic Server (1MB): 10 Concurrent Requests



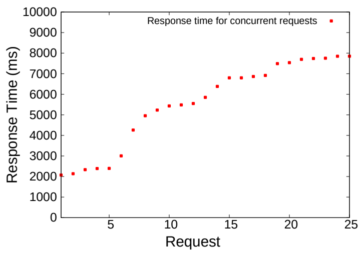
(C) Dynamic Server (1MB): 15 Concurrent Requests



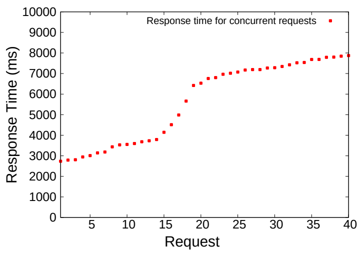
(D) Dynamic Server (1MB): 20 Concurrent Requests



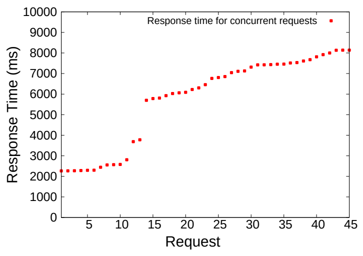
(E) Dynamic Server (1MB): 25 Concurrent Requests



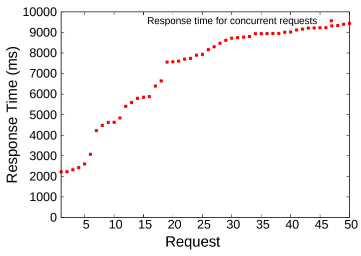
(F) Dynamic Server (1MB): 30 Concurrent Requests



(G) Dynamic Server (1MB): 40 Concurrent Requests



(H) Dynamic Server (1MB): 45 Concurrent Requests



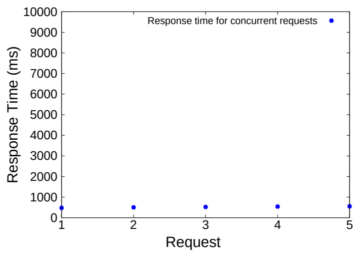
(I) Dynamic Server (1MB): 50 Concurrent Requests

FIGURE 4.7: Recorded Response Time Behavior of Dynamic Server: Image Size 1MB

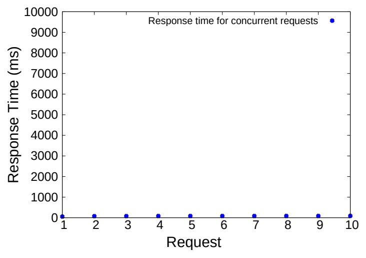
concurrent requests. Hence, we specify the condition using function *average*() in Algorithm 1. Subsequently, the capacity planning table has entries between 5 concurrent requests and 45 concurrent requests. We keep an entry in the capacity planning table having maximum resources reserved for concurrent requests greater than the 45 requests.

Case III: Static image download website with image size 1MB: We show the capacity planning table for this case in Table 4.7 and the response time behavior of the site for various concurrent requests in Figure 4.8. The target application in this case is a static image download web-site, therefore, the capacity planning table in this case has large number of entries with concurrent requests as high as 500 concurrent requests meeting the required average response time of 3 seconds (using *average*() function in Algorithm 1). In this case, the next higher number of requests (600 concurrent requests) could not be met by the maximum resources. Therefore, we keep an entry for the maximum resources for any number of concurrent requests beyond 500 concurrent requests.

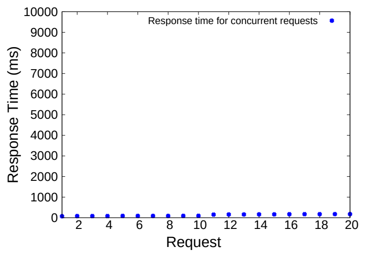
Case IV: Static image download website with image size 2MB: We show



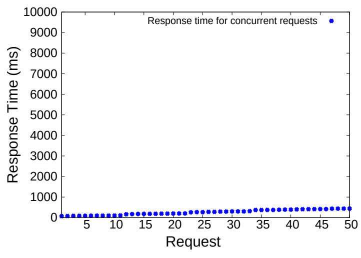
(A) Static Server (1MB): 5 Concurrent Requests



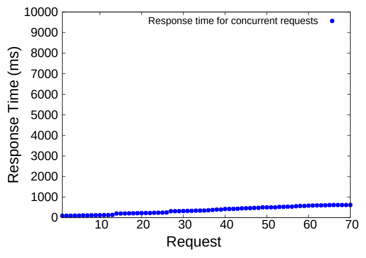
(B) Static Server (1MB): 10 Concurrent Requests



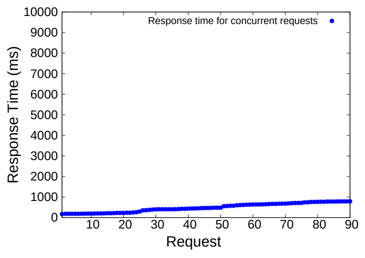
(C) Static Server (1MB): 20 Concurrent Requests



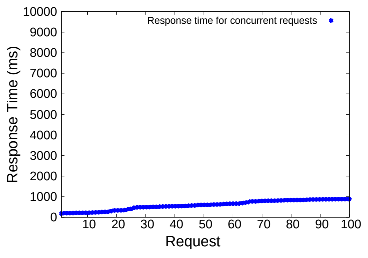
(D) Static Server (1MB): 50 Concurrent Requests



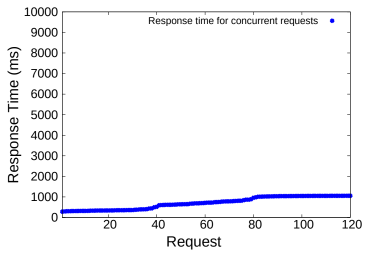
(E) Static Server (1MB): 70 Concurrent Requests



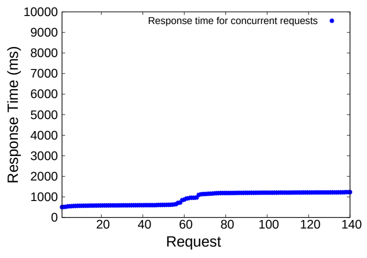
(F) Static Server (1MB): 90 Concurrent Requests



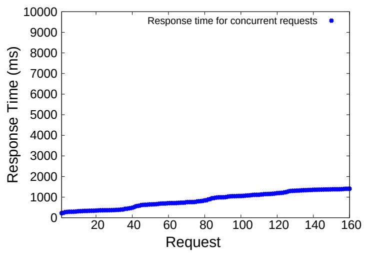
(G) Static Server (1MB): 100 Concurrent Requests



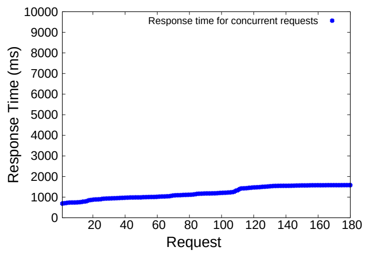
(H) Static Server (1MB): 120 Concurrent Requests



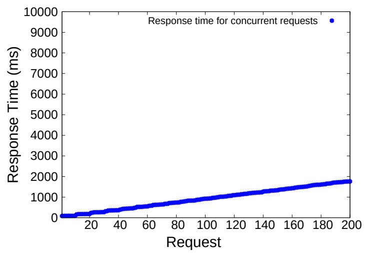
(I) Static Server (1MB): 140 Concurrent Requests



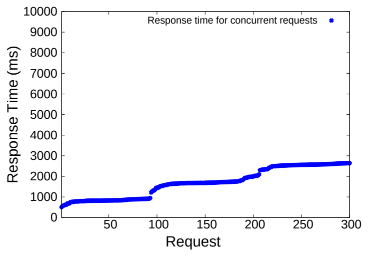
(J) Static Server (1MB): 160 Concurrent Requests



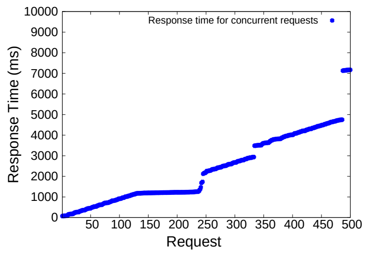
(K) Static Server (1MB): 180 Concurrent Requests



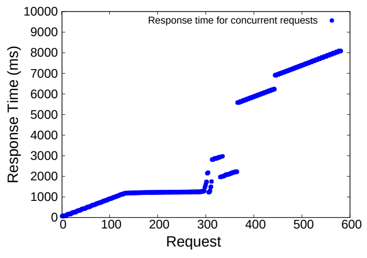
(L) Static Server (1MB): 200 Concurrent Requests



(M) Static Server (1MB): 300 Concurrent Requests



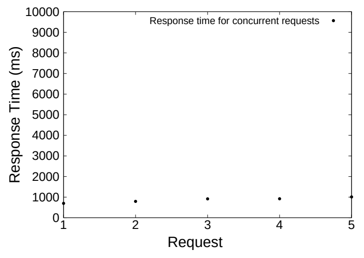
(N) Static Server (1MB): 500 Concurrent Requests



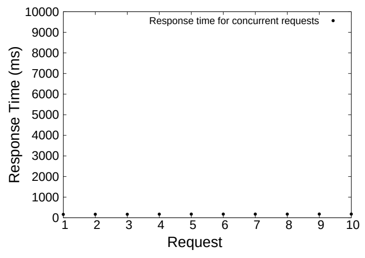
(O) Static Server (1MB): 600 Concurrent Requests

FIGURE 4.8: Recorded Response Time Behavior of Static Server: Image Size 1MB

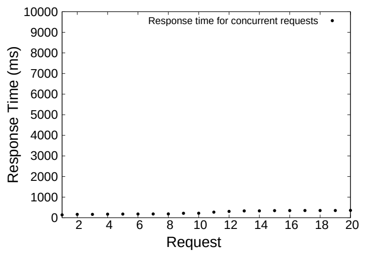
the capacity planning table for this case in Table 4.8 and the response time behavior of the site for various concurrent requests in Figure 4.9. In this case, we wanted the target web-site's capacity planning table should provide resources for a required response time of 4 seconds using *all_below*() function. The capacity planning table in this case has entries as high as 200 concurrent requests. In this case,



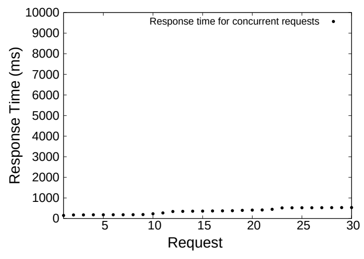
(A) Static Server (2MB): 5 Concurrent Requests



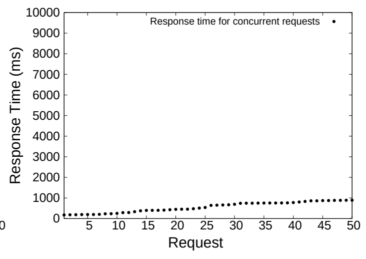
(B) Static Server (2MB): 10 Concurrent Requests



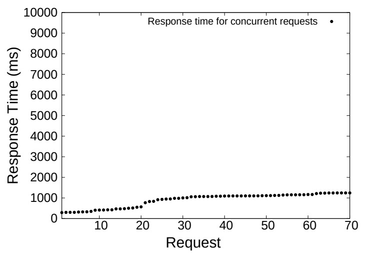
(C) Static Server (2MB): 20 Concurrent Requests



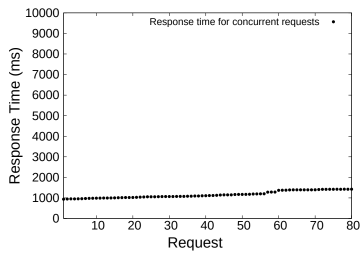
(D) Static Server (2MB): 30 Concurrent Requests



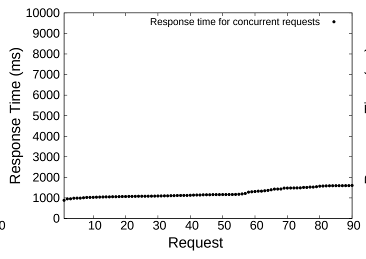
(E) Static Server (2MB): 50 Concurrent Requests



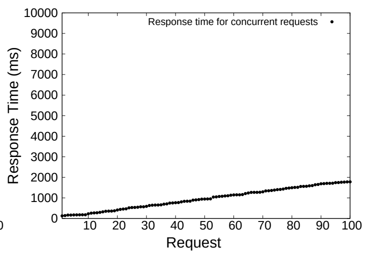
(F) Static Server (2MB): 70 Concurrent Requests



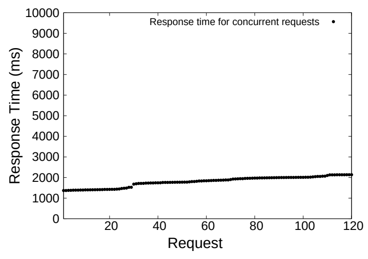
(G) Static Server (2MB): 80 Concurrent Requests



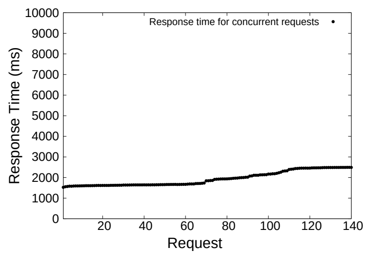
(H) Static Server (2MB): 90 Concurrent Requests



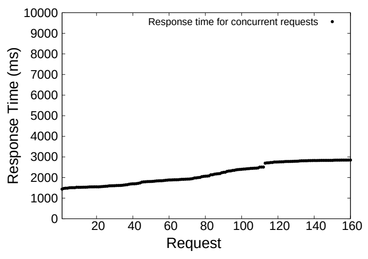
(I) Static Server (2MB): 100 Concurrent Requests



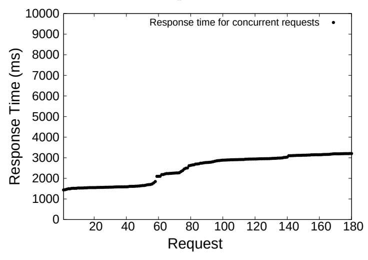
(J) Static Server (2MB): 120 Concurrent Requests



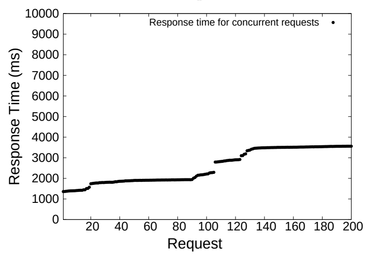
(K) Static Server (2MB): 140 Concurrent Requests



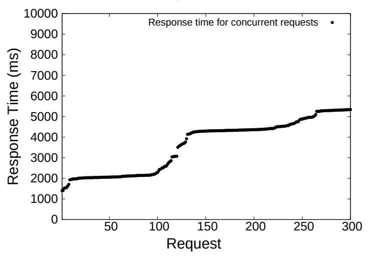
(L) Static Server (2MB): 160 Concurrent Requests



(M) Static Server (2MB): 180 Concurrent Requests



(N) Static Server (2MB): 200 Concurrent Requests



(O) Static Server (2MB): 300 Concurrent Requests

FIGURE 4.9: Recorded Response Time Behavior of Static Server: Image Size 2MB

the maximum resources (R_{max}) could not meet the requirements of 300 concurrent requests. We keep a resource entry in the capacity planning table for the cases of concurrent requests beyond 200 requests with resource assignment of maximum resources. The user can readily use the prepared capacity planning tables with the modified auto-scaling algorithm which relies on incoming legitimate traffic instead

	Item/Resource	**Configuration Settings/Values**
Attacker Settings	Physical Server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 4 processors (12 cores each) memory 96 GB
	Attack service	ApacheBench2
	Configuration	4 vCPUs and 4 GB
	Traffic rate	500/200 concurrent requests (Total 10000 requests)
Victim Settings	Physical server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 4 processors (12 cores each) memory 96 GB
	Victim service	Dynamic image conversion service
	Configuration	As per DARAC policy

DARAC Auto-scaling Policy	Initial Resources	vCPUs=4 and Memory=8GB (1MB case)
		vCPUs=2 and Memory=4 GB (500KB case)

	Min to Max vCPUs Min to Max vCPUs	2 vCPU to 16 vCPUs (500KB case) 4 vCPU to 16 vCPUs (1MB case)
	Min to Max Memory Min to Max Memory	4 GB to 16 GB (500KB case) 8 GB to 16 GB (1MB case)
	Monitoring Period	60 seconds
	Condition for Overload	70% Utilization
	Condition for Underload	30% Utilization
	Increase-Decrease Factor	As per ACPM Table 4.5 ((500KB case)) As per ACPM Table 4.6 ((1MB case))
Other Settings	Benign Service	4 vCPUs and 4 GB
	Traffic rate	1/10/20/50 concurrent web requests (Total 100 requests)
	Operating System	Ubuntu 14.04
	Network	1Gbps
	Hypervisor	XenServer 6.5

Table 4.9: Experimental Configuration for the Evaluation of DARAC (DDoS Aware Resource Allocation) Approach in the Presence of DDoS Attacks

of fully relying on a resource utilization based indirect mechanism.

4.5 Evaluation of DARAC

We now detail the experimental evaluation of our proposed DDoS aware resource allocation mechanism. To achieve this, we use our propose approach in the presence of DDoS attacks and record various metrics to observe its performance and behavior. As we focus on the auto-scaling mechanism's performance during DDoS attacks, we show various results related to the DDoS aware resource allocation mechanism which uses our capacity planning modules detailed in Section 4.4.3 and Section 4.4.4. For a comprehensive experimental evaluation, we use the attack settings and configuration of various resources as detailed in Table 4.9. The most important parameter in these settings is the modified DDoS aware auto-scaling algorithm. The modified algorithm in this case works as per the steps described in Figure 4.4 in Section 4.4.1. Figure 4.4 shows that the DARAC algorithm on receiving an overload trigger ($>$ 70% CPU utilization) it will ask the traffic evaluator module about the presence of the DDoS attack and the segregated legitimate traffic. DARAC will then ask the ACPM capacity planning module about the resource requirement of the legitimate traffic. DARAC then applies the resultant response received in the form of resources to the victim service.

We use two kinds of attack traffic sequences in the evaluation experiments: 500 requests concurrency and 200 request concurrency. We configured both the attack traffic sequences to send the concurrent requests for a total 10000 requests. In our evaluation experiments, the attack requests are sent simultaneously with the benign requests. The attack traffic in this case is sent from a single attacker machine for effective attack detection time calculations and subsequent comparison among the number of experimental sets. We send benign traffic from a benign sender VM with four different traffic frequencies (1/10/20/50 concurrent requests for a total of 100 requests).

To have good coverage of the attack experiments, we use two different dynamic websites as attack targets (victim service). In one set, the target website performs the image conversion of 500KB image. In another experimental set, the target website performs the same operation on images of 1MB size on each request. We provide initial resources to the VMs hosting these target services by following the capacity planning tables described in Section 4.4.4. For 500KB service, we set the initial resources to the planned capacity for the minimum C_{min} entry in Table 4.5. Hence, we set it to the resource entry for 5 concurrent requests (2vCPUs, 4GB and 100Mb/s). On the other hand, we set the initial resources of the 1MB victim service as per Table 4.6 which shows resource entry (4vCPUs, 8GB and 100Mb/s) for 5 concurrent requests. Similarly, subsequent resource changes and the maximum resources are also set using the capacity planning tables in Table 4.5 (for 500KB service) and Table 4.6 (for 1MB services).

We conduct a total of 16 attack experiments where we perform 8 experiments each on 1MB and 500KB service. Out of these 8 attack experiments, we set the attack frequency of four attack experiments to 200 concurrent requests for a total of 10000 requests. Other four experiments have attack frequency set to 500 concurrent requests for a total of 10000 requests. We perform each of these four experiments using benign frequencies with a rate of 1/10/20/50 concurrent requests for a total of 100 requests.

We observe following important metrics to ascertain the efficacy of our proposed approach.

1. **Resource Increment Trigger:** The time at which the first auto-scaling trigger comes up in the form of resource over-load trigger due to the average utilization of more than 70% in last one minute.

2. **Resource Decrement Trigger:** The time at which the first auto-scaling trigger comes up in the form of resource under-load trigger due to the average utilization of less than 30% in last one minute.

3. **Attack Detection Time:** The time at which the traffic evaluator (using tool `ddos-deflate`) detects the incoming DDoS attack and adds the attack sources to the blacklist (`iptables` firewall in our experimental setup).

4. **Resources after the Increase and Decrease Resource Triggers:** We see the resultant resources after the increase and decrease trigger received by the modified DDoS aware auto-scaling algorithm.

5. **Response Time to the Benign Traffic:** As described earlier, we send four different benign traffic sets with frequencies 1/10/20/50 concurrent requests for a total of 100 requests. We record the response time behavior to these benign concurrent requests.

4.5.1 Results

In this section, we present the detailed experimental results and the related inferences. We show the resource increase and decrease trigger times, attack detection times, and resource changes after the triggers for different experiments in tables 4.10, 4.11, 4.12, and 4.13. Similarly, we show the response time behavior and the resource change behavior for different experiments in the graphs shown in figures 4.12, 4.10, 4.13, and 4.11.

Benign Requests	Detection Time	Increase Trigger	vCPUs (after)	Memory (after)	Decrease Trigger	vCPUs (after)	Memory (after)
1	38.39s	83s	2	4GB	340s	2	4GB
10	36.93s	82s	2	4GB	341s	2	4GB
20	37.34s	84s	4	6GB	340s	2	4GB
50	37.40s	84s	10	10GB	400s	2	4GB

Table 4.10: Various Metrics during an Attack using 200 Request Concurrency and Total 10000 Requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 500KB

In the case of 500KB target service, we show various metrics in Table 4.10 where the attack traffic is set to the attack frequency of 200 concurrent requests for a

Benign Requests	Detection Time	Increase Trigger	vCPUs (after)	Memory (after)	Decrease Trigger	vCPUs (after)	Memory (after)
1	41.43s	83s	4	8GB	400s	4	8GB
10	40.12s	85s	4	8GB	341s	4	8GB
20	40.46s	84s	7	8GB	383s	4	8GB
50	41.52s	84s	16	16GB	350s	4	8GB

Table 4.11: Various Metrics during an Attack using 200 Request Concurrency and Total 10000 Requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 1MB

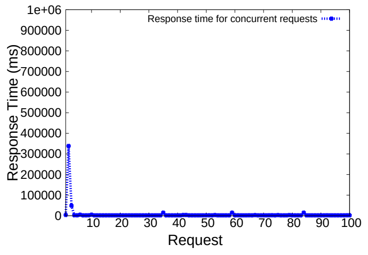
Benign Requests	Detection Time	Increase Trigger	vCPUs (after)	Memory (after)	Decrease Trigger	vCPUs (after)	Memory (after)
1	40.93s	84s	2	4GB	945s	2	4GB
10	39.41s	80s	2	4GB	934s	2	4GB
20	38.40s	83s	4	6GB	956s	2	4GB
50	38.40s	83s	10	10GB	956s	2	4GB

Table 4.12: Various Metrics during an Attack using 500 Request Concurrency and Total 10000 Requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 500KB

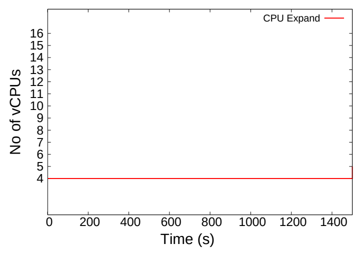
Benign Requests	Detection Time	Increase Trigger	vCPUs (after)	Memory (after)	Decrease Trigger	vCPUs (after)	Memory (after)
1	40.23s	84s	4	8GB	900s	4	8GB
10	38.40s	83s	4	8GB	893s	4	8GB
20	39.86s	80s	7	8GB	851s	4	8GB
50	42.35s	84s	14	16GB	955s	4	8GB

Table 4.13: Various Metrics during an Attack using 500 Request Concurrency and Total 10000 Requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 1MB

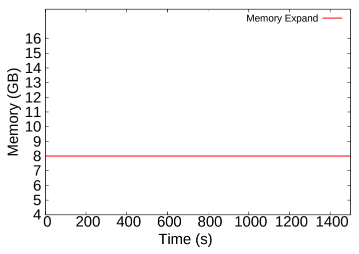
total of 10000 requests. As soon as the attack is launched the CPU and memory utilization of the target service reaches beyond ($> 70\%$) and a resource increase trigger is generated after the first one minute of the attack (83s). The DARAC triggers the traffic evaluation module. The traffic evaluation module could detect the attack in around 38s. After the attack detection, the attack sources are added to the firewall and already established connections are removed. The DDoS aware resource allocation algorithm after consultation with the traffic evaluator module gets the legitimate traffic rate which is (1/10/20/50) in four different attack cases. Practically, the received traffic values from the traffic evaluator module are not exactly the same as the sent traffic, however they are quite similar with minor differences due to their different arrival times to the interface. The DARAC resource allocation mechanism shows desired results by changing the resources in consonance to the benign frequency. In the case of benign traffic of 1 request and 10 request concurrency, we did not change the resources after receiving the



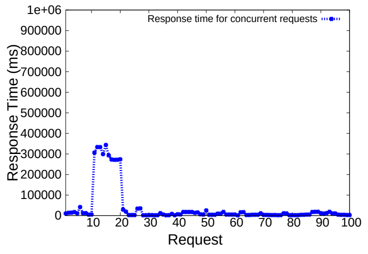
(A) Benign Behavior: 200 requests (Attack) 1 request (Benign) Service (1MB)



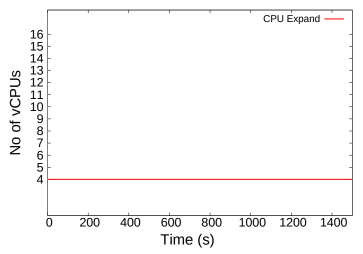
(B) DARAC vCPUs : 200 requests (Attack) 1 request (Benign) Service (1MB)



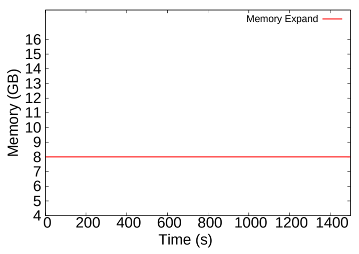
(C) DARAC Memory : 200 requests (Attack) 1 request (Benign) Service (1MB)



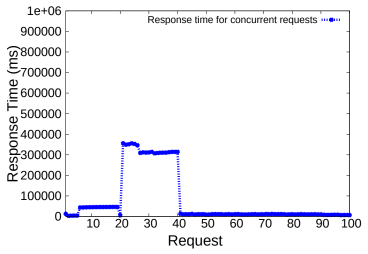
(D) Benign Behavior: 200 requests (Attack) 10 request (Benign) Service (1MB)



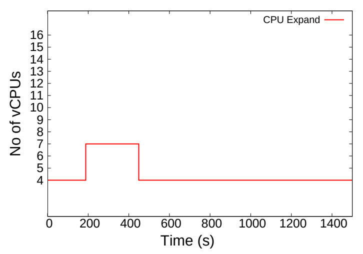
(E) DARAC vCPUs : 200 requests (Attack) 10 request (Benign) Service (1MB)



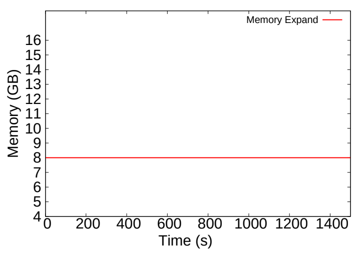
(F) DARAC Memory : 200 requests (Attack) 10 request (Benign) Service (1MB)



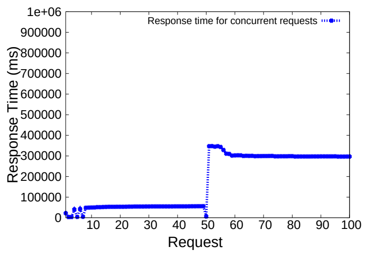
(G) Benign Behavior: 200 requests (Attack) 20 request (Benign) Service (1MB)



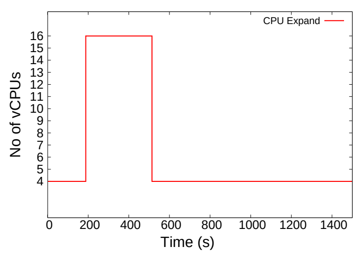
(H) DARAC vCPUs : 200 requests (Attack) 20 request (Benign) Service (1MB)



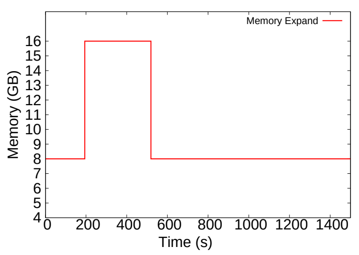
(I) DARAC Memory : 200 requests (Attack) 20 request (Benign) Service (1MB)



(J) Benign Behavior: 200 requests (Attack) 50 request (Benign) Service (1MB)



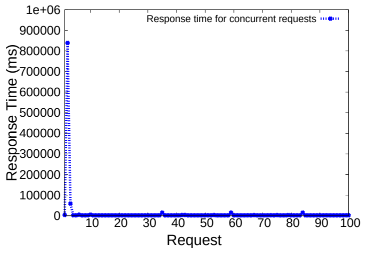
(K) DARAC vCPUs : 200 requests (Attack) 50 request (Benign) Service (1MB)



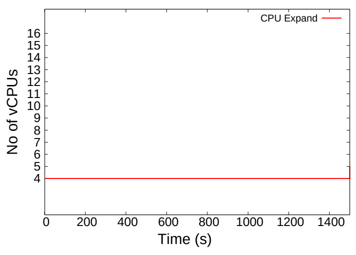
(L) DARAC Memory : 200 requests (Attack) 50 request (Benign) Service (1MB)

FIGURE 4.10: Response Time and Auto-scaling Behavior during an Attack using 200 Request Concurrency and a Total 10000 requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 1MB

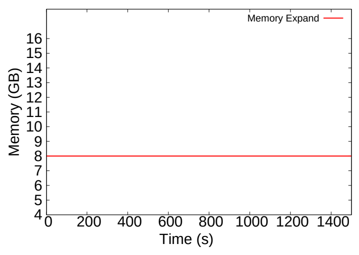
increase trigger. On the other hand, in the case of 20 and 50 requests, we change the resources to the equivalent entry of 20 and 50 concurrent requests by using the capacity planning table shown in Table 4.5. We show the response time behavior of the target service in different graphs in Figure 4.12. The service under the attack becomes unavailable due to the attack and request response time for the initial requests become huge. Even after, the attack detection at 38s, the attack effects do not disappear and the resource utilizations remains high. We keep on receiving the auto-scaling increase triggers after every minute of observance. We see that the resource utilization triggers a "lower utilization trigger" (using resource utilization below 30%) between 340s to 400s in different cases. The first



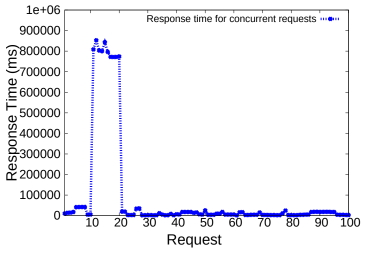
(A) Benign Behavior: 500 requests (Attack) 1 request (Benign) Service (1MB)



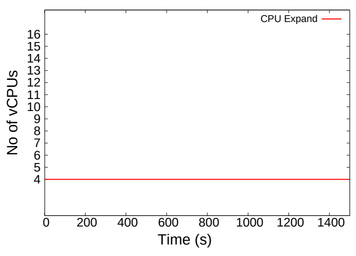
(B) DARAC vCPUs : 500 requests (Attack) 1 request (Benign) Service (1MB)



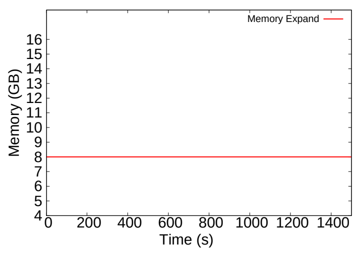
(C) DARAC Memory : 500 requests (Attack) 1 request (Benign) Service (1MB)



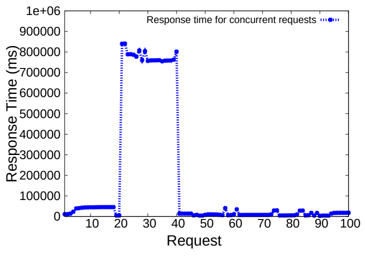
(D) Benign Behavior: 500 requests (Attack) 10 request (Benign) Service (1MB)



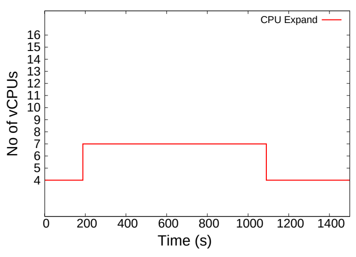
(E) DARAC vCPUs : 500 requests (Attack) 10 request (Benign) Service (1MB)



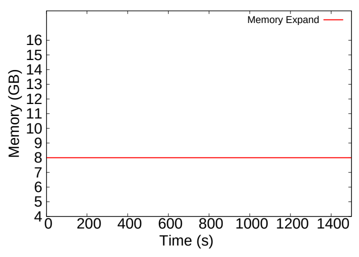
(F) DARAC Memory : 500 requests (Attack) 10 request (Benign) Service (1MB)



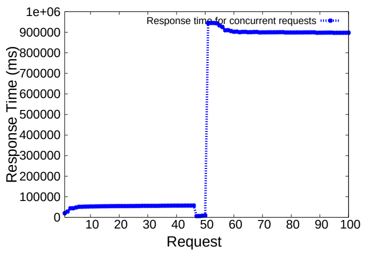
(G) Benign Behavior: 500 requests (Attack) 20 request (Benign) Service (1MB)



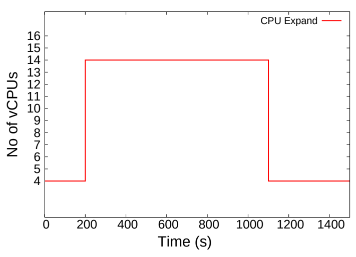
(H) DARAC vCPUs : 500 requests (Attack) 20 request (Benign) Service (1MB)



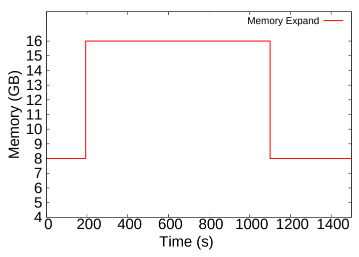
(I) DARAC Memory : 500 requests (Attack) 20 request (Benign) Service (1MB)



(J) Benign Behavior: 500 requests (Attack) 50 request (Benign) Service (1MB)



(K) DARAC vCPUs : 500 requests (Attack) 50 request (Benign) Service (1MB)

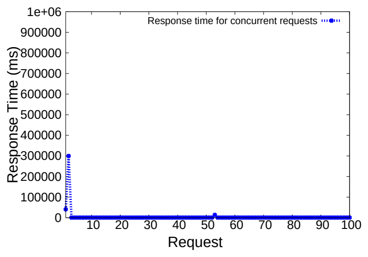


(L) DARAC Memory : 500 requests (Attack) 50 request (Benign) Service (1MB)

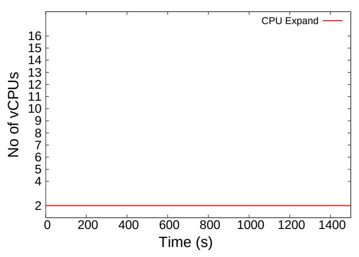
FIGURE 4.11: Response Time and Auto-scaling Behavior during an Attack using 500 Request Concurrency and a Total 10000 requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 1MB

batch of concurrent benign requests are served in a relatively short period of time as the attack effects were not completely formed and visible. After 340s to 400s the response time behavior becomes normal and the service becomes available to serve the requests. The response time behavior graphs show the response time behavior of the benign traffic which has initial peaks of around 340s to 400s. We gather these peaks by configuring the benign sender to collect the response for a longer time (∼1000s).

