MODELING AND MITIGATION OF ECONOMIC DENIAL OF SUSTAINABILITY (EDoS) ATTACKS IN CLOUD COMPUTING

BY

Fahd Al-Haidari

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

November 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by **FAHD ABDUALSALAM AL-HAIDARI** under the direction of his dissertation advisor and approved by his dissertation committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHIOLOSOPHY in COMPUTER SCIENCE AND ENGINEERING**.

Dissertation Committee

[illegible]
Dr. Mohammed Sqalli
Thesis Advisor

[illegible]
Dr. Khaled Salah
Co-advisor

[illegible]
Dr. Shokri Selim
Member

Ashraf
Dr. Ashraf Mahmoud
Member

[illegible]
Dr. Farag Azzedin
Member

[illegible]
Dr. AL-Turki
Dr. Adel Ahmed
Department chairman

[illegible]
Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

5/1/13
Date

©Fahd Al-Haidari

2012

Dedication

To my loving

I dedicate this dissertation to my parents,

my wife, my kids (Suha, Amr, Marwh),

my brothers, and my sister.

ACKNOWLEDGMENTS

I thank Allah for granting me the guidance, patience, health, and determination to successfully accomplish this work.

I would like to express deep appreciation and gratitude to my dissertation advisor **Dr. Mohammed Sqalli** and co advisor **Dr. Khaled Salah** for their constant help, guidance, encouragement and invaluable support. Thanks are due to my dissertation committee members, **Dr. Shokri Selim, Dr. Ashraf Mahmoud,** and **Dr. Farag Azzedin**, for their cooperation, comments and support.

I would like to thank King Fahd University of Petroleum and Minerals for sponsoring me throughout my graduate studies.

I also would like to thank my parents, my wife and my kids who always support me with their love, patience, encouragement and constant prayers. I would like to thank my brothers and sister for their love and support throughout my study.

Finally, I would like to thank all my friends, mostly those in the ICS department and all others who helped in this dissertation work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EDoS	:	Economic Denial of Sustainability
DDoS	:	Distributed Denial of Service
APIs	:	Application Programming Interfaces
NIST	:	National Institute of Standards and Technology
SaaS	:	Software as a Service
PaaS	:	Platform as a Service
IaaS	:	Infrastructure as a Service
EC2	:	Elastic Cloud Computing
S3	:	Simple Storage Service
CapEx	:	Capital Expenditures
OpEx	:	Operational Expenditures
IDC	:	International Data Corporation
XSS	:	Cross-Site Scripting
CSRF	:	Cross-Site Request Forgery
REST	:	Representational State Transfer
SOAP	:	Simple Object Access Protocol

VMBR	:	Virtual-Machine Based Rootkit
LKS	:	Availability –Localized Knowledge Spillovers
IAM	:	Accountability – Independent Accountability Mechanism
SLA	:	Service Level Agreements
QoS	:	Quality of Service
VM	:	Virtual Machines
AMI	:	Amazon Machine Image
AWS	:	Amazon Web Services
IDS	:	Intrusion Detection System
CDN	:	Content Distribution Networks
sPoW	:	Self-verifying Proof of Work
SQS	:	Amazon Simple Queue Service
LB	:	Load Balancer
RDS	:	Relational Database Service
SRIOV	:	Single Root I/O Virtualization
MOO	:	Multi Objective Optimization
WS	:	Weighted Sum

NBI	:	Normal Boundary Intersection
NC	:	Normal Constraint
PMMLCG	:	Prime Modulus Multiplicative Linear Congruential Generator
Req/sec	:	Request per second
GPU	:	Graphics Processing Unit
SOS	:	Secure Overlay Services
CAPTCHA	:	Completely Automated Public Turing test to tell Computers and Humans Apart
VF	:	Virtual Firewalls
V-Nodes	:	Verifier cloud Nodes
VSG	:	Virtual Security Gateway
ELB	:	Elastic Load Balancer
ECU	:	EC2 Computing Units
TTL	:	Time-To-Live
SWITCH	:	Swiss Academic Research Network
IP2HC	:	IP to Hop-Count
PID	:	Path Identification

ABSTRACT

Full Name	:	Fahd AbdulSalam Al-Haidari
Thesis Title	:	MODELING AND MITIGATION OF ECONOMIC DENIAL OF SUSTAINABILITY (EDOS) ATTACKS IN CLOUD COMPUTING
Major Field	:	Computer Science and Engineering
Date of Degree	:	November, 2012

Cloud computing has become one of the fastest growing segments of IT industry. Due to its flexibility, pay per use, elasticity, scalability, and other attributes promised by this paradigm, cloud computing has gained the interest of large organizations and corporations for hosting their services. Cloud computing has been identified as the technology with the potential to have the most significant impact on organizations in 2011. However, the benefits foreseen of the cloud bear several unaddressed issues of high importance, especially with regards to the level of security provided by a cloud computing service model. Security has been identified clearly as the greatest challenge to cloud computing. The cloud introduces resource-rich computing platforms, where adopters are charged based on the usage of the cloud's resources, known as "pay-as-you-use" or utility computing. With this model, a conventional Distributed Denial of Service (DDoS) attack on server and network resources is transformed in a cloud environment to a new breed of attacks that targets the cloud adopter's economic resources, namely Economic Denial of Sustainability attack (EDoS). EDoS occurs when zombie machines (part of a botnet) send a large amount of service requests towards the cloud, exploiting the cloud's scalability, to charge a cloud adopter's bill an exorbitant extra amount of cost, leading to large-scale service withdrawal or bankruptcy. In this research, we study the EDoS attacks and their impact on cloud computing cost, resources and performance. We also propose effective countermeasures and mitigation techniques against such attacks. The effectiveness of the proposed mitigations techniques is evaluated analytically and using simulation.

خلاصة الرسالة

الاسم الكامل: فهد عبد السلام الحيدري

عنوان الرسالة: نمذجة ومكافحة هجمات ال (EDoS) في الحوسبة السحابية

التخصص: علوم وهندسة الحاسب الآلي

تاريخ الدرجة العلمية: نوفمبر, 2012

الحوسبة السحابية هي حاليا واحدة من من أسرع القطاعات نموا في تكنولوجيا المعلومات. بسبب مرونتها، واليات الدفع المالي بحسب الاستخدام "pay per use", وقابليتها للتوسع, وغيرها من الصفات التي وعد بها هذا النموذج, اكتسب هذا القطاع رغبة الكثير من المنظمات الكبيرة والشركات لاستضافة خدماتها على نظام الحوسبة السحابية . وفقاً لشركة أبحاث جارتنر, الولايات المتحدة, تم تصنيف الحوسبة السحابية كواحدة من اهم عشر تقنيات تكنولوجيا المعلومات التي لها الأثر الأكبر على المنظمات والشركات في عام 2011. ومع ذلك، فإن الحوسبة السحابية وبالرغم من فوائدها, حملت العديد من الاشكالات الهامة, خاصة فيما يتعلق بمستوى الأمان التي يتم توفيرها بواسطة خدمة نموذج الحوسبة السحابية. وفقا لدراسة حديثة أجرتها مؤسسة البيانات الدولية (IDC)، ورد الأمن بوصفه أكبر تحد للحوسبة السحابية. الحوسبة السحابية تتمتع بوفرة الموارد والتي تعتبر واحدة من المميزات لهذا النموذج حيث يتم الدفع الى مزود الخدمة بحسب استخدام الموارد. مع هذا النموذج، يتم تحويل هجوم (DDoS) التقليدي على موارد الخادم والشبكة في بيئة الحوسبة السحابية لسلالة جديدة من الهجمات التي تستهدف الموارد المالية للمستفيد من الحوسبة السحابية (Adopter), وهو ما يطلق عليه " Economic Denial of Sustainability " او (EDoS). هذا النوع من الهجمات, يحدث عندما ترسل ألياً كمية كبيرة من طلبات الخدمة نحو الحوسبة السحابية من قبل اجهزة مخترقة وموجهة من المهاجم (bots)، حيث يتم استغلال قابلية السحابة للتوسع لغرض شحن فاتورة المستفيد مبالغ إضافية باهظة, مما يدفع المستفيدين إلى الانسحاب من الخدمة على نطاق واسع أوقد يؤدي الى الإفلاس. في هذا البحث، ندرس هجمات EDoS وتأثيرها على الحوسبة السحابية. أيضا نطور تدابير مضادة جديدة وتقنيات تخفيف من مثل هذه الهجمات. كذلك يتم تقييم فعالية تقنيات التخفيف المقترحة باستخدام المحاكاة و التحليل الكمي.

CHAPTER 1

INTRODUCTION

Cloud computing is currently one the most hyped information technology innovations and has become the fastest growing segment of IT industry. Due to the flexibility, pay per use, elasticity, scalability and other attributes promised by this paradigm, it gained the interest of large organizations and corporate to host their services onto the cloud. Cloud computing as group of technologies has been on the top 10 lists for a few years. US research company, Gartner, has identified cloud computing as first of the top 10 technologies with the potential for significant impact on organizations in 2011 [1].

The term cloud computing as defined by NIST [2] is “A model for on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction”. The computing resources that constitute the power of the cloud system are a collection of services, applications, information, and infrastructure that could be sold on-demand and accessed over the Internet in a fast, cost-effective way. The services are fully managed by the provider and users can have as much or as little of a service as they want at any given time.

One main misconception is that the cloud is one thing. However, cloud computing is the combination of many preexisting technologies that include aspects of grid computing and virtualization combined with application programming interfaces (APIs) and utilities to supply access to the virtualized environments [3]. New advances in processors, virtualization technology, disk storage, broadband Internet connection, and fast, inexpensive servers have combined to make the cloud a more compelling solution. However, as always the introduction of new technologies presents certain risks, and it could open an organization to security vulnerabilities and threats. With security being one of the top concerns that hinders cloud computing [4, 5, 6, 7], for that reason, it has become a major field of study.

1.1 Research Problem Statement

The ability to respond to security threats and events is listed as one of the main issues of concern in the cloud computing area. In this research, we study the Economic Denial of Sustainability (EDoS) attacks and their effect on cloud computing. An identification of possible EDoS attacks is carried out. Effective solutions to such EDoS attacks will be introduced. The effectiveness of the proposed mitigation techniques will be evaluated using simulation that is verified through quantitative analysis. The results obtained will allow us to better secure the cloud networks that will provide or use cloud computing services in the near future.

1.2 Research Objectives

The ultimate objective of this research is to develop and evaluate new countermeasures and mitigation techniques against Economic Denial of Sustainability (EDoS) attacks in Cloud Computing. A classification of EDoS attacks and countermeasures will also be proposed to benefit the cloud computing research community. The primary objectives of the research can be summarized as follows.

- Explore EDoS attacks and their effect on cloud computing.
- Classify and propose a robust taxonomy of EDoS attacks, and its variants.
- Develop and implement new countermeasures and mitigation techniques against the EDoS attacks.
- Test the effectiveness of the proposed techniques from Objective 3 above, using simulation that is verified through quantitative analysis.