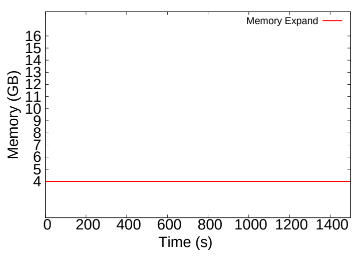
We show the results of attack set using the higher attack frequency of 500 concurrent requests with a total of 10000 requests in Figure 4.13 and Table 4.12. In



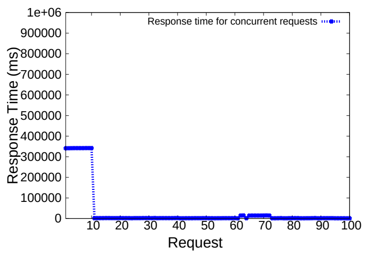
(A) Benign Behavior: 200 requests (Attack) 1 request (Benign) Service (500KB)



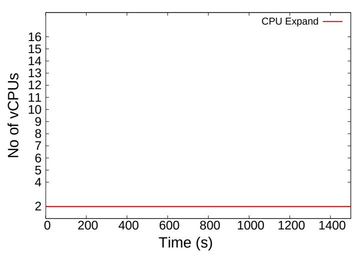
(B) DARAC vCPUs : 200 requests (Attack) 1 request (Benign) Service (500KB)



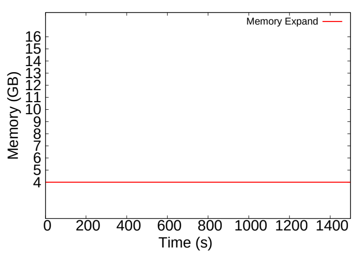
(C) DARAC Memory : 200 requests (Attack) 1 request (Benign) Service (500KB)



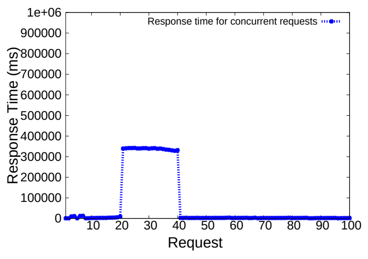
(D) Benign Behavior: 200 requests (Attack) 10 request (Benign) Service (500KB)



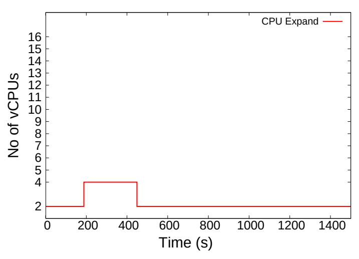
(E) DARAC vCPUs : 200 requests (Attack) 10 request (Benign) Service (500KB)



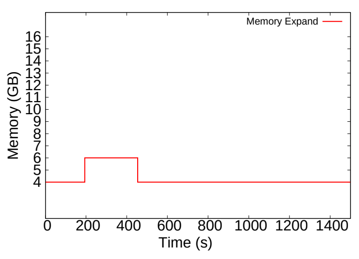
(F) DARAC Memory : 200 requests (Attack) 10 request (Benign) Service (500KB)



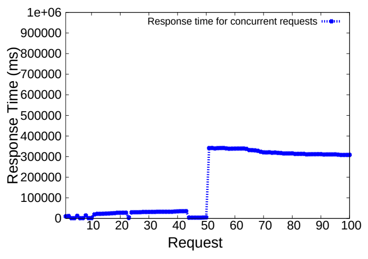
(G) Benign Behavior: 200 requests (Attack) 20 request (Benign) Service (500KB)



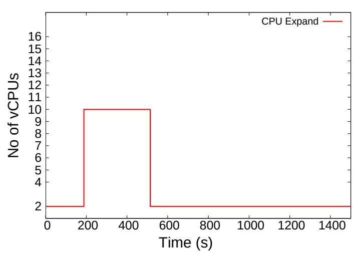
(H) DARAC vCPUs : 200 requests (Attack) 20 request (Benign) Service (500KB)



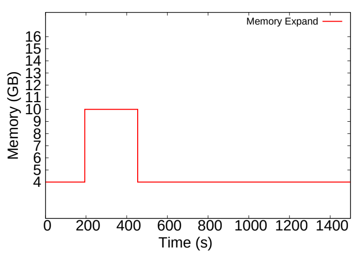
(I) DARAC Memory : 200 requests (Attack) 20 request (Benign) Service (500KB)



(J) Benign Behavior: 200 requests (Attack) 50 request (Benign) Service (500KB)



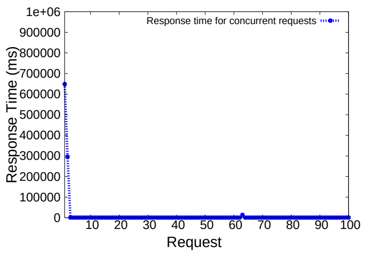
(K) DARAC vCPUs : 200 requests (Attack) 50 request (Benign) Service (500KB)



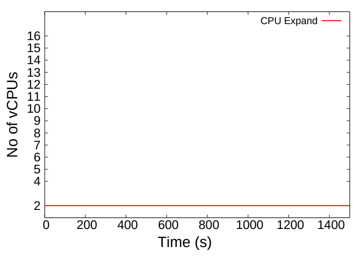
(L) DARAC Memory : 200 requests (Attack) 50 request (Benign) Service (500KB)

FIGURE 4.12: Response Time and Auto-scaling Behavior during an Attack using 200 Request Concurrency and a Total 10000 requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 500KB

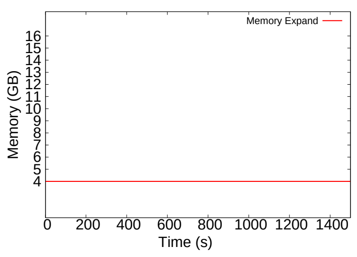
these attack cases, most of the observed metrics related to the resource changes are quite similar to the 200 request concurrency case discussed above with 500KB target site. However, we make two important observations during these results. The response time behavior shows peaks which are quite huge as compared to the case of 200 request concurrency. These peaks reaches the response time values as high as 950s. The other important observation is related to the legitimate request behavior. After receiving the first trigger at $\sim$ 84s, the legitimate requests were not regularly received from the victim service. We configured the victim service in such a manner that the traffic evaluator send the legitimate traffic values after



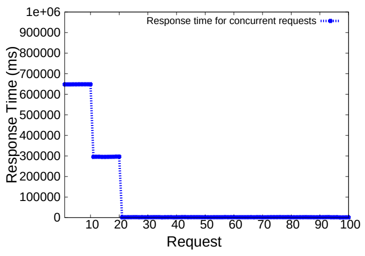
(A) Benign Behavior: 500 requests (Attack) 1 request (Benign) Service (500KB)



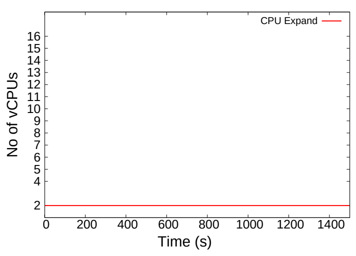
(B) DARAC vCPUs : 500 requests (Attack) 1 request (Benign) Service (500KB)



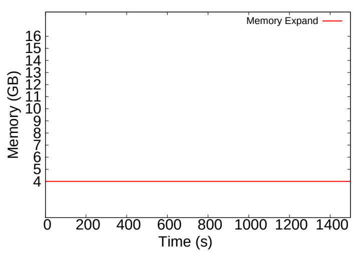
(C) DARAC Memory : 500 requests (Attack) 1 request (Benign) Service (500KB)



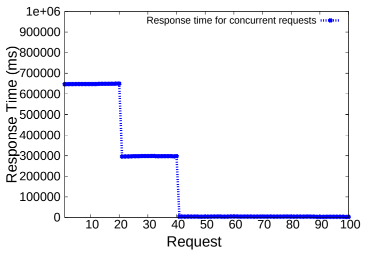
(D) Benign Behavior: 500 requests (Attack) 10 request (Benign) Service (500KB)



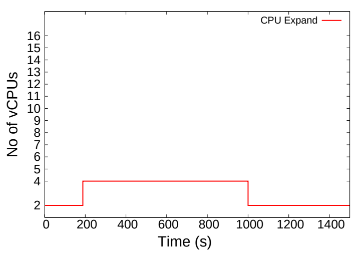
(E) DARAC vCPUs : 500 requests (Attack) 10 request (Benign) Service (500KB)



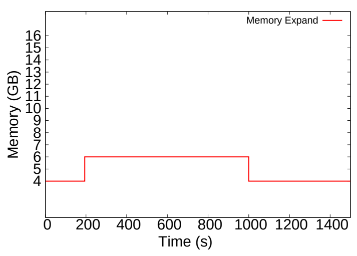
(F) DARAC Memory : 500 requests (Attack) 10 request (Benign) Service (500KB)



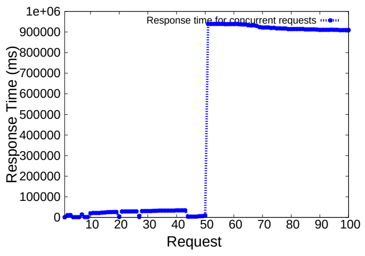
(G) Benign Behavior: 500 requests (Attack) 20 request (Benign) Service (500KB)



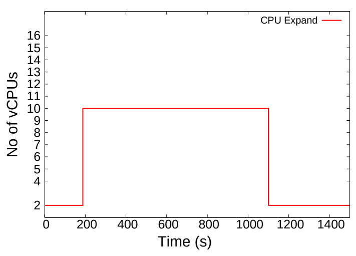
(H) DARAC vCPUs : 500 requests (Attack) 20 request (Benign) Service (500KB)



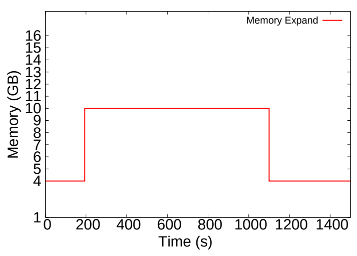
(I) DARAC Memory : 500 requests (Attack) 20 request (Benign) Service (500KB)



(J) Benign Behavior: 500 requests (Attack) 50 request (Benign) Service (500KB)



(K) DARAC vCPUs : 500 requests (Attack) 50 request (Benign) Service (500KB)



(L) DARAC Memory : 500 requests (Attack) 50 request (Benign) Service (500KB)

FIGURE 4.13: Response Time and Auto-scaling Behavior during an Attack using 500 Request Concurrency and a Total 10000 requests with DDoS Aware Resource Allocation on a Dynamic Web-site with an Image Size of 500KB

each minute. However, in the attack case of 500 request concurrency, we could not receive these values regularly but intermittently. The resource trigger and the subsequent resource changes are quite similar to the above case and the DARAC does not increase the resources in the case of benign requests with 1/10 request concurrency.

Now, let us see the experimental results of the victim service having 1MB size for the image conversion. In brief, the observed metrics were quite similar to the above cases of 500KB website. The DDoS aware resource allocation algorithm makes correct auto-scaling decisions and we could see that the resource changes are in conformance to the capacity planning tables for 1MB dynamic victim service. We show the observed metrics in the Table 4.11 and response time and resource changes graphs in Figure 4.10 for the attack frequency of 200 concurrent requests. The attack detection time, increase trigger, and decrease trigger are quite similar to the case of 500KB victim size for the same attack frequency. The response time behavior shows response peaks of $\sim$ 350s to 400s. The DARAC resource allocation algorithm shows the desired results by changing resources as per the capacity planning tables. The DARAC algorithm do not increase the resources in the cases of 1/10 concurrent requests and changes the resources in the case of 20/50 concurrency.

In the case of attack frequency of 500 concurrent requests for 1MB victim service, the results are quite similar to the similar case of 500KB victim service (Table 4.13). However, the response time peaks are quite higher and reaches up to $\sim$900s. We show this behavior in Figure 4.11. The DARAC behavior also performs the desired behavior by not changing the resources in the cases of 1/10 concurrent requests. On the other hand, resource changes (increases) are made in the cases of 20/50 concurrent requests using the capacity planning table shown in Table 4.6. On the other hand, we see a denial on legitimate request values (sent by the traffic evaluator module running in the victim service) are not received regularly. The frequency of getting these values is even smaller as compared to the 200 concurrent request case. In this case, our proposed approach in the current configuration does not make any decision and stick to the current resources.
After observing the effect of denial on the legitimate request frequency information transmission by traffic evaluator, we could see that the victim VM under attack is also non-responsive for most of the duration. Post attack analysis reveal that the DDoS attack resulted in to a lot of heavy resource usage and resultant resource contention. This heavy resource contention did not allow the computation of current legitimate frequency and its transmission. On the other hand, in the current attack experiments, DARAC could achieve the desired results by having resource changes as per the capacity planning tables prepared using ACPM module preparation algorithm (Algorithm 1). DARAC could also show the required response time for the benign requests after the attack effects disappear.

4.6 DARAC Defense: Discussion and Issues

In this section, we will elaborate few very important issues related to the working of DARAC and associated aspects. These issues also describe salient strengths as well as shortcomings of our proposal.

1. **DDoS aware resource allocation :** We could observe in the experimental evaluation detailed in Section 4.5 that the DARAC approach achieves important requirements set in the Section 4.3. We meet the segregation accuracy using the traffic evaluator module ($R1$). We fulfill the requirement $R2$ by identifying the actual resource requirement using automated capacity planning module. The ACPM module also helps in ascertaining the availability to the genuine/legitimate users. We meet the requirement of giving genuine auto-scaling triggers using legitimate requests received.

2. **Response time behavior :** The response time behavior is up to the expected level once the attack effects are over. During the attack (even after the attack detection time), the attack effects were visible and the response time to the legitimate requests was also not within the required response time limit. However, once the attack effects are over, the response times are within the required response time limits specified during the capacity planning table preparation.

3. **ACPM module issues :** We believe that the the current implementation of ACPM module has few important limitations, which on rectification may improve the performance further. We presented a calculation about the time required for the meta-data generation (in capacity planning table preparation) in Section 4.4.4. We save the recorded response time values in different files for each step. Though, we delete this temporary meta-data, once we create the final capacity planning table. We still believe that a temporary database can save a lot of efforts of generating and saving files. Similarly, our ACPM module in the current implementation supports similar kind of (or a single kind) concurrent requests. In reality, the dynamic web-servers offer different kind of request-response at the same time. The current implementation of ACPM does not support the mixed set of concurrent requests.

4. **Traffic evaluator module :** We are using a generic traffic evaluator module in our experiments. We believe that it can be replaced with any other traffic evaluation module with more sophisticated DDoS detection mechanisms as per the need.

5. **Legitimate request retrieval :** We believe that the current implementation of the mechanism to retrieve the legitimate request concurrency can be improved. Mechanism with the support of hypervisors or network devices (SDN switches) may help to have quick and accurate evaluation in the presence of service downtime. These mechanisms may also become helpful when the legitimate request values are intermittently unavailable. Additionally, the current mechanism asserts that the recent past behavior of concurrent requests will be repeated in the recent future. However, in the case when the traffic features (such as variable frequency of legitimate traffic) are changing, this assertion may not hold.

6. **Undetectable Attacks :** The proposed DDoS aware resource allocation mechanism is dependent on the accurate traffic segregation. For the cases, where the attacks remain undetected, the segregation will also report a huge number of requests as legitimate requests. In that case, it will quickly reach to the maximum resources leading to DDoS.

7. **Service downtime :** The service downtime in the experimental evaluation ranges between 300s to 1000s. On the other hand, the attack requests were detected and blocked in a small time $\sim$ 40s. We see that we require mechanisms to clear the established attack connections and the heavy resource contention to decrease the downtime. We discuss more methods to further decrease the service downtime in Chapter 5, Chapter 6, and Chapter 7.

8. **Controlled auto-scaling effects on providers:** At first glance, it may appear that the controlled auto-scaling results in monetary and business losses for the cloud service providers (CSP). However, the minimization of attack effects and intelligent dynamic scaling would result in attack effect free cloud network. We saw in Chapter 3 that DDoS attacks may result in collateral damages affecting many stakeholders of cloud computing infrastructure including network and co-hosted VM tenants.

9. **Overhead of attack awareness:** The attack awareness requires a negligible effort in terms of any computation complexity or performance perspective as the auto-scaling decisions making logic and the traffic filtering logic is not changed. However, there might be a delay where a genuine resource requirement appears while there is no attack. This may also result in a few application performance issues.

10. **Horizontal Scaling:** The experimental evaluation in this work has only considered the cases of vertical scaling. However, the evaluation is equally

applicable to the case of horizontal scaling where more VM instances can be added and removed as per the auto-scaling strategy (Equation 4.1).

11. **Dynamic scaling limitations:** Many production solutions on DDoS attack resiliency advocate of immediate resource scaling to expedite the overall attack absorption and mitigation [5] [139]. However, scaling may bring additional costs which may result in a huge penalty for undetectable attacks with longer durations.

4.7 Conclusion

In this chapter, we argue that the on-demand resource allocation capability also known as auto-scaling is an important process. Auto-scaling may spread the effects of DDoS attacks and increase the financial damages in the cloud computing environments. We performed real attack experiments to measure the overall attack effects on a victim service and observed that the fake resource utilization and subsequent resource addition results in EDoS, ultimately converging to DDoS. Based on these experiments, we analyze the requirements of DDoS attack mitigation in the cloud computing environment. These requirements form the basis of our proposed approach DARAC, which helps in customizing auto-scaling to minimize the attack effects.
There are three important aspects of DARAC which make it quick and effective DDoS mitigation solution for cloud computing: (a) the capacity planning module to ascertain the real resource requirements based on the legitimate requests, (b) correct DDoS aware auto-scaling, and (c) service availability to legitimate users during attack. The novelty of our work lies in the auto-scaling strategy with capacity planning for the services. we show detailed experimental results with wide coverage of attacker and benign traffic sets. Evaluation results show the efficacy of DARAC in the attack detection, blocking attacker traffic, managing the response time behavior of legitimate users, and stopping EDoS culmination by DDoS aware resource allocation. As discussed in Chapter 3, the collateral damages in the cloud are mostly visible due to the multi-tenant nature of the cloud. We show in the Chapter 5 that at the fine grain level, these collateral damages mostly occur due to the shared resources and resultant resource contention. In the next chapter, we extend the issue of collateral damages among VMs to the level of victim operating system.

Chapter 5

Service Resizing for Quick DDoS Mitigation

5.1 Introduction

Many of the recent DDoS mitigation solutions utilize a large amount of resources available in cloud using quick resource scaling [46][54][97][92]. Cloud service providers also recommend the use of resource scaling techniques to perform efficient and quick DDoS mitigation [63]. In resource scaling techniques, more and more resources are given to the affected victim service such that it can handle the incoming attack and carry out the attack mitigation quickly.

The resource scaling comes with the cost of additional resources that subsequently has impact on the budget/sustainability of the victim enterprise. Controlling cost is very important for the SMEs (Small and Medium Enterprises) owing to the limitations on their budgets. Practically, the DDoS attack mitigation costs should be less than the losses arising due to the attack without having mitigation in place. Additionally, the resource scaling strategies should be able to identify the real resource requirements. The auto-scaling service should be able to discard the fake resource alarms generated due to DDoS attacks. In Chapter 4, we show a DDoS aware resource allocation algorithm that only scales when there is a real requirement.

In this chapter, we critically examine much finer grain performance issues of DDoS mitigation process. These performance issues arise at the level of the victim service running on top of an operating system in the form of a process. These performance issues are related to the resource availability during DDoS attacks and the difference in the resource usage of variety of DDoS attacks. We show the DDoS attack dynamics and highlight important issues by doing real attack experiments on cloud services.

Based on our attack experiments, we observe that DDoS attacks may take different shapes based on the attack features and available resources on the server. In most of the real incidents, DDoS attacks take a form of “extremely unavailable DDoS (extreme DDoS)”. In this case, all the services (including the victim service) become inaccessible to perform the attack mitigation and subsequent service recovery. In these cases, owing to the heavy resource contention among the attack request processing and other system services, the DDoS mitigation methods may not get a chance to act and perform timely mitigation. Adding more and more resources to the victim service as part of resource scaling may not always help in the mitigation. In cloud computing, resources always come with a cost and the mitigation methods should spend resources carefully.

In this chapter, we provide a novel DDoS mitigation support and resource management framework which provides support services to the DDoS mitigation mechanisms to perform quick and sustainability aware mitigation on cloud services. With the proposed framework, we aim to minimize the service downtime, the mitigation time, and attack cooling down period. We address the “sustainability” or the cost aspect by using the available resources within the service instead of resource scaling. We argue that during “extreme DDoS”, sacrificing the resources by the victim web-service and utilizing those freed resources for the DDoS mitigation service can provide a quick, sustainability aware in-resource mitigation. We achieve these performance goals by providing two important features, (i) Resource shrinking and expanding and (ii) TCP tuning.

The rest of the chapter is organized as follows: We detail the DDoS attack mitigation and attack dynamics in cloud computing in Section 5.2. In the Section 5.3, we illustrate the various attack instance on a cloud-based service and different aspects of DDoS attacks in cloud computing. Our novel contributions are highlighted in detail in Section 5.5. In Section 5.6, we evaluate the proposed technique with various experiments and analyze the results. Finally in Section 5.7, we conclude and discuss the possible future directions.

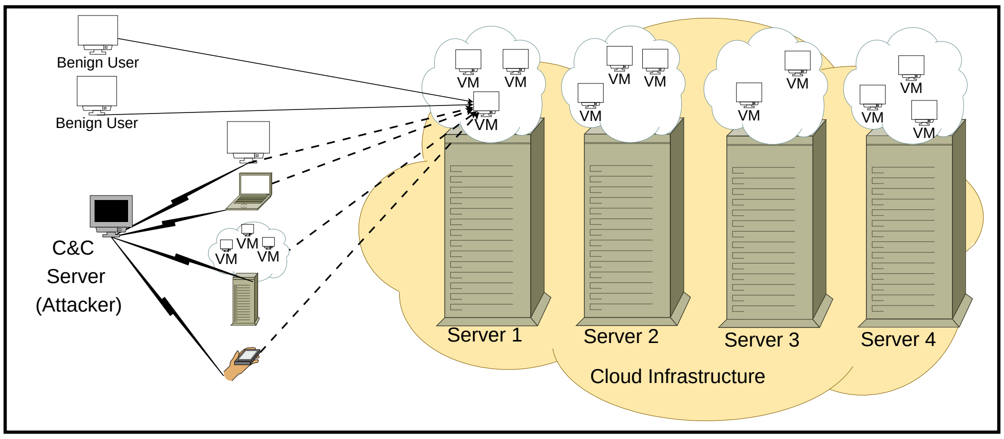


FIGURE 5.1: DDoS Attack in Cloud

5.2 DDoS Attack and Mitigation in Cloud Services: State of the Art

We show the DDoS attack scenario in Figure 5.1. Cloud computing infrastructure consists of multiple high capacity physical servers connected with high speed networking. The physical servers are managed by a cloud middle-ware framework to support mechanisms such as resource allocation and management, fault tolerance, and resource accounting. In most of the cases, the physical servers are virtualized to host and run virtual machines. These VMs are resource abstractions of physical servers to support easy deployment, multi-tenancy, improved resource utilization and other services like service migration, backup-recovery and cloning. Cloud VMs run a variety of services such as web, ftp, and database servers to HPC applications. The DDoS attackers targeting cloud services may follow a traditional DDoS attack model, where a command and control (C&C) server directs large number of bots to send the attack traffic. The consequence of such attack is usually “service denial” to the legitimate users. There are incidents of using the “DDoS-for-Hire” services [140] as an inexpensive attack infrastructure to launch the DDoS attack. In Figure 5.1, we also include cloud originated attacks to show the attackers utilizing cloud capabilities [39]. Other attack infrastructures include malware infected computers, phones and servers.

The attack effects and the losses distinguish a cloud targeted DDoS attack from a DDoS attack targeted at fixed on-premise service. These effects include sustainability/economic losses owing to the auto-scaling, the collateral damages due to multi-tenancy to non-targets, and the additional costs of attack mitigation [92]. There is a number of attack incidents, in which the victim services face heavy economic losses during and after the attack. For most of the small and medium enterprises (SMEs), a huge cost of DDoS attack mitigation cannot be justified. If we see the trends of 2015, more than 84% of the reported DDoS attacks were having peak bandwidth less than 1 Gbps. Similarly, more than 46% of the target services were web-services offering business and enterprise web-services [2].

A very similar trend is shown by "DDoS for Hire" services, where the maximum attack bandwidth is around 1 Gbps [140]. At the same time, more than 33% of the reported DDoS attacks target cloud infrastructure based services. Additionally, most of the organizations are running less than 50 VMs in public clouds based on their scale and adoption patterns [141]. These facts provide very important insight which is useful for designing DDoS mitigation solutions for a majority of DDoS victims.

As most the DDoS attacks with small attack footprint target SMEs which have a limited infrastructure based on their requirements and budgets. Therefore, we require DDoS mitigation solutions that are aware of organizational sustainability. These solutions should continue to provide safety from the attack related effects with minimum downtime of the services.

5.2.1 Generic Defense Architecture

DDoS solutions in the research literature and the marketplace mostly follow a generic solution architecture. Most of the DDoS mitigation methods are traffic evaluation based processes where the principal aim of the process is to differentiate attackers and the victim source addresses by a segregation function `F(Traffic)`. We show a generic DDoS mitigation mechanism in Figure 5.2.

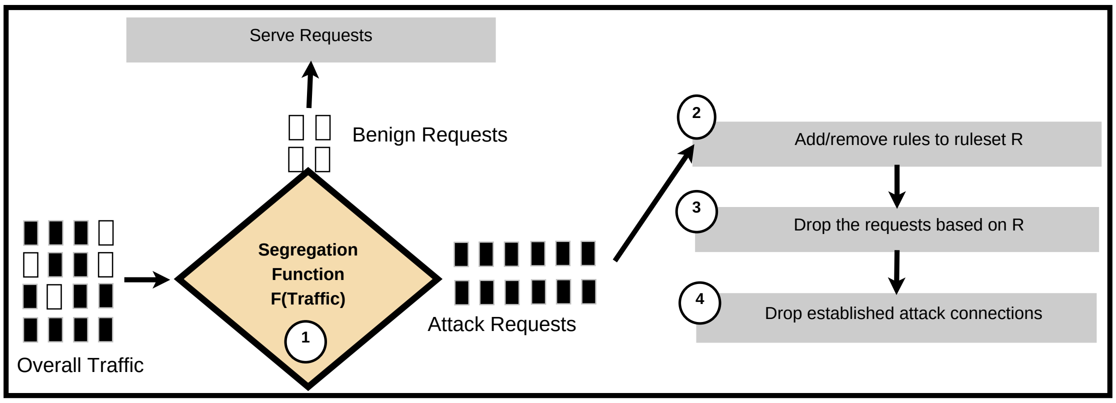

Serve Requests
Benign Requests
Overall Traffic
Segregation Function F(Traffic)
1
Attack Requests
2
Add/remove rules to ruleset R
3
Drop the requests based on R
4
Drop established attack connections

FIGURE 5.2: Generic Architecture of DDoS Defense Mechanisms

A DDoS mitigation method will have following major activities:

1. Segregation: `F(Traffic)` is usually an online algorithm which processes the incoming traffic and filters the traffic based on features such as request and connection patterns. These features of benign or attack traffic behavior help in filtering out the attack traffic. The remaining traffic is assumed to be legitimate traffic and served by the service. There is a large number of such segregation functions which are given in surveys on DDoS mechanisms [16].
2. Add/Remove rules: When the function `F(Traffic)` identifies the attackers, mitigation mechanism will add the attacker source addresses in the block rule-set `R` to enforce the "drop" of incoming attack connections. The firewall services such as `iptables` and APF [142] offer services to perform the "block" and "drop" activities.
3. Drop attack requests: Based on the rules maintained by `R`, the firewall that is also a part of the segregation function drops the attack requests.
4. Drop established attack connections: The final important activity that is dependent upon the output of the segregation function (`F(Traffic)`) would close all the established connections involving the attacker addresses. This activity is usually performed by sending a connection close (e.g. reset) packet using mechanisms offered by TCP.

The above four activities are online and may run simultaneously, especially for attack instances that are repetitive in nature or with changing attack vectors. We would like to highlight the fact that most DDoS mitigation solutions based on some traffic evaluation follow the above generic architecture. We would also like to emphasize the fact that our solution framework is flexible and open to use any segregation function `F(Traffic)`.

5.3 DDoS Attack and its Mitigation: A Real Time Experimental Case Study

In this section, we provide a discussion on the results of few interesting attack experiments launched and targeted to a set of example web services. To conduct attack experiments, we create attack instances on a service with the configuration given in Table 5.1. These attack instances help us in answering few important questions related to the DDoS attack defense in cloud services. We deal with these questions later in this section (with a detailed discussion in Section 5.6) with an effort to find out the answers in our experiments. The infrastructure available to the web-service is similar to a "C4 Extra Large" instance on Amazon EC2 which has 4 vCPUs (64 bit), 7.5 GB RAM. We design the attack traffic by following the classical work in [41]. The web-service under attack is a representative service of most of the modern web services. This dynamic web service runs an image conversion program which converts an image from one format to the other. We are converting a .jpeg image to .gif images for our experiments. As discussed in Section 5.2, we are not using a particular mitigation mechanism but a generic representative of many mitigation methods.

Therefore, the DDoS mitigation mechanism or segregation function (`R`) becomes a supplement to our discussions. Any other mitigation method can be used in place of the mechanism used in this work. We are mostly interested in the cost-resource dynamics while the mitigation approach carries its activity in the presence of an attack. In all the experiments, we are using a popular open source DDoS mitigation mechanism, DDoS-Deflate [87]. This tool is a connection count based filter mechanism working on top of `netstat` utility to count connections coming

Resource	**Configuration**
Physical Server	Dell PowerEdge Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
Total CPUs	8 Processors (4 cores each)
Total Memory	96GB
Hypervisor	XenServer 6.5
Vicitm/Attacker/Benign OS	Ubuntu 14.04
Victim Service	Dynamic Web service Apache2-PHP
Victim Configuration	4CPUs and 8 GB
Attacker Configuration	2 CPUs and 1 GB
Benign Configuration	2 CPUs and 1 GB
Attacker/Benign Application	ApacheBench2
Attack Traffic	500 concurrent requests (Total 5000 requests)
Benign Traffic	1 concurrent request (Total 100 requests)
Network	1 Gbps

Table 5.1: Attack Setup

from each sender. We used this tool in its default configuration which flags an address as an"attacker" if the address tries to establish more than 150 concurrent connections. Now, we consider the three attack instances and their impact on a benign user which is accessing the service.

In the first three attack instances (figures 5.3, 5.4, and 5.5), the victim service is converting a 500KB file on each incoming request. If the service is not under an attack, each request usually takes a response time of around∼900ms. Figure 5.3 shows the victim service behavior experienced at benign user's end when there were resources similar to C4 Extra Large instance.

In all the experiments, we schedule the attack and benign traffic arrival in such a manner that they start sending the requests at the same time. Additionally, we made the request timeout values very high (10000s) to see the attack impacts without missing any responses to the requests. The attack starts its impact on the service from the very first request and makes the victim service unavailable instantly. The DDoS mitigation mechanism starts its work with a frequency run of each 5 seconds. The attack gets detected and reported after∼36 seconds of

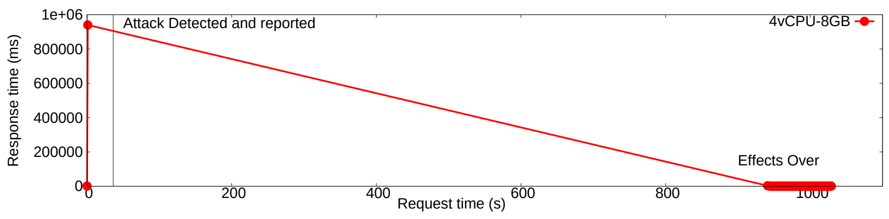


FIGURE 5.3: Request Response Behavior during a DDoS Attack (Resources on Victim Service=4vCPU-8GB)

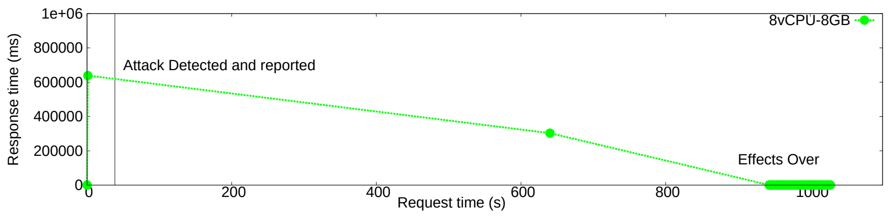


FIGURE 5.4: Request Response Behavior during a DDoS Attack (Resources on Victim Service=8vCPU-8GB)

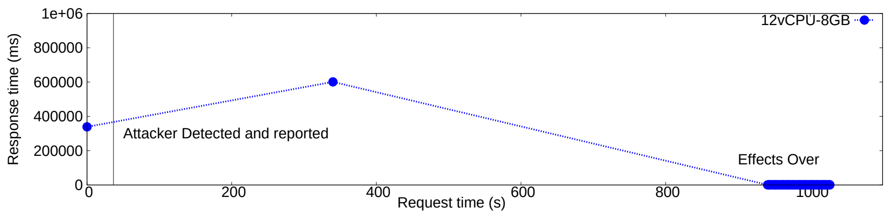


FIGURE 5.5: Request Response Behavior during a DDoS Attack (Resources on Victim Service=12vCPU-8GB)

the launch of the attack. The total downtime of the victim service is 939s. We repeat the same attack by giving more resources to the victim service (shown in figures 5.4 and 5.5). We made this "resource increase" in the CPU resources to see if the attack mitigation gets fastened. "Resource increase" is also motivated by the recommendations made by many DDoS mitigation mechanisms which advocate to use resource scaling during the attack [65, 92]. In the 8vCPUs-8GB and 12vCPU-8GB victim instances, we see a insignificant difference from the perspective of the attack detection time, the attack reporting time and the total victim service downtime.

Now, we initiate the same attack with a change in the victim service's behavior. Instead of processing a 500KB image, we use the image size of 2MB. This change is in consonance with the average page size of the web pages across the globe from a popular survey [143]. A single request to the web server for this image takes around 4.5s while there is no attack. We performed two experiments with the image size of 2MB and the attack configuration as shown in Table 5.1. We show the resultant request graphs in figures 5.6 and 5.7. In these attack cases, the service behavior changes completely and the response time becomes very high owing to the image size. Another important factor of these attack instance is that we were unable to know the time when the mitigation service detects the attack. Mitigation service reported the attack only when the attack effects were completely over.

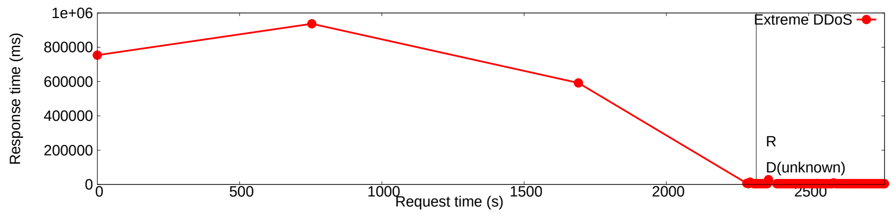


FIGURE 5.6: Request Response Behavior during a Extreme DDoS Attack (R= Time of reporting and D = Time of detection)

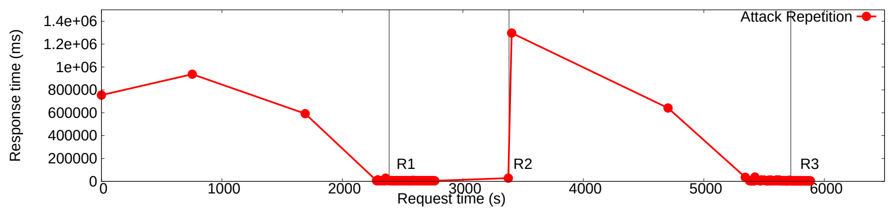


FIGURE 5.7: Request Response Behavior during Repeated Extreme DDoS Attacks (R1= Time of reporting attack 1, R2 = Start of attack 2, and R3= Time of reporting attack 2)

The resource usage in the "extreme" attack instances (shown in figures 5.6 and 5.7) was maximum (CPU and memory usage reaching to 100%). Similarly, no service was available during the whole attack period, till the time when all the attack effects subsides. On the other hand, in all the three attack instances (figures 5.3, 5.4 and 5.5), we were able to use the victim server for other services, during the victim service downtime. Another crucial fact to observe, as in many of the attack instances, the victim server's other services (e.g. *ssh*) also becomes unavailable owing to the intensity of the attack and the heavy resource usage. We observe that the attack had lead to the resource starvation for all the services on the machine.

It is also worth highlighting that the resource starvation or the "resource race" (as pointed in Chapter 3) was so severe that the mitigation service could not even report the detection of the attack. We configured the mitigation service to write the time of mitigation activities such as detection and reporting in a file. The reporting of these activities was very comfortably performed in the initial three attack instances. However, in the case of Figure 5.6, the mitigation service could not even get the file in the memory and write into it. We term this attack scenario as "extreme DDoS" as there is an extreme service denial at the server.

Figure 5.7 shows a repetitive attack instance where we repeated the attack after the effects of the first attack are over. Recent attack reports show attack incidents where attacks are repeated in a "stop-start-cycle" with attack repetition after few minutes to hours [2]. Repetition may come with changes in the attack vector, attack sources, and the attack size. In the case of repetition, once the service is recovered from the first attack, we start the next attack after 10 minutes. In attack repetition, we see a repetition of effects of the "extreme DDoS". In the attack repetition cases, the detection time of the attack is not known and service is only available after the attack effects are completely over. We summarized the results of all attack instances in Table 5.2. Attack repetition results are given in Table 5.3. It is visible that in the repeated attack instances are equivalent to "two" individual extreme DDoS attacks with a large attack cooling down period.

5.4 DDoS Mitigation Requirements: Discussion

Based on the attack instances and the experimental outcomes, we discuss and design a set of five important requirements related to DDoS attacks on cloud services. These requirements are equally relevant to the cases of DDoS attacks to "fixed" infrastructure services. Based on these observations and design requirements, we propose our DDoS mitigation framework in the next section.

5.4.1 R1: DDoS Mitigation in the Presence of Attack

DDoS attacks aim to create "denial" of the victim service. Victim service becomes unavailable due to the lack of resources and more and more incoming requests. In the presence of the attack both DDoS mitigation method and the victim service need more resources that are not readily available. The attack mitigation behavior is quite visible when we differentiate a "DDoS" attack and an "extreme DDoS" attack. In DDoS attack (Figure 5.3), the mitigation mechanism was able to work in the presence of attack and could perform the mitigation activities like adding rules, dropping the subsequent connection and terminating the established attack connections while reporting the attack. On the other hand, in the case of extreme DDoS attack getting the required resources to perform the activity was very competitive as resources like CPU time and memory were heavily used by the web service under attack. Therefore, it becomes difficult for the victim service owner to monitor the state of the victim service. Providing additional support in the form of fault-tolerance and recovery is also difficult without accessing the service. Usually, additional support is given by providing additional instances, other essential resources like memory, and by monitoring the situation manually for attacks and vulnerabilities. These support and recovery mechanisms always require information from the victim server, as without knowing the state and gaining the access, no other supports are useful. We need mechanisms which can help

Attack	**Resources**	**Time of Attack Detection**	**Time of Attack Reporting**	**Downtime (Victim Service)**	**Downtime (Other Services)**	**No. of Attack requests served before detection**
DDoS	4vCPU-8GB	36s	0m36s	939s	0s	45
DDoS	8vCPU-8GB	39s	0m39s	943s	0s	394
DDoS	12vCPU-8GB	37s	0m37s	941s	0s	452
Extreme DDoS	4vCPU-8GB	Unknown	2315s	2294s	2294s	27

Table 5.2: Various Attack Metrics

Attack in Repetition	**Time of Attack Detection**	**Time of Attack Reporting**	**Downtime (Victim Service)**	**Downtime (Other Services)**	**No. of Attack requests served before detection**
Attack 1	Unknown	2388s	2290s	2290s	40
Attack 2	Unknown	2345s	2259s	2259s	42

Table 5.3: Attack Repetition Results

in providing access to other services and resources for mitigation mechanisms even in the presence of the extreme DDoS attacks.

5.4.2 R2: Victim Service Availability After the Attack Mitigation

Attack mitigation has multiple facets such as attackers identification, blocking, and clearing established connections. The time taken by each of these activities is important to estimate the overall downtime and subsequent service availability time. We see in Table 5.2 that even though the attack was detected at 38s the service became available only after a much longer time (940s). We term this time period as "Attack Cooling Down Period, T_C" which is the total time taken by the services to recover after the attack detection. There are a number of contributions available to perform quick DDoS mitigation; however, there are no contributions towards quantifying or reducing the T_C. Our work makes novel contributions in the direction of minimizing T_C.

5.4.3 R3: Availability of Other Services During the Attack

Most administrators report about the unavailability of any access channel (including manual terminal access) to the victim service during the attack. Even cloud-based mitigation methods require a channel to perform the mitigation at the victim side. In all the extreme attack instances, the interactive services to access the victim server were unavailable. We are also interested in the performance of other critical services in the presence of an attack to the victim web-service. Performance of these services can be monitored by considering their availability (or intermittent availability) and the response time. We detail the availability aspect of other services in Section 5.6.1.

5.4.4 R4: Effect of Scaling on Mitigation and Sustainability/costs

Most of cloud hosting service providers propose to mitigate DDoS attacks by scaling the service under attack [54] [91] [92]. In DDoS attack cases, we did not observe significant change in overall downtime, attack detection and reporting time even with scaled up resources. On the other hand, an interesting statistics was observed which is related to the the total "attack" requests served during an onslaught. If we have more and more resources available on the server, then the attack requests entering the service queue will rise up before they get classified as "attack" requests. In this case, the service will try to respond to more and more attack requests and make the detection difficult due to the "resource race".

We look at the size of outgoing bandwidth spent on serving the attack requests which is directly proportional to the no. of attack requests entered into the system. In any sophisticated attack, if the attack detection does not take place even after a long time, and if we infuse more and more resources (anticipating a quick detection and mitigation), the system will have large number of requests resulting in massive attack mitigation costs. The attack mitigation costs include the cost of additional resources in terms of additional physical resources or VM instances and the cost of the mitigation software. These direct costs exclude the other costs such as business losses and the cost penalties owing to the service downtime. The incoming bandwidth to a cloud-based service is free up to an extent; however, the more expensive outgoing bandwidth may result in severe economic losses [63].

There is a high probability of detecting the attack quickly with very large amount of resources [92]. Though cloud computing utility models follow hourly pricing model, still, it may reach to a multi-fold sum of plain hosting costs without any presence of attacks. Costs of losses may become significant if the attacks continue for a long period or repeated or launched with additional stealth and sophistication. On the other hand, a secure remote accessibility to the victim machines is still needed to employ the support mechanisms. In the case of network layer/overlay based detection this might not be entirely true; however, most of the application layer methods will surely need the access.

5.4.5 R5: Repeated/prolonged Attacks and Variable Attack Vector

The survey reports in [2] suggests that there were many attack incidents, where the attacks continue for longer duration and bring variations in attack features. At times, sophisticated attacks try to defeat the mitigation mechanism by stealth [144]. As per the attack reports by Arbor Networks [2], repeated attacks may come in a "start-stop cycle" after some intervals. We showed a case of repeated attack instance in Figure 5.7 and Table 5.3. It is very difficult to anticipate the attack repetition to prepare the defense. The mitigation mechanisms should be able to circumvent the attacks as quickly as possible in the presence of repetition. Repeated attacks may bring variations from the perspective of attack rate, type, packets, sources, and the attack duration.

5.5 Proposed Mitigation Framework

In our attack experiments, we observed that an attack with the same "frequency" and "intensity" may bring completely different manifestations on two different victims. On a victim running a light web-service it was mere DDoS, where the mitigation was quick and without many hurdles. On the other hand a web service that does more "resource utilization (work) per request" gets deteriorated to have "extreme DDoS" attack. The survey reports [2] also suggests that most of the reported DDoS attacks have a peak attack bandwidth of∼1 Gbps with most attack targets as SMEs. These small targets are very cautious about sustainability and concerned about the cost of the security solutions. Additional resource scaling should only be used when required in the mitigation. In the following, we propose a solution having the following objectives. We decide these objectives considering the design requirements detailed in the Section 5.4.

1. Minimizing the "Attack Cooling Down Period", T_C.
2. Mitigation with the available resources. Acquiring more resource only when needed.
3. Handling the prolonged attacks and the attack repetitions.
4. Providing quick resources from the available resources for mitigation in the presence of an attack.
5. Minimizing the attack consequences like bandwidth costs and isolation penalties.

5.5.1 Shrink-Expand Based Service Resizing

"Resource race" formed owing to DDoS attacks is mostly due to the CPU and memory resources. To achieve the five objectives listed above, we propose a novel "service resizing" method which provides more resources to DDoS mitigation methods in the presence of an attack. Our proposed method frees the resources by shrinking the "resource intensive" victim web-service to a minimal set of resources thus reducing the attack surface. We use OS level processor affinity methods [145][146] to affine the services to few or more processors dynamically. These methods are accessible using the affinity utilities such `taskset`.

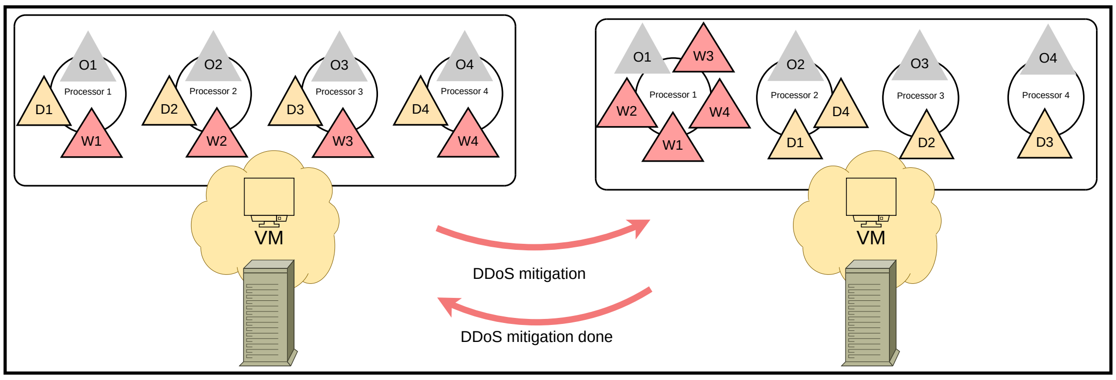


FIGURE 5.8: Resizing Services for DDoS Mitigation

We illustrate the service resizing mechanism in Algorithm 4 and a related function RESIZE() in Algorithm 5. We also show the process of service resizing in Figure 5.8. This algorithm works by regularly monitoring two important service parameters, (i) Request response time, T_{req} and (ii) Number of established connections = N_{est}. We assume that the victim service has a defined set of service capacity parameters that show the maximum supported connections as per the capacity (N_{max}) and an acceptable request timeout at the client end (T_{to}). Our algorithm checks these parameters at regular intervals. Attack detection time would decide the time taken by overall attack mitigation time. The proposed algorithm checks if both the parameters are under control using a condition that tests them ($T_{req} >= T_{to}$ && $N_{est} >= N_{max}$). If this attack condition becomes true (as the attack is present) in that case the algorithm will immediately go for service resizing.

In case of the C4 instance, the resources available to the VM are 4 vCPUs and 8GB of memory. We resize the services utilizing the CPU affinity utilities available for compute resources (vCPUs). We will assign the minimum resource `MinR` = 1 vCPU to the victim web-service and `R-MinR`=3 vCPUs to the DDoS Mitigation Service (`DDoSMS`) and other services. We argue that the resource shrinking confined to the minimum resources (in this case 1 vCPU) provides free resources to the mitigation methods. The presence of the extreme attack is an important information to proceed with the decision of shrinking. Resource shrinking and expansion will allow the DDoS mitigation service `DDoSMS` to get the maximum compute power which is also isolated from the victim service. After resizing, the attack requests will be limited to just `MinR` resources, and the `DDoSMS` will be able to perform its activities comfortably using the dedicated resources. Once `DDoSMS` detects the attack it will perform all the related activities. Once the attack cooling down

Algorithm 4: Service Resizing Algorithm (Shrink-Expand)

```
SHRINK-EXPAND;
Data: Request response time = $T_{req}$,
Request timeout =$T_{to}$,
Number of established connections = $N_{est}$,
Maximum connections as per the capacity = $N_{max}$,
Victim service = $WebService$
DDoS mitigation service = $DDoSMS$;
Total resources = $R$,
Minimum resources to run the WebService = $MinR$,
Result: Attack Mitigation Successful
initialization;
while ($T_{req} < T_{to}$ && $N_{est} <= N_{max}$) do
    Nothing
end
RESIZE(WebService, MinR);
RESIZE(DDoSMS, R-MinR);
while ($T_{req} >= T_{to}$ && $N_{est} >= N_{max}$) do
    Nothing;
end
RESIZE(WebService, R);
RESIZE(DDoSMS, R);
Show “Attack Mitigation Successful”
```

Algorithm 5: Resize() function

```
RESIZE();
Data: Service $S$
Resources = $M$,
Result: Service S’s new affinity is M
initialization;
Find out all the instances of S;
Change affinity of S to M;
```

period T_C passes the algorithm will succeed in mitigation and resize the services back to their original form (resources `R`).

To support our claims, we conducted the experiments again with the proposed algorithms (Section 5.6). We also performed additional experiments to demonstrate how an “operating system level resizing” may help using separate and isolated resources. As discussed in the Section 5.2, adding rules to the firewall is an important activity during the overall DDoS mitigation activity. In the presence of an attack (similar to Figure 5.3), we add and remove a number of rules to the firewall. First, we perform this operation on shared resources with the victim web service. We also perform the same operation on separate resources to see the impact of isolation and separation. We show the results of this experiment in Table 5.4. It is visible that resource sharing with the victim service under attack results in heavy performance penalties which are as large as four times the actual time taken using separate resources (e.g. time required to add/remove 500 rules).

No. of Rules	Shared Resources		Separate Resources	
	Addition	Deletion	Addition	Deletion
100	0m0.381s	0m0.547s	0m0.090s	0m0.097s
500	0m2.095s	0m2.069s	0m0.582s	0m0.590s
1000	0m5.479s	0m4.450s	0m1.505s	0m1.614s
5000	0m50.811s	0m49.552s	0m30.045s	0m30.200s
10000	3m5.560s	2m59.750s	2m22.881s	2m22.743s
20000	11m28.439s	11m29.298s	9m50.675s	9m55.632s

TABLE 5.4: Firewall: Shared Resources vs. Separate Resources

5.5.2 Minimizing Attack Cooling Down Period using TCP Tuning

We saw in Section 5.3 that the attack cooling down period has a major contribution in the overall service downtime. The process of clearing up "established" attack connections is an important step in the overall mitigation activity. These established connections may have both the attack connections and the benign user connections. However, in the extreme attack cases, the downtime results into successive timeouts for benign users. Usually, the connection removal activity is performed by identifying the sequence number and sending an RST (e.g. tcpkill). We propose a novel methods of "TCP tuning" in which we supported the connection removal activity by tuning two important TCP parameters which help in clearing the established attack connections quickly. However, to maintain the service quality, we reset these parameters back to their original values once the attack downtime is over. These two parameters are, *tcp_fin-timeout* and *tcp_retries2* [147]. We set their values to "10s" and "1 retry" respectively.

1. *tcp_fin-timeout*: This parameter decides the time for which sockets will be in state FIN-WAIT-2. It is an important parameter to assist in the early removal of attack connections as the victim service has already closed the connections.

2. *tcp_retries2*: This parameter decides the number of retries to be performed before killing an alive connection.

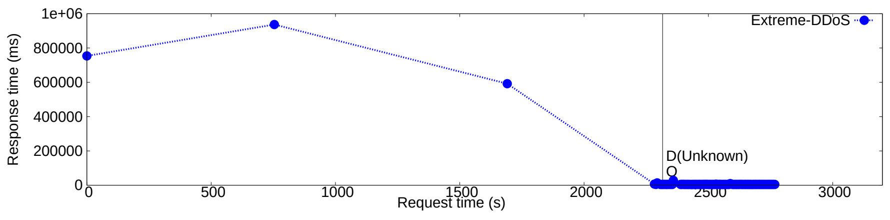
By setting the above parameters, we may lose some benign connections; however, loosing some benign connections during attack downtime is reasonable. Victim service looses the benign connections during the extreme attacks. By employing the proposed techniques we show that the reduction in overall downtime, also results in reduction in the loss of benign connections.

5.6 Evaluation and Results

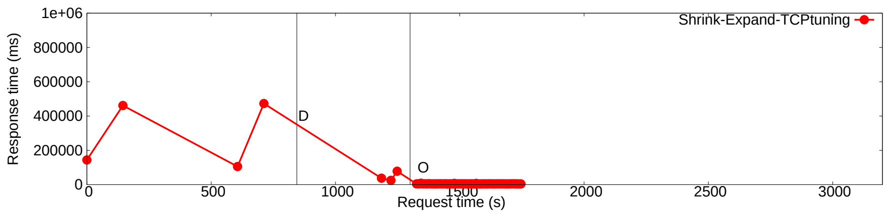
We perform the following attacks to evaluate the effectiveness of proposed service resizing algorithm and the TCP tuning technique. We show the detailed results in Figure 5.9, Figure 5.10 and Table 5.5.

1. Extreme DDoS attack with Shrink-Expand and TCP Tuning
2. Extreme DDoS with Shrink-Expand and without TCP Tuning
3. Extreme DDoS without Shrink-Expand and with TCP Tuning
4. Repeated extreme DDoS attack with Shrink-Expand and TCP Tuning

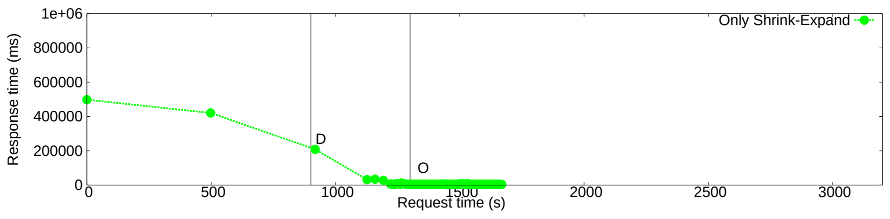
We compared the results of attacks mentioned above (point 1,2 and 3) with the extreme DDoS attack instance, discussed in Section 5.3 and Figure 5.6. Similarly, we compare the attack described in point 4 with the attack incident shown in Figure 5.7. We show the attack outcome without applying any of our proposed techniques in Figure 5.9a. We use both “Shrink-Expand” and “TCP Tuning” techniques to support the DDoS mitigation process in Figure 5.9b. The results show improvement in all the important parameters such as the attack detection time, the attack reporting time, the downtime for the victim service and the other services, and the total number of attack requests served by the victim. When we only use “Shrink-expand”, the attack detection time is increased to 901s that was 845s when both the techniques were employed. We can also see in the Figure 5.9b about the response time downfall during the downtime for few requests that is not available in Figure 5.9c. On the other hand, in case of Figure 5.9d, the attack detection time is increased to more than 950s owing to the unavailability of “Shrink-expand” mechanism. However, “Only TCP Tuning” setting results into very small improvement (2118s as compared to 2294s in Table 5.5) in downtime of victim service which signifies the requirement of isolated resources for the DDoS mitigation service. We see that quick connection release during the attack (using TCP tuning) support the isolated resource availability in the presence of the extreme DDoS attacks.



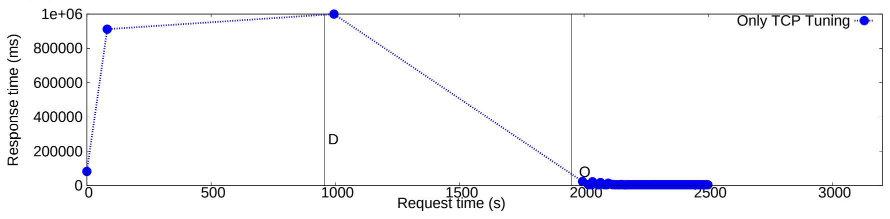
(A) Extreme DDoS Attack (No Shrink-Expand and No TCP Tuning)



(B) Extreme DDoS Attack with Shrink-Expand and TCP Tuning



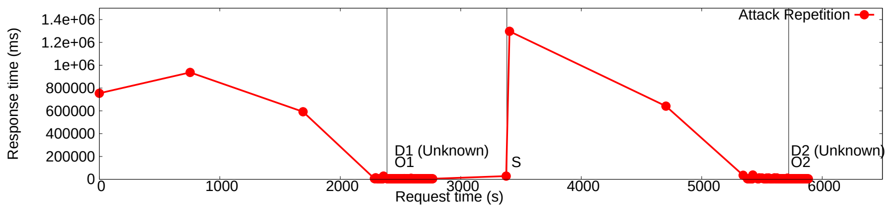
(C) Extreme DDoS Attack with Shrink-Expand and without TCP Tuning



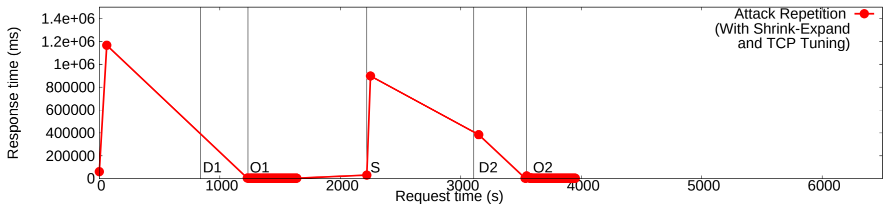
(D) Extreme DDoS Attack with TCP Tuning and without Shrink-Expand

FIGURE 5.9: Evaluation and Results of Shrink-Expand and TCP Tuning (D= Time of Attack Detection and O= Attack Effects Over)

For the cases of repeated attack incidents, we compared the performance of the proposed techniques (Figure 5.10b) with the cases in which we do not employ these techniques (Figure 5.10a and Table 5.3). In these attack cases, we achieve a performance improvement in the mitigation process with a quick attack detection and attack reporting. The case of repeated attacks one after another (shown in Table 5.6), show similar performance metrics to the extreme attack cases (in Table 5.5).



(A) Attack Repetition with no Shrink-Expand and no TCP Tuning



(B) Attack Repetition with Shrink-Expand and TCP Tuning

FIGURE 5.10: Evaluation and Results of Shrink-Expand and TCP Tuning in Case of Repeated Attacks (D1= Attack 1 Detection Time, O1= Attack 1 Effects Over, S= Start of Attack 2, D2= Attack 2 Detection Time, and O2= Attack 2 Effects Over)

Attack	**Resources**	**Time of Attack Detection**	**Time of Attack Reporting**	**Downtime (Victim Service)**	**Downtime (Other Services)**	**No. of Attack requests served before detection**
Extreme	4vCPU-8GB	Unknown	2315s	2294s	2294s	27
Extreme	4vCPU-8GB (Shrink-Expand and TCP Tuning)	845s	845s	1326s	1326s	10
Extreme	4vCPU-8GB (Only Shrink-Expand)	901s	901s	1278s	1278s	12
Extreme	4vCPU-8GB (Only TCP Tuning)	956s	956s	2118s	2118s	34

TABLE 5.5: Attack Results after Applying Shrink-Expand and TCP Tuning

Attack in Repetition	**Time of Attack Detection**	**Time of Attack Reporting**	**Downtime (Victim Service)**	**Downtime (Other Services)**	**No. of Attack requests served before detection**
Attack 1	840s	840s	1234s	1234s	8
Attack 2	851s	851s	1321s	1321s	12

TABLE 5.6: Attack Repetition Results after Applying Shrink-Expand and TCP Tuning

5.6.1 Discussion and Issues

Extreme DDoS attacks occur owing to the heavy resource sharing at the level of operating systems. The virtual machines are usually highly isolated as compared to the processes at the level of an operating system. The proposed techniques should not be used and are rather not useful where these resource contentions are not severe. We discuss following important issues about the proposed scheme considering the evaluation results.

Deciding when to resize : The service resizing is only needed when the service is facing an extreme attack. Anticipating a DDoS attack to take the shape of extreme attack depends on the service and the amount of efforts it spends on each request. Resizing may also result in downtime for benign users in the presence of low rate DDoS attacks. In this case, we would like to adopt step-wise resizing (e.g. freeing just 1vCPU for the DDoS Mitigation).

Attack requests in the system : The number of attack requests entered into the system are directly proportional to the time it takes to detect the attack source. Therefore, attack detection time will grow due to the increased downtime, increased network bandwidth spent on attack, and the larger attack cooling down period.

Network bandwidth : An admission control on the attack requests helps in achieving the network isolation which leads to the minimization of the collateral damages and the energy consumption.