1.3 Main Contributions

The main contributions of this dissertation as follows:

- Conducting an extensive literature survey about the security of cloud computing.
- Studying the impact of the EDoS attacks on the cloud computing services using a simulation model verified by an analytical model.
- Proposing and describing a novel architecture, namely EDoS-Shield, which is deployable as an on-demand cloud-based EDoS mitigation technique.

- Developing a novel and practical approach used as an enhancement to the EDoS-Shield by using a Graphical Turing test and TTL values in order to mitigate EDoS attacks originating from spoofed IP addresses.
- Developing discrete simulation and analytical models to verify the effectiveness of the mitigation techniques.
- Studying and simulating the impact of setting both the upper utilization threshold and the scaling size of the provisioning technique on the performance of the cloud services.
- Formulating and solving optimization problems for tuning both the upper utilization threshold and the scaling size of the provisioning technique in cloud computing.

1.4 Organization of Dissertation

This dissertation is organized as follows. Chapter 2 presents an extensive literature survey of the cloud computing system and the EDoS attacks. In Chapter 3, we evaluate the impact of the EDoS attack on the cloud service by using a discrete event simulation as well as an analytical model. Chapter 4 presents a novel mitigation technique against EDoS attack namely EDoS-Shield evaluated by both the simulation and analysis models. An Enhanced EDoS-Shield mitigation technique is discussed in Chapter 5. Finally, Chapter 6 includes the conclusion and the future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we give an overview about all the related topics that will be discussed throughout this work. In Section 1, we broadly look into the definition of the cloud and the different layers in a typical cloud computing environment. Then, we classify the different security concerns and summarize what has been presented in the literature in Section 2. In Section 3, we present the ways in which attacks can compromise the cloud by presenting taxonomy about the cloud attacks. Then, we describe the Economic Denial of Sustainability (EDoS) in details along with its literature in Section 4.

2.1 Cloud Computing

Cloud computing is still an evolving paradigm. Its definitions, use cases, underlying technologies, issues, risks, and benefits will be refined in a spirited debate by the public and private sectors. These definitions, attributes, and characteristics will evolve and change over time. However, the National Institute of Standards and Technology (NIST) in the U.S., whose publication is generally well accepted, has defined cloud computing by describing five essential characteristics, three cloud service models, and four cloud deployment models [2]. They are summarized in a visual form in Figure 2.1.

2.1.1 Essential Characteristics

Because cloud computing is related to a number of other technologies, it is best defined by the presence of a number of characteristics. The Essential Characteristics of Cloud Computing are those things that are required of a service to make it qualify as true “Cloud Computing”.

On-demand self-service. Cloud customers can provision cloud services, such as server time, network storage, software, process, or more from the service provider as needed automatically without going through a lengthy process. The used resource can be automatically deprovisioned.

Broad network access. The technology is available over the network and can be accessed through standard mechanisms by both thick and thin clients. All of the servers are connected to a high-speed network that allows data to flow to the Internet, as well as between computing and storage elements.

Resource pooling. Computing resource pooling allows a cloud provider to serve its consumers via a multi-tenant model by which physical and virtual resources are assigned and reassigned according to consumer demand. An example of such characteristics is about Amazon EC2 which allows users to share machine images, and start up servers based on their needs.

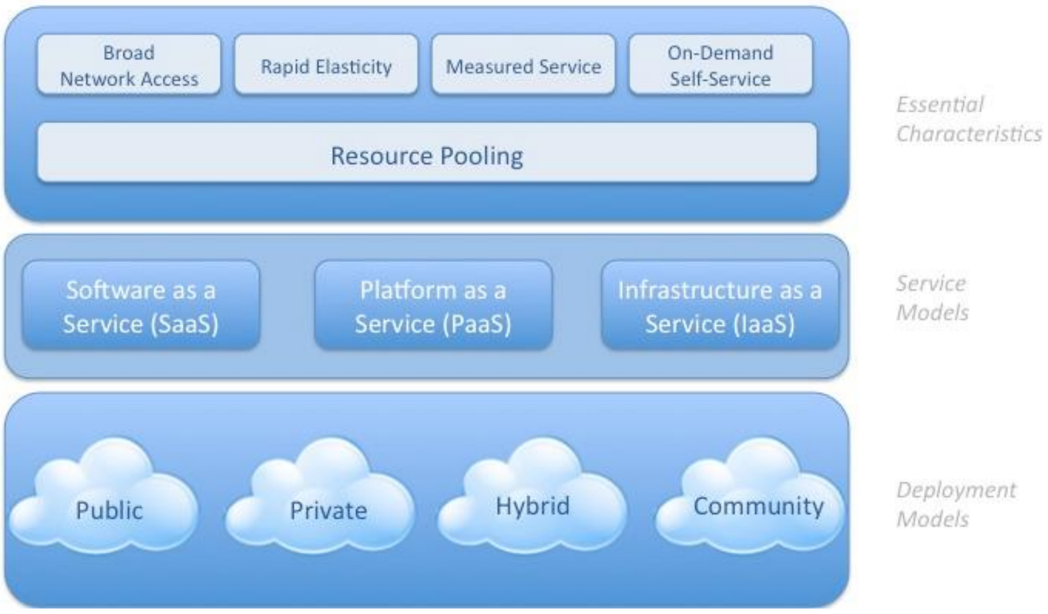


Figure 2.1: Cloud computing visual model of NIST [3]

Rapid elasticity. This characteristic means that resource allocation can be scaled out or in depending on demand. Elasticity enables scalability, which means that the cloud can scale upward for peak demand and downward for lighter demand. The Scalability means that an application can scale when adding users and when application requirements increase. For systems, based on loads to the system, or usage of the system, the system's capabilities automatically can be scaled to satisfy the system requirement. An example is about Amazon’s S3 service that allows the customer to continue to add to the system, with a seemingly endless supply of storage.

Measured Service. This means that cloud computing is a usage-driven. Business units only pay for the computational resources they use. Resource usage is metered based on appropriate use of the technology and also reported based on metering to providing transparency for both the provider and consumer of the utilized service. Examples are measuring the storage, bandwidth, and computing resources consumed and charging for the number of active user accounts per month. Amazon EC2's users are charged based on size of server, operating system, and inbound/outbound data. In addition, Cloud Watch provides reporting features regarding the usage of EC2 instances.

2.1.2 Service Models

One main misconception is that cloud is one thing. However, cloud deforms into many shapes and designs. Each cloud computing provider has his own design and architecture that fits his business module. Cloud is a multi-dimensional layer and provides services generally at three different levels [9]: 1- Computing layer, 2- Business model layer, 3- Application domain layer. A company that uses software as a service provider is likely to have a different security and use cases than a company that would use an infrastructure as a service provider such as Amazon EC2. Figure 2.2 shows the different cloud layers, their targeted consumers, and the level of control provided.

Cloud Applications (Software as a Service SaaS). This form of service has been around for many years, and it is the most visible layer to the end users. SaaS leverages Web 2.0 in traits to help export the computational work from the users' terminal to data centers where the cloud applications are deployed [9]. Thus, it lessens the restrictions on the hardware requirements needed at the users' end. Moreover, providers of the cloud applications are able to upgrade and roll new features without disturbing the users with requests to install major updates or service packs. Web-portals, Google Apps [10], EBay, and web email providers are examples of SaaS providers.

Cloud Software Environment (Platform as a Service PaaS). This is the next layer up, where a cloud service provider offers a computing platform service which may include databases, programming tools, security software and other applications that make up a complete development platform. The provider offers such services along with a well-defined APIs that facilitate the interaction between the environments and the cloud applications. Cloud applications' developers utilize these services to implement their applications on the cloud. The advantage of PaaS is that the developer can buy a fully functional development and production environment.

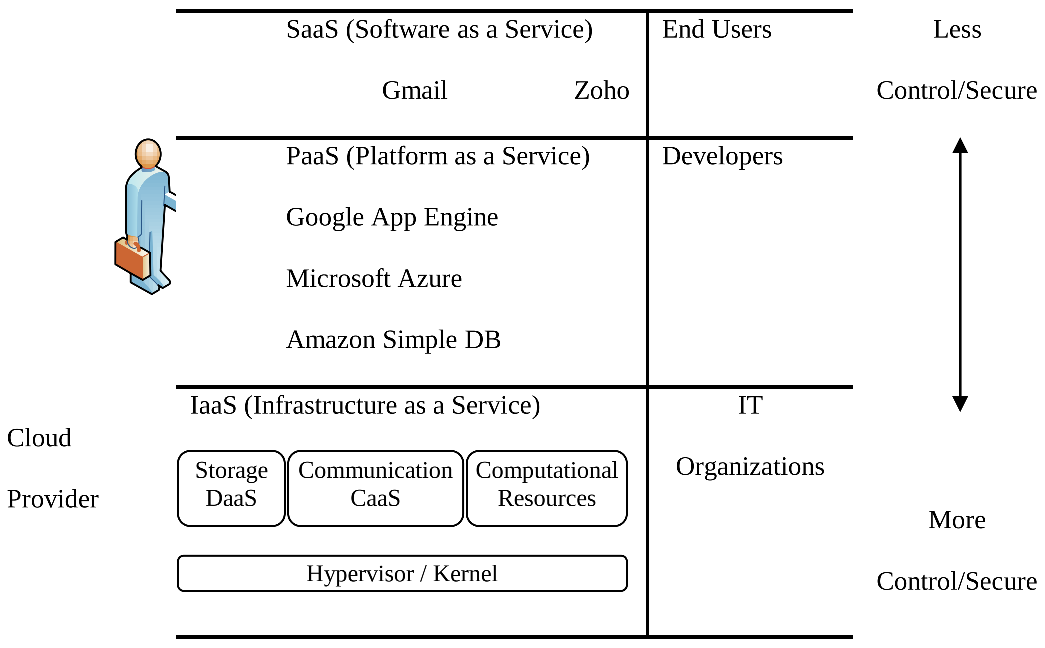


Figure 2.2: Cloud computing layers [11].

Cloud Software Infrastructure (Infrastructure as a Service IaaS). This layer is at the most basic level of the Cloud Computing offerings, and it provides virtual machines and other abstracted hardware and operating systems that may be controlled through a service API. Typical examples include Amazon EC2 (Elastic Cloud Computing) Service [12] and S3 (Simple Storage Service) [13], Terremark Enterprise Cloud, Windows Live Sky drive, and Rackspace Cloud. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls) [14].

2.1.3 Deployment Models

Public Clouds: resources including infrastructure, platform and software are available to the public through services that are usually accessible via internet. Amazon Web Services is an example of a company that provides cloud computing capabilities through a public cloud.

Private Clouds. resources are made only available to specific customers. It is typically hosted on the company's own servers, within their own network infrastructure and are completely isolated from the public internet. An example of this could is VMWare with the virtual servers hosting given applications.

Community Clouds. "The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations)" [3]. It may be managed by the member organizations, such as federated model, or a third party, such as brokered model [15].