Attack strength : The service resizing tries to help in one critical aspect which is the impact of attack strength. If the attack comes with a minimum rate and achieves the “extreme DDoS”, increasing the attack rate further will not have any adverse impact on the service under the attack. Resizing will always bring the victim web service to the `MinR` resources.

Availability issues : In the cases of extreme DDoS attacks without using the proposed techniques, victim service faces a huge downtime. With the help of the “Shrink-Expand” approach and the TCP Tuning technique, the downtime is reduced to achieve the service availability. After the attack is over, the services are resized back to the original resources to maintain the availability.

Attack repetition : There is no direct and obvious way by which we can know that an attack is going to be repeated in the future. Therefore, after completing the mitigation requirements through shrinking, we will again expand the resources. If there is another attack before this expansion than the web service will remain shrunk with the minimum resources.

Attacks during downtime: There may be other DDoS attacks (may be even with a changed attack vector) once the service downtime is reached. In this situation, the attack mitigation will be quick as compared to the case where “Shrink-Expand” or TCP Tuning are not utilized because of the resource availability. In case the attack detection or mitigation is not possible within the available resources, the traditional auto-scaling methods may be used to scale the service.

Overhead of resizing : Shrink-Expand overhead will be similar to the overhead of moving tasks from one CPU to another CPU using context switches used in preemption and global load balancing.

Availability of other services : Other services were not completely available (intermittently available) in the case of resizing of applications. This is mostly due to heavy memory usage by the victim service under attack. The service resizing at the level of memory can ensure the availability of other services. However, owing to a large decrease in downtime, the services were restored quickly.

Resources available : Traditionally, the resource requirement is not among the primary aims while designing DDoS mitigations solutions. We could see that giving more resources may not help in few attack instances unless the resource contention issues get solved. On the other hand, we could spare 3 vCPUs using resizing which is equivalent to having 75% of C4 Compute resources (4 vCPUs) without any additional costs.

5.7 Conclusion

DDoS attacks on cloud services see various trends owing to the nature of the business model supported by cloud computing. We see a conversion of DDoS attacks'"arms-race" into a "resource-race" owing to the emergence of cloud computing services. There is an immense need of methods to characterize and mitigate these attacks on cloud environment.

In this chapter, we conduct real attack experiments on cloud services to critically observe the overall mitigation activity at the much fine grain level, i.e., at the resource level. DDoS attacks being resource based attack turn into the "extreme DDoS" attacks for services with high resource utilization per request. We characterized these extreme DDoS attacks and observed that the resource contention created by the victim service under an attack may also compromise the DDoS mitigation service itself. Additionally, in these extreme DDoS attacks, availability after the attack detection is also affected owing to a longer attack cooling down period.

To circumvent these problems, we proposed a framework to support the overall mitigation activity. Our supporting framework puts efforts to provide enough resources such that the mitigation mechanism can perform its task even in the presence of extreme DDoS attacks. For this purpose, we perform attack experiments and highlight the need for methods to minimize the service downtime after the attack detection. We proposed a novel supporting framework which employs our processor affinity-based service resizing and the TCP tuning technique during the attack period to serve two important purposes, (i) providing required resources to mitigation mechanism, and (ii) minimizing the overall downtime.

We perform detailed experiments to show the efficiency and efficacy of our scheme. The novelty of our scheme opens up multiple directions of research to visualize the inter-service relationship on an operating system. Additionally, the behavior of other unrelated services and providing access to attack mitigation techniques involving resource scaling, are few other directions which are open and relevant. Isolation and separation of victim services concerning other basic resources such as memory, disk, and bandwidth, is another direction which may be extended to support the proposed work.

In the next chapter, we extend our contributions to minimize the resource contention generated owing to the "extreme DDoS" further to guarantee various resource to ensure the availability of other critical services.

Chapter 6

Victim Service Containment to Minimize Internal Collateral Damages

6.1 Introduction

DDoS attacks are usually considered as a huge pool of resource intensive requests which overload the target service resources. The target resources are usually the basic resources such as CPU cycles, memory and swap usage, I/O operations, and network bandwidth. Additional application level resources include the number of simultaneous connections, ports, concurrent sessions, application buffers and other temporary identifiers. All the co-located processes share the server resources to achieve their goals. A victim web-service, DDoS mitigation service, logging and scheduling processes are few examples of such processes running on top of an operating system.

In this chapter, we provide a novel observation related to the operating system level resource contention which may arise among the co-located services/processes. We argue that the services co-located with the DDoS victim service (say a web-server process) may not provide the expected processing and timely outcome due to the extensive resource contention formed by the incoming DDoS attack. These co-located services include all the important services such as DDoS mitigation service, firewall, and internal security policy services (e.g. SELinux). We show this phenomenon in our experiments and term it as "internal collateral damage" which severely affects the co-located non-target services. In this chapter, we show that

DDoS Mitigation methods may not provide the expected outcome and delivery due to this phenomenon.

To approach the "internal collateral damage" problem, we provide an analysis of DDoS attacks from the perspective of OS level resource managements. Cloud computing infrastructures usually employ virtual machines (VMs) to ensure a strong performance and resource isolation. However, the internal operating system level isolation cannot be governed by the isolation provided by the virtual machines.

In this chapter, we argue that resource isolation among co-located services may provide quick and efficient DDoS mitigation. We propose a novel approach "Victim Service Containment" to achieve resource isolation of victim web-service resources. We perform real-time attack experiments to show that the proposed approach provides resources availability to all the co-located important services and also helps in minimizing the overall attack effects and costs. We provide attack cases and model resource requirements to provide solutions based on the resource control groups [148]. We also extend the discussion to limit DDoS effects to just the target service by eliminating or minimizing the additional collateral damages.

The rest of the chapter is organized as follows. Section 6.2 provides the details about DDoS attack protection at various levels of the cloud architecture. Section 6.3 provides details of the experiments showing the novel "Internal Collateral Damages" phenomenon. We provide a detailed resource management model to showcase the resource contention problem in Section 6.4. Section 6.5 provides the proposed design to eliminate the internal collateral damages. Section 6.6 provides the details of the evaluation to show the efficacy of the proposed solution. We also discuss the features of the proposed solution and various aspects related to DDoS mitigation in Section 6.2. We provide conclusions in Section 6.7.

6.2 DDoS Attack and Protection in the Cloud

A command and control (C & C) attack server coordinates the DDoS attacks with the help of a large group of malware infected network of computers. These computers are also known as "bots". The modern cloud infrastructure can also be used as a attack infrastructure with the emergence of pay-as-you-go "DDoS for hire" services. The impact of these attacks depend upon various attack dimensions. These attack dimensions include attack frequency, attack duration, type of attack packets, number of sources, target victim services, etc. On the other hand, the primary objective of the victim web-service is to remain available to serve the customers. In Figure 6.1, we consider three important cloud architecture junctions, where the DDoS detection and mitigation mechanisms are usually employed. These three junctions are cloud network edge, host physical server's virtualized network and the victim VM's own network and application.

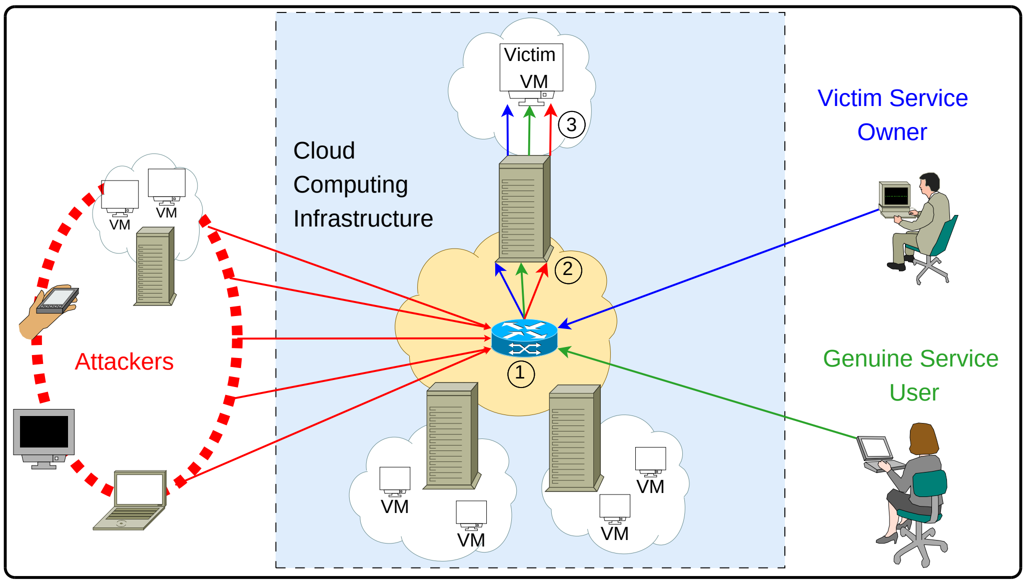


FIGURE 6.1: DDoS attacks targeted at cloud services

Most of the DDoS attack detection mechanisms work on top of the traffic filters, access pattern anomaly detections or other detection mechanisms based on attack related behaviors. There are many recent detection methods which in addition to relying on traffic filters also utilize the cloud resource management and auto scaling capabilities [92] [46]. We propose the following essential requirements for effective DDoS mitigation in victim cloud services:

R1. Quick attack detection and mitigation with minimum downtime
R2. Sustainability/budget aware mitigation
R3. Minimum collateral damages

Requirements R1 and R2 are quite important from the perspective of the mitigation costs and the availability of resources. The victims' organizations are much more concerned about the costs of DDoS mitigation solutions as the mitigation costs directly affect their IT and security budgets. On the other hand, we can achieve the minimization of the service downtime and the subsequent savings through quick DDoS mitigation. The requirement R3 is important from the perspective of minimizing the effects to non-targets and attack spread. We showed in Chapter 3 that DDoS attacks in cloud computing may also affect the co-hosted

VMs and the associated services run by these VMs. We also discussed that the effective mitigation can be achieved by minimizing the overall attack effects using stronger isolation and victim service resource separation.

We extend the notion of resource isolation from hypervisor level to individual VM and OS level. Each server operating system usually runs a number of utilities to support the overall working of the victims' service in addition to its protection, maintenance and recovery. We show a typical set of services provided by each server operating system in Figure 6.2. On a server OS, these co-located services support the main service such as a web-service by offering services related to the service health, backup, and monitoring. None of the co-located services are

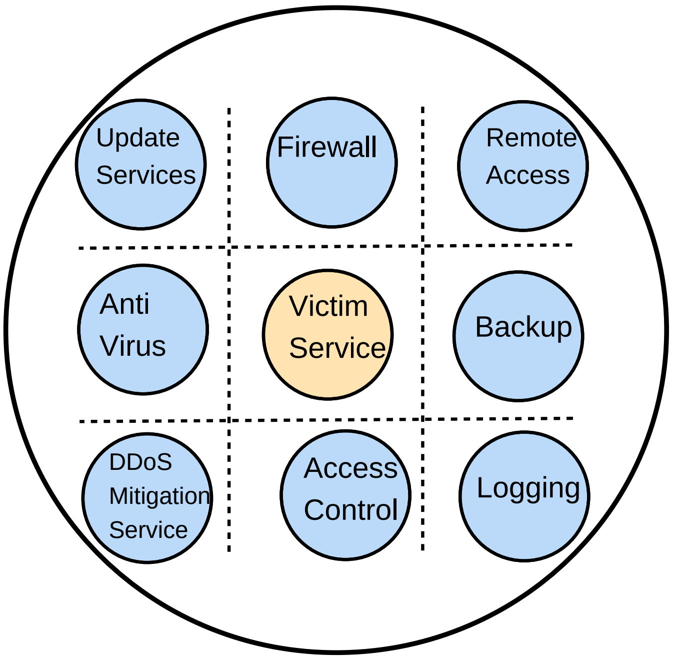


FIGURE 6.2: Various essential services on a typical server operating system

directly accessible by a web-service user or an attacker. The VM owner accesses required services using a "remote access" service. These services include update services for server software, security software patches, firewall, DDoS mitigation service, backup services, logging services, and all the other services available on the operating system. Many of these services are critical to the system availability, security, and fault tolerance. Ideally, these services should always be available and remain unaffected by the DDoS attacks on victims' web-service. The availability of these co-located services even during the DDoS attack period would assist in the following activities:

I. System state awareness: The administrator can identify the real cause of web-service unavailability with the help of other services. If the services such as remote access becomes unresponsive and unavailable, the administrator has no other methods available to access the VM. Additionally, if the services such as DDoS mitigation service is unavailable owing to heavy resource usage by victim's web-service, we may observe unwanted delay in the overall mitigation activity leading to increased losses. Therefore, the system state awareness is one of the most important and primary requirement to achieve any mitigation objectives.

II. Helping the critical situation: The co-located services also help in the critical state formed by the attack or other faults. The availability of such co-located services helps in providing additional assistance such as fault tolerance by backup servers, additional resources using auto scaling, VM migration, load balancing, and application backups. At times, the mitigation process may also require manual intervention to recover from the attack. There are many recent DDoS attacks, known as "Smoke-screening attacks" [149] that launch other severe attacks such as "data theft" behind the massive DDoS attacks. The success of smoke-screening attacks lies in the heavy consumption of victim resources during the attack. These consumed resources may include the computing resources as well as manpower resources to help the attack mitigation.

The above requirements may also help in minimizing the smoke-screening attacks if other services looking after data breaches remain available during the DDoS attack and react to the situation. For example, assume DDoS attack `A1` is targeting a service `s1` running on a VM. Service `s2` is there to take care of some other important attacks such as data breaches. There is a smoke-screening attack `A2` that bypasses the service `s2` on the same VM. As the service `s2` is no more available owing to the internal collateral damages. In this case, the attack `A2` might become successful as the service `s2` is unavailable for the required protection. We discuss the resource contention scenario further in Section 6.4.

6.3 DDoS Attack Mitigation: A Closer Look

In this section, we extend our discussion on DDoS attack mitigation and its resource requirements. To quantify this, we perform experiments to critically analyze the attack launch and subsequent attack mitigation. We show the experimental setup in Figure 6.3 and experimental configuration settings in Table 6.1. In the experimental setup, we have a victim VM hosted on one physical server and an attack VM hosted on another physical server. The main motivation to conduct this experimental attack study is to see the effects of DDoS attacks on important and critically required co-located services. These co-located services are running on victim operating system in the form of processes. We prepare the target service by giving resources equivalent to an Amazon EC2 "C4 Extra Large" instance [150]. Our target services is representative service of most of the dynamic web services. This dynamic web service converts images of specified size from JPEG format to PNG format. For our experiment, we are using the input image of size of 1 MB. We used other image sizes to perform detailed comparison with the proposed solution in Section 6.5. These image sizes are representative sample of websites across the globe [143].

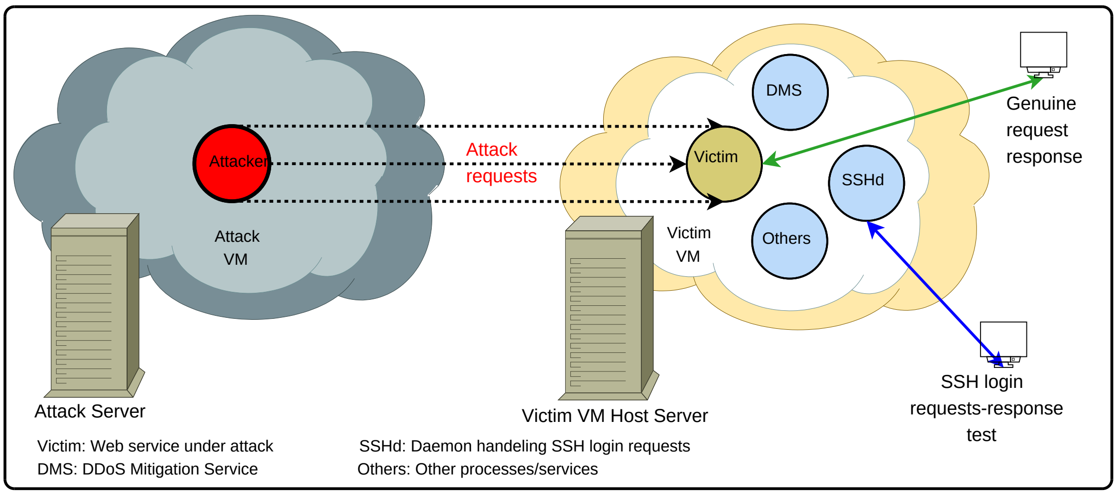

Attacker
Attack VM
Attack Server
Attack requests
Victim
Victim VM
Victim VM Host Server
DMS
SSHd
Others
Genuine request response
SSH login requests-response test
Victim: Web service under attack
DMS: DDoS Mitigation Service
SSHd: Daemon handeling SSH login requests
Others: Other processes/services

FIGURE 6.3: Experimental setup for DDoS attack analysis

Component/Resource	**Configuration Settings**
Victim host physical server	Dell PowerEdge R630, Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
Resources on physical server	8 processors (4 cores each), total memory 96GB
Virtual Machine Monitor/Hypervisor	XenServer Version 6.5
Victim, attacker, benign operating systems	Ubuntu Version 14.04
Victim service	Image conversion web service
Victim VM configuration	Processors 4 vCPUs and 8 GB Memory
Attacker VM configuration	Processors 2 vCPUs and 4 GB Memory
Benign VM configuration	Processors 2 vCPUs and 4 GB Memory
Attacker and benign traffic launch application	ApacheBench2
Attack traffic rate	500 concurrent requests (for a total of 5000 requests)
Benign traffic rate	1 concurrent request (for a total of 100 requests)
Network	1Gbps
Other services for availability test	SSH to have remote login-logout one after another during attack period

TABLE 6.1: Configuration settings for the attack experiments

To see the impact on the availability of other services inside the victim VM, we use a DDoS mitigation service to detect and report the attack. Instead of using any sophisticated DDoS mitigation service, we use `DDoS-Deflate` [87] which is a connection count based attack detection and filtering service. We use this tool with its default settings. These settings allow this tool to identify attackers who are establishing more than 150 connections with the server. The reason behind using this simple detection mechanism is to maintain the applicability of our experiments to any other tool. Therefore, any other DDoS mitigation service can also be used in place of DDoS-Deflate. We performed this attack experiment to closely observe the attack effects such as the attack detection time, overall service downtime and effects on other services.

Instead of sending attack requests from large number of sources, we are sending attack requests from a single source. This enables us to see invariability of attack detection time values reported in the experiments. In case of multiple sources the attack detection times vary depending upon the connection establishment times and share of established connections by each source. We consider "SSH" service as an important co-located service for impact evaluation. SSH is also an important utility for remote monitoring and support system for the administrator. In SSH test, the VM owner sends a SSH session request to the victim server and if the session is granted, it logs out from the session immediately. This test is done for 500 SSH login-logout cycles and starts at the same time of the attack launch.

We tried to keep the attack scenario real by using a dynamic service attacked by a DDoS attack and the victim owner trying to access the victim's machine using SSH service. To check the availability of the target machine, a genuine user sends a request, one after another to the victim's service for a total of 100 requests during the attack period. We launch the DDoS attack on the victim's service by sending 500 concurrent attack requests. We create the attack traffic on the basis of the requirements given in [41]. The attack traffic, genuine traffic and SSH traffic are sent simultaneously to the victim's VM. We show the attack results and associated quantified metrics in Table 6.2. In Figure 6.4a, we illustrate the response time behavior of the victim web service. We show the SSH login-logout completion time in Figure 6.4b. The DDoS mitigation technique could detect the attack source

Attack Reporting Time	**Victim Service Unavailability Time**	**SSH Unavailability Time**	**Maximum Response Time (SSH)**	**Minimum Response Time (SSH)**	**Average Response Time (SSH)**
39s	945s	517s	186.830s	0.129s	1.226s

Table 6.2: Various Metrics: DDoS attack experimental study

based on its policies after 39s of the attack being launched. Subsequently, the victim service becomes unavailable for a period of 945s. The graph in Figure 6.4a represents the request-response behavior at the benign senders' end. The benign sender sends a total of 100 requests with one concurrent request at a time. These graphs show the times when each request (1^{st} request to 100^{th} request) is served one

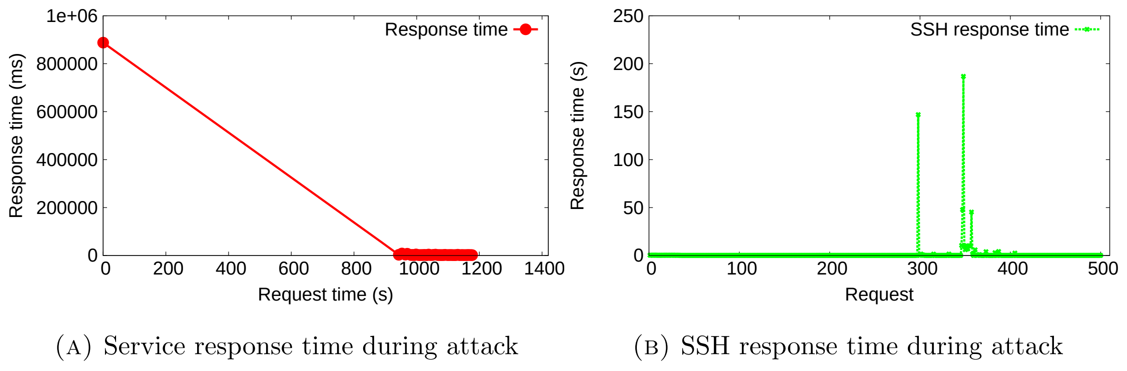


FIGURE 6.4: Behavior of co-located services during attack

after another. The X-axis in Figure 6.4a shows the "Request time" at which the benign sender sends the request. The DDoS mitigation service performs following major activities during the attack period:

1. Count based filtering based on the number of connections
2. Adding rules to the firewall to reject the attack traffic involving the detected sources
3. Cleaning the established connections involving detected attack sources

After detecting the attack, the mitigation service adds rules to the firewall and drops all the tcp connections involving the attackers. The SSH service was not available for more than 500s which includes few intermittent responses near average response time ($\sim$ 1s) (also see downtime spikes in Figure 6.4b). This gives a clear indication of unwanted effects on the other important services. In addition to all these observations, the victims' VM was not responsive on the XenCenter interface that makes the service unavailable even for direct interactions in the data center. We attribute resource contention among services as the major reason behind such attack effects.

We term these attack effects as "internal collateral damages". These services are co-located at the level of operating system and the fair sharing provided by CPU or device scheduler does not provide the required resource isolation among these services. The severity of resource race created by the DDoS attack is such that, even the VM interface is not available. We attribute the observed resource contention to the most basic resources such as CPU, memory, network and disk resources which are required by all the services, though in different proportions at different times. In Section 6.4, we model the operating system level service requirements which will be used in the design of our proposed solution.

6.4 DDoS Attack Mitigation: Modeling OS level requirements

In this section, we provide an abstraction of operating system level resource race and contention formed among the running OS services. We also elaborate the resource exhaustion states and the resulting resource contention. This discussion is also applicable to the traditional operating systems that are not running on top of virtual machines/clouds. A cloud infrastructure runs a number of physical servers (P_i, i = 0, 1, ... n). These servers host various services (S_k, k = 0, 1, ... m). A service S_k in turn runs various VM instances (I_j, j =0, 1, ... p) to host and manage the cloud based services. A Victim's VM which is affected by a DDoS attack would result into a stressing condition of one or more of its resources. The exhaustive set of these bottleneck resources include CPU, memory, swap, disk I/O operations, bandwidth, and number of connections/ports.

The DDoS attacks may also choose to exploit any general and weak target resource or utility which is an application level resource such as a temporary buffer created by a programmer. This buffer may be exhausted by some specific requests or a higher frequency of requests even in the presence of availability of other ample resources. Each physical server has following shared resources: CPU, memory, bandwidth, and disk. We represent these resources using `C` for CPU, `M` for memory, `B` for bandwidth, and `D` for disk. Any other resource of interest is represented by `O` for other. We define a function *Resources*(*entity*) which provides the available resources of an entity such as physical server and a virtual server.

$$Resources(P_i) =< C_i, M_i, D_i, B_i, O_i > . \tag{6.1}$$

Resources available on each VM instance is a subset of the resources available on physical server.

$$Resources(I_j) =< C_j, M_j, D_j, B_j, O_j > . \tag{6.2}$$

The type of virtualization and resource sharing techniques decide the actual resource allocation to VMs. To have a generic view, the allocation of the CPU (`C`) resources are usually represented as number of processors. In case, the VMs share processors, the `C` resource can represent the CPU shares or the CPU time. On the other hand, resource such as memory (`M`) and disk (`D`) are allocated among the hosted VM instances with clear limits on the size. Bandwidth (`B`) is the total throughput on NIC which is limited and shared using network virtualization techniques. The requirements of the hosted instances can only be met, if the sum of individual instance requirements are less than or equal to the resources available on the host physical server. If there are two VM instances I_1 and I_2 to be hosted on P_i, following Equation 6.3 must hold:

$$Resources(P_i) \geq \sum_{j=1}^{2} Resources(I_j) + Resources(H). \tag{6.3}$$

The $Resources(H)$ represents the requirements of the host operating system H on a physical server. Equation 6.3 should also hold true for individual resources such as the sum of the individual resource requirements (for example CPUs, C) of each instance should be less than or equal to the individual instance's CPU requirements. If this equation does not hold true, the auto-scaling algorithm will be triggered to find and provide more resources. These resources can be acquired using migration or creating more VM instances at other physical servers. After observing the effects of DDoS attack cases (Section 6.3), we estimate that the resource race taking place at the level of individual processes should also be taken into account. On the other hand, each VM Instance I_j runs a complete operating system to support the service. The two most important primitives at the level of operating system are processes and resources. These processes include the system processes, daemons, and other application processes. We represent these processes or services by p_l, l = 0, 1,...q. Among these processes, victim service (p_v), DDoS mitigation service (p_d) and other set of processes p_o, are of primary interest to our discussion. Let us take an example of a DDoS attack which impacts the resources such as CPU, memory, and a number of network connections. At any given point in time during the attack, the resources available with the victim process under attack are:

$$Resources(p_v) =< C_v, M_v, D_v, B_v, O_v > . \tag{6.4}$$

The above resource requirement does not show resources occupied by the service at all times. It just represents the need of a service during a given point in time. During the attack, the victim's service will be stressing one or more resources such as CPU, memory, and connections to their maximum. At the same time, the VM owner would want the DDoS mitigation service to act on the situation and perform the detection. Similarly, the other important services such as remote access (ssh) or other linked services such as firewall need to act on time. The resource requirement of DDoS Mitigation Service (p_d) and other important set of processes p_o during the attack period would be:

$$Resources(p_d) =< C_d, M_d, D_d, B_d, O_d > . \tag{6.5}$$

and

$$Resources(p_o) =< C_o, M_o, D_o, B_o, O_o > . \tag{6.6}$$

We can also write

$$Resources(I_j) \geq Resources(p_v) + Resources(p_d) + Resources(p_o). \tag{6.7}$$

If we rewrite Equation 6.7 for individual resources during the attack duration t.

$$C_j \geq C_v + C_d + C_o. \tag{6.8}$$

$$M_j \geq M_v + M_d + M_o. \tag{6.9}$$

$$D_j \geq D_v + D_d + D_o. \tag{6.10}$$

$$B_j \geq B_v + B_d + B_o. \tag{6.11}$$

$$O_j \geq O_v + O_d + O_o. \tag{6.12}$$

As the resources such as memory or CPU are usually not dedicated in their entirety to a single process (unless it is pinned). Not fulfilling one or more of the above equations, is the real cause of delayed action by DDoS mitigation (p_d) and unavailability of other services (p_o) during the attack. In addition, the resource consumption behavior of a DDoS mitigation service is also multi-resource (using CPU, memory, disk, and network). Therefore, the resource contention occurs among these services. From the CPU scheduling perspective, it appears that the resources will be fairly given to each process based on their priority, however the multi-resource and interlinked resource requirements show the internal collateral damages.

The major reason for adoption of virtualization technology as an enabler for cloud computing is owing to the features such as performance isolation among VMs and multi-tenancy. However, virtualization of resources has no major role to play, while the internal performance isolation among processes is considered. The operating system scheduling strategies and resource management activities remains mostly same for operating systems running on both virtual machines and physical machines.

Cloud platforms use the auto-scaling algorithms to govern and facilitate the on-demand dynamic resource allocation in terms of horizontal and vertical scaling. These algorithms monitor various resource usage metrics and based on the resource triggers, they provide resources or withdraw the idle resources. Giving more resources to solve the problem of internal collateral damages among processes, would not be suitable for the following reasons:

I. Auto-scaling algorithms operate as cloud-level or hypervisor-level resource allocation primitives. The resource requirements of individual processes inside the VM, will not be visible to the auto-scaling methods. Additionally, auto-scaling algorithms add additional resources on top of a VM and not to individual processes or services running inside a VM.

II. Many of the existing solutions [92] advocate the application resource scaling to absorb the DDoS attack. However, the resource scaling may not necessarily help to mitigate quickly and provide timely resources to p_d and p_o. The additional resources and instances will also face a similar situation as the victim's service under attack and will also overload the added resources. The additional resource can surely become useful for the mitigation processes if the p_d process can use it readily.

DDoS attack attributes such as attack duration and attack scale also decide the attack effects and resultant impact on the victim service. Based on our experimental study and the above discussions, below are the major requirements for an effective solution to internal collateral damage.

I. **Strong Internal Isolation**: An effective solution should provide a strong resource and performance isolation among processes. Performance isolation should be guaranteed for processes that are used in ensuring the availability and attack mitigation capabilities in a victim's server. Strong isolation can also be ensured by having resource usage limits.

II. **Resource Availability**: Resource requirements to maintain availability is directly related to the requirement I above on strong internal isolation. However, the availability of required resources should be ascertained for each process. Capacity planning and server consolidation areas have a large number of approaches [135] dealing with the actual resource requirements of co-located virtualized servers. Similar solutions can be extended to ensure the timely resource availability to DDoS mitigation and other related services.

III. **Service Performance**: The victim service performance for benign users should not be affected owing to the solution targeting the above requirements. This metric is important as the service quality should not be hampered by the implementations for isolation and availability.

We will use these requirements as the basis of our proposed solution design in the next section.

6.5 Victim Service Containment: Proposed Solution

Based on the effective solution requirements discussed in the last section, we propose a novel solution using the resource containment of the victim process groups at the level of operating system. The primary idea of this solution lies in ascertaining the required resource share for each service. However, in the presence of DDoS attacks, the real requirement is to ascertain and fix the minimum resources required for services (except victim service), to remain available and produce the required and timely outcome. The resource contention situation arising out of DDoS attacks is the major cause behind the unavailability. Operating system schedulers try to achieve these requirements by employing various scheduling primitives and access primitives such as process groups. However, they do not provide the strong guarantee about the control against situations such as "internal collateral damages". The proposed algorithm is a proactive approach for resource calculation. This algorithm evaluates the resource requirements of all the services (except the victim service), while the service is not under attack. Resource containment limit is defined by keeping the resource requirements of all the services (except victim service) as minimum required resources for the rest of the system. The remaining resources become the upper limit or resource containment limit for the victim web-service. The proposed method "Victim Service Containment" is detailed in Algorithm 6. We describe our proposed solution using Figure 6.5. We show a rectangle to represent all the available resources on the virtual machine. In Figure 6.5(i), while the status is "no attack presence", there are ample resources available on the virtual machine.

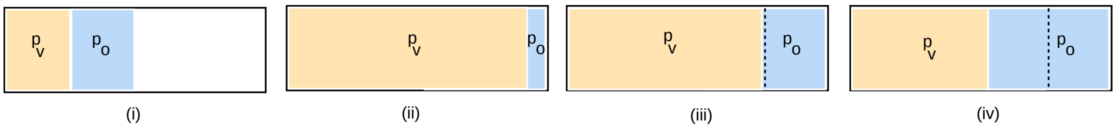


FIGURE 6.5: Victim Service Containment (V is victim service and O is all other services), (i) Normal operation, (ii) During DDoS attack, (iii) After "Victim Service Containment" during DDoS attack, and (iv) No containment for other services during normal operations

Algorithm 6: Victim Service Containment

```
Victim Service Containment;
Data: Total available resources $Resources(I_j) =< C_j, M_j, D_j, B_j, O_j >$
Victim Service $p_v$
Result: Resource limit, $L$ for victim service ($p_v$)
initialization;
Show $L = Resources(I_j) - ResourcesNoAttack(I_j)$;
end
```

During a DDoS attack, the resource usage reaches its maximum for one or more resources and the victim's service occupies or uses most of the resources and hence the other services do not get the necessary resources in a timely manner. Once the resource containment method is applied, we define a maximum limit of resource usage for the victim's service which it will not be allowed to exceed (dashed line in Figure 6.5(iii)). It is important to note that this limit has no effects on the other services and they may still use the available resources (Figure 6.5(iv)).

For our DDoS attack experiments, we evaluated resource requirements for CPU, memory, disk and network bandwidth. Similarly, the proposed algorithm 6, can be extended to evaluate the resource requirements for other fine grain resources. The most important phase of the algorithm is to identify the resource requirements of all the services prior to a real DDoS attack. Server consolidation strategies in cloud computing provide optimization algorithms to allocate resource to VMs. Our major concern is to provide performance isolation and resource availability inside a VM, therefore, we calculate the resource requirements (limit $L = Resources(I_j) - ResourcesNoAttack(I_j)$ in the algorithm) taking these aspects into consideration. Each resource can be controlled and shared using multiple dimensions. For example, a CPU can be distributed using "shares" and also as dedicated CPUs. Similarly a disk can be divided from the perspective of disk size based distribution as well as maximum read and write speed shares. We take the specific example of the VM under consideration in the attack experiment shown in Section 6.3. We also show the specific resource limits, we enforced.

I. CPU: The CPU requirements of the complete operating system are around 5%-20% for each of the four CPUs while there is no attack. This sums to around 20%-80% out of the total 400% covering all the four processors which equates to one processor out of four processors for other services. Therefore, to have a stronger isolation, we give a limit of 3 processors to the victim's service so that one processor is always available for all the other services.

II. Memory: The memory limit for the victim's service is defined by keeping the minimum memory usage by the operating system, when there is no attack. We did not include the requirements of victim web-service into consideration while calculating the memory requirements of the complete system. The total VM memory is 8GB. The memory usage while there is no attack is 1.6 - 2GB which gives an overall memory limit of 6GB.

III. Disk: We controlled the disk read and write speeds for the victim's service. The maximum read and write speeds of the storage disks using IO benchmarks is around 75-80 MBps. Looking at the memory occupancy (75%), we decided to limit the speeds to 60 MBps which is 75% of the maximum read and write speed of 80 MBps.

IV. Network: Network traffic can only be prioritized using the control group facilities. Therefore, we gave top priority to all the other traffic and the next priority to victim's service' traffic.

Applying the proposed algorithm also changes the the overall service performance. Once the limit L is defined by the victim service containment algorithm, the resource limit can be enforced in the following two ways.

A. Limiting from the available resources: We used this approach to limit the available resources in the resource limits we calculated previously. One disadvantage of this approach is that it may limit the performance of the victim's service owing to the resource constraints which restricts to lower amounts of resources as compared to the earlier limit (or no limit) for all the resources.

B. Scale to get more resources: In this case, the resources required for the whole operating system including all the services for attack free cases can be added over and above the VM resources. The victim's web service can be limited to the resources which were already available. The additionally scaled resources (on top) can be used by the other services as a minimum resource guarantee. This will provide both the advantage of victim's service performance as well as resource containment. The additional cost of added resources is a factor while opting for this approach. We also showcase the effects of scaled resources in Section 6.6.1.

Note the proposed solution as initially described only applies when one service is attacked. We address the situation where other services are also attacked in Section 6.6.2.

6.6 Results and Discussion

We evaluate the "Victim Service Containment" algorithm by observing various metrics related to the victim service and the co-located services during the DDoS attack. We employed the experimental setup as shown in Figure 6.3. Our major focus is towards DDoS mitigation service and SSH service. To achieve this, we first launch the attacks against target VM, while there is no victim service containment in place. We use three different services (500KB, 1MB and 2MB image size for conversion) to see these effects. After this, we employed our proposed approach (algorithm 6) to apply the resource containment on victim's web-service. We show the collective results of these attack experiments in Table 6.3 and Table 6.4.

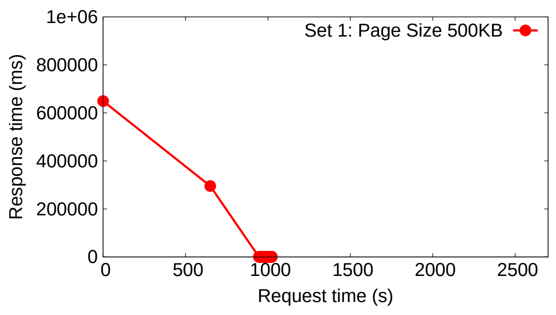
Target Web Service Type	**Attack Reporting Time** (No VSC)	**Attack Reporting Time** (With VSC)	**Service Down Time** (No VSC)	**Service Down Time** (With VSC)	**SSH Down Time** (No VSC)	**SSH Down Time** (With VSC)
500KB	37s	40s	943s	346s	0s	0s
1MB	39s	38s	945s	375s	517s	0s
2MB	2040s	41s	2383s	2801s	1959s	0s

Table 6.3: Various performance metrics: before and after the "Victim Service Containment (VSC)"

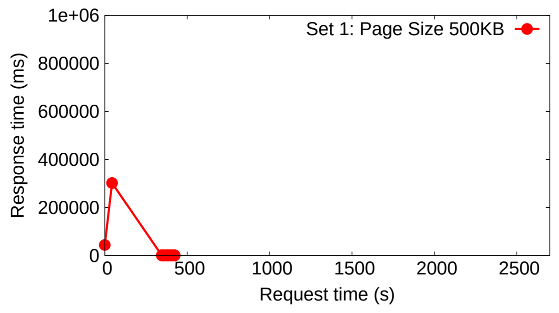
We show the victim's service response time and SSH response time in Figure 6.6 and Figure 6.7 respectively. We discuss the results and observations with respect to important evaluation metrics in the following:

Target Service Type	**Max. Response Time** No VSC	**Max. Response Time** With VSC	**Min. Response Time** No VSC	**Min. Response Time** With VSC	**Avg. Response Time** No VSC	**Avg. Response Time** With VSC	**VM Interface** No VSC	**VM Interface** With VSC
500KB	0.363s	0.267s	0.127s	0.131s	0.148s	0.152s	Not available	Available
1MB	186.8s	0.284s	0.129s	0.130s	1.226s	0.149s	Not available	Available
2MB	765.2s	0.298s	0.129s	0.134s	4s	0.154s	Not available	Available

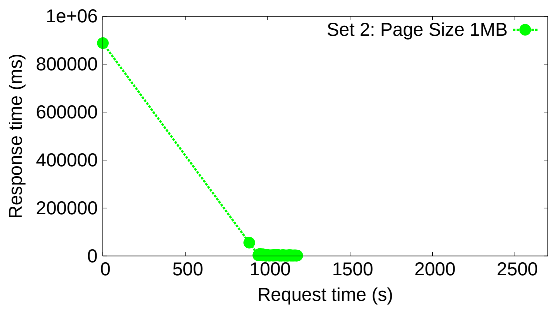
TABLE 6.4: SSH performance: before and after the "Victim Service Containment (VSC)"



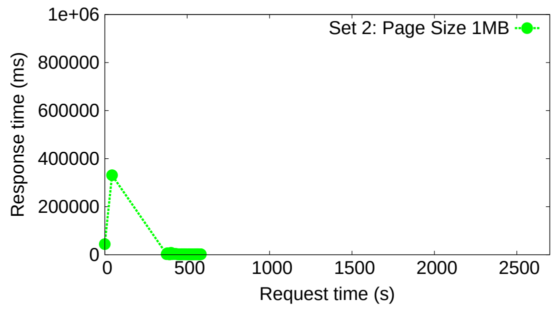
(A) Service response time during attack



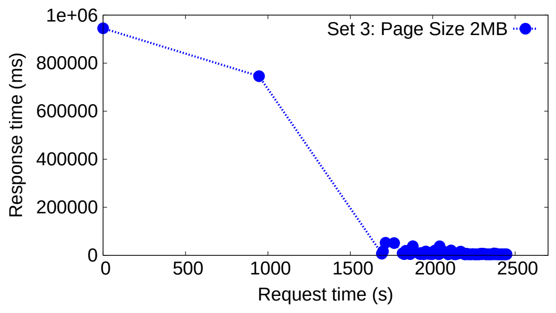
(B) Service response time during attack (After Containment)



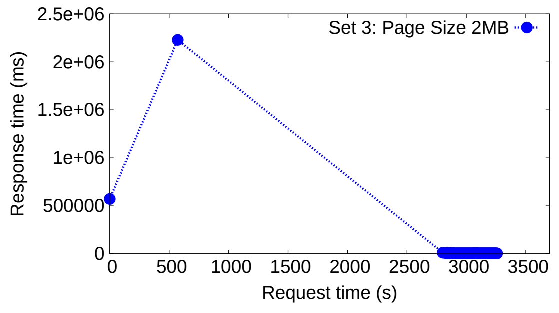
(C) Service response time during attack



(D) Service response time during attack (After Containment)



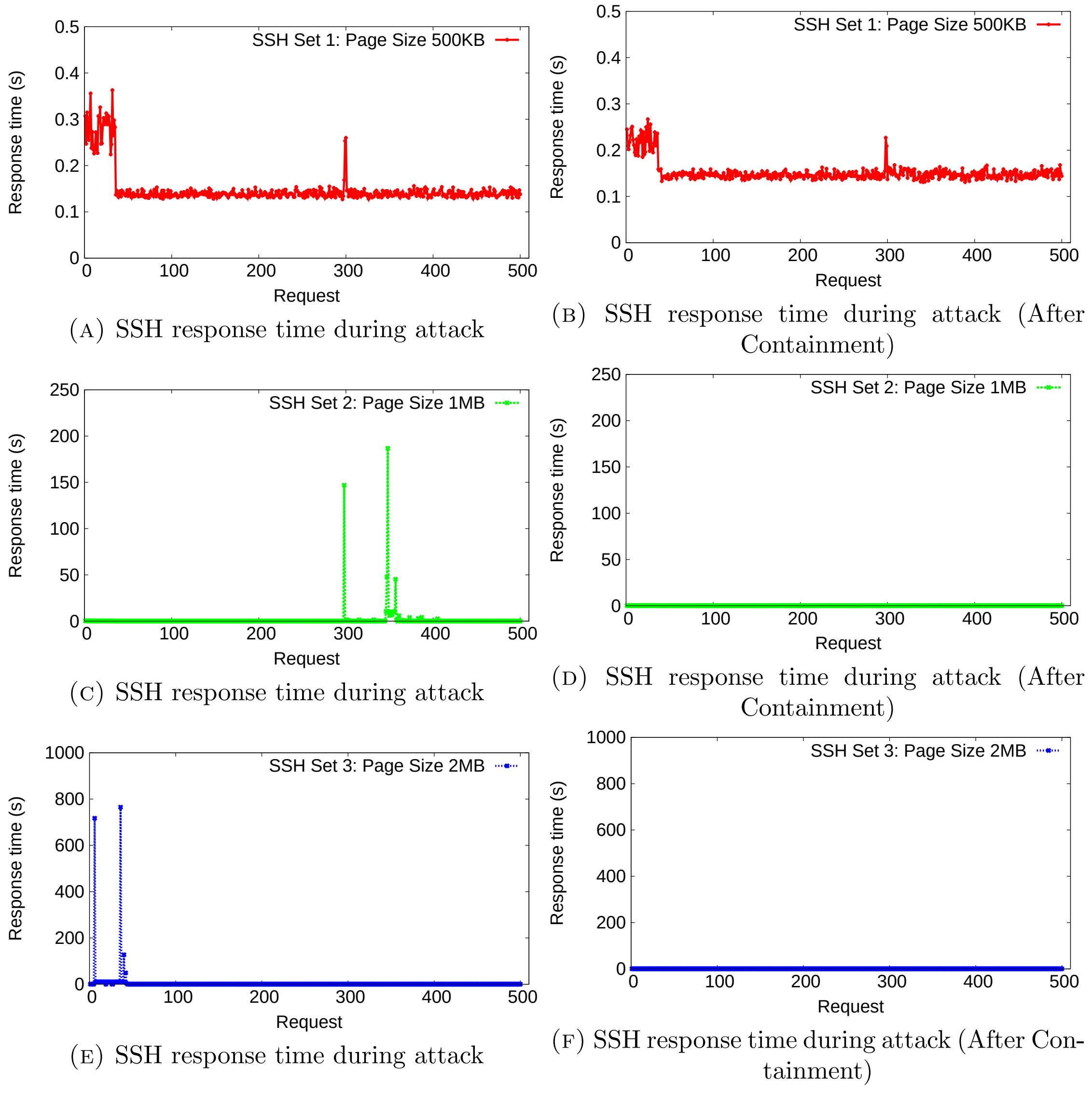
(E) Service response time during attack



(F) Service response time during attack (After Containment)

FIGURE 6.6: Service response time during attack before and after containment

A. DDoS mitigation service performance: To see the performance of DDoS mitigation service, we monitor the attack reporting time and victim service availability time. Attack reporting time is when the DDoS mitigation service detects the attacker IPs and adds them to the firewall. After the containment, the attack



(A) SSH response time during attack

(B) SSH response time during attack (After Containment)

(C) SSH response time during attack

(D) SSH response time during attack (After Containment)

(E) SSH response time during attack

(F) SSH response time during attack (After Containment)

FIGURE 6.7: SSH response time during attack before and after containment

reporting time in these attack cases should be small as the detection criteria is immediately fulfilled by the number of connection requests launched by the attacker. However, the attack reporting time is quite high (2040s) for page size 2MB which gives a clear indication of resource contention faced by DDoS mitigation service (Figure 6.6e and Figure 6.6f). As the amount of resource contention generated by victim's service, serving image conversion for 500KB and 1MB is not significant, the attack reporting time for these services is only around 40s. The victim's service availability time is greatly minimized in the cases of victim service serving 500KB and 1MB conversions (figures 6.6a, 6.6b, 6.6c and 6.6d). However, for 2MB image size, the victim's web service is unavailable for an additional period as compared to the case of "no service containment". This is due to the fact that the resource limits imposed on victim's service provides limited amount of resources compared to the resources available in the case of “no service containment”. This issue can be resolved by selecting for approach “Scale to get more resources”. We also performed the attack experiments using the scaled resources and detailed the results in next subsection 6.6.1. On the other hand, the victim’s service becomes available after the attack reporting time by terminating the attack connections in all the three cases. Additionally, the victim service availability time for the cases of 500KB and 1MB image conversion services has reduced from 943s to just 345s after the service containment.

B. SSH service performance: Our proposed solution solved the issues with the availability of co-located services during the attack period. We show the availability aspect by the performance of SSH service “login-logout” response time in Figure 6.7. We show the average, minimum and maximum response times observed for the SSH service in Table 6.4. In the extreme resource contention case (2MB image conversion service), the SSH service was not available for more than 1959s out of the total downtime of 2383s. Similarly, for the case of 1MB service, the SSH unavailability time is around 517s. On the other hand, the SSH unavailability time for attack case on 500KB image service is 0s which means that the SSH services was available throughout the attack period. However, the SSH service has been affected in having peaks in the response time during start of the attacks (Figure 6.7a). These peaks were resolved up to a certain extent after service containment (Figure 6.7b).

C. Availability of VM interface: We also use the responsiveness of the VM interface on the server side (XenCenter Interface) as an important criteria during the attack period. As the attack mitigation often requires remote as well as manual intervention to the victim’s computer to see the real cause of the situation. We showed the availability/responsiveness of the VM interface during various attack cases in Table 6.4. We observed that the availability issue of VM interface is completely solved by the resource limits posed by the victim’s service containment.

6.6.1 Scale to get more resources

We observed that the resource containment approach adversely affects a much important performance metric (service downtime) in the case of 2MB image size (Figure 6.6f and Table 6.3). In this section, we show the results of additional experiment to see the effects of “scaled resources”. These scaled resources are equivalent to the resources deducted or removed from the victim service during the resource containment mechanism. Now, we assign five vCPUs to the victim VM with a containment limit of four vCPUs to the victim web service process groups. Additionally, we assign 10 GB of main memory to the victim VM with a containment limit of 8GB to the victim web service process groups. We remove the containment limits on disk read and write speeds and maintain the network priority as configured in the containment. By adding the resources on top, we equated the victim VM's resources to what it already had before the victim service containment. We show the attack experiment results in Figure 6.8. We also present a detailed comparison of various performance metrics in all the three configurations in Table 6.5. These three configuration cases are related the service performance before the containment, with the containment, and with the scaled resources and containment. We observe that the additional resources help in resolving the adverse effect on the overall service downtime. The service downtime is decreased from 2801s to 1920s that is even better than the service downtime before (2383s). The other performance metrics are quite similar to the cases of victim service containment alone (Table 6.5).

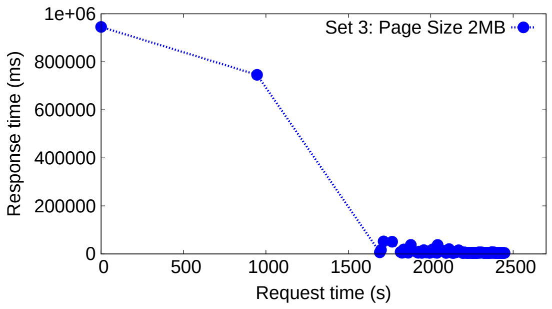
Performance Metric	**Before**	**With VSC**	**With VSC and Scaled Resources**
Attack Reporting Time	2040s	41s	41s
Service Downtime	2383s	2801s	1920s
SSH Downtime	1959s	0s	0s
Maximum Response Time (SSH)	765.2s	0.298s	0.368s
Minimum Response Time (SSH)	0.129s	0.134s	0.147s
Average Response Time (SSH)	4s	0.154s	0.164s
VM Interface Availability	Not available	Available	Available

Table 6.5: Various performance metrics before and after VSC and scaled resources

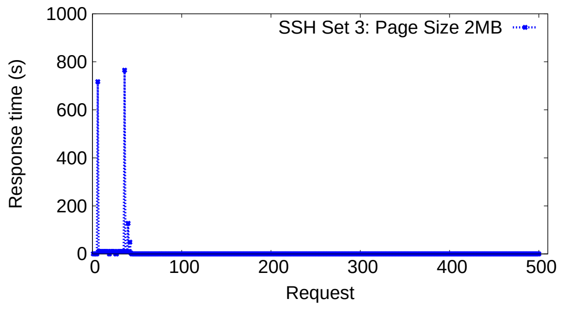
6.6.2 Other important aspects

Now, we discuss various aspects related to the DDoS mitigation process in connection with the proposed service containment algorithm.

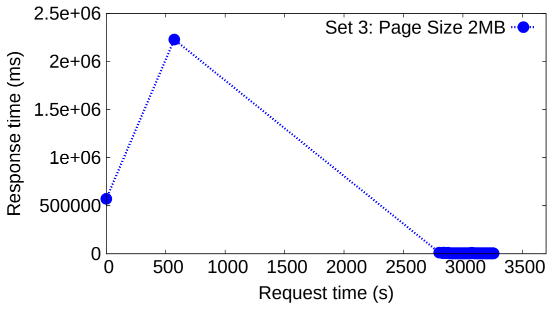
I. Disadvantages of resource limits: The resource limits we put during containment, might not be the most appropriate limit for the different applications. The proposed algorithm is flexible to use with different requirements for different applications, however, the resource limit will be independent of the incoming attack features. One obvious disadvantage of the containment is on the performance of the victim's service, if the resource limit is applied on the available resources. However, this can be easily resolved by having additional resources on top of the VM resources already available with additional cost (also discussed in Section6scaledresources). On the other hand, all the other co-located services have



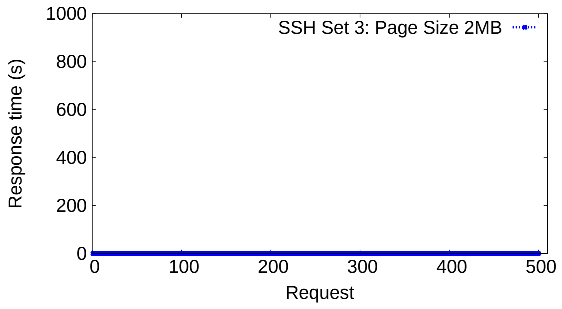
(A) Service response time before



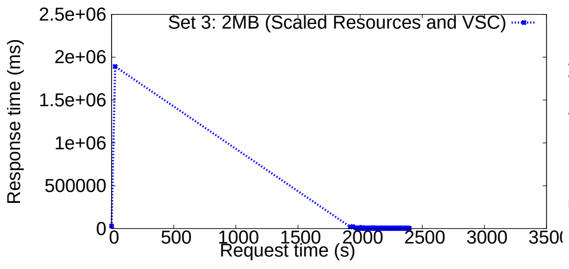
(B) SSH response time before



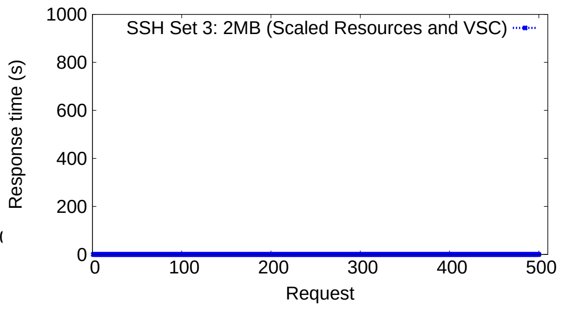
(C) Service response time after containment



(D) SSH response time after containment



(E) Service response time after containment and scaled resources



(F) SSH response time after containment and scaled resources

FIGURE 6.8: Service/SSH response time during attack before and after scaled resources

no resource limits as they can use any amount of the available resources. The resource limits for other process groups can also be decided, if there are incidents of resource contention by them. Previous overhead studies show that there is a small overhead of memory containment limits owing to the fine grain monitoring [151].

II. Cloud level collateral damages: The collateral damages shown in Chapter 3 are due to the resource race and contention among co-hosted VMs. Most of the resource contentions are owing to the shared or virtualized nature of the resources. We showed that the elimination or minimization of resource contention at operating system level, may also help in achieving positive results outside the VM. Additionally, we provided the attack cases, where the resource contention is extreme for CPU, memory and disk resources. In these attack cases, resolving internal collateral damages, may lead to the minimization or elimination of the collateral damages among the co-hosted VMs.

III. Victim separation: The collateral damages due to the DDoS attacks are also extended owing to the victim's service running multiple VM instances of the same service on different servers using load balancing. Victim's VM separation may give the desired results of minimization of collateral damages owing to the removal of resource contention among VMs. However, in the proposed approach, the effects are minimized to the victim's VM itself. We extended the victim separation and minimized the effects to the other services.

IV. Fixed Infrastructure Protection: The proposed solution also applies to services not running on virtualized or cloud platforms. Resource containment can be directly applied to the host operating system to perform quick DDoS mitigation and ascertain the availability of all the other services.

V. Proactive vs. Reactive Implementation: In the proposed victim's service containment approach, we used the proactive implementation, where the resource containment limits are calculated and imposed on the victim's service. Hence, the method "Victim Service Containment" is enabled all the time. Triggering the "Victim Service Containment" approach just during the attack (reactive implementation) can also be used if the resource limits are already calculated for other services. However, applying containment limits while the resource in consideration (such as memory) is already at the maximum utilization is difficult to achieve as it requires the usable memory for the victim process to be shrunk. Therefore, we used a proactive approach in our approach.

VI. Smoke-screening attacks: There are multiple recent incidents of "smoke-screening attacks", where DDoS attacks are used to bring down the organizational resources. Behind the scene, other severe cyber attacks are launched on the victim services. We relate the problem of "internal collateral damages" with these attacks, as in the presence of DDoS attacks almost all the resources are used up to their maximum limits and create a contention for services. We have firewall and other access control primitives such as SELinux to stop and report these attacks. However, these services may not be available and may not give the desired and timely outcome owing to the heavy resource contention. The proposed resource containment methods take care of these attacks, by ascertaining the availability of services, which have capabilities to stop/detect the real attack underneath the DDoS attack.

VII. Contention formed by other services: In the proposed approach, we aimed at the resource containment of the victim's service. However, there might be instances where the resource contention is developed by the services other than the victim's service. This might be due to the faulty code, resources or a real attack on the other co-located service. There are also recent attack incidents, where DDoS attacks are targeted at DDoS detection tools [152] to slowdown the mitigation. In this case, multiple resource containment groups may be formed to help in this situation by anticipating the possible target services.

6.7 Conclusion

In this chapter, we show a novel resource contention phenomenon, "internal collateral damage" which is observed in the presence of DDoS attacks on the cloud services. Our real-time attack experiments demonstrate that these attacks may severely affect the timely and expected outcome of the critical services such as the DDoS mitigation service. We observed the attack effects on co-located SSH and VM interface services which are mandatory tools for last resort manual access for control. Both the services also become unavailable owing to the resource contention generated by the victim service. The external auto-scaling methods working from outside the target VM, would not help in the mitigation due to the resource contention problems at the process level.

We model the DDoS mitigation activity as an OS level resource management problem. We show that the victim service is responsible for the resource contention due to its heavy resource usage for one or more resources such as CPU, memory, disk, and bandwidth including other application resources. To overcome these issues, we propose a novel victim resource containment method by which the rest of OS services including critical services such as DDoS mitigation service and remote log-in service remain available irrespective of the presence of any severity of DDoS attacks. We develop an illustrative example of "Victim Service Containment algorithm" and address the service unavailability problem. Experimental attack results demonstrated the efficacy of our proposed solution leading to the overall improvement of attack reporting and service availability. This novel attack characterization also opens up multiple issues related to DDoS mitigation, resource management activity and availability of other services.

In the next chapter, we detail our novel resource management approach which sacrifices the contended resources as soon as the possibility of the attack is detected. We make the resources available with the help of minimizing the metric resource utilization factor. These resources help in absorbing, detecting and mitigating the attack as quickly as possible and recover the service back to operation.

Chapter 7

Scale Inside-out: Rapid DDoS Attack Absorption and Mitigation

7.1 Introduction

The distributed denial of service (DDoS) attacks in cloud computing requires quick absorption of incoming attack data. DDoS defense methods primarily work online and perform a threshold based anomaly detection in the incoming traffic which follows learning based on past attacks/traffic. Attack data absorption is an important milestone before triggering any analytical threat intelligence activity [153] [154] [155]. Many of the recent solutions are based on the dynamic resource scaling which advocate to use quick deployment of enormous resources to absorb the attack as quickly as possible [5] [139]. The resource scaling comes with an additional cost which may prove to be a huge disruptive cost in the cases of longer, sophisticated, and repetitive attacks. These solutions achieve the quick attack absorption by minimizing the attack surface. We discussed in Chapter 5 and Chapter 6) that the effects of the attack depend upon the target application resource usage, as the attacks may result into an extreme resource contention which may affect the attack absorption and subsequent threat mitigation adversely.

In this chapter, we address an important and related problem, whether the resource scaling during attack, always result in rapid DDoS mitigation? For this purpose, we conduct real-time DDoS attack experiments to study the attack absorption and attack mitigation for various target services in the presence of dynamic cloud resource scaling. We found that the activities such as attack absorption which provide timely attack data input to attack analytics, are adversely compromised by the heavy resource usage generated by the attack. We show that the operating system level local resource contention, if reduced during attacks, can expedite the overall attack mitigation. The attack mitigation would otherwise not be completed by the dynamic scaling of resources alone. We conceived a novel relation which terms "resource utilization factor" for each incoming request as the major component in forming the resource contention. To overcome these issues, we propose a new "Scale Inside-out" approach which during attacks, reduces the resource utilization factor to a minimal value for quick absorption of the attack. The proposed approach sacrifices victim service resources and provides those resources to mitigation service in addition to other co-located services to ensure resource availability during the attack. We also provide a formal model of attack absorption and outline its major requirements in this chapter.

The rest of the chapter is organized as follows. Section 7.2 shows the relationship between cyber threat intelligence and attack absorption in attack mitigation. We show our initial attack experiments to prepare the requirements of effective mitigation based on the important observations in the attack outcomes in Section 7.3. We give a formal model of resource utilization factor and its role in the attack mitigation process in Section 7.4. In Section 7.5, we describe the proposed approach and its detailed working. We give experimental evaluation in Section 7.6. Section 7.6 also provides a detailed discussion on the outcomes and their applicability. Finally, we conclude our work with a light on the future work in Section 7.7.

7.2 Cyber Threat Intelligence in the Cloud

In Figure 7.1, we present a generic information flow diagram showing the DDoS attack source, participants, target infrastructure and cyber threat intelligence to combat any possible attacks. A command and control server coordinates these DDoS attacks using a large number of possible attack sources in the form of bot run zombie computers. The attack sources may include phones, servers, desktops, PDAs, cloud VMs or more recently other Internet of Things devices (IoT). After the emergence of cloud computing, there are attack incidents which are even utilizing cloud resources (BotClouds) to originate such attacks as a service [39]. Cyber threat intelligence related to the class of DDoS attacks may intend to provide a complete security framework which involves implementing methods to prevent, detect and recover from DDoS attacks.

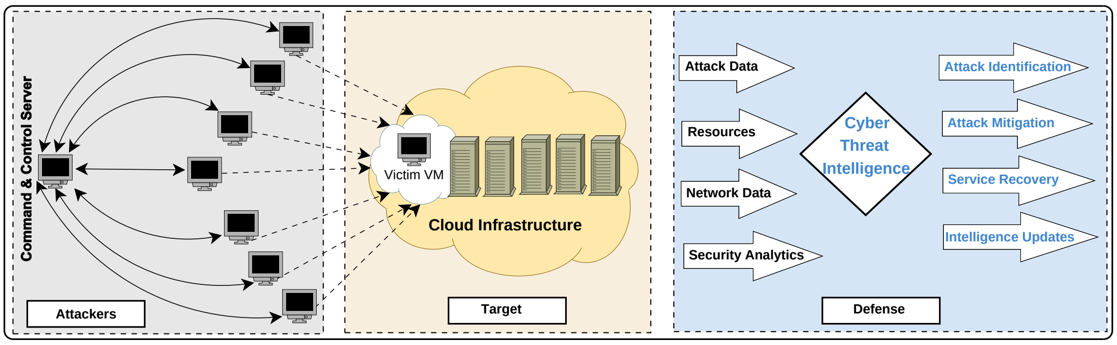


Figure 7.1: Cyber Threat Intelligence and the Role of Attack Absorption

7.2.1 Input to Threat Intelligence

The detection methods are mostly functions used to identify patterns in the incoming traffic and logs [153]. Attack data collected from the victim server file system/logs may be supplemented by other collected data from other parts of the overall network. Additional important inputs to the threat intelligence include security analytics in the form of security knowledge based updates about the newer threats and methods. Additional on-demand compute, storage, and bandwidth resources might also be provided as inputs to perform the attack analytics, associated computation, maintain the availability, fasten the attack absorption, and recovery. In many of the large scale and sophisticated attacks, human intervention is also required to achieve the goals of threat intelligence. Attack absorption is one of the most important aspect for our work and a key ingredient to the success of timely and efficient attack mitigation and subsequent service availability.

7.2.2 Threat Intelligence Outcomes

The outcomes of the threat intelligence activity include the identified attacks, the attack sources, and the attack vulnerabilities [153]. Attack mitigation methods will block the attack sources to send any incoming traffic further and drop any already established attack connections. Service recovery to gain availability may require some time due to a range of attack mitigation activities. If the attacks remain present for longer duration or remain undetected due to the stealthy and sophisticated nature, it may even delay the recovery further. We show in Figure 7.1 about additional outcomes in the form of new and incremental "threat knowledge" which updates the existing knowledge base of threat intelligence. To extend our discussion towards the major findings of our work, we expand the discussion of the generic cyber threat intelligence towards real attack analytics. We detail the real attack experimental study with critical aspects of attack absorption, mitigation and recovery in Section 7.4.

7.3 Real-time Attacks: Critical Aspects

The cyber threat intelligence flow illustrated in Figure 7.1 shows attacks origin, generation, attack target, and attack mitigation. In this section, we describe the attack dynamics from its origin to detection and mitigation with the help of attack experiments. To conduct these experiments, we use an attacking cloud and a victim cloud running VMs as attack and defense infrastructure respectively. We show various configuration settings related to the attack experiments in Table 7.1 and the experimental setup in Figure 7.2. The victim service is a dynamic file conversion website which converts an image from one format to another format on each service request. In this set of experiments, we are using `.jpeg` images of size 2MB which gets converted to a `.png` image. By using an image conversion application as a test site we could approximate a real web-server behavior. The test web application uses most of the server resources such as CPU, memory, disk and the bandwidth.