Hybrid cloud. Resources in such cloud are offered by two or more clouds (private, community, or public). Such model may be used by a customer who desires the ease of use of a public cloud but desires some level of dedicated resources afforded by a private cloud. For example, an organization might use a public cloud service, such as Amazon Simple Storage Service (Amazon S3) for archived data but continue to maintain in-house storage for operational customer data.

2.2 Security of the Cloud

Because of its lower total cost of ownership, scalability, competitive differentiation, reduced complexity for customers, and faster and easier acquisition of services, the cloud computing tends to reduce Capital Expenditures (CapEx) and lower the Operational Expenditures (OpEx) costs [16,17]. For the CapEx, the cloud reduces the hardware costs in the IaaS since it is not required anymore to engineer the own data center to overcome many problems related to the resources' utilization; unlike the traditional IT where engineering own resources is mandatory to satisfy the peak performance cases. The "pay-as-you-go", as a key characteristic of cloud computing, lowers the operation expenses costs since the computing resources in a cloud environment are typically charged for on a fine-grained usage basis.

However, these cloud gained benefits are coming along with severe concerned issues raised by the Cloud Computing, especially regarding the security level provided by the Cloud [18]. According to a recent survey conducted by the International Data Corporation (IDC) [19], security ranked first as the greatest challenge of cloud computing as Figure 2.3 shows that about 87% of IT executives cited security as the top challenge preventing their adoption of the cloud services model. Security concerns have led organizations to hesitate to move critical resources to the cloud. Corporations and individuals are often concerned about how security and compliance integrity can be maintained in this new environment. In addition, moving critical applications and sensitive data to public and shared cloud environments is of great concern for corporations since they their data center’s network boundary defense is not on hand.

To understand why cloud computing security is ranked as the first concern, one has to highlight what changes in the Cloud that expose vulnerabilities to the cloud environment.

Normally, organizations physically own their servers and infrastructure, which are dedicated to serving the purposes of the individual organization. Thus, the traditional computing model can protect the dedicated organization's infrastructure by dividing physical and logical security zones. With cloud computing, the hardware is shared among hundreds, or thousands, or possibly millions of competing users and applications by means of the virtualization which raises many security concerns in the cloud [20].

One of such main concerns raised by the virtualization is the complexity of keeping track of security on two tiers: the physical host security and the virtual machine security [20]. More specifically, compromising the physical host server in the cloud will affect all of the virtual machines residing on that particular host server. Similarly, a compromised virtual machine might also be used to compromise the physical host server and then affecting all of the other virtual machines running on that same host.

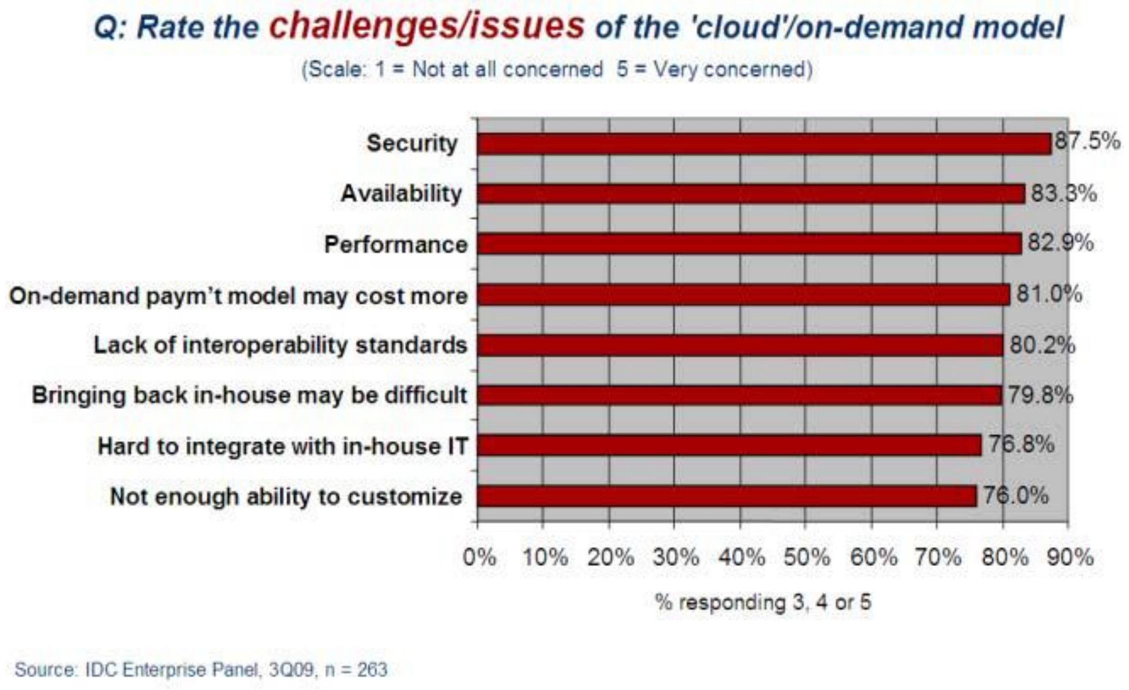


Figure 2.3: Results of IDC ranking security challenges [19]

In traditional IT environment, the data is stored in the local network and the organization has the full control on their sensitive data. However, in the cloud computing, there are many concerns regarding the stored data including data location [21], data sanitization, and the control over the remote data [22]. The users are concerned about the data location since they have no knowledge about exactly where their data is hosted, and what country it will be stored in [21]. Also, they are always concerned about their data is it safe from outsiders who should not have access to it [22]. The providers have to ensure that customer data is not exposed to unauthorized individuals.

In the cloud computing world, the user is required to transfer data throughout the cloud in order to access the computing power provided by the virtual environment. Consequently several security concerns arise including the confidentiality, integrity, and availability of data [23].

Moreover, the next-generation hackers are positioning themselves to take advantage of the eagerness shown by organizations wishing to move to the cloud, and are developing strategies and tactics to steal the organization's data from the cloud. Thus, organizations are sharing the cloud with the next-generation hackers, and the next-generation hackers are using the cloud to gain access to organizations' applications and data [24].

The strength of a chain lies in its weakest link; the same applies to cloud computing. A cloud's vulnerability comes from the components and configurations that compose the cloud. Any breach in any component could jeopardize the security of the whole cloud. Braking down the cloud and applying the best security practices to each and every component is the best bet for a secure cloud. The Cloud Security Alliance's report [25] presents 13 different security domains and the processes that need to be followed in an overall cloud deployment. We categorize the security concerns into four main categories based on the three cloud layers namely: SaaS, PaaS, IaaS; in addition to the special policies and procedures that should be taken in considerations when moving to the cloud architecture. Figure 2.4 shows the taxonomy of the cloud security. In the following section, we touch on the most common security threats in the different layers that compose the cloud environment.

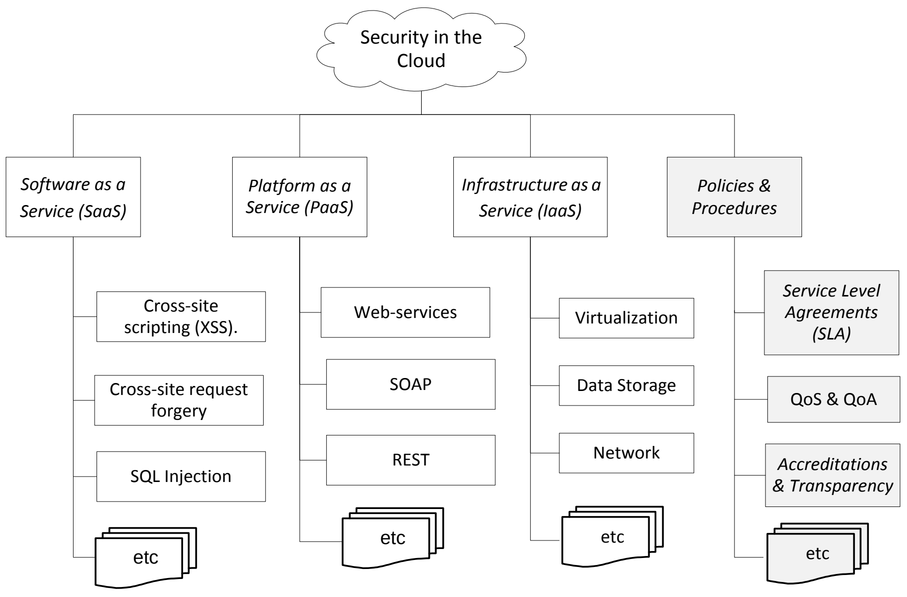

Security in the Cloud
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Policies & Procedures
Cross-site scripting (XSS).
Cross-site request forgery
SQL Injection
etc
Web-services
SOAP
REST
etc
Virtualization
Data Storage
Network
etc
Service Level Agreements (SLA)
QoS & QoA
Accreditations & Transparency
etc

Figure 2.4: Taxonomy of cloud security [10]

2.2.1 SaaS Security

Vulnerabilities could be Software-level, such as an SQL-injection or cross-site scripting vulnerability [26]. There is nothing new in the nature of these vulnerabilities and it is based on the application itself. Researchers have been looking at this level of vulnerabilities and came up with many ways to mitigate them. A summary of the most famous threats is as follows.

Cross-site scripting (XSS). In cross-site scripting (XSS), an attacker exploits the trust a web client (browser) has for a trusted server and executes injected script on the browser with the server's privileges. Web applications have XSS vulnerabilities because the validation they perform on un-trusted input does not suffice to prevent that input from invoking a browser's JavaScript interpreter, and this validation is particularly difficult to get right if it must admit some HTML mark-up. Effective checking algorithms provide an extensive evaluation that finds both known and unknown vulnerabilities in real-world web and are already demonstrated by Wassermann et al. [27].

Cross-site request forgery. Cross-Site Request Forgery (CSRF) is an attack against Web application users in which an adversary causes a victim's browser to perform an unwanted action on a trusted website via a malicious link or other content [28].

SQL injection. An SQL injection attack targets interactive web applications that employ database services. These applications accept user inputs and use them to form SQL statements at runtime. During an SQL injection attack, an attacker might provide malicious SQL query segments as user input which could result in a different database request. By using SQL injection attacks, an attacker could thus obtain and/or modify confidential/sensitive information. An attacker could even use SQL injection vulnerability as a rudimentary IP/Port scanner of the internal corporate network. Several papers in literature have proposed ways to prevent SQL injection attacks in the application layer by examining dynamic SQL query semantics at runtime [29].

2.2.2 PaaS Security

We can argue that most of the vulnerabilities mentioned in the SaaS layer, are also valid in the PaaS layer, such as the SQL-injection or cross-site scripting vulnerabilities. However, when talking about platform as a service, web-services come into the picture with two architecture styles using Representational State Transfer (REST) or Simple Object Access Protocol (SOAP).

XML Signature Element Wrapping. A well-known type of attacks on protocols using XML Signature for authentication or integrity protection [30]. This of course applies to Web Services and therefore also for Cloud Computing. A number of countermeasures as well as attacks circumventing these countermeasures have been published. However, mostly due to the rare usage of WS-Security in business applications, these attacks remained theoretical. However, in 2008 it was discovered that Amazon’s EC2 services were vulnerable to wrapping attacks [31]. Gruschka et al. [31] reviewed the available work in light of the discovered Amazon EC2 vulnerability and provided a practical guideline for achieving a robust and effective SOAP message security validation mechanism.

2.2.3 IaaS Security

Virtualizations. One of the security threats in this area is VM-level attacks. Potential vulnerabilities in the hypervisor or VM technology used by cloud vendors are potential problems. Vulnerabilities have appeared in VMWare [32] where it was possible for a program running in the guest to gain access to the host's complete file system and create or modify executable files in sensitive locations. Multiple vulnerabilities have also emerged in Xen [33], and Microsoft’s Virtual PC and Virtual Server [34]. Vendors mitigate potential VM-level vulnerabilities by watching guest-to-guest communication, turning off file sharing between guests and hosts, and setting strict firewalls.

Virtual-machine based rootkit (VMBR) is another type of malicious software that gains qualitatively more control over a system. VMBR works by installing a virtual-machine monitor underneath an existing operating system and hosts. King and Chen [35] demonstrated how attackers can gain a clear advantage over intrusion detection systems running in a target OS. Also, they showed how attackers can leverage this advantage to implement malicious services that are completely hidden from the target system and to enable easy development of general purpose malicious services.

Live Virtual Machine Migration is also one of the poorly explored areas in this field. An empirical study illustrated that a malicious party can exploit the latest versions of the popular Xen and VMware virtual machine monitors. Once it got exploited, an attacker could automate the manipulation of a guest operating system’s memory during a live virtual machine migration. Oberheide et al. [36] empirically demonstrated how two of the most popular and widely deployed VMMs, Xen and VMware are vulnerable to practical attacks targeting their live migration functionality.

Side-Channel Attacks arise from sharing physical infrastructure between mutually distrustful users, even when their actions are isolated through machine virtualization as within a third-party cloud compute. Ristenpart et al. [37] demonstrated the risk of such attacks and suggests a number of approaches for mitigating such risk.

Data Storage. Several issues arise when dealing with data storage in the cloud, and are stated as follows.

Physical location. Since customers will not know where their data will be stored, it is important that the Cloud provider commit to storing and processing data in specific jurisdictions and to obey local privacy requirements on behalf of the customer.

Customer isolation and segregation. Data and services need to be isolated for other customers that use the same service. In many cases, it is usually the application logic that provides this separation. Especially as you look in storage, PaaS, or SaaS, they often share the same disk; so application logic becomes critical in enforcing this segmentation.

Encryption and key management. Key management is an important challenge facing IT in general, as well as cloud computing specifically. Would you manage your keys or hand them over to your provider? What would be the case when there are multiple providers, multiple services and workloads?

Many of the data security techniques are mature and widely used today, but many organizations are not using them in operations and especially at the scale for cloud kind of operation. Examples of techniques already available for key management:

1. Confidentiality - Encryption, and sanitization.
2. Integrity - Digital fingerprint and signatures.
3. Authenticity – Digital signature.
4. Availability –Localized Knowledge Spillovers (LKS), P2P, Multi-Cloud
5. Accountability – Independent Accountability Mechanism (IAM), Audit Trails

Wang et al. [1] investigated the problem of data security in cloud data storage and proposed a distributed scheme with explicit dynamic data support, including block update, delete, and append. The scheme relies on an erasure correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure-coded data, the scheme achieves the integration of storage correctness insurance and data error localization.

Network. The areas of network security, and encryption and authentication techniques are fairly mature. And with cloud computing, network security is no different. However, it becomes vital to use existing mature techniques to secure the Cloud. The CSA Cloud Security Alliance [25] describes in details the common techniques, e.g., the use of VPN with standard network encryption such as SSL, digital certificates, and Kerberos are well known techniques that could be utilized.

2.2.4 Policies and Procedures

Despite the advancements in computer cryptography, the clouds will probably never be secure just by applying technologies; policies and standards are critical variables that must be acknowledged. QoS, SLA agreements with the provider, transparency, certifications prerequisites ISO 27001, ISO 27002, and SAS 70 Type II should all be considered when evaluating cloud providers.

Service Level Agreements (SLA). A service level agreement is a document which defines the relationship between two parties: the provider and the recipient. The SLA has to describe different levels of security and their complexity based on the services to make the customer understand the security policies that are being implemented. There has to be a standardized way to prepare the SLA irrespective to the providers. Kandukuri et al. [38] put forward some security issues that have to be included in the SLA.

Quality of Service QoS. As a cloud must provide services to many users at the same time, and different users have different QoS requirements, the scheduling strategy should be developed for multiple workflows with different QoS requirements. Xu et al. [39] have looked into this and introduced a Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW) to address this problem.

2.3 Cloud Computing Attacks

In this section, we present a taxonomy that outlines the existing attacks targeting the cloud computing system by utilizing the vulnerabilities of its architecture as a group of technologies. Attacks that target the technologies that constitute the cloud computing system like grid computing, virtualization, and utility computing still could target the cloud computing system. Similarly, insecure applications that run in the cloud are identical to insecure applications that run on standalone, dedicated servers [6]. Issues such as buffer overflows, SQL injection, cross-site scripting, and other common application-level vulnerabilities still be presented in the cloud applications.

However, the cloud architecture and applications running in the cloud also bring up a new set of security concerns. The taxonomy here presents the new attacks raised because of the cloud computing system and targeting the cloud computing system. It is worth to note that some of such attack might be similar to the traditional ones with some modification in their pattern, purpose, or tactics.

Figure 2.5 shows the presented taxonomy which classifies the cloud computing attacks based on the compromised surface in the cloud including the virtualization-based attacks, the network-based attacks, and the service-based attacks.

2.3.1 Virtualization-Based Attacks

The virtualization as it was described earlier in section 2.2.3 is the most common form for providing computational resources to the cloud users [40]. Vax [42] discusses the virtualization security risks and the cloud computing issues. The report discusses the benefits gained by the virtualization in the cloud computing in addition to the risks showed up because the virtualization. The author concluded that compromising the hosting virtualization platform puts all of the virtual guests at risk and hence introducing a new set of identity, access, and user activity reporting problems.

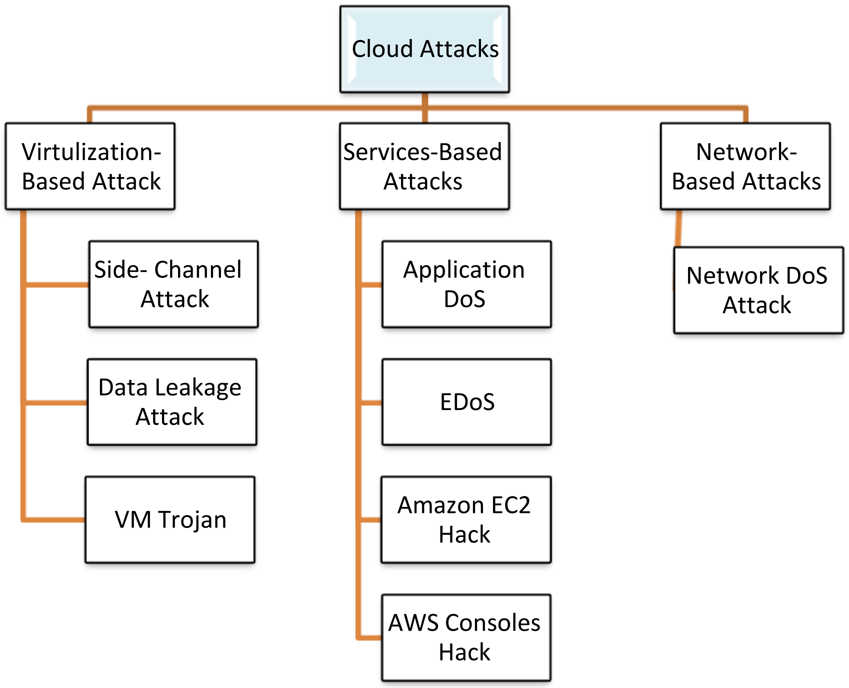


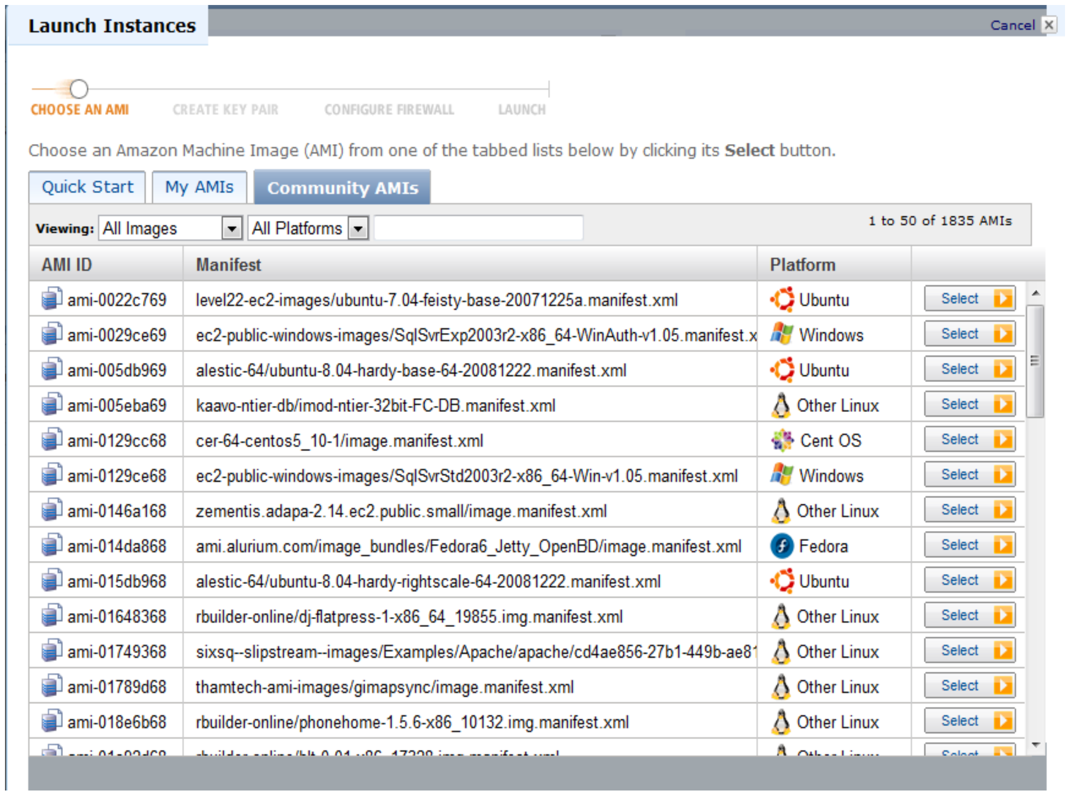
Figure 2.5: Taxonomy of cloud attacks

The question is why the virtualization could pose the cloud or expose it to some threats. Two main issues must be considered about the adoption of the virtualization technology in the cloud computing system. The first is about protecting an individual virtual machine from tampering and the second is about the configuration and the source of the virtual machines (VM).