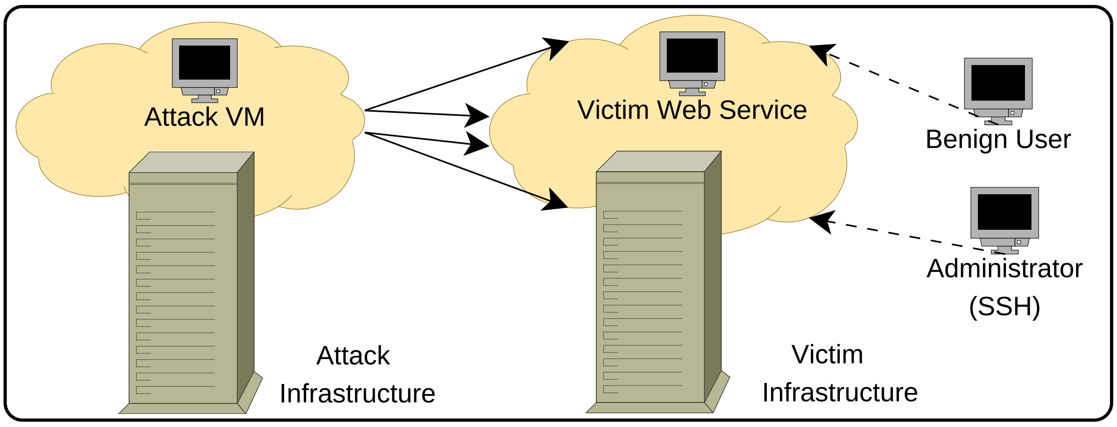


FIGURE 7.2: Experimental Setup

We extend the test service to its variants in later experiments in Section 7.6 to showcase different aspects of cloud computing resource scaling during attacks. We send attack traffic with 500 concurrent requests from a single attack VM. We do not send the attack traffic from multiple sources to achieve a deterministic attack detection time over multiple repetitions of the experiments. In case of multiple sources, the attack detection time varies depending upon the connection share of each attack source in the total number of established connections at victim server end. Moreover, the main focus of our experiments is on the attack absorption

	Resource	**Configuration Settings**
Attack Settings	Physical server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 4 processors (12 cores each) total memory 96GB
	Service	ApacheBench2
	Configuration	2 vCPUs and 4 GB
	Traffic rate	500 concurrent requests (Total 5000 requests)
Victim Settings	Physical server	Dell PowerEdge R630 Intel(R) Xeon(R) CPU E5-2670 v3, 8 processors (4 cores each) total memory 96GB
	Service	Image conversion web service
	Configuration	4 vCPUs and 8 GB (Set 1) 8 vCPUs and 16 GB (Set 2) 16 vCPUs and 32 GB (Set 3)
Other Settings	Benign Service	2 vCPUs and 4 GB
	Traffic rate	1 concurrent web request (Total 100 requests) SSH request response cycles (Total 500 requests)
	OS	Ubuntu 14.04
	Network Speed	1Gbps
	Hypervisor	XenServer 6.5

Table 7.1: Attack Experiment Details

and subsequent attack mitigation at the victim side processes. The launch of the attack is motivated by [41], where authors made important comments about the major portion of a large number of attack traces (around 12000 traces). More than 38% of uniform random attack events and 46% of all the attack events had an estimated rate of 500 requests per second or more. We see similar attack experiments in many other contributions such as [92].

To demonstrate the service availability behavior of the victim service under attack, we use a host to send benign traffic by sending one concurrent request for a total of 100 requests. We are also interested in the behavior and performance of other critical services which are co-hosted with victim service on the same VM. For this purpose, we design a test to assess the availability of secure shell (ssh) by sending a ssh session request, establishment, and session close cycle. SSH test represents the communication medium by which the VM/victim service owner connects to the service. We scheduled the attack experiments in a manner such that the attack traffic, benign traffic, and ssh requests are launched at the same time. We also aim at analyzing the performance dynamics of the attack absorption activity and other important attack mitigation and availability concerns while the victim VM gets more resources. For this purpose, in Set 1 experiments, we provide resources equivalent to an EC2 "C4" instance [150]. In Set 2, we expand these resources to see the attack dynamics with resources twice of C4 resources. We term Set 2 resources equal to two C4 instances. We expand these services to as high as four times as that of the basic C4 resources in Set 3 which are equivalent to four C4 instances. Instead of scaling the resources dynamically while the attack is present, we do three separate attack experiments to observe the effects on service downtime, attack detection time, and performance of other services. We study the attack mitigation behavior by placing a generic DDoS mitigation mechanism in place.

We use "`DDoS-Deflate`" which is one of the popular open source DDoS mitigation tools that detects the attacks sources on the basis of a threshold on maximum number of connections [87] by a source. The detected attack sources are then suitably blocked by a firewall such as `iptables` [142].

While performing the attack experiments, we keep a close eye on attack absorption behavior of the victim VM and the DDoS mitigation service. We achieve this by keeping track of various metrics such as attack detection time, attack reporting time, downtime of victim service, service response time, number of attack requests served before the attack detection, downtime of ssh service, and response time behavior. One more behavior for which we are particularly interested to see the attack absorption behavior is the number of established connections during the attack period. Connection establishment depends on the server resources availability and the duration of connection cycle which consists of connection-establishment and connection-close cycle. We will detail about this important factor while discussing the attack dynamics in Section 7.4.

7.3.1 Observations

Figure 7.3a shows the behavior of victim service response time for the requests sent by the benign sender. As soon as the attack starts, service reaches the downtime and the response times become very high (Table 7.2). In the experimental setting, we changed the behavior of benign sender in such a manner that it waits for the longer responses to come without getting timed out. The response time behavior is having a large downtime and huge peaks in responses in Set 1. Set 2 with

Resources	Time of Attack Detection (s)	Time of Attack Reporting (s)	Downtime of Victim Service (s)	Time to complete the requests (s)	Maximum Response Time (s)	Minimum Response Time (s)	Average Response Time
Set 1	Unknown	2481	2481	2911.34	1507.68	4.053	29.11
Set 2	Unknown	931	931	1367.19	593.371	4.065	13.67
Set 3	41.58	41.58	971	1384.13	956.895	4.071	13.8413

Table 7.2: Performance of DDoS mitigation service during attacks

Resources	Downtime of ssh service (s)	Time to complete ssh requests (s)	Maximum Response Time (ssh) (s)	Minimum Response Time (ssh) (s)	Average Response Time (ssh) (s)
Set 1	2269.46	2373.15	706.4	0.15	4.7463
Set 2	864.2	940.657	254.88	0.147	1.88131
Set 3	0	83.309	0.475	0.145	0.166618

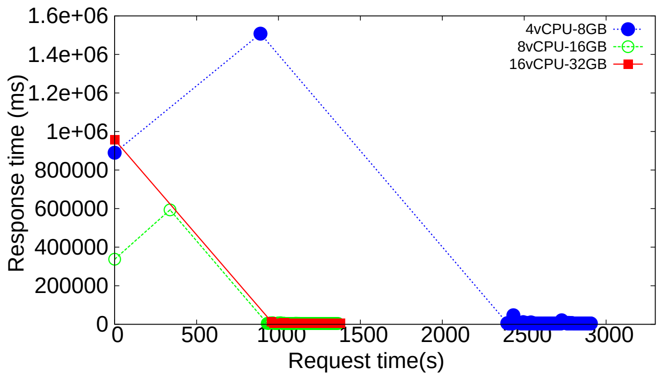
Table 7.3: Performance of SSH request-response behavior during attacks

increased resources settles the downtime from a huge 2481s to 931s which is quite similar in the case of Set 3. An important factor to consider related to the attack mitigation performance is attack reporting time. Set 1 and Set 2 victim VM is not responsive for the whole downtime and the reporting of attack detection time is unknown.

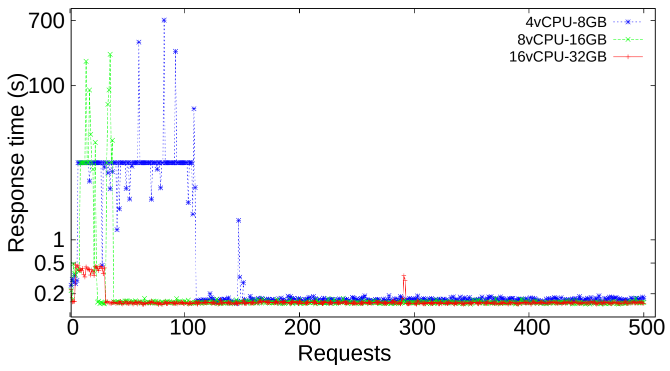
We configured the DDoS mitigation service to notify the attack detection in an attack log file once the attackers are identified from the traffic. However, the huge resource usage during the attack did not allow the access to VM interface and the reporting time remains unknown till all the attack effects are over. DDoS mitigation service achieved the attack reporting in the case of Set 3 resources.

The behavior of other co-hosted services (inside the same victim VM) as demonstrated by Figure 7.3b and various performance metrics in Table 7.3. The SSH service though not the real direct target of the DDoS attack, faces downtime and performance penalties in the later part of the web-service downtime. SSH service downtime and response time see a decremental trend from Set 1 to Set 3.

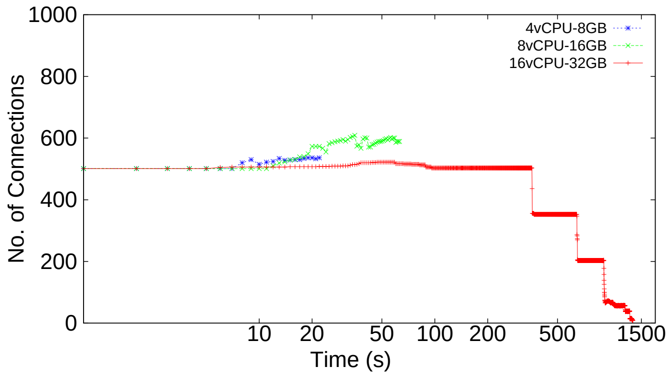
Set 3 having the highest resources (four times the Set 1 resources) sees no downtime of the SSH service, though there are response times which are as high as thrice of the minimum response time. In Figure 7.3c, we show the total number of connections at the victim service end since the start of the attack. Number of connections remains around 500-600 for all the three sets. However, for the Set 1 and the Set 2, the number of connections are unknown as the victim VM became non-responsive and values were not retrieved after 40s-50s. Set 3 stays at$\sim$500 connections for most of the duration and then a step-wise decrease is observed till all the attack effects are over. We made the following observations during attack experiments.



(A) Web request-response



(B) SSH request-response



(C) Number of connections

FIGURE 7.3: Attack effects with varying resources

1. Resource Contention during attack: Non-responsive VM, unknown attack detection time, unknown number of connections, and huge peaks in SSH request-response cycles are the number of observations which are the result of a heavy resource contention formed by the DDoS attack. In addition to just service denial during the downtime, these effects transform the DDoS attack in a “extreme DDoS” attack (also discussed in Chapter 5). Resource contentions occur due to heavy resource usage by victim service and no timely resource availability to other important processes at the operating system level. Resource contention and the absence of performance isolation are important issues while designing operating system and virtual machine monitors [48, 156]. We argue that resource contention should be minimized as this is an important effect of DDoS attacks and also a major contributor to the other three observations.

2. Downtime Post-attack Detection: The attack detection time is unknown in the cases of Set 1 and Set 2, due to the heavy resource contention. Though in Set 3, the attack detection time is around 41s. Still in Set 3, the service downtime is 971s. After detecting the attack, the DDoS mitigation service, with the help of the firewall, drops all the incoming attack connections from the identified attackers. In addition to this, the DDoS mitigation services tries to remove all the connections which are established by the attackers. Connection removal is achieved by resetting the connections by utilities such as `tcpkill`. Removal of these connections, also takes up a considerable amount of time due to heavy resource usage, as all these connections are already having requests which are currently processed by the victim service. In order to minimize the overall downtime, the attack cooling down period should also be minimized. The attack cooling down period is the time elapsed after the attack detection time to bring the service availability back.

3. Number of Connections: In all the three set of experiments, we observe the stability of number of connections to 500-600, which is the major reason behind the high attack cooling down period. We also increased the environment variables such as maximum number of connections and port-ranges at the server end, still the number of connections remains stable. Number of connections decide the overall attack surface on the victim and should be maximized to absorb the attack traffic as soon as it arrives [5].

4. Collateral Effects on Other Services: We see adverse performance impact on the SSH service in Figure 7.3b. SSH service is very light resource usage utility as compared to the co-hosted dynamic website. Still, the SSH request-response cycle is severely affected by the large resource contended downtime. We observe similar effects by non-responsive VM, unknown attack detection time, and unknown number of connections.

We will use the above observations in preparing the requirements for the proposed approach.

7.4 Formalizing Defense Requirements

We formally define important requirements keeping the four important observations we made during the attack experiments in Section 7.3. For this purpose, we first define a typical DDoS Attack mitigation activity as a three stage process in Figure 7.4.

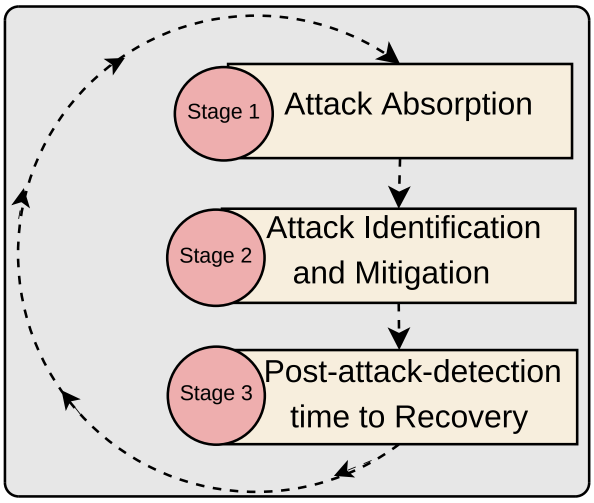

Stage 1
Attack Absorption
Stage 2
Attack Identification and Mitigation
Stage 3
Post-attack-detection time to Recovery

FIGURE 7.4: Stages of DDoS Attack Mitigation

Stage 1. Attack Absorption: In attack absorption, the mitigation method tries to absorb the incoming traffic as soon as it arrives on the network interface. Traffic input comes through the network connections and absorption gets affected if the victim service is not able to create and release connections quickly. This stage on attack absorption helps the next two stages by absorbing the attack traffic as soon as it arrives. Our proposed method (Section 7.5) makes sure that the absorption does not form the resource contention with the help of reduction of resource utilization factor. However, the most important reason for having timely attack absorption is to expedite the attack detection which may only succeed if the attack features are absorbed as soon as they appear.

Stage 2. Attack Identification and Mitigation: Attack absorption is an online activity which always runs to ensure the timely traffic assessment and threat analytics. Attack identification and mitigation (blocking and dropping the attack connections) are completely causally dependent on the attack absorption. Resource availability to DDoS mitigation service and other critical services must be ensured to have resource contention free attack mitigation. Each incoming request is the contributor to the resource usage, and a huge number of requests during the DDoS attack forms the contention.

Stage 3. Post-Attack Detection to Recovery: Quick service recovery is dependent on the timely attack absorption and mitigation. Quick removal of established attack connections is the key to minimize the time required to achieve service availability post-attack detection. The performance of the attack absorption stage is heavily dependent on the attack absorption delay.

Attack absorption delay (T_{absorb}) is the difference between the time at which the traffic flow reaches the victim interface ($T_{interface}$) and the time when it is actually within the reach of the DDoS mitigation service (T_{DMS}). We define attack absorption delay with the help of the following equation.

$$T_{absorb} = T_{DMS} - T_{interface}. \tag{7.1}$$

Victim VMs have certain basic resources such as vCPUs (C), memory (M), disk (D) and the bandwidth (B). We define the resource allocation to a VM as its capacity.

$$Resources(V_i) =< C_i, M_i, D_i, B_i > . \tag{7.2}$$

The actual behavioral capacity of a victim server running inside the VM is usually represented by the total number of simultaneous requests it can serve at the same time. The number of requests (r) which the service will be able to serve depend upon the resource requirements of each request. We define the resource requirements of each request r with the help of the resource utilization factor S_r, which is the total resource requirement to complete the request r. Processing a request requires access to few or all of the basic resources.

$$S_r =< C_r, M_r, D_r, B_r > . \tag{7.3}$$

We define the capacity of the victim VM by the following equation where the VM, V_i is able to serve N_{max} requests.

$$Capacity(V_i) = N_{max} * S_r. \tag{7.4}$$

We assume that the victim service has a defined set of requirements for achieving a minimum quality of service (QoS) in order to have timely and correct output. In order to extend the notion of resources, we also utilize the maximum number of connections (C_{max}) as an important resource of the victim service. We assume that we can only maximize the N_{max} value if we are able to establish as many connections (maximizing C_{max}) as possible (Equation 7.5).

$$N_{max} \propto C_{max} \tag{7.5}$$

It is also evident from Equation 7.4 that if we want to maximize the number of maximum requests (N_{max}) which can be served by the victim service, we need to either increase the $Capacity(V_i)$ or reduce the S_r. Following are the few possible requirements of a mitigation method by which the attack absorption can be expedited:

R1. T_{absorb} **Reduction:** During an attack, the number of connection requests coming to the victim service are huge. The time at which the traffic reaches the interface and transforms in the form of established connections is an important delay which is actually the attack absorption delay (T_{absorb}). The attack absorption delay will be higher for the cases where the mitigation methods filter the traffic on the basis of more fine grain features such as application level features such as movement among the web-pages, time spent on the page, and click-pattern. The attack absorption delay gets extended due to the serving of the initial requests (index pages of the websites) and then allowing the additional requests to enter the service. These requests might be waiting to enter the service queue or they may be even retransmissions. We argue that the delay T_{absorb} can be minimized if the connection life-cycle and the delay in providing the incoming traffic features to the DDoS mitigation service can be reduced.

R2. S_r **Reduction:** Resource utilization factor S_r, denotes the resource usage of individual request processing and responding with a suitable request outcome. S_r comprises of multiple activities at the service end for each incoming request. As per Equation 7.4, reduction in S_r will lead to increase in N_{max} and reduction in T_{absorb}. This relationship can also be given by:

$$N_{max} \propto \frac{1}{S_r} \propto \frac{1}{T_{absorb}} \tag{7.6}$$

Additionally, reduction in S_r may also lead to minimizing the resource contention effects, as all the three stages of mitigation process may run in parallel. Resource contention shows adverse effects on the parallel and similar activities [48].

R3. $Capacity(V_i)$ **Scaling:** The auto-scaling capabilities of cloud computing facilitate dynamic resource scaling on the go without any noticeable downtime in VM stop-resume. Capacity scaling can be achieved with the help of vertical or horizontal resource scaling as described in [19][110]. By increasing $Capacity(V_i)$, it is assumed that the attack absorption will be quick and the T_{absorb} will be minimized by increasing the N_{max} (in turn C_{max}). In next Section 7.5, we detail our approach by addressing these three requirements.

7.5 Our Proposal: Scale Inside-out

We now address the attack mitigation requirements detailed in Section 7.4 by describing the novel solution elements incorporated in the proposed approach. The main contribution of our work lies in providing quick attack mitigation by combating the four important attack observations made in the attack experiments in Section 7.3. We address the problem of reduction in attack absorption delay (T_{absorb}) by reducing the resource utilization factor, S_r. In the client server architecture of web-services such as the one which we used in our experiments, there are number of activities related to each request. Receiving a connection request, reading the request, processing the request, preparing a reply, sending a response, and closing the connection are few essential activities which a server-end normally performs during a request-response cycle. Processing a request requires utilizing various resources such as CPU, memory, disk, and the bandwidth. We show all these activities as part of the resource utilization factor in Figure 7.5.

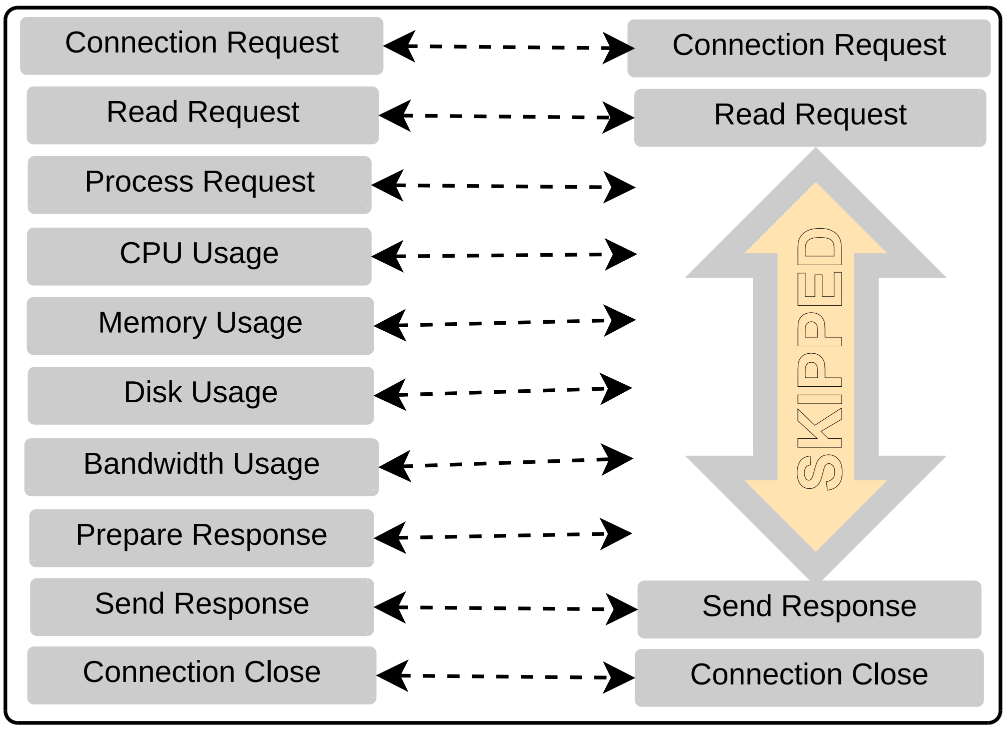


FIGURE 7.5: Scale Inside-out the "Resource Utilization Factor"

During DDoS attacks, huge number of requests to victim, create heavy resource usage. Resource utilization of these requests adds up to form the resource contention with all the basic resources such as CPU and memory. Though DDoS mitigation service in our experiments is a very low resource usage based service, still, the resource contention severely affects the overall attack mitigation process. Additionally, we saw in attack experiments in Section 7.3, scaling of victim service resources to four times does not solve the problem of attack downtime, specially the downtime due to the attack cooling down period. Meeting requirement R3 alone does not solve the local operating system level resource contention completely and comes with an additional cost. We calculated the cost of various web services in the form of an estimate of their resource utilization factor and summarized in Table 7.4. We show the amount of time taken and the total number of

Web Service Type	**Execution Time (real)**	**Execution Time (user)**	**Execution Time (system)**	**Total number of system calls executed**
2KB	0m0.036s	0m0.028s	0m0.004s	771
50KB	0m0.042s	0m0.036s	0m0.004s	811
100KB	0m0.155s	0m0.144s	0m0.008s	945
500KB	0m0.792s	0m0.788s	0m0.000s	1827
1MB	0m1.937s	0m1.904s	0m0.028s	3227
2MB	0m4.078s	0m4.032s	0m0.040s	5974

Table 7.4: Resource Utilization Factor of different services

system calls executed by the victim web service using different image sizes. We can see that the utilization of various resources is reflected in the time taken to perform the request and it increases with the size of the image. We use this fact in our novel approach "Scale Inside-out", where in addition to capacity scaling, we argue to perform an internal application scaling to reduce the resource utilization factor during a DDoS attack. We term our approach as "Scale Inside-out" as it reduces the request processing (by reducing S_r) to increase N_{max} and C_{max}, which in turn minimizes, T_{absorb}.

To achieve this goal, we skip all the activities between "read request" and "send response" as shown in Figure 7.5. By doing the reduction in S_r during the attack period, we argue that the resource contention and all the subsequent problems like attack absorption delay, attack cooling down period, and collateral damages are resolved. We show the working of this approach using algorithms 7 and 8.

We test the applicability of our method by a quick and dirty check in which we keep checking response time of the service and the total number of connections available on the victim service. Figure 7.6 shows the flow of the proposed approach. We also

Algorithm 7: Scale Inside Out Algorithm

```
SCALE-INSIDE-OUT;
Data: $T_{response}$= Response time,
$T_{timeout}$=Request timeout,
$C_{now}$= No. of connections,
$C_{attack}$ = Minimum connections during an attack,
$S_{attack}$ = Resource Utilization Factor during attack,
$S_r$ = Resource Utilization Factor without an attack,
$V$ = Victim service;
Result: Attack Mitigated
Start;
while ($T_{response} < T_{timeout}$ && $C_{now} <= C_{attack}$) do
    Nothing
end
ScaleInsideOut($V$, $S_{attack}$);
while ($T_{response} >= T_{timeout}$ && ($C_{now} >= C_{attack}$) do
    Nothing;
end
ScaleInsideOut($V$, $S_r$);
Show "Attack Mitigated";
end
```

Algorithm 8: Function ScaleInsideOut()

```
ScaleInsideOut();
Data: Web service $W$
Resource utilization factor = $U$,
Result: Updated Resource Utilization Factor
initialization;
Apply $U$ to web service $W$;
end
```

show the flow of the DDoS mitigation activity which progresses in parallel with the “Scale Inside-out” approach. Using this figure, we show that the proposed approach is working as a support framework to provide immediate resource help to the victim to combat the DDoS attacks.

To demonstrate the applicability, we use thresholds $T_{timeout}$ and C_{attack} which best describe the attack situation by having a timeout value and the minimum number of connections during an attack. Whenever the response time becomes higher than the acceptable response time and the number of connections (C_{now}) become more than or equal to C_{attack}, the “Scale Inside-out” approach changes the resource utilization factor of the victim web-service from S_r to S_{attack}. After “Scale Inside-out” reduces the S_r to S_{attack}, both attack as well as genuine requests get the responses with reduced resource utilization factor S_{attack}. We observe the resource

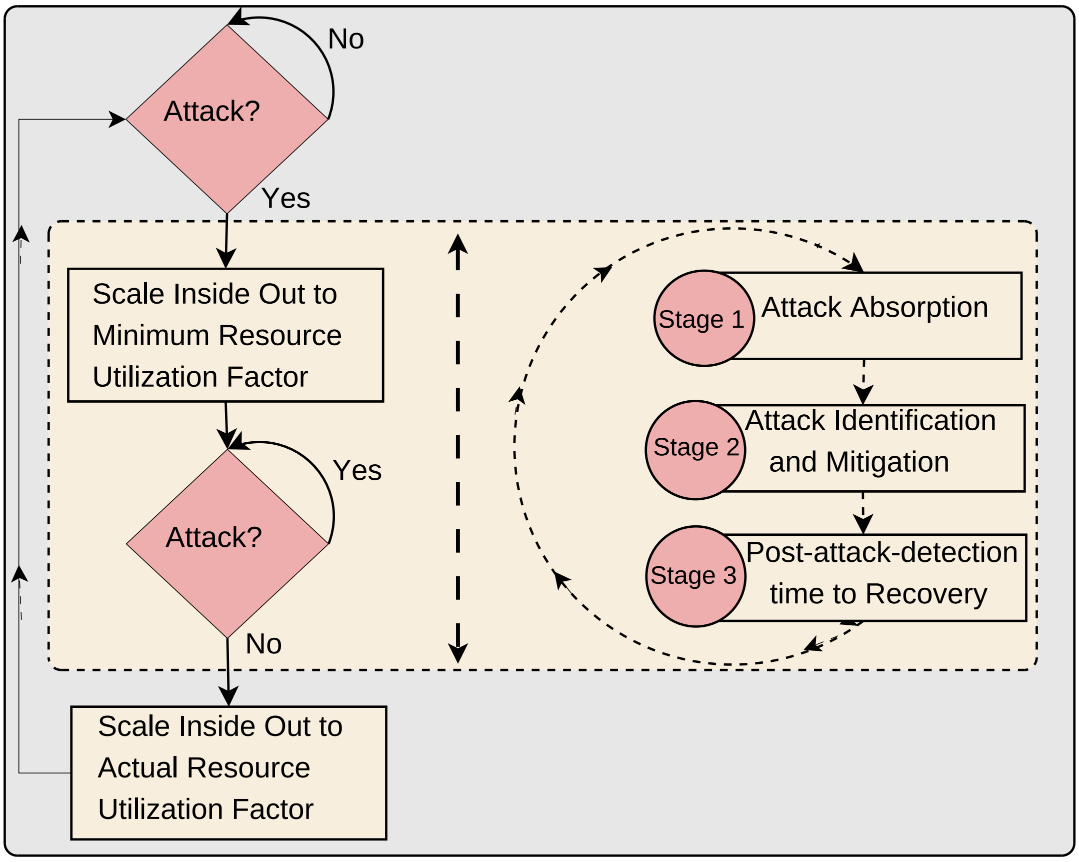

Attack?
No
Yes
Scale Inside Out to Minimum Resource Utilization Factor
Attack?
Yes
No
Scale Inside Out to Actual Resource Utilization Factor
Stage 1
Attack Absorption
Stage 2
Attack Identification and Mitigation
Stage 3
Post-attack-detection time to Recovery

FIGURE 7.6: Scale Inside-out flow running concurrently with the DDoS mitigation flow

utilization factor changing back to its original value once the service is restored back.

The major motivation behind "Scale Inside-out" approach is that during the attack downtime, it is fine to sacrifice the victim service resources to expedite all the three stages of attack mitigation if the mitigation method minimized the downtime and other effects. "Scale Inside-out" works as a supporting resource framework which tries to minimize the attack downtime, minimize the effects on the legitimate users with an aim of working with any kind of DDoS mitigation service.

7.6 Experimental Evaluation and Discussion

We implement the reduction in resource utilization factor by not performing the request processing activities in the server side scripts. There are only three parameters in the Algorithm 7 which are chosen as follows:
(1) Minimum connections during an attack, C_{attack} is set to 500 connections. This value of 500 connections is chosen based on the important seminal works in the

DDoS research area [41].
(2) Victim application response timeout, $T_{timeout}$ is set to 10 seconds.
(3) Reduced resource utilization factor, S_{attack} is set by automatically changing the server side scripts, which after the reduction do not perform the activities for each request during the attack period.
The major goal of this contribution is to showcase the effects of "extreme DDoS" attacks and the above values in 1) and 2) help (basically a quick and dirty check) in determining the presence of the attack. We argue that in case of "extreme DDoS" attacks, it is important to identify the presence of the attack. "Presence of the attack" is determined by the quick check in the Algorithm 7 and then based on the values of the two parameters (C_{attack} and $T_{timeout}$), the decision to apply the reduction in S_r is taken. Out of the three parameters used in the Algorithm 7, $T_{timeout}$ and C_{attack} are used in taking the decision about the "Scale Inside-out". They decide whether the "Scale Inside-out" should be performed or not. The third parameter S_{attack} is used specifically to support the generic DDoS mitigation mechanism (DDoS Deflate) which is a connection based detection mechanism. Varying the parameter S_r to other values of S_{attack} would mean that we perform some activities out of the activities such as "Read Request", "Process Request", "CPU, Memory, Disk, and Bandwidth Usage", "Prepare Response", and "Send Response". This may help some of the mitigation mechanisms which also see some of the application layer features while detecting the attack. However, it will not help the mitigation provided by our generic DDoS mitigation method (DDoS-Deflate) that does not use any application layer features while filtering the traffic. In order to analyze the efficacy of the "Scale Inside-out" (SIO) approach, we perform exhaustive real time attack experiments. We use various image sizes (2KB, 50KB, 100KB, 500KB, 1MB, and 2MB) and resources (Set 1, Set 2, and Set 3) and apply the "Scale Inside-out" approach to evaluate various metrics such as attack detection time, attack reporting time, service downtime, and the attack cooling down period. Similarly, we collect various metrics for other critical services to observe the contention and collateral damages.

The major motivation of using three sets of resources is to show the effects of capacity scaling on the mitigation performance. The experimental settings are similar to what we see in Table 7.1 and Figure 7.2 in Section 7.3. We show the detailed results on effects of "Scale Inside-out" on victim web service performance in Table 7.5. Table 7.6 shows the attack effects on SSH service.