In the context of virtualization, a single hardware can be partitioned into multiple VMs and each partition must be protected from the attacks otherwise there are much possibilities to compromise the whole hosting system. In addition, virtual machines add an additional layer of software that could become a single-point of failure [36].

The configuration and the source of the VM are another big concern that could poses the cloud [41]. It is important to know who configured the virtual machine and where the virtual machine came from. To sign up for a cloud service with Amazon's EC, one has to select an Amazon-configured Amazon Machine Image (AMI) from a pool of community-shared AMIs, or upload its own AMI.

Figure 2.6 shows the various community AMIs available on Amazon's website [12]. These AMIs are created, configured, and uploaded by other Amazon EC2 users. Even Amazon trust those creator, still there is a possibility that a well-intentioned AMI creator could have introduced security issues by inserting hidden malicious logic such as installing an outdated/insecure library or software package, altering the security configuration/setting, enabling an inherently insecure service (e.g., Telnet), introducing inadvertent application-level security issues, or reusing cryptographic secrets (private keys).



Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Viewing: All Images | All Platforms

1 to 50 of 1835 AMIs

AMI ID	Manifest	Platform	
ami-0022c769	level22-ec2-images/ubuntu-7.04-feisty-base-20071225a.manifest.xml	Ubuntu	Select
ami-0029ce69	ec2-public-windows-images/SqlSvrExp2003r2-x86_64-WinAuth-v1.05.manifest.x	Windows	Select
ami-005db969	alestic-64/ubuntu-8.04-hardy-base-64-20081222.manifest.xml	Ubuntu	Select
ami-005eba69	kaavo-ntier-db/imod-ntier-32bit-FC-DB.manifest.xml	Other Linux	Select
ami-0129cc68	cer-64-centos5_10-1/image.manifest.xml	Cent OS	Select
ami-0129ce68	ec2-public-windows-images/SqlSvrStd2003r2-x86_64-Win-v1.05.manifest.xml	Windows	Select
ami-0146a168	zementis.adapa-2.14.ec2.public.small/image.manifest.xml	Other Linux	Select
ami-014da868	ami.alurium.com/image_bundles/Fedora6_Jetty_OpenBD/image.manifest.xml	Fedora	Select
ami-015db968	alestic-64/ubuntu-8.04-hardy-rightscale-64-20081222.manifest.xml	Ubuntu	Select
ami-01648368	rbuilder-online/dj-flatpress-1-x86_64_19855.img.manifest.xml	Other Linux	Select
ami-01749368	sixsq--slipstream--images/Examples/Apache/apache/cd4ae856-27b1-449b-ae81	Other Linux	Select
ami-01789d68	thamtech-ami-images/gimapsync/image.manifest.xml	Other Linux	Select
ami-018e6b68	rbuilder-online/phonehome-1.5.6-x86_10132.img.manifest.xml	Other Linux	Select

Figure 2.6: Community AMIs available on Amazon’s website [12]

According to Computerworld, a side-channel attacker looks at indirect information related to the computer, such as the electromagnetic emanations from screens or keyboards to determine what is going on in the machine.

Recently, a report [37] conducted by researchers at the University of California-San Diego and the Massachusetts Institute of Technology shows that an attacker could attempt to compromise the cloud by placing a malicious virtual machine in close proximity to a target cloud server and then launching a side channel attack. Authors show that it is practical to hire and lunch additional VMs to get a high chance of co-residence with the target VM. The researchers argued that a VM resident on the same physical server as a target VM could monitor shared resources on the server to extract information about the target VM in the same machine.

Cross-VM Side Channel Attacks. The report specifically concentrated on the case of Amazon's EC2 in which they were able to locate the physical server and then gathered data by placing their own software there and launching a side-channel attack. The discussed side-channel attacks include measuring cache usage and CPU utilization on the physical machine, detecting co-residence by gathering information about the load without relying on sending any network probes, estimating traffic rates which can be used to deduce targets activity patterns, and keystroke timing attack.

Data Leakage. In context of virtualizations, one of the key security issues is the data leakage which caused by multiple tenants sharing physical resources. There are some vulnerabilities of VM that could lead to such dangerous security issue. One area of such vulnerabilities is that while the VM, using a virtual file system, is in an inactive state, the

VM is susceptible to data modification [43]. By manipulating the virtual file system, an attacker could modify the configuration of the VM and then alter the data stored on the VM.

Another type of vulnerabilities is related to the virtual storage in the cloud. The client data is stored in virtual block devices that accessed by the CPU/RAM of each cloud server. Thus, when a customer deletes their relative drive and then a new customer creates a new drive, there is a chance for the new customer to try and image off the previously written data by other customers [44].

Virtual Machine Trojan. Castro [45] has released an open source Virtual Machine Trojan VMT called ViMTruder. According to Castro such types of Trojan comes embedded within a virtual machine downloaded from the internet. The objective of VMT is similar to the one of the normal Trojan which is to remotely take control of the machine for malicious purposes. However, in a virtual environment like cloud computing system, such Trojan has access to the local network at which the host machine is resident. Thus a VMT can lunch several types of attack differ than the normal Trojan including sniffing, scanning services and machines in the local network, exploiting the local network and the services there such as *ftp* and *ssh*, and of course lunching DoS attacks against the local network, or external hosts.

2.3.2 Service-Based Attacks

This kind of attacks concentrate on those are utilizing or targeting the cloud-based services such as Amazon EC2’s cloud-based services, Google App services, Amazon S3 services or any other cloud web-based services. The attacks use the vulnerabilities of the cloud services to poison the cloud or the users running a cloud service.

***Application* DDoS Attacks.** As the cloud computing infrastructure is shared by millions of users, DDOS attacks have much greater impact than against single tenanted architectures, argued in [46, 47].

The common way that such attack could uses to denial a service is to saturate the target machine with external communications requests. As a result, the target system cannot respond to legitimate traffic at all or in some cases it responds so slowly at which the service is considered unavailable. Here we will focus only on those attacks utilizing the cloud computing services to lunch the DoS or DDoS.

Amazon EC2 Hack. Gruschka and Iacono [31] discussed how the cloud like Amazon EC2 is vulnerable due to a Signature Wrapping Attack described in [30]. They used forged SOAP body along with XML Re-Writing attack to manipulate Amazon EC2 request. In The XML Re-Writing attack, additional nodes are injected to replace signed nodes at different level in XML tree such that it results in successful signature verification [31]. Utilizing this type of attack to modify an eavesdropped EC2 request enables an attacker to execute arbitrary own machine commands on behalf of a legitimate cloud user. Once the attacker be able to execute such hidden commands, it is possible to perform a DoS that targeting the availability of the users services or to create a botnet that targeting the cloud's economy by charging the EC2 costs on the user’s bill which is described later in this section as EDoS.

EDoS. This is an application DoS targeting a hosted service in the cloud. It was reported by Hoff [48]. The aim of such attack is to make the cloud unsustainable by fading the cloud billing mechanism to charge the cloud user's bill for the attack's activities.

In [49], the authors discuss some of the research challenges regarding the cloud computing including technological issues of data centers and some other security issues that cloud computing raises. One of these discussed concerns is the DoS attack that target a cloud application and by which the user end-up with paying for the attack through their increased resource usage. In fact, Jesper [8] recently reported this scenario with their application running on Amazon’s AWS cloud without mentioning whether they got charged for the usage generated or Amazon waived the extra costs incurred as a result of the attack. The details about such attack are presented in Section 5.

AWS Management Consoles Hack. Amazon Web Services (AWS) is the most widely used method for controlling and managing the AMIs. This helps users manage their EC2 instances easily. EC2 users have to be authenticated to AWS console which is used to administer and manage the various AMIs running in a user’s account. The three most common methods of authentication [50] are a username/password combination, an Access Key ID/Secret Access Key combination, and X.509 certificates.

The web management console asks the user to provide the user name and password via a login page hosted in Amazon.com domain. Such web pages are susceptible to the web application vulnerabilities by which the attacker could gain access to the user's session [51]. Once the attacker gets accessed the user's session, he could do many malicious activities against the user account. One of such possible activities, rebated to the DoS, is that the attacker can force the generation of a new access key for the EC2 user's session making the old key obsolete and no longer be used to authenticate to the application. In such case, the administrator must update all the applications to use the newly generated key and thus the attacker doing a temporary denial of service.

2.3.3 Network-Based Attacks

Here in this category, we focus on those attacks dealing with the network elements in the cloud computing system including links, traffic, routers, and network protocols.

Network DoS. What are new regarding the Flooding attacks in the cloud computing environment is that such attacks are utilizing the vulnerabilities along with the properties of the cloud to launch huge flooding messages that aim at resource exhaustion as it was reported in [16] that the attacker may use a cloud for sending his flooding messages.

In [52], Liu describes a new form of DoS by which exploits the network under-provisioning in a cloud infrastructure. First, the attacker starts by gaining access to a set of hosts by launching VM using the cloud API. The authors have stated that by exploiting the VM assignment algorithm, it is possible to launch enough hosts in a subnet in a single trial. Then the attacker has to derive the topology which can be done by choosing one host as the sink and the rest of hosts as sources and then sending a sequence of packets back-to-back to the sink at the same time. The topology could be learned from the sink host's perspective. After that, the attacker has to identify a bottleneck link in the network to be attacked. Finally, the attacker transmits enough traffic from a few hosts in the subnet to hosts in other subnets in order to saturate the uplink.

2.4 Economic Denial of Sustainability attack (EDoS)

In this section, we present details about the EDoS starting by differentiating the EDoS from DoS. Then, we define the EDoS along with the ways that could be used by the hackers to compromise victim machines in the cloud computing environment and lunch the EDoS attack. Finally, the existing migration techniques will be discussed in this Section.

2.4.1 From DoS to EDoS

Distributed Denial of Service (DDoS) and DoS attacks are blunt force shock attacks [53]. The goal, regardless of motive, is to overwhelm the targeted victim's infrastructure making its offered services unavailable. The key behind the difficulty of dealing with such attacks is that their requests have to be processed by the targeted service in order to determine its invalidity resulting in a certain amount of workload per attack request. For example, the attacker could activate distributed botnets to lunch asynchronous fake requests targeting a victim website and thus making it unavailable due to an exhaustion of its resources handling such requests [54].

However, cloud computing allows us to scale our servers to high orders, to serve a large number of requests for service. The introduction of resource-rich cloud computing platforms, where adopters are charged based on the usage of the cloud’s resources, known as “pay-as-you-use” or utility computing, has transformed the DDoS attack problem in the cloud to a financial one. This new type of attack targets the cloud adopter’s economic resources, and is referred to as Economic Denial of Sustainability (EDoS) attack [48]. In other words, the attack is making the cloud unsustainable by fading the cloud billing mechanism to charge the cloud user's bill for the attack's activities.

Unlike conventional DDoS attacks, EDoS attacks target the financial constraint of an organization, not its physical network/server constraints. A well-known tactic taken by EDoS attacks is to remotely control zombies to smoothly (with low rate to avoid triggering security alarms) flood a targeted cloud service by undesired requests. Zombies are compromised nodes attached to the Internet including computers infected by way of viruses/worms/malware and, in some cases, powerful unprotected servers. As a result of such undesired requests, and because of the cloud elasticity notion, the service usage will be scaled up to satisfy the on-demand requests. And because of the “pay per use” notion, a cloud adopter’s bill will be charged for those undesired requests, leading to service withdrawal or bankruptcy. For example, a company taps into Amazon EC2 or Google App Engine for their application, and pays for the computing and bandwidth costs based on its usage. Depending on the traffic, the service usage can be scaled up or scaled down. Attackers could send malformed requests blended with legitimate ones to the applications running on these services. Since these requests are consuming the computing resources including inbound and outbound bandwidth traffic, and the CPU cycles for processing such requests, the applications vendor’s cloud bill gets charged an excessive amount of expenditure. Unless the application vendor or the cloud providers have a smart technique to stop such risk, a sustained attack like this could ramp up the applications vendor’s cloud infrastructure bill.

Like DoS attacks, EDoS attacks can potentially disrupt cloud services for an extended period resulting in poor user experience and service-level impacts. However, an EDoS attack has its goal to drain a company’s cloud services budget. EDoS attacks on pay-as-you-go cloud applications increase the usage of network bandwidth, CPU, and storage consumption under the application provider's account [55]. And, this results in a dramatic increase in a company’s cloud utility bill. What makes this more disastrous is that it is extremely difficult to selectively filter the malicious traffic without impacting the service as a whole. This also means that any proposed mitigating technique must be highly intelligent; otherwise, the technique itself could be utilized by the attackers as a source of DoS or EDoS attacks.

2.4.2 EDoS Classification

Based on the way that could be used maliciously to utilize the cloud customer's resources in order to drive up costs for the customer, EDoS could be classified into internal and external attacks.

The internal attack scenario is based on the vulnerabilities related to the virtualization in the provider’s infrastructure in which the hijacked accounts can be used to stage internal EDoS attacks within the cloud provider’s infrastructure in order to damage the customer economically. Ristenpart et al. [37] discuss some cloud's vulnerabilities that could be used to compromise an AMI. Once an AMI get compromised, many malicious activities could be done. For example, a CSRF attack could launch a huge number of instances of an AMI attacking other cloud applications under victim’s EC2 account. Because of the rapid elasticity as a property of the cloud computing, the attacked AMIs will scale up by create new instances to satisfy the increased load. In such scenario, the victim is paying for the computing and networking resources used by both attacking AMIs and attacked AMIs.

In the external attack scenario, an attacker uses a public channel to use up the customer's metered resources. EDoS in such scenario can utilize distributed controlled zombies as well as a single entity to smoothly flood a targeted cloud service by undesired web requests. The customer cloud adopter’s bill will be charged for those undesired requests, leading to a denial of sustainability [20, 56]. For example, an attacker could target a botnet to visit a website that deal with ecommerce purchases in order to lunch legitimate requests but of course without making purchases. Those requests will arrive to the queues of the service located in the cloud eating some computational resources and thus the vendor has to pay the cloud provider for the use of these resources [48].

2.4.3 EDoS Mechanism

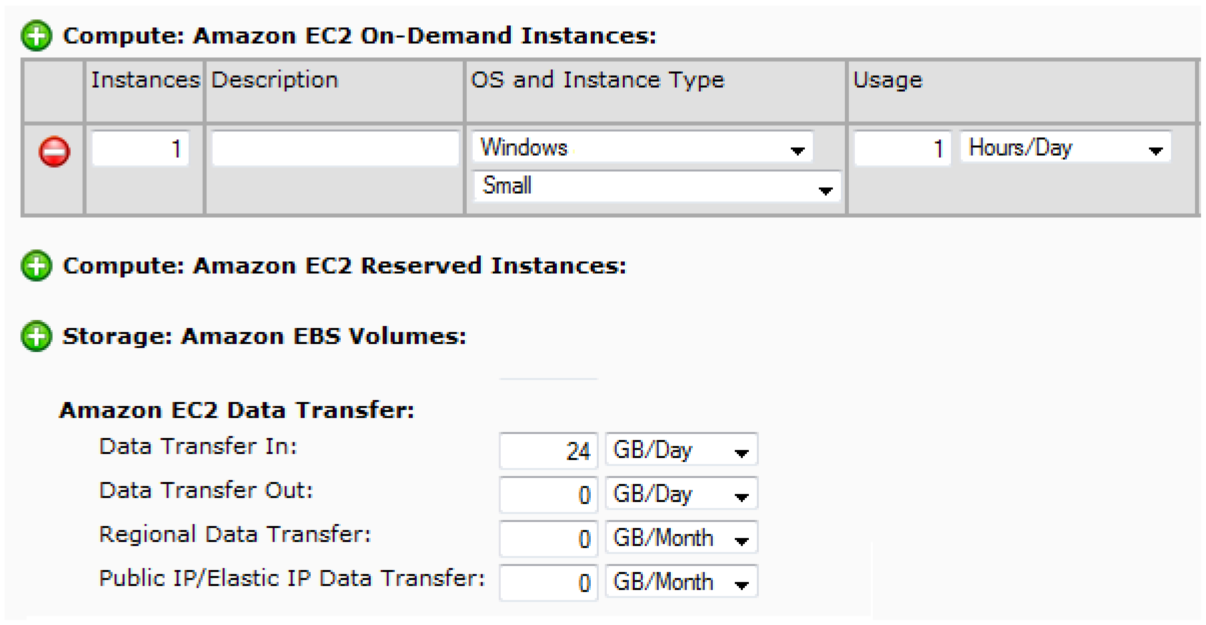
Regardless of the type of EDoS attack, the attack target the cloud's billing mechanism to charge the vendor for the attack's activities. One of the main important notions of the cloud is its Measured Service by which the metered billing allows the cloud provider to charge based on computing resources consumed by the customer. This enables companies to turn indirect costs into that of variable costs or operational costs by which the companies could satisfy the growth of the business and optimize margins.

The cloud computing service providers apply a utility computing with specific billing details which vary among the providers. However, Leong [57] presented the most common general pricing models which include per-instance charge, capacity pool charge, and resource charge. In a per-instance charge model, the customer has to pay for the provisioned instances by the hour or by the month regardless of the amount of used resources. In a capacity pool charge model, the providers charge the customer by the hour or by the month for the overall amount of resources like CPU power and RAM instead of provisioned instances. The payment in such model varies depending on the providers. For example, the customer may pay for the actual using or for the guaranteed capacity. The third model, a resource charge, is a combination of the previous models in which the customer has to pay by the unit of usage for abstracted measures of capacity.

Considering that most cloud providers base their billing rates on CPU and bandwidth consumption, and also that EDoS attack can last for an extended time [58], scaling to meet the demands of the attack can be extremely costly. To figure out how much does this economically affect the cloud user, Figure 2.7 and 2.8 Show an example of the Amazon EC2 pricing calculator offered by Amazon [59] as an estimator of the customer monthly bill for using its EC2 services.

Figure 2.7 shows a specific expected usage of one instance of AMI where a windows as an operating system is required for such an on-demand instance which only used for one hour per day. As well as, the amount of network data transfer expected into the user's Amazon EC2 account from outside Internet in a month is assumed to be 1 GB/h. Figure 2.8 shows the estimated monthly payment for the given expected usage shown in Figure 2.7. The estimated cost is about $4 for the computation and about $70 for the bandwidth, all per month. With a little calculation, one can get that the cost per hour is about $0.1 for the computation (one instance) and about $2 for the network bandwidth (1GB).

To show how much impact EDoS has on the cloud user bill, we present a scenario given by Armbrust et al. [60], but with the recent prices reported in Amazon for small on-demand EC2 instances running on a Windows operating system [81]. Assuming an EC2 instance that could handle 500 bots; this instance costs$0.1 per hour. If an attacker succeeds in controlling and having 500000 bots generate an extra 1 GB/s of bogus network bandwidth, it will cost the victim an extra $100 per hour for computation (1,000 instances) and an extra $720 per hour for the network bandwidth. Therefore, the attack would cost the victim about $10,000 if it lasts 12 hours. In fact, when creating an Amazon account for launching EC2 instances, the maximum number of allowed instances for an account is 20 instances by default. However, one can increase the quota upon sending a request to Amazon especially when the application is expected to have a heavy workload as it is the case with most of the popular websites. The above example shows that, although one of the most important capabilities of cloud-based offerings is their ability to scale to meet abnormal spikes in traffic and load, this ability can have disastrous impact on the owner of the applications in the cloud.



Compute: Amazon EC2 On-Demand Instances:

	Instances	Description	OS and Instance Type	Usage
	1		Windows / Small	1 Hours/Day

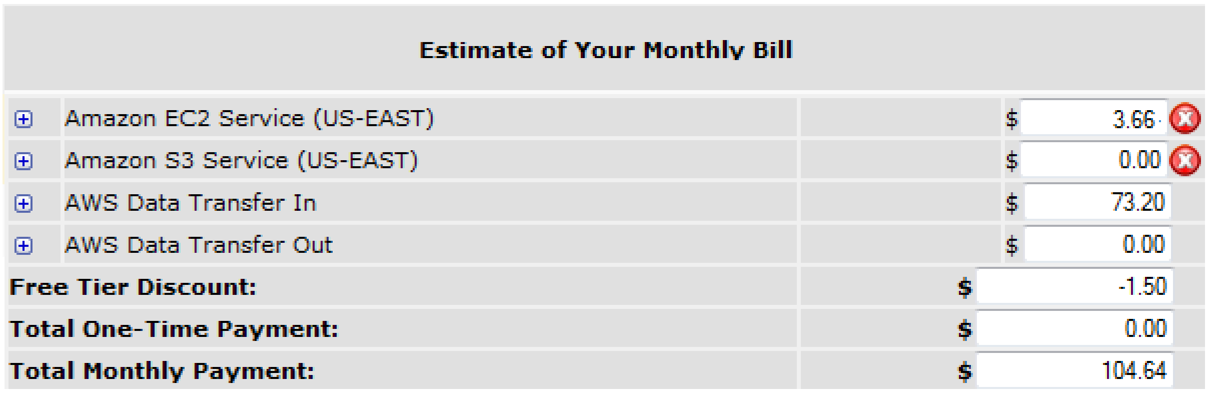
Compute: Amazon EC2 Reserved Instances:

Storage: Amazon EBS Volumes:

Amazon EC2 Data Transfer:

Data Transfer In:	24	GB/Day
Data Transfer Out:	0	GB/Day
Regional Data Transfer:	0	GB/Month
Public IP/Elastic IP Data Transfer:	0	GB/Month

Figure 2.7: Amazon EC2 calculator



Estimate of Your Monthly Bill

Amazon EC2 Service (US-EAST)	$ 3.66
Amazon S3 Service (US-EAST)	$ 0.00
AWS Data Transfer In	$ 73.20
AWS Data Transfer Out	$ 0.00
Free Tier Discount:	**$** -1.50
Total One-Time Payment:	**$** 0.00
Total Monthly Payment:	**$** 104.64

Figure 2.8: Amazon EC2 monthly bill estimator

2.4.4 EDoS Attack Mitigation Techniques

Mitigating EDoS is one such approach for protecting the cloud infrastructure against the rippling effect of cost incurred on legitimate users through EDoS attacks. Not much work has been done in the past to address this ever-important issue of concern. In fact, there is some work that analyzes if the existing network-level and application-level DDoS mitigation mechanisms can be adopted to handle EDoS on-demand, by deploying these solutions on cloud platforms. Here, we discuss some of these approaches based on the classification presented above.