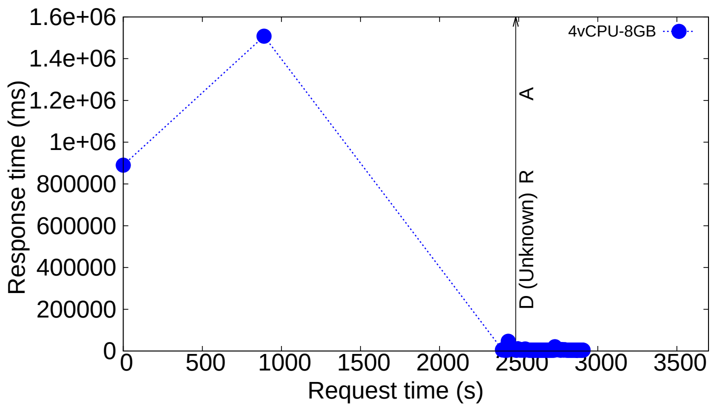
We will first describe the patterns in the results related to the victim service performance with the help of Figure 7.7. Figure 7.7a shows the response time

Resources	Type	Time of Attack Detection (no SIO)	Time of Attack Detection (with SIO)	Time of Attack Reporting (no SIO)	Time of Attack Reporting (with SIO)	Downtime of Victim Service (no SIO)	Downtime of Victim Service (with SIO)	Time to complete the requests (no SIO)	Time to complete the requests (with SIO)
Set 1	**2KB**	37.24	38.13	37.24	38.13	43	38	45.859	5.376
	50KB	41.25	39.17	41.25	39.17	645	38	647.4	5.621
	100KB	41.3	41.21	41.3	41.21	944	45	957	5.834
	500KB	38.28	42.23	38.28	42.23	945	38	1024.31	6.949
	1MB	39.35	41.24	39.35	41.24	875	40	1053.64	7.758
	2MB	Unknown	40.29	2481	40.29	2481	53	2911.34	14.139
Set 2	**2KB**	41.49	41.14	41.49	41.14	47	37	49.5	4.921
	50KB	41.44	41.14	41.44	41.14	643	44	645.87	4.882
	100KB	39.45	42.14	39.45	42.14	940	37	953.26	4.977
	500KB	41.55	38.14	41.55	38.14	946	39	1023.82	5.416
	1MB	40.36	42.14	40.36	42.14	947	40	1137.54	5.98
	2MB	Unknown	43.18	931	43.18	931	40	1367.19	6.081
Set 3	**2KB**	36.54	41.14	36.54	41.14	43	37	44.212	4.921
	50KB	36.54	41.14	36.54	41.14	638	44	640.711	4.882
	100KB	36.7	42.14	36.7	42.14	937	37	949.46	4.977
	500KB	38.63	38.14	38.63	38.14	940	39	1017.6	5.416
	1MB	38.53	42.14	38.53	42.14	943	40	1131.63	5.98
	2MB	41.58	43.18	41.58	43.18	971	40	1384.13	6.081

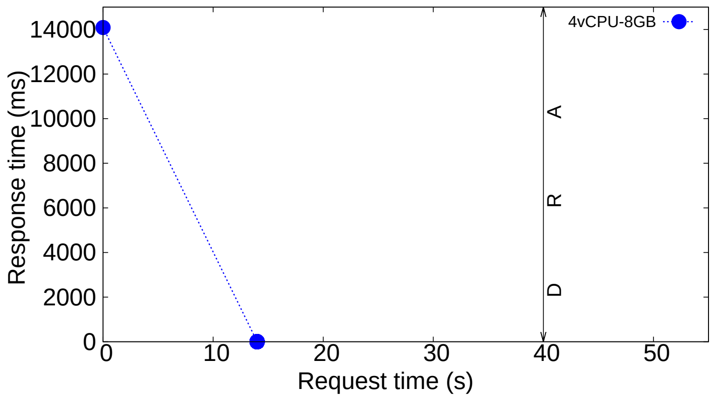
Table 7.5: DDoS mitigation Performance during attacks (without and with "Scale Inside-out" (all in seconds))

behavior to the benign user while we do not employ the proposed approach for Set 1 resources for 2MB service. In this case, we observe that the attack detection time ('D' as shown after the vertical arrow on the graph) is not known till the attack effects are completely over. Attack reporting time ('R') and attack detection time ('D') are only observed after the victim service achieves the service availability state ('A'). The heavy resource contention also resulted in the unavailability of VM interface as well as longer peaks in the response time behavior. On the other hand after applying the "Scale Inside-out" approach to help the attack mitigation expedition, the web service response is similar to the Figure 7.7b. The three important metrics, the attack detection time (D), the attack reporting time (R) and the service availability (A), are heavily reduced to around 40s. Additionally, the attack cooling down period is ∼0s in the cases where "Scale Inside-out" is used. The attack cooling down period is the time taken by the victim service to become available after the attack is detected. We represent this factor in Figure 7.7b where both "Attack Detection Time (D)" and "Service Availability Time (A)" are shown at 40 seconds. The attack cooling down period is essentially the difference between these two events. On the other hand, the attack cooling down period was quite high in the cases without using "Scale Inside-out" (also shown in Figure 7.7a).

After the reduction of the resource utilization factor from S_r to S_{attack}, all the benign requests are given a response which is not useful for them. Therefore, the response time behavior as shown in the Table 7.6 shows a huge reduction in various metrics. However, the reduction in S_r helps in achieving a huge reduction in the service downtime. Except the 2KB service, which has the least S_r, all the other services have a service downtime ranging between∼600s-2400s.



(A) Web request response (without Scale Inside-out)



(B) Web request response (with Scale Inside-out)

FIGURE 7.7: Web response behavior (before and after Scale Inside-out) for Set 1 resources and 2MB service, the vertical arrow shows attack detection time ('D'), attack reporting time ('R') and service availability time ('A')

Another important observation regarding the service downtime in the cases when "Scale Inside-out" is used, is related to the achieved stability in the service downtime to∼40s. Coming to the impact of "Scale Inside-out" on the attack absorption, we show the results in the form of various graphs in Figure 7.8. With a huge reduction in the resource utilization factor from S_r to S_{attack}, the number of connections N_{max} reaches to a value as high as∼50000 connections to absorb the attack as soon as possible.

Victim web services where the S_r is already very low (2KB-100KB) show number of established connections ranging between∼2000-20000 while we do not use "Scale Inside-out". However, after employing our approach, the rise in number of connections also helps in achieving important goals of reducing the overall downtime and other related effects. We show the effects of our proposed approach on the collateral damages to other critical services in Table 7.7. For better comprehension, we show the performance of SSH test in the form of SSH response times

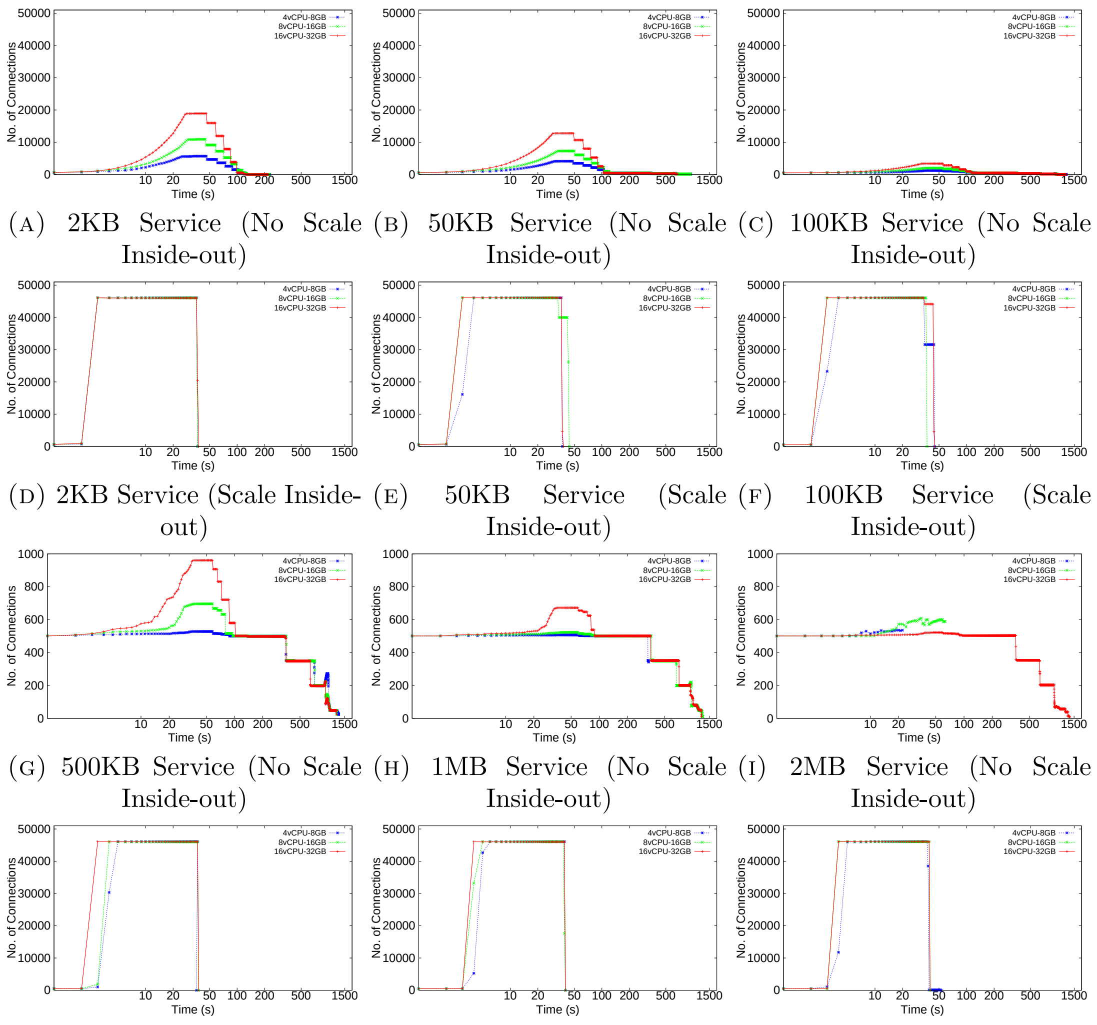
Resources	Type	Maximum Response Time (no SIO)	Maximum Response Time (with SIO)	Minimum Response Time (no SIO)	Minimum Response Time (with SIO)	Average Response Time (no SIO)	Average Response Time (with SIO)
Set 1	**2KB**	6.425	2.261	0.024	0.019	0.46	0.05376
	50KB	602.6	2.165	0.031	0.021	6.47	0.05621
	100KB	597.98	1.926	0.134	0.022	9.57	0.05834
	500KB	942.127	1.508	0.769	0.018	10.24	0.06949
	1MB	822.437	7.732	1.907	0.018	10.53	0.07758
	2MB	1507.68	14.088	4.053	0.022	29.11	0.14139
Set 2	**2KB**	6.561	2.194	0.024	0.019	0.49	0.04921
	50KB	602.732	2.058	0.031	0.021	6.45	0.04882
	100KB	603.4	2.636	0.134	0.014	9.53	0.04977
	500KB	642.263	1.57	0.776	0.014	10.2382	0.05416
	1MB	944.613	2.346	1.919	0.019	11.37	0.0598
	2MB	593.371	3.017	4.065	0.018	13.67	0.06081
Set 3	**2KB**	6.369	2.194	0.024	0.019	0.44	0.04921
	50KB	602.4	2.058	0.03	0.021	6.4	0.04882
	100KB	900.869	2.636	0.134	0.014	9.4946	0.04977
	500KB	902.034	1.57	0.776	0.014	10.176	0.05416
	1MB	641.035	2.346	1.91	0.019	11.3163	0.0598
	2MB	956.895	3.017	4.071	0.018	13.8413	0.06081

TABLE 7.6: Performance of DDoS mitigation service during attacks (without and with "Scale Inside-out" (all in seconds))

by various graphs in Figure 7.9. Proposed approach removes the collateral effects that were formed by the victim service on SSH service. The huge peaks in the response times, specially in the cases of web services with a higher S_r reduce to acceptable values.

Additionally, it is also visible that the collateral effects on SSH service seems to disappear in case of Set 3 resources without employing the proposed approach. However, a close look on the maximum time taken in the request-response cycle in SSH is more than double in all the cases where we use the proposed approach. These results also prove that the capacity scaling alone may not solve all the collateral effects as the resource contention highly depends upon the application type and its resource utilization factor S_r. Therefore, even the Set 3 resources may become resource contended for heavier services or attacks. However, "Scale Inside-out" always shows the stable results for various metrics for critical co-located services. In addition to all these metrics, we also show an additional metric which is quite important to understand the attack absorption dynamics. We show the number of attack requests served by the victim service under attack, before the attack is actually detected in Table 7.8. This metric is important from two perspectives, first, the cost of outgoing bandwidth spend on responding to these attack requests will be huge if the attack is not detected in time.

Second, is the severity of resource contention, which remains high if the victim service keeps serving the attack requests without the mitigation service timely identifying them. The attack requests served before detection, are heavily reduced in the cases when the "Scale Inside-out" is used. In the cases of higher S_r, the number of attack requests served before attack detection looks quite similar to



(A) 2KB Service (No Scale Inside-out)
(B) 50KB Service (No Scale Inside-out)
(C) 100KB Service (No Scale Inside-out)
(D) 2KB Service (Scale Inside-out)
(E) 50KB Service (Scale Inside-out)
(F) 100KB Service (Scale Inside-out)
(G) 500KB Service (No Scale Inside-out)
(H) 1MB Service (No Scale Inside-out)
(I) 2MB Service (No Scale Inside-out)
(J) 500KB Service (Scale Inside-out)
(K) 1MB Service (Scale Inside-out)
(L) 2MB Service (Scale Inside-out)

FIGURE 7.8: Number of connections during attack (without and with Scale Inside-out)

after "Scale Inside-out" is applied. This is because a certain minimum time is required to complete the quick and dirty check by the proposed approach.

On the other hand, the capacity scaling (Set 2 and Set 3) shows negative impact here as the scaled resources serve more and more requests before getting a detection trigger. After discussing the detailed results of the evaluative experiments, we will now discuss various aspects of the proposed approach. Following are various important usage aspects related to the "Scale Inside-out" approach:

1. Attack Downtime: Many of the recent DDoS attacks show the attack downtime ranging between few days to weeks and the unavailability causes enormous losses [12]. During the downtime, the legitimate customers will not get served, hence, we utilize the idea of sacrificing the victim service resources and make the resources available for the DDoS mitigation service and other critical services.

2. Variable S_r: The real web services may not always have a fixed resource

Resources	Type	ssh down time (no SIO)	ssh down time (SIO)	Time to complete ssh test (no SIO)	Time to complete ssh test (SIO)	Max. ssh Time (no SIO)	Max. ssh Time (SIO)	Min. ssh Time (no SIO)	Min. ssh Time (SIO)	Average ssh Time (no SIO)	Average ssh Time (SIO)
Set 1	2KB	0	0	81.616	77.846	0.341	0.231	0.144	0.144	0.163	0.155692
	50KB	0	0	82.016	81.045	0.332	0.283	0.145	0.146	0.164	0.16209
	100KB	0	0	85.139	80.685	0.374	0.316	0.145	0.146	0.17	0.16137
	500KB	0	0	84.779	81.127	0.409	0.489	0.146	0.146	0.169	0.162254
	1MB	445.75	0	549.26	80.722	120.07	0.321	0.145	0.146	1.09853	0.161444
	2MB	2269	0	2373.15	81.928	706.4	0.346	0.15	0.146	4.7463	0.163856
Set 2	2KB	0	0	84.54	78.389	0.458	0.223	0.148	0.147	0.169	0.156778
	50KB	0	0	84.48	78.308	0.566	0.218	0.146	0.147	0.168	0.156616
	100KB	0	0	83.622	78.347	0.441	0.331	0.145	0.146	0.167244	0.156694
	500KB	0	0	84.262	78.977	0.474	0.47	0.146	0.146	0.168	0.157954
	1MB	0	0	83.536	78.581	0.431	0.334	0.145	0.147	0.167072	0.157162
	2MB	864.2	0	940.657	77.219	254.88	0.489	0.147	0.146	1.88131	0.154438
Set 3	2KB	0	0	81.1	78.389	0.419	0.223	0.145	0.147	0.162	0.156778
	50KB	0	0	82.221	78.308	0.476	0.218	0.145	0.147	0.164442	0.156616
	100KB	0	0	83.196	78.347	0.491	0.331	0.147	0.146	0.166392	0.156694
	500KB	0	0	83.394	78.977	0.506	0.47	0.147	0.146	0.166788	0.157954
	1MB	0	0	83.191	78.581	0.501	0.334	0.144	0.147	0.166382	0.157162
	2MB	0	0	83.309	77.219	0.475	0.489	0.145	0.146	0.166618	0.154438

TABLE 7.7: SSH request-response behavior during attack (without and with "Scale Inside-out" (all in seconds))

Resources	Type of the Victim Service					
	2KB	**50KB**	**100KB**	**500KB**	**1MB**	**2MB**
Set 1 (No SIO)	5581	4115	794	30	6	4
Set 1 (SIO)	323	179	44	13	13	8
Set 2 (No SIO)	10820	6916	1542	198	24	20
Set 2 (SIO)	515	313	109	58	13	8
Set 3 (No SIO)	18825	12778	2920	487	170	20
Set 3 (SIO)	793	415	80	19	13	12

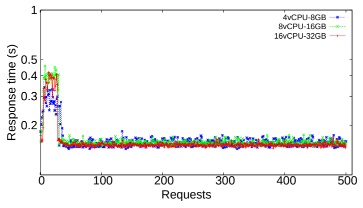
TABLE 7.8: Number of Attack Requests Served Before Attack Detection

utilization factor for all the requests (except in the cases of static services). In the experimentation, we change the S_r to S_{attack} in all the cases, however, variable S_r values for different resource users/dependent services may be used to ensure a minimum availability during the downtime.

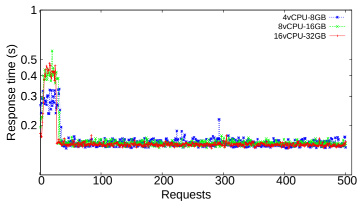
3. Effects on other DDoS mitigation methods: The proposed approach is independent of the DDoS mitigation method used as it aims at providing resources to the mitigation service. For most of the DDoS mitigation methods which work at the network or the application level, proposed approach will be helpful as it is able to absorb the attack data timely. However, application layer filtering approaches which are based on more fine-grain user inputs may require additional support.

4. Attack Absorption: Our major aim and contribution in this work is to expedite the attack absorption with minimizing the attack absorption delay (T_{absorb}). Methods requiring additional data in mitigation may require other mechanisms.

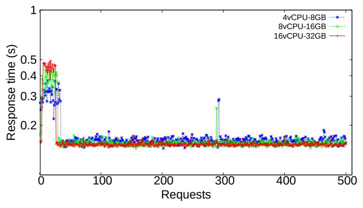
5. Attackers are not interested in response: One important argument in support of the reduction in S_r, is that the attackers being mostly bots do not have any interest in the service response. However, the responses with reduced S_r may trigger the bots to automatically increase/change the attack vectors (guessing that



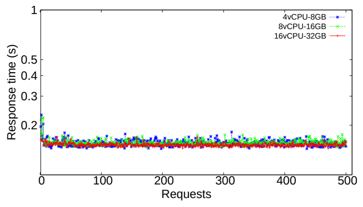
(A) 2KB Service (No Scale Inside-out)



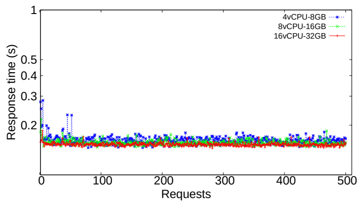
(B) 50KB Service (No Scale Inside-out)



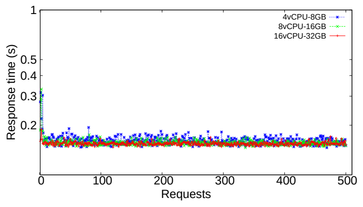
(C) 100KB Service (No Scale Inside-out)



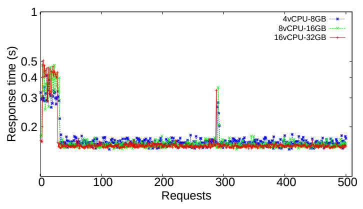
(D) 2KB Service (Scale Inside-out)



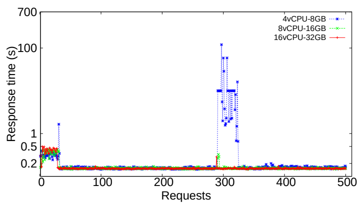
(E) 50KB Service (Scale Inside-out)



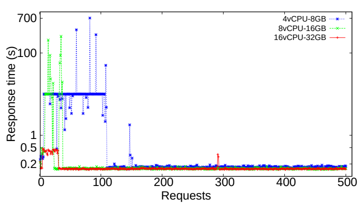
(F) 100KB Service (Scale Inside-out)



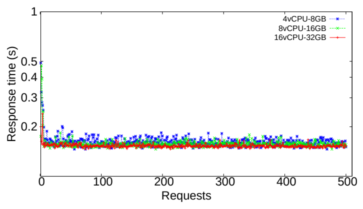
(G) 500KB Service (No Scale Inside-out)



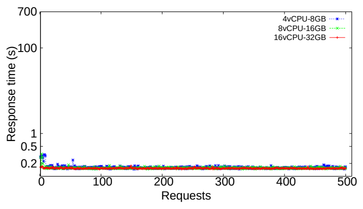
(H) 1MB Service (No Scale Inside-out)



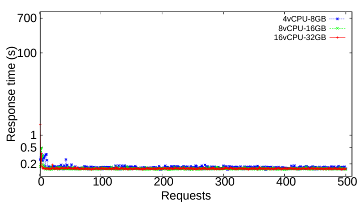
(I) 2MB Service (No Scale Inside-out)



(J) 500KB Service (Scale Inside-out)



(K) 1MB Service (Scale Inside-out)



(L) 2MB Service (Scale Inside-out)

FIGURE 7.9: SSH request-response behavior during attack (without and with Scale Inside-out)

the service is not down yet). This instigation may also help the mitigation method to get the upcoming attacks quickly for quick detection.

6. Legitimate users: At first instance it may appear that the service is giving unusable responses to the benign users. However, due to a heavy reduction in downtime, the legitimate users are served right after the attack detection.

7. Repeated/stealthy/sophisticated attacks: As discussed in point number 5, the attacks which are undetectable or requires additional efforts (at times manual efforts) should show reduced effects if the proposed approach is used as the S_{attack} will be minimum during the downtime. If an attack is not detected, it will have same impact on the service availability in both the cases (with "Scale Inside-out" and without "Scale Inside-out") and attack will remain successful. However, in the cases when "Scale Inside-out" is applied, the server will have better probability of detection with higher attack absorption without performing activities while serving the requests (reduced S_r). If the attacks are repeated during the downtime, the detection will be quick as the service is already using S_{attack}.

8. Sacrifice vs losses: In order to achieve the loss minimization and quick availability, we apply a forceful downtime using "Scale Inside-out". The sacrifice of service resources by reducing the S_r results in minimizing the losses due to the attack. The losses due to the DDoS attack without using our proposed approach are higher than the losses incurred due to the reduction from S_r to S_{attack} using our proposed approach. In the experimental results, directly visible losses include service downtime, denial of service to legitimate requests, resource scaling costs etc. There are indirect losses which include business losses and reputation losses, and are directly related and contribute to the service downtime. We provide a detailed discussion on direct and indirect losses due to the service downtime in Chapter 2.

9. Costs: Capacity scaling comes up with the additional cost and in case of repeated/stealthy/sophisticated attacks it may become quite high which may even question the sustainability of the victim organization. However, "Scale Inside-out" approach also works well without the scaling methods.

10. "Scale Inside-out" vs. Shutdown: Proposed approach may look like victim service/machine shutdown, as in the case of shutdown the victim service administrator anticipates that the attack will go away soon. However, without shutting the service, we collect all the attack data and use the resources to detect and mitigate the attack as quickly as possible.

11. "Scale Inside-out" for Fixed infrastructure: Our proposed approach does not advocate or deny the use of resource scaling in the attack mitigation. Proposed approach works well with both the auto-scaling enabled approaches and fixed resource servers.

7.7 Conclusion

In this chapter, we show that during massive and extreme DDoS attacks, the victim service resources get stressed heavily. The legitimate traffic coming to the victim service is not serviced during the downtime due to these attacks. In addition, collateral damages due to the heavy resource contention delays the overall mitigation process. The traffic flow in these attacks is really huge that cannot be timely absorbed by the victim machine having the limited resources. Most of the existing DDoS solutions use the dynamic scaling capabilities of cloud computing to scale service instances to absorb the attack quickly.

The resource scaling capabilities provided by the profound and on-demand resources provide a great help in mitigating the attacks quickly to restore back the services. Resource scaling allows quick collection of attack features to expedite the attack detection and mitigation. The heavy resource contention created during the presence of the attack should be minimized so that the overall mitigation activity can function smoothly. In addition to the dynamic scaling of resources, we argue that a reorganization and in-resource scaling of critical bottleneck resources may fasten the overall mitigation process further. Most of the DDoS mitigation tools/frameworks rely on the incoming traffic features such as traffic rate or connections to evaluate and segregate the traffic.

We show that the mitigation activity could be expedited if the attack absorption delay, that is the time between the request arrival and request evaluation by the mitigation method can be minimized. To minimize the absorption delay, we reduce the "Resource Utilization" per request during an attack. To showcase the efficacy of our claims we also take the case of "number of connections" as a critical resource as it directly affects the overall time and efficiency of traffic data absorption during segregation process. To increase the number of connections, we propose to reconfigure the victim service to have a reduced resource utilization per request during the attack.

This essentially means that for each incoming request (during the attack), the server does not perform the requested activity which leads to the resource contention. The implementation hastens the incoming traffic arrival and evaluation since the reconfiguration results into the establishment of a huge number of connections which helps in absorbing the attack as quickly as possible. We perform real time attack experiments on cloud services to showcase the effectiveness of our scheme. The proposed scheme also works well with the services having resource/budget limitations. We also believe that our proposed technique opens up a new direction of "in-resource" scaling by which the attack data collection lead to early mitigation.

Chapter 8

Conclusion and Future Work

In this thesis, we present our novel contributions and results related to the characterization and mitigation of DDoS attacks in cloud computing.

First, we propose a comprehensive taxonomy of contributions related to DDoS defense mechanisms in cloud computing. We also present detailed attack and threat models of DDoS attack in cloud computing to understand various attack features and threats. We believe that this survey would provide a directional guidance towards requirements of DDoS defense mechanisms and a guideline towards unified and effective solutions. We found that there are only few contributions in the recent past which target cloud-specific features like resource allocation, on-demand resources, BotCloud detection, and network reconfiguration.

Later, we conduct extensive attack experiments to see the real attack effects of DDoS attack on various stakeholders of cloud computing infrastructure. In addition to the obvious targets such as a victim server or a network, we showed that almost all the components and stakeholders of a cloud architecture are affected by a DDoS attack. We developed a system model of cloud computing resource allocation to help in understanding the role of auto-scaling algorithms during a DDoS attack. Furthermore, we identified important features which multiply the impact of DDoS attack in virtualized infrastructure clouds. These features include auto-scaling, migration, multi-tenancy, resource race, performance interference, and isolation. We show that multiple unrelated and non-targeted VMs, servers, and users are also affected by DDoS attacks in the cloud. We also discussed other related issues such as attack effect spread, migrant selection, overhead of cloning, and victim migration during DDoS attacks.

We make four important contributions which help in minimizing the DDoS attack effects and expedite the overall DDoS mitigation process.

In the first contribution, we showed through preliminary experiments that the fake resource utilization and subsequent resource addition due to the attack, results into EDoS attacks, which ultimately converges to DDoS. This has motivated us to design our mitigation system, DARAC, which is a DDoS aware resource allocation method for cloud. There are three important aspects of DARAC which make it a quick and effective DDoS mitigation solution for cloud computing. The attacker and benign traffic segregation, intelligent auto-scaling for real users and quality services to benign users during the attack, are three significant contributions of our work. The novelty of our work lies in the wise auto-scaling strategy with server capacity planning for the services. We show detailed experimental results with wide coverage of attacker and benign traffic sets. Our evaluation results show significance of DARAC in the detection and blocking of attacker traffic, stopping EDoS culmination, and capacity planning based DDoS aware resource allocation.

In the second contribution, we conduct real attack instances on cloud services to critically see the overall mitigation activity at fine grain level, i.e., at the resource level. DDoS attacks target victim resources and turn into "extreme DDoS" attacks for services with high resource utilization per request. We characterize these extreme DDoS attacks and observe that the resource contention created by the victim service under an attack may also compromise the DDoS mitigation service itself. Additionally, in these extreme DDoS attacks, availability after the attack detection is also affected due to a longer attack cooling down period. To circumvent these problems, we provide a framework to support the overall mitigation activity desirable from any mitigation tool. Our supporting framework puts efforts to provide enough resources such that the mitigation mechanism can perform its task even in the presence of extreme attacks. For this purpose, we perform attack experiments and highlight the need for methods to minimize the downtime, post-attack detection. We propose a novel supporting framework based on "service resizing" which employs processor affinity-based service resizing and TCP tuning techniques during the attack period to fulfill two important aims, (i) providing required resources to the DDoS mitigation activity and (ii) minimizing overall downtime.

In the third contribution, we model the DDoS mitigation activity as an OS level resource management problem. We show that the victim service is responsible for the resource contention due to its heavy resource usage for one or more resources like CPU, memory, disk, and bandwidth including other application resources. To overcome these issues, we propose a novel victim resource containment approach by which the rest of OS services including critical services like DDoS mitigation service and remote log-in service remain available irrespective of the presence of any severity of DDoS attacks. We develop an illustrative example of "Victim Service Containment" algorithm and address the service unavailability problem. Our experimental attack results demonstrated the efficacy of our proposed solution leading to the overall improvement of attack reporting and service availability.

Lastly, in the fourth contribution, we show that the attack mitigation activity could be expedited if the attack absorption delay that is the time between the request arrival and request evaluation by the mitigation method can be minimized. To minimize the absorption delay, we reduce the "resource utilization" per request during an attack using our proposed "Scale Inside-out" approach. To showcase the efficacy of our claims we also take the case of "number of connections" as a critical resource as it directly affects the overall time and efficiency of traffic data absorption during segregation process. To increase the number of connections, we propose to reconfigure the victim service to have a reduced resource utilization per request. We perform real time attack experiments on cloud services to showcase the effectiveness of our scheme. The proposed scheme also works well with the services having resource/budget limitations. We also believe that our proposed technique opens up a new direction of "in-resource" scaling by which the attack data collection lead to early mitigation.

Our novel contributions in terms of attack characterization and mitigation solutions open up multiple research issues related to DDoS mitigation, attack repetition, OS level resource management, and service resilience. Along with the novelty of contributions emanating from our schemes, directions for future research emerge to visualize the inter-service relationship on an operating system.

Additionally, the behavior of other unrelated services, resource scaling, and attack repetition are few other issues which remain challenging and relevant. Isolation and separation of victim services concerning network resources, is also an open problem. Smoke-screening, malware spread within cloud, security attack inclusive SLA design, and attack surface minimization and absorption are some of the other important aspects of the attacks that are need be investigated while looking at DDoS attacks in cloud computing.

Publications

1. G Somani, MS Gaur, D Sanghi, M Conti, and M Rajarajan, “Scale Inside-out: Rapid Cloud DDoS Attack Absorption and Mitigation”, IEEE Transactions on Dependable and Secure Computing, 2017 (Accepted, In press).
2. G Somani, MS Gaur, D Sanghi, M Conti, M Rajarajan, and R Buyya, “Combating DDoS Attacks in Cloud: Current Requirements and Future Trends”, (IEEE) Cloud Computing, vol. 4, no. 1, pp. 22-32, Jan.-Feb., 2017.
3. G Somani, MS Gaur, D Sanghi and M Conti, “DDoS attacks in Cloud Computing: Collateral Damage to Non-targets”, (Elsevier) Computer Networks, vol. 109(2), 2016.
4. G Somani, MS Gaur, D Sanghi, M Conti, and R Buyya, “Resizing Services to Support Quick DDoS Mitigation in Cloud Computing Environment”, (Springer), Annals of Telecommunications, vol. 72, 2017.
5. G Somani, MS Gaur, D Sanghi, M Conti, and R Buyya, “DDoS Attacks in Cloud Computing: Issues, Solution Taxonomy, and Future Directions”, (Elsevier) Computer Communications, Volume 107, 2017.
6. G Somani, MS Gaur, D Sanghi, M Conti, and M Rajarajan, “DDoS Victim Service Containment to Minimize Internal Collateral Damages in Cloud Computing”, (Elsevier) Computers and Electrical Engineering, vol. 59, 2017.
7. G Somani, MS Gaur, and D Sanghi, “DDoS Protection and Security Assurance in Cloud”, Computer Communications and Networks, “Guide to Security Assurance for Cloud Computing”, Springer, 171-191, 2015.
8. G Somani, MS Gaur, and D Sanghi, “DDoS Attacks in Cloud: Affecting everyone out there!”, 8^{th} International Conference on Security of Information and Networks, ACM SIN 2014, 169-176, Sochi, Russia, 2015.
9. G Somani, A Johari, M Taneja, U Payne, MS Gaur and D Sanghi, “DARAC: DDoS Mitigation using DDoS Aware Resource Allocation in Cloud”, 11^{th} International Conference on Information Systems Security, ICISS 2015, 263-282, Springer, Kolkata, India, 2015

Bibliography

[1] Kaspersky Labs. Global IT Security Risks Survey 2014 - Distributed Denial of Service (DDoS) Attacks. http://media.kaspersky.com/en/B2B-International-2014-Survey-DDoS-Summary-Report.pdf, 2014.