2.4.4.1 Internal Based EDoS Mitigation Techniques

Ristenpart et al. [37] argue that fundamental risks arise from sharing a physical infrastructure between mutually distrustful users, even when their actions are isolated through machine virtualization, as is the case within a third-party cloud compute service. They discussed a number of general approaches for mitigating this risk. One approach focuses on the internal structure of cloud providers' services and their placement policy to be more obfuscated in order to make exploiting placement more difficult. However, as they argued, such approaches might only slow down, and not entirely stop, a dedicated attacker. Another approach is to inhibit the side-channel attacks by employing blinding techniques to minimize the information leakages. However, there are many channels' vulnerabilities that must be considered like cache-based side, network access side, CPU branch predictors, etc. Therefore, for such approach to be practical, it requires to anticipate all possible side-channels. Ultimately, to mitigate the co-residence problem in a virtualization environment such as a cloud computing system, they proposed that the provider, e.g., Amazon EC2, has to patch all placement vulnerabilities and this can simply be done by allowing the users to decide about the placement of their virtual machines.

2.4.4.2 External Based EDoS Mitigation Techniques

Analytical Based Approach. Singh and Kumar [61] proposed an analytical approach to address the DDoS attack problem. Their approach is based on mathematical equations to compute the number of malicious packets malicious. They also propose an algorithm which is a refined method of the traditional hop count inspection mechanism, to mitigate the effect of these identified malicious packets on the underlying networking infrastructure. These malicious packets accompany legitimate data from the attacker’s side, and can cause significantly degrade network performance.

Cloud Based ID Approach. Bakshi and Y. Dujodwala [20] discussed a countermeasure to secure the cloud from DDoS Attacks using an Intrusion Detection System (IDS), installed on a virtual switch. The idea is to log the network traffic into a database where the IDS is used to examine the packets for a specific attack types based on predefined rules that define the pattern of specified attacks. The IDS could determine the nature of the attack, and is capable of notifying the virtual server of the extent of security risk involved.

Auto Scaling Based Approach. Rodero et al. [62] discussed some of the limitations related to the interfaces offered by the providers to their clients in order to manage the cloud services. The current clouds offer interfaces too close to that infrastructure which means that the clients cannot manage the services based on services' status. Thus, the authors proposed to cite a new abstraction layer closer to the lifecycle of services that allows automatic deployment and escalation depending on the service status. Such layer would allow the configuration of certain business rules like making limits on the maximum expenses, the Service Providers is willing to pay, so that the business does not go bankrupt due for example to EDoS attacks.

One of the control technologies that will reduce the effects of the EDoS is the Auto Scaling technique, enabled by Amazon CloudWatch, which is a web service that provides monitoring for AWS cloud resources. Amazon clients will be able to define boundaries that would limit the elasticity of their cloud platforms and thus reducing the effect of the EDoS. However, this cannot be considered as a practical solution for the EDoS attack since the attack could still charge the user account even to some extent defined by the boundaries.

Proof of Work. Some protocols are asymmetric as they consume more resources on the server side than on the client side. These protocols can be misused for denial of service attacks. The attacker generates many service requests and overwhelms the server's resources. If the protocol is such that the resources are released after a certain period of time, the attacker simply repeats the attack to keep the server's resources constantly occupied. One approach to protect against attacks on such asymmetric protocols is to redesign the protocols by introducing another asymmetric step before committing the server's resources. In such approach, the server requires a proof of work from the client, before committing its resources to the client.

HinKhor and A. Nakao [56] described a self-verifying proof of work, sPoW, which is a unilaterally deployable “pay-as-you-use” cloud-based EDoS mitigation mechanism. The proposed scheme offers network-level and application-level EDoS protection to servers deployed in clouds, or DDoS protection to servers in general (outside a cloud). The strategy is to employ a system that enables packets' origins to compete with each other for the servers' resources by expending their own resources to generate and embed a "signal" within their headers. Figure 2.9 shows the steps involved in such approach summarized as follows.

1- A client performs a DNS resolution to obtain the location of the client plug-in on the content distribution networks (CDN), (1) and (2).
2- The client plug-in then performs a puzzle request to the sPoW name resolution specifying the server hostname to obtain the crypto-puzzle for the destination server (3).
3- The sPoW name server forwards the puzzle request to the server plug-in, which in turn, generates a puzzle for that domain name; creates a temporary encrypted server channel; sends back both the encrypted details as well as the encryption key as the crypto-puzzle. (4), (5), and (6).
4- The client plug-in recovers the server channel details and submits an initial connection request through that server channel. The request includes a randomly generated shared key encrypted using the server's public key. (7).
5- The server then uses the client generated shared key to create an encrypted communication channel and sends the information back to the client plug-in (10).
6- Finally, the communication between the client plug in and the server plug-in can proceed using the established communication channel.

One limitation of this work is the cost for changing queue names which is done frequently. With a sharp increase of the cost, sPoW is no longer deployable. The sPoW relies on the Amazon Simple Queue Service (Amazon SQS) to drop the EDoS traffic without charging the cloud adopter since the messages are targeted at queue name not at the cloud adopter as a destination.

The above techniques are very limited in their scope for providing a robust and effective platform to protect the cloud against the risks associated with EDoS attacks. In addition, variant forms of such attacks have not been identified, and may pose very strong threats to cloud-based services in the near future. We intend to identify, formalize, design and test several techniques for protecting the cloud against such attacks, as part of this work.

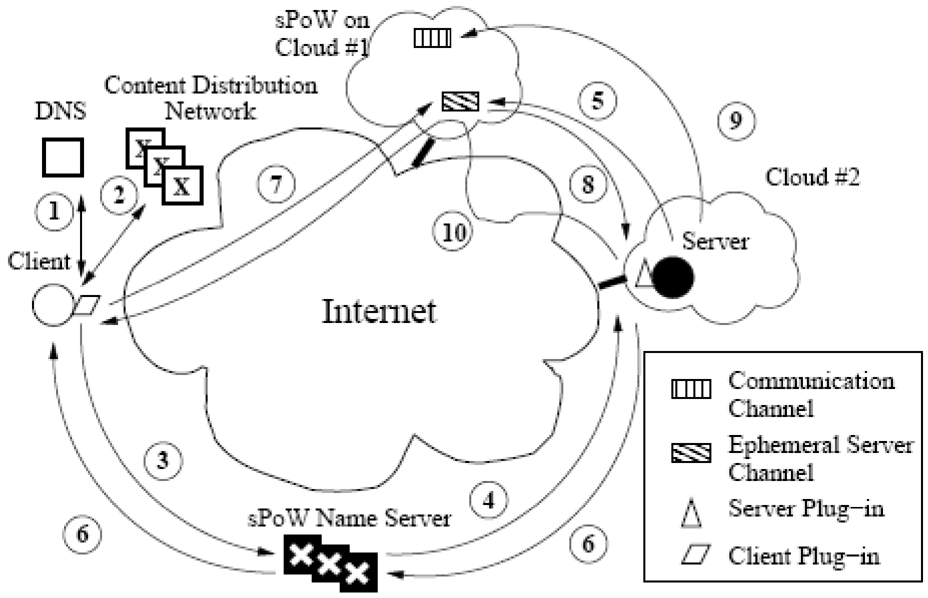


Figure 2.9: A walkthrough of sPoW. [56]

CHAPTER 3

ANALYTICAL AND SIMULATION MODELS TO EVALUATE THE IMPACT OF EDOS ATTACK AGAINST CLOUD COMPUTING SERVICES

3.1 Introduction

The introduction of resource-rich cloud computing platforms, where adopters are charged based on the usage of the cloud’s resources, known as “pay-as-you-use” or utility computing, has transformed the Distributed Denial of Service (DDoS) attack problem in the cloud to a financial one. This new type of attacks targets the cloud adopter’s economic resources, and is referred to as Economic Denial of Sustainability (EDoS) attack [48]. In other words, the attack is making the cloud unsustainable by having the cloud user pay for the attack's activities.

In this chapter, we present an analytical model to study the impact of EDoS attacks on a Cloud service. The model considers a number of performance metrics. These metrics include end-to-end response time, utilization of computing resources being consumed, and the incurred cost resulting from the attack. Such model is convenient to show the impact of EDoS attack on both the performance and cost of the cloud computing services. Although we concentrate on modeling the EDoS attack against cloud computing, the proposed model is also suitable for other similarly behaving attacks such as DDoS.

The rest of the chapter is organized as follows. In Section 3.2, we discuss the related works and in Section 3.3 we present the EDoS attack. Section 3.4 presents the proposed analytical model for a cloud service and Section 3.5 presents the proposed analytical model for the cloud service under EDoS attack. In Section 3.6, we present the provisioning technique parameters tuning. The experimental setup including the simulation model and results are discussed in Sections 3.7 and 3.8. Finally, the conclusion and the future work are presented in Section 3.9.

3.2 Related work

There are several proposed works in the literature to model a service exposed by the cloud computing.

Xiong et al. [84] have proposed a queuing theory based method for studying the performance of a cloud service in terms of the response time. In their model, the cloud computing service has been modeled as a tandem of two *M/M/1* queues representing the Web server and the service center for single-class customers.

Chen and Li [63] have proposed a queuing-based model for dynamically provisioning cloud computing virtual machines to meet the service level agreement (SLA). They have proposed using *M/M/S/k* queuing model to capture the architecture of the web application in the cloud computing. In their work, they have considered the web application as a centralized queue and the virtual machines as service centers with finite caches and buffers. Similarly, Hu et al. [86] have proposed modeling computing resources in the cloud as a multi-server queuing model, *M/M/S*. The purpose of such model is to guide resource allocation decisions in terms of the minimum number of servers that should be allocated to each application environments to meet the SLA.

Bi et al. [87] have proposed a dynamic provisioning model for virtualized multi-tier applications in Cloud data centers based on open queuing networks. The virtualized multi-tier application in cloud computing is deployed on multiple VMs for each tier that provides certain functionality to its preceding tier (e.g., web server, database). They constructed a hybrid model that represents the scheduling tier as an *M/M/s* queuing system, and multiple *M/M/1* queuing systems for the other tiers.