[2] Arbor Networks. Worldwide Infrastructure Security Report Volume XI., 2015.

[3] Tara Seals. Q1 2015 DDoS Attacks Spike, Targeting Cloud. http://www.infosecurity-magazine.com/news/q1-2015-ddos-attacks-spike/, 2015.

[4] Teri Robinson. Series of DDoS Attacks Plague Linode Data Centers, Infrastructure. http://www.scmagazine.com/, 2015.

[5] Amazon Web Services. AWS Best Practices for DDoS Resiliency. https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf, 2015.

[6] Balachandra Reddy Kandukuri, V Ramakrishna Paturi, and Atanu Rakshit. Cloud Security Issues. In *Services Computing, 2009. SCC '09. IEEE International Conference on*, pages 517–520, Sept 2009.

[7] Lori M Kaufman. Can public-cloud security meet its unique challenges? *IEEE Security & Privacy*, 4(8):55–57, 2010.

[8] Lori M. Kaufman. Data Security in the World of Cloud Computing. *IEEE Security & Privacy*, 7(4):61–64, July 2009.

[9] Dimitrios Zissis and Dimitrios Lekkas. Addressing Cloud Computing Security Issues. *Future Generation Computer Systems*, 28(3):583 – 592, 2012.

[10] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, April 2004.

[11] Steve Mansfield-Devine. The growth and evolution of DDoS. *Network Security*, 2015(10):13–20, 2015.

[12] P Nelson. Cybercriminals Moving into Cloud Big Time, Report Says. http://www.networkworld.com/article/2900125/malware-cybercrime/criminals-/moving-into-cloud-big-time-says-report.html, 2015.

[13] SPAMfighter News. Survey - With DDoS Attacks Companies Lose around Euro 100k/Hr. `http://www.spamfighter.com/News-19554-Survey-With-DDoS-/Attacks-Companies-Lose-around-100kHr.htm`, 2015.

[14] Reuven Cohen. Cloud Attack: Economic Denial of Sustainability (EDoS). `http://www.elasticvapor.com/2009/01/cloud-attack-economic-denial-of.html`, 2009.

[15] Shui Yu. *Distributed Denial of Service Attack and Defense.* Springer Briefs in Computer Science. Springer, 2014.

[16] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Comput. Surv.*, 39(1), April 2007.

[17] Qiao Yan, Richard Yu, Qingxiang Gong, and Jianqiang Li. Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges. *Communications Surveys Tutorials, IEEE*, PP(99):1–1, 2015.

[18] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67:147 – 165, 2016.

[19] Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. Resource Allocation Algorithms for Virtualized Service Hosting Platforms. *Journal of Parallel and distributed Computing*, 70(9):962–974, 2010.

[20] Joseph Idziorek and Mark Tannian. Exploiting Cloud Utility Models for Profit and Ruin. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 33–40. IEEE, 2011.

[21] XiaoFeng Wang and Michael K Reiter. Mitigating Bandwidth-exhaustion Attacks Using Congestion Puzzles. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 257–267. ACM, 2004.

[22] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 287–300. USENIX Association, 2005.

[23] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[24] Akamai Technologies. Akamai's State of the Internet Q4 2013 Executive Summary, Volume 6, Number 4. `http://www.akamai.com/dl/akamai/akamai-soti-q413-exec-summary.pdf`, 2013.

[25] Neustar News. DDoS Attacks and Impact Report Finds Unpredictable DDoS Landscape. http://www.neustar.biz/about-us/news-room/press-releases/2014/neustar-2014-ddos-attacks-and-impact-report-finds-unpredictable-ddos-/landscape#.U33B_nbzdsV, 2014.

[26] Prolexic Technologies. http://www.prolexic.com/, 2014.

[27] Arbor Networks. Understanding the nature of DDoS attacks. http://www.arbornetworks.com/asert/2012/09/understanding-the-nature-of-ddos-attacks/, 2014.

[28] Lee Munson. Greatfire.org faces daily $30,000 bill from DDoS attack. https://nakedsecurity.sophos.com/2015/03/20/greatfire-org-faces-daily-/30000-bill-from-ddos-attack/, 2015.

[29] Akamai. Akamai Quarterly Report Q4. https://www.akamai.com/us/en/about/news/press/2015-press/akamai-releases-fourth-quarter-2014-state-/of-the-internet-report.jsp, 2014.

[30] Infosecurity. Evolving DDoS Tactics Hijack Internet and Cause Attack Surge, 22 APRIL, 2014. http://www.infosecurity-magazine.com/view/38077/evolving-ddos-tactics-hijack-internet-and-cause-attack-surge/, 2014.

[31] Gartner. Gartner Says Cloud Computing Will Become the Bulk of New IT Spend by 2016. http://siliconangle.com/blog/2014/01/27/20-cloud-/computing-statistics-tc0114/, 2014.

[32] Jack Woods. 20 cloud computing statistics every CIO should know. http://siliconangle.com/blog/2014/01/27/20-cloud-computing-statistics-tc0114/, 2014.

[33] ReviewMyLife.co.uk. Amazon CloudFront and S3 maximum cost. http://www.reviewmylife.co.uk/blog/2011/05/19/amazon-cloudfront-and-s3-maximum-cost/, 2011.

[34] S VivinSandar and Sudhir Shenai. Economic Denial of Sustainability (EDoS) in Cloud Services Using HTTP and XML Based DDoS Attacks. *International Journal of Computer Applications*, 41(20):11–16, 2012.

[35] Natalia Vlajic and Armin Slopek. Web Bugs in the Cloud: Feasibility Study of a New Form of EDoS Attack. In *Globecom Workshops (GC Wkshps), 2014*, pages 64–69. IEEE, 2014.

[36] Fahd Al-Haidari, M Sqalli, and Khaled Salah. Evaluation of the Impact of EDoS Attacks Against Cloud Computing Services. *Arabian Journal for Science and Engineering*, 40(3):773–785, 2014.

[37] Khaled Salah, Jose M. Alcaraz Calero, Sherali Zeadally, Sameera Al-Mulla, and Mohammed Alzaabi. Using Cloud Computing to Implement a Security Overlay Network. *IEEE Security & Privacy*, 11(1):44–53, 2013.

[38] Jelena Mirkovic, Max Robinson, and Peter Reiher. Alliance Formation for DDoS Defense. In *Proceedings of the 2003 workshop on New security paradigms*, pages 11–18. ACM, 2003.

[39] Reza Memarian Mohammad, Conti Mauro, and Leppanen Ville. EyeCloud: A BotCloud Detection System. In *In Proceedings of the 5th IEEE International Symposium on Trust and Security in Cloud Computing (IEEE TSCloud 2015), Helsinki, Finland.* IEEE, 2015.

[40] Paolo Farina, Enrico Cambiaso, Gianluca Papaleo, and Maurizio Aiello. Understanding DDoS Attacks From Mobile Devices. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 614–619. IEEE, 2015.

[41] David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

[42] Chris Burt. Large Volume DDoS Attacks See Exceptional Growth in First Half of 2014: Arbor Networks. `http://www.thewhir.com/web-hosting-news/large-volume-ddos-attacks-see-exceptional-growth-first-half-2014-arbor-/networks`, 2014.

[43] Zhang Xu, Haining Wang, and Zhenyu Wu. A Measurement Study on Co-residence Threat inside the Cloud. In *24th USENIX Security 15*, pages 929–944, Washington, D.C., August 2015. USENIX Association.

[44] AUSWEB. SLA Service Level Agreement. `https://my.ausweb.com.au/knowledgebase/112/SLA-Service-Level-Aggreement.html`, 2015.

[45] BudgetVM. Service Level Agreement (SLA). `https://www.budgetvm.com/sla.php`, 2015.

[46] Joseph Latanicki, Philippe Massonet, Syed Naqvi, Benny Rochwerger, and Massimo Villari. Scalable Cloud Defenses for Detection, Analysis and Mitigation of DDoS Attacks. In *Future Internet Assembly*, pages 127–137, 2010.

[47] Jens Lindemann. Towards Abuse Detection and Prevention in IaaS Cloud Computing. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 211–217, Aug 2015.

[48] Diwaker Gupta, Ludmila Cherkasova, Rob Gardner, and Amin Vahdat. Enforcing Performance Isolation Across Virtual Machines in Xen. In *Middleware 2006*, pages 342–362. Springer, 2006.

[49] Soon Hin Khor and Akihiro Nakao. sPoW: On-demand Cloud-based EDDoS Mitigation Mechanism. In *HotDep (Fifth Workshop on Hot Topics in System Dependability)*, 2009.

[50] Madarapu Naresh Kumar, P. Sujatha, Vamshi Kalva, Rohit Nagori, Anil Kumar Katukojwala, and Mukesh Kumar. Mitigating Economic Denial of Sustainability (EDoS) in Cloud Computing Using In-cloud Scrubber Service. In *Proceedings of the 2012 Fourth International Conference on Computational Intelligence and Communication Networks*, CICN '12, pages 535–539, Washington, DC, USA, 2012. IEEE Computer Society.

[51] Fahd Al-Haidari, Mohammed H Sqalli, and Khaled Salah. Enhanced EDoS-Shield for Mitigating EDoS Attacks Originating from Spoofed IP Addresses. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1167–1174. IEEE, 2012.

[52] Mohammed H Sqalli, Fahd Al-Haidari, and Khaled 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*, pages 49–56. IEEE, 2011.

[53] Wael Alosaimi and Khalid 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)*, pages 116–121, 2013.

[54] Huangxin Wang, Quan Jia, Dan Fleck, Walter Powell, Fei Li, and Angelos Stavrou. A Moving Target DDoS Defense Mechanism. *Computer Communications*, 46:10–21, 2014.

[55] Tarun Karnwal, T Sivakumar, and G Aghila. A Comber Approach to Protect Cloud Computing Against XML DDoS and HTTP DDoS Attack. In *Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on*, pages 1–5. IEEE, 2012.

[56] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet Denial-of-Service with Capabilities. *ACM SIGCOMM Computer Communication Review*, 34(1):39–44, 2004.

[57] Muddassar Masood, Zahid Anwar, Syed Ali Raza, and Muhammad Ali 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*, pages 37–42, Dec 2013.

[58] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou, and Walter Powell. Catch Me If You Can: A Cloud-Enabled DDoS Defense. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pages 264–275. IEEE, 2014.

[59] N Jeyanthi and PC Mogankumar. A Virtual Firewall Mechanism Using Army Nodes to Protect Cloud Infrastructure from DDoS Attacks. *Cybernetics and Information Technologies*, 14(3):71–85, 2014.

[60] Zubair A. Baig and Farid Binbeshr. Controlled Virtual Resource Access to Mitigate Economic Denial of Sustainability (EDoS) Attacks Against Cloud Infrastructures. In *Proceedings of the 2013 International Conference on Cloud Computing and Big Data*, CLOUDCOM-ASIA '13, pages 346–353, Washington, DC, USA, 2013. IEEE Computer Society.

[61] Bhavna Saini and Gaurav Somani. Index Page Based EDoS Attacks in Infrastructure Cloud. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 382–395. Springer, 2014.

[62] Amazon. Amazon CloudWatch. `https://aws.amazon.com/cloudwatch/`, 2014.

[63] AWS Discussion Forum. https://forums.aws.amazon.com, 2016.

[64] P. Du and A. Nakao. DDoS Defense as a Network Service. In *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, pages 894–897, April 2010.

[65] Gaurav Somani, Abhinav Johri, Mohit Taneja, Utkarsh Pyne, Manoj Singh Gaur, and Dheeraj Sanghi. DARAC: DDoS Mitigation using DDoS Aware Resource Allocation in Cloud. In *11th International Conference, ICISS, Kolkata, India, December 16-20, 2015, Proceedings*, pages 263–282, 2015.

[66] Drew Dean and Adam Stubblefield. Using Client Puzzles to Protect TLS. In *USENIX Security Symposium*, volume 42, 2001.

[67] David Leggett. CAPTCHAs tough on sales common way to test user tolerance. `http://www.uxbooth.com/articles/captchas-tough-on-sales-common-way-to-/test-user-tolerance/`, 2009.

[68] Joseph Idziorek, Mark Tannian, and Doug Jacobson. Detecting Fraudulent Use of Cloud Resources. In *Proceedings of the 3rd ACM workshop on Cloud computing security*, pages 61–72. ACM, 2011.

[69] Mohd Nazri Ismail, Abdulaziz Aborujilah, Shahrulniza Musa, and AAmir Shahzad. Detecting Flooding Based DoS Attack in Cloud Computing Environment Using Covariance Matrix Approach. In *Proc. of the 7th International Conf. Ubiquitous Information Management and Communication*, page 36. ACM, 2013.

[70] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical Approaches to DDoS Attack Detection and Response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314. IEEE, 2003.

[71] Pourya Shamsolmoali and Masoumeh Zareapoor. Statistical-based Filtering System Against DDoS Attacks in Cloud Computing. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 1234–1239. IEEE, 2014.

[72] Juan Francisco Gómez-Lopera, José Martínez-Aroza, Aureliano M Robles-Pérez, and Ramón Román-Roldán. An Analysis of Edge Detection by using the Jensen-Shannon Divergence. *Journal of Mathematical Imaging and Vision*, 13(1):35–56, 2000.

[73] Steven J Templeton and Karl E Levitt. Detecting Spoofed Packets. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 164–175. IEEE, 2003.

[74] Qi Chen, Wenmin Lin, Wanchun Dou, and Shui Yu. CBF: A Packet Filtering Method for DDoS Attack Defense in Cloud Environment. In *Dependable, Autonomic and Secure Computing (DASC), IEEE Ninth Int. Conf. on*, pages 427–434. IEEE, 2011.

[75] N Jeyanthi, N Ch SN Iyengar, PC Mogan Kumar, and A Kannammal. An Enhanced Entropy Approach to Detect and Prevent DDoS in Cloud Environment. *International Journal of Communication Networks and Information Security (IJCNIS)*, 5(2), 2013.

[76] Thomas Vissers, Thamarai Selvi Somasundaram, Luc Pieters, Kannan Govindarajan, and Peter Hellinckx. DDoS Defense System for Web Services in a Cloud Environment. *Future Generation Computer Systems*, 37:37–45, 2014.

[77] Anusha Koduru, TulasiRam Neelakantam, Saira Bhanu, and S Mary. Detection of Economic Denial of Sustainability Using Time Spent on a Web Page in Cloud. In *Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on*, pages 1–4, Oct 2013.

[78] Shui Yu, Wanlei Zhou, Song Guo, and Minyi Guo. A Feasible IP Traceback Framework Through Dynamic Deterministic Packet Marking. *IEEE Transactions on Computers*, 65(5):1418–1427, 2016.

[79] Ashley Chonka, Yang Xiang, Wanlei Zhou, and Alessio Bonti. Cloud Security Defence to Protect Cloud Computing Against HTTP-DoS and XML-DoS Attacks. *Journal of Network and Computer Applications*, 34(4):1097–1107, 2011.

[80] Sérgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto, and Ronaldo M Salles. Botnets: A Survey. *Computer Networks*, 57(2):378–403, 2013.

[81] Lanjuan Yang, Tao Zhang, Jinyu Song, JinShuang Wang, and Ping Chen. Defense of DDoS Attack for Cloud Computing. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 2, pages 626–629. IEEE, 2012.

[82] Opeyemi A Osanaiye. IP Spoofing Detection for Preventing DDoS Attack in Cloud Computing. In *Intelligence in Next Generation Networks (ICIN), 18th International Conf on*, pages 139–141. IEEE, 2015.

[83] Jelena Mirković, Gregory Prier, and Peter Reiher. Attacking DDoS at the source. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 312–321, Nov 2002.

[84] VS Huang, Robert Huang, and Ming Chiang. A DDoS Mitigation System with Multi-stage Detection and Text-Based Turing Testing in Cloud Computing. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 655–662. IEEE, 2013.

[85] Hongbin Luo, Yi Lin, Hongke Zhang, and Moshe Zukerman. Preventing DDoS Attacks by Identifier/locator Separation. *Network, IEEE*, 27(6):60–65, 2013.

[86] Terence KT Law, John Lui, and David KY Yau. You Can Run, But You Can't Hide: An Effective Statistical Methodology to Trace Back DDoS Attackers. *Parallel and Distributed Systems, IEEE Transactions on*, 16(9):799–813, 2005.

[87] DDoS Deflate. https://github.com/jgmdev/ddos-deflate, 2016.

[88] Baohui Li, Wenjia Niu, Kefu Xu, Chuang Zhang, and Peng Zhang. You Cant Hide: A Novel Methodology to Defend DDoS Attack Based on Botcloud. In *Applications and Techniques in Information Security, Communications in Computer and Information Science*, pages 203–214. Springer Berlin Heidelberg, 2015.

[89] Mark Graham, Winckles Adrian, and Sanchez-Velazquez. Erika. Botnet Detection Within Cloud Service Provider Networks Using Flow Protocols. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on. IEEE*, pages 1614–1619, 2015.

[90] Hammi Badis, Guillaume Doyen, and Rida Khatoun. A Collaborative Approach for a Source Based Detection of Botclouds. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 906–909. IEEE, 2015.

[91] Siqin Zhao, Kang Chen, and Weimin Zheng. Defend Against Denial of Service Attack with VMM. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, pages 91–96. IEEE, 2009.

[92] Shui Yu, Yonghong Tian, Song Guo, and Dapeng Oliver Wu. Can We Beat DDoS Attacks in Clouds? *Parallel and Distributed Systems, IEEE Transactions on*, 25(9):2245–2254, 2014.

[93] Peng Xiao, Wenyu Qu, Heng Qi, and Zhiyang Li. Detecting DDoS attacks against data center with correlation analysis. *Computer Communications*, 67:66–74, 2015.

[94] Jian Zhang, Ya-Wei Zhang, Jian-Biao He, and Ou Jin. A Robust and Efficient Detection Model of DDoS Attack for Cloud Services. In *Algorithms and Architectures for Parallel Processing*, pages 611–624. Springer International Publishing, 2015.

[95] M. Du and F. Li. Atom: Automated tracking, orchestration and monitoring of resource usage in infrastructure as a service systems. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 271–278, Oct 2015.

[96] Alqahtani Sarra and Gamble Rose. DDoS Attacks in Service Clouds. In *48th Hawaii International Conference on System Sciences*. IEEE Computer Society, 2015.

[97] Gilad Yossi, Herzberg Amir, Sudkovitch Michael, and Goberman Michael. CDN-on-Demand: An Affordable DDoS Defense via Untrusted Clouds. In *Network and Distributed System Security Symposium (NDSS)*, 2016.

[98] Rishikesh Sahay, Gregory Blanc, Zonghua Zhang, and Hervé Debar. Towards Autonomic DDoS Mitigation using Software Defined Networking. In *SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies*. Internet society, 2015.

[99] Xiulei Wang, Ming Chen, and Changyou Xing. SDSNM: A Software-Defined Security Networking Mechanism to Defend against DDoS Attacks. In *Frontier of Computer Science and Technology (FCST), 2015 Ninth International Conference on*, pages 115–121. IEEE, 2015.

[100] Bing Wang, Yao Zheng, Wenjing Lou, and Y Thomas Hou. DDoS Attack Protection in the Era of Cloud Computing and Software-defined Networking. *Computer Networks*, 81:308–319, 2015.

[101] Qiao Yan and F Yu. Distributed denial of service attacks in software-defined networking with cloud computing. *Communications Magazine, IEEE*, 53(4):52–59, 2015.

[102] Soon Hin Khor and Akihiro Nakao. DaaS: DDoS Mitigation-as-a-Service. In *Applications and the Internet (SAINT), 11th Int. Symp. on*, pages 160–171. IEEE, 2011.

[103] Fouad Guenane, Michele Nogueira, and Guy Pujolle. Reducing DDoS Attacks Impact Using a Hybrid Cloud-based Firewalling Architecture. In *Global Information Infrastructure and Networking Symposium (GIIS), 2014*, pages 1–6. IEEE, 2014.

[104] Hiroshi Fujinoki. Dynamic binary user-splits to protect cloud servers from ddos attacks. In *Proceedings of the Second International Conference on Innovative Computing and Cloud Computing*, ICCC '13, pages 125:125–125:130, New York, NY, USA, 2013. ACM.

[105] Yichuan Wang, Jianfeng Ma, Di Lu, Xiang Lu, and Liumei Zhang. From high-availability to collapse: quantitative analysis of "cloud-droplet-freezing" attack threats to virtual machine migration in cloud computing. *Cluster Computing*, 17(4):1369–1381, 2014.

[106] Shuen-Chih Tsai, I-Hsien Liu, Chien-Tung Lu, Chan-Hua Chang, and Jung-Shian Li. Defending cloud computing environment against the challenge of ddos attacks based on software defined network. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the Twelfth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Nov., 21-23, 2016, Kaohsiung, Taiwan, Volume 1*, pages 285–292. Springer, 2017.

[107] Luis M Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically Scaling Applications in the Cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[108] Fahd Al-Haidari, M Sqalli, and Khaled Salah. Impact of CPU Utilization Thresholds and Scaling Size on Autoscaling Cloud Resources. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 256–261. IEEE, 2013.

[109] Ming Mao, Jie Li, and Marty Humphrey. Cloud Auto-scaling with Deadline and Budget Constraints. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 41–48. IEEE, 2010.

[110] Chiwook Jeong, Taejin Ha, Jaeseon Hwang, Hyuk Lim, and JongWon Kim. MARS: Measurement-based Allocation of VM Resources for Cloud Data Centers. In *Proc. of Student workhop*, pages 63–66. ACM, 2013.

[111] TagMan. Just One Second Delay In Page-Load Can Cause 7% Loss In Customer Conversions. `http://www.tagman.com/mdp-blog/2012/03/just-one-/second-delay-/in-page-load-can-cause-7-loss-in-customer-conversions/`, 2013.

[112] Sherice Jacob. Speed Is A Killer - Why Decreasing Page Load Time Can Drastically Increase Conversions. `https://blog.kissmetrics.com/speed-is-a-killer/`, 2011.

[113] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[114] Ming Luo, Tao Peng, and Christopher Leckie. CPU-based DoS Attacks Against SIP Servers. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 41–48. IEEE, 2008.

[115] Huan Liu. A New Form of DOS Attack in a Cloud and its Avoidance Mechanism. In *Proc. of 2010 workshop on Cloud computing security*, pages 65–76. ACM, 2010.

[116] Zhang Xu, Haining Wang, Zichen Xu, and Xiaorui Wang. Power Attack: An Increasing Threat to Data Centers. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, volume 14, 2014.

[117] Zahid Anwar and Asad Waqar Malik. Can a DDoS Attack Meltdown My Data Center? A Simulation Study and Defense Strategies. *IEEE Communications Letters*, 18(7):1175–1178, 2014.

[118] Yusen Li, Xueyan Tang, and Wentong Cai. On Dynamic Bin Packing for Resource Allocation in the Cloud. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 2–11. ACM, 2014.

[119] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *Proceedings of the 4th USENIX Networked Systems Design and Implementation*, NSDI, pages 17–17, Berkeley, CA, USA, 2007.

[120] Francesco Palmieri, Sergio Ricciardi, and Ugo Fiore. Evaluating Network-Based DoS Attacks under the Energy Consumption Perspective: New Security Issues in the Coming Green ICT Area. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 374–379. IEEE, 2011.

[121] Varun Bhardwaj, Anamika Sharma, and Gaurav Somani. Client-side Verifiable Accounting in Infrastructure Cloud. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 361–366. IEEE, 2015.

[122] Vyas Sekar and Petros Maniatis. Verifiable Resource Accounting for Cloud Computing Services. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 21–26. ACM, 2011.

[123] Ki-Woong Park, Jaesun Han, JaeWoong Chung, and Kyu Ho Park. Themis: A Mutually Verifiable Billing System for the Cloud Computing Environment. *Services Computing, IEEE Transactions on*, 6(3):300–313, 2013.

[124] Joseph Idziorek, Mark Tannian, and Douglas Jacobson. Attribution of Fraudulent Resource Consumption in the Cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 99–106. IEEE, 2012.

[125] Luigi Clemente. Auto Scaling on AWS: an overview. `http://www.luigiclemente.com/scalable-websites-on-aws-an-overview/`, 2013.

[126] Mor Sides, Anat Bremler-Barr, and Elisha Rosensweig. Yo-Yo Attack: Vulnerability In Auto-scaling Mechanism. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 103–104. ACM, 2015.

[127] Amazon Web Services Documentation. Manual Scaling. `http://docs.aws.amazon.com/autoscaling/latest/userguide/as-manual-scaling.html`, Retrieved on April 7, 2017.

[128] Microsoft Azure Documentation. Autoscaling: Types of scaling. `https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling`, Retrieved on April 7, 2017.

[129] Amazon Web Services Documentation. Scheduled Scaling. `http://docs.aws.amazon.com/autoscaling/latest/userguide/schedule_time.html`, Retrieved on April 7, 2017.

[130] Rackspace Support. Rackspace Auto Scale Control Panel User Guide - Create a scaling policy. `https://support.rackspace.com/how-to/rackspace-auto-scale-control-panel-user-guide-create-a-scaling-policy/`, Retrieved on April 7, 2017.

[131] Amazon Web Services Documentation. Dynamic Scaling. `http://docs.aws.amazon.com/autoscaling/latest/userguide/as-scale-based-on-demand.html`, Retrieved on April 7, 2017.

[132] Google Cloud Platform: Compute Engine Documentation. Load Balancing and Scaling . `https://cloud.google.com/compute/docs/load-balancing-and-autoscaling#policies`, Retrieved on April 7, 2017.

[133] William G. Morein, Angelos Stavrou, Debra L. Cook, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. Using Graphic Turing Tests to Counter Automated DDoS Attacks Against Web Servers. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 8–19, New York, NY, USA, 2003. ACM.

[134] Wanchun Dou, Qi Chen, and Jinjun Chen. A Confidence-based Filtering Method for DDoS Attack Defense in Cloud Environment. *Future Generation Computer Systems*, 29(7):1838–1850, 2013.

[135] Srimathy Mohan, Ferdous M Alam, John W Fowler, Mohan Gopalakrishnan, and Antonios Printezis. Capacity Planning and Allocation for Web-Based Applications. *Decision Sciences*, 45(3):535–567, 2014.

[136] John Stevens . How Slow is Too Slow in 2016? . `https://www.webdesignerdepot.com/2016/02/how-slow-is-too-slow-in-2016/`, 2016.

[137] Hagen Petters. Why You Should Focus On Page Load Speed in 2017. https://www.shoutmeloud.com/why-you-should-focus-on-page-load-speed.html, 2017.

[138] Howard Frazier-Broadcom, Schelto Van Doorn-Intel, Robert Hays-Intel, Paul Kolesar-CommScope, and Geoff Thompson-Nortel. IEEE 802.3 ad Link Aggregation (LAG). 2007.

[139] Jeffrey Lyon. How to Help Prepare for DDoS Attacks by Reducing Your Attack Surface. https://blogs.aws.amazon.com/security/blog/tag/DDoS, August 26, 2015.

[140] José Jair Santanna, Roland van Rijswijk-Deij, Rick Hofstede, Anna Sperotto, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras. Bootersan analysis of ddos-as-a-service attacks. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 243–251. IEEE, 2015.

[141] Kim Weins. Cloud Computing Trends: 2015 State of the Cloud Survey. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-/state-cloud-survey, 2015.

[142] Netfilter/iptables Project. www.netfilter.org, 2016.

[143] HTTP Archive. Compare stats. httparchive.org/compare.php, 2016.

[144] Massimo Ficco and Massimiliano Rak. Stealthy Denial of Service Strategy in Cloud Computing. *IEEE transactions on cloud computing*, 3(1):80–94, 2015.

[145] Michael Kerrisk. SCHED_SETAFFINITY. http://man7.org/linux/man-pages/man2/sched_setaffinity.2.html, 2016.

[146] Robert M. Love. Taskset Command. http://www.linuxcommand.org/man_pages/taskset1.html, 2016.

[147] Oskar Andreasson. Ipsysctl tutorial 1.0.4. https://www.frozentux.net/ipsysctl-tutorial/chunkyhtml/tcpvariables.html, 2016.

[148] Paul Menage. CGroups. https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt, 2016.

[149] Ilia Kolochenko. DDoS Attacks: a Perfect Smoke Screen for APTs and Silent Data Breaches. http://www.csoonline.com/article/2986967/advanced-/persistent/-threats/ddos-attacks-a-perfect-smoke/-screen-for-apts-and-/silent/-data-/breaches.html, 2015.

[150] Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/, 2016.

[151] Jerome Petazzoni. Gathering LXC and Docker Containers Metrics. https://blog.docker.com/2013/10/gathering-lxc-docker-containers-metrics/, 2013.

[152] Phil Muncaster. DDoS-ers Take Down Mitigation Tools in Q1. http://www.infosecurity-magazine.com/news/ddos-ers-take-down-mitigation/, 2016.

[153] CERT-UK. An Introduction to Threat Intelligence. `https://www.cert.gov.uk/wp-/content/uploads/2015/03/An-introduction-to-threat-intelligence.pdf`, 2015.

[154] Dave Shackleford. SANS Security Analytics Survey. `https://www.sans.org/reading-room/whitepapers/analyst/security-analytics-survey-34980`, 2013.

[155] Ponemon Institute. Big Data Analytics in Cyber Defense. `http://www.ponemon.org/local/upload/file/Big_Data_Analytics_in_Cyber_Defense_V12.pdf`, February 2013.

[156] Jeanna Neefe Matthews, Wenjin Hu, Madhujith Hapuarachchi, Todd Deshane, Demetrios Dimatos, Gary Hamilton, Michael McCabe, and James Owens. Quantifying the Performance Isolation Properties of Virtualization Systems. In *Proceedings of the 2007 workshop on Experimental computer science*, page 6. ACM, 2007.

Biography

Gaurav Somani has completed his PhD in Computer Science and Engineering at Department of Computer Science and Engineering at Malaviya National Institute of Technology, Jaipur, INDIA. Currently, he is an assistant professor at Department of Computer Science and Engineering at Central University of Rajasthan, India. Earlier, he served as a lecturer at the LNM Institute of Information Technology, Jaipur, INDIA. He has completed his MTech in information and communication technology from DAI-ICT, Gandhinagar, India, and BE in information technology from the University of Rajasthan, India, both with a distinction. His research interests include Cloud Computing, Virtualization, Distributed Computing, Network and System Security, and Security Engineering. He is a recipient of Teacher Fellowship from University Grants Commission (UGC), INDIA.

He has published several papers in various international conferences and journals and served as a reviewer for many top journals and conferences. Some of his top papers are published at IEEE Transactions on Dependable and Secure Computing, Computer Communications, Computer Networks, Annals of Telecommunications, Future Generation Computer Systems, Computers and Electrical Engineering, IEEE International Conference on Cloud Computing, and IEEE Cloud Computing. So far, he has supervised 10 masters and 14 bachelors theses. He has edited a book on “Research Advances in Cloud Computing (Springer)” with Sanjay Chaudhay and Rajkumar Buyya. He is also a guest editor of a special issue of Journal of Software: Practice and Experience on “Integration of Internet of Things, Cloud and Big Data Analytics”. He is also editing a book on “Versatile Cybersecurity (Springer)” with Mauro Conti and Radha Poovendran. He is also part of program committees of various conferences and was a keynote and tutorial chair for ICISS 2016. He is a member of IEEE and ACM.