Shi et al. [92] have developed efficient energy saving methods in the cloud datacenter by dynamically allocating resources based on utilization analysis and prediction. The main prediction scheme that has been used in their work was an *M/M/1* queuing model that captures the Cloud-based web service. Similarly, Calheiros et al. [65] have proposed an adaptive provisioning technique for Cloud-based resources to deliver Cloud-based applications that meet QoS targets based on queuing network system model and workload information. They model each virtualized application instance as an *M/M/1/k* queuing model, where *k* refers to a finite queue of length *k*.

However, we believe that the structure of a cloud is similar to multiple queues rather than a centralized queue with multiple servers; since each cloud instance has its own network interface, computing resources, memory, and storage [72]. Several studies have modeled a web service as a network of queues, in which each machine in the distributed system is modeled as a single queue [75, 87, 96, 97]. A VM in the cloud computing system can be viewed as a machine in a distributed system offering a web service, and which could be modeled as a single queue [88]. Thus, modeling each cloud instance as an *M/M/1* is closer to the real world deployments.

In addition, according to the elasticity of the Cloud computing system, a cloud-based service usually has multi cloud instances (like EC2) offering the service to the Cloud users. Thus, modeling a cloud service by multi *M/M/1* queuing model is closer to the reality.

3.3 EDoS Attack

Cloud computing is designed to scale computation resources and servers in magnitude and availability based on demand and requested usage by end users. Moreover, adopters of the cloud service model are charged based on a pay-per-use basis of the cloud's server and network resources, which is widely known as utility computing. Such a service model may appear to overcome the effects of a DDoS attack, i.e., resource bottlenecks are eliminated. However, these clouds merely transform a conventional DDoS attack on server and network resources to a new breed of attacks that target the cloud adopter's economic resource, originally labeled as Economic Denial of Sustainability attack (EDoS) by Hoff [48]. Therefore, unlike conventional DDoS attacks, EDoS targets the financial constraint of an organization, but not its physical network or server constraints.

EDoS occurs when zombie machines (part of a botnet) send a large amount of undesired traffic towards the cloud, exploiting the cloud’s scalability, to chalk up an exorbitant amount of cost on a cloud adopter’s bill. In other words, the attack is making the cloud unsustainable by fading the cloud billing mechanism to charge the cloud user's bill for the attack's activities.

3.4 Proposed model

Our study in this chapter focuses on the evaluation of the EDoS attack on the SaaS cloud service such as a web application service which could be considered as a single-class service in which there is only one kind of application service provided in the data center. The attack utilizes the scalability nature of the cloud to charge the cloud adopters extra cost for the attack activities. For the attack to have more malicious influences on the cloud economics, it is required to flood the cloud service by heavy work load forcing the cloud provision technique to add more instances to manage this workload and satisfy the SLA requirements for the target cloud service.

Figure 3.1 shows a cloud-based web service architecture drawn based on the given specifications and architecture of most cloud computing providers like Amazon Web Application Hosting [67]. The main components are the Load Balancer (LB) service, VM instances, and Storage service.

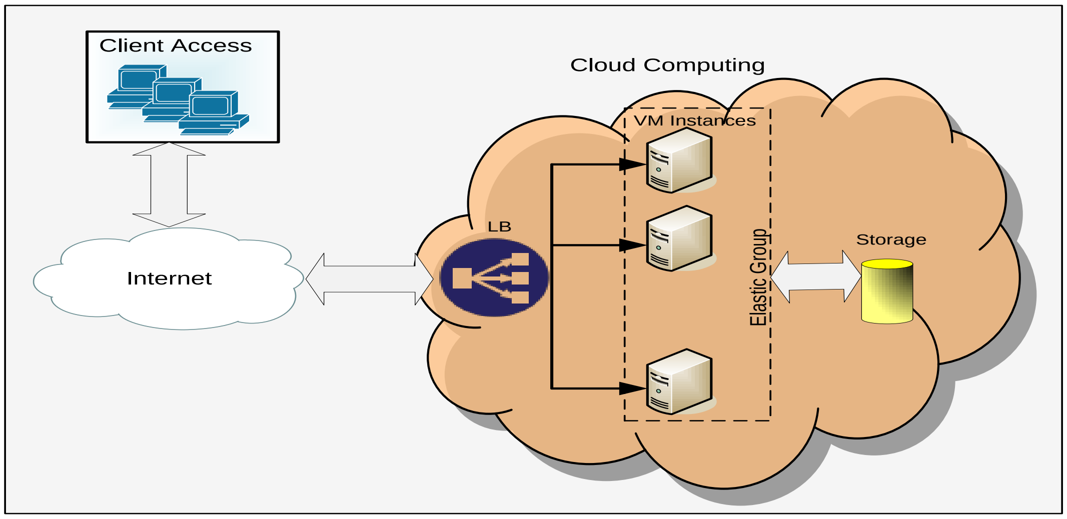


Figure 3.1: Cloud-Based web application architecture

The LB passes the clients requests through to a pool of available VM instances that represent the web/application service [66]. VM instances are clustered in elastic groups to which users associate triggers that will automatically scale VM resources based on bandwidth or CPU utilization measured by a monitoring system such as Amazon CloudWatch web service. LB ensures even distribution of the incoming load among all running VM instances in a group [68, 69].

VM instances run simultaneously as web application service centers, each potentially having a queue to process client requests [72]. The scalability of the service can be controlled by varying automatically the size of the group based on parameters such as the average CPU utilization of the running instances [70]. For example, when the average CPU utilization for a group exceeds an upper threshold, a trigger is fired to create a new instance that will be attached to the group and registered at the LB.

Like most web application architectures, a cloud-based web application service has a database server to be used for caching and storing configuration information [71]. For example, a Relational Database Service (RDS) could be used to enable a web application caching tier.

For most of the web applications, the number of client requests and the service rate are considered to be random variables having Poisson distribution [63, 73, 74, 75]. Similarly, it is an adequate and reasonable assumption for a cloud-based web application to consider the requests arrival rate and their service rate to follow a Poisson distribution. Several studies have assumed exponential distribution for both the requests inter-arrival time and the requests service time in a cloud service [64, 65, 91, 92]. In addition, we are focusing on the EDoS attack targeting a single-class service where all cloud customers' requests have the same processing procedure as it is in the web service that delivers content, such as web pages, using the Hypertext Transfer Protocol (HTTP) over the Internet. Thus, considering a Poisson distribution for the service rate in our case is a valid assumption that also helps in simplifying our performance model.

The cloud-based web application architecture shown in Figure 3.1 can be reasonably approximated using an open queuing network. Figure 3.2 shows an open queuing network model that mimics the required cloud-based web application stages including the load balancing service, computing tier represented by a group of parallel VM instances, and the cloud storage tier.

The LB is modeled as *M/M/1* queuing model, considering the use of randomized algorithm to balance the load amongst the available instances so that each instance has an equal probability of receiving a request [72].

VM instances are modeled as parallel queuing models each of *M/M/1*. It is worth noting that, in reality, a cloud instance has a bounded buffer queue, i.e., *M/M/1/k*, but for approximation and convenience, we use *M/M/1*. Such approximation is highly accurate for systems with large finite buffers, such as cloud computing systems, where the probability of overflow of the buffer is negligible [105, 106].

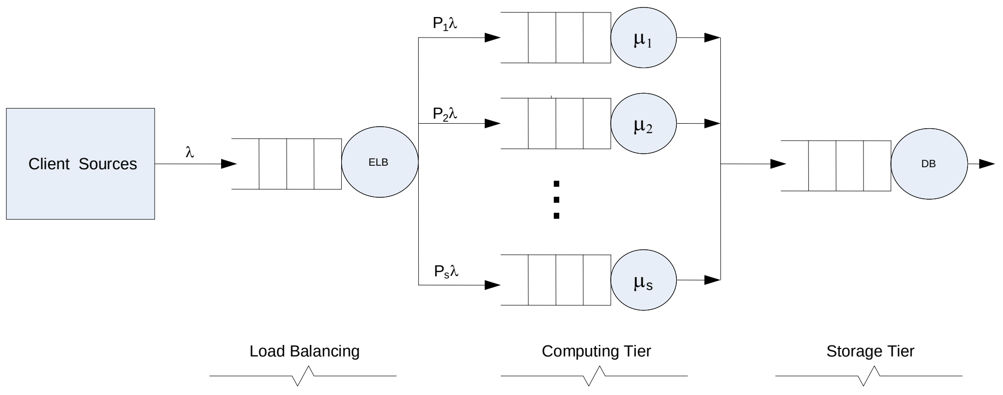


Figure 3.2: Queuing Model for the cloud-based web application

Modeling each compute instance as *M/M/1* corresponds to the architecture of the instance as it has its own network interface (NIC), computing resources (CPUs), memory, and storage [72, 88]. For example, with using Single Root I/O Virtualization (SRIOV)-capable Network Interface Card (NIC) [99], a VM could be bound to a Virtual Function (VF) driver that provides an abstraction of a dedicated NIC. Thus, when a packet is routed to the VM, it will be copied to the local queue assigned to the VM by the VF that gets executed with the virtual interrupt controlled by the SRIOV [100]. In addition, the LB is considered to be efficient and fast, otherwise, it will be a bottleneck at the cloud as it is the public access point of the cloud services. Thus, the other side (receiver), which is the computing instance, should have a queue to hold the arriving requests. Moreover, several studies have modeled a web service as a network of queues, in which each machine in the distributed system is modeled as a single queue [75, 96, 97]. Other studies have also proposed modeling each VM as a single queue for different purposes [65, 72, 87, 92, 98].

The storage tier can be implemented using RDS, such as Amazon RDS which provides a managed relational database in the cloud as a web service. RDS offers several capabilities such as scaling up the compute and storage resources, monitoring database health, point-in-time recovery for the DB Instance, and managing automated backups. Accordingly, RDS is considered to be vertically scalable on the database tier, and it is not able to scale-out by adding database servers due to its classic architecture [90]. Thus, the Cloud Storage service with such characteristics could be modeled as a single M/M/1 queue as it also has been discussed in [88, 95].

However, since the EDoS attack mainly utilizes the scalability of the computing resources to maliciously charge the user, we concentrate on modeling the computing layer of the cloud service to analyze the impact of such attacks on the cost and performance of the targeted cloud service. Furthermore, the utilization of Cloud storage can be ignored by assuming that all data requirements for the execution of a request is met by the virtual web application instance that has its networking, computing, and storage resources to process the requests [65, 72]. Thus, the problem can be reduced to the queuing model presented in Figure 3.3 as an open queuing network model.

Since the LB ensures even distribution of the traffic among the instances, the transition probability, P_i, will be equal to *1/S* for all the computing instances, where *S* is the number of running instances in the Auto-Scaling group. Assuming that all the computing instances have the same capacity of computing power, $\mu_i=\mu$, and the arrival rate at each instance is $\lambda_i = \lambda / S$, the equations of such model will be as follows.

The mean computing utilization *U* is calculated as follows:

$$U = \frac{\sum_{i}^{S} \frac{\lambda_i}{S\mu}}{S} = \frac{\lambda}{S\mu} \tag{3.1}$$

The mean response time, based on the given open queuing network model of S parallel single queues [79], can be calculated as follows.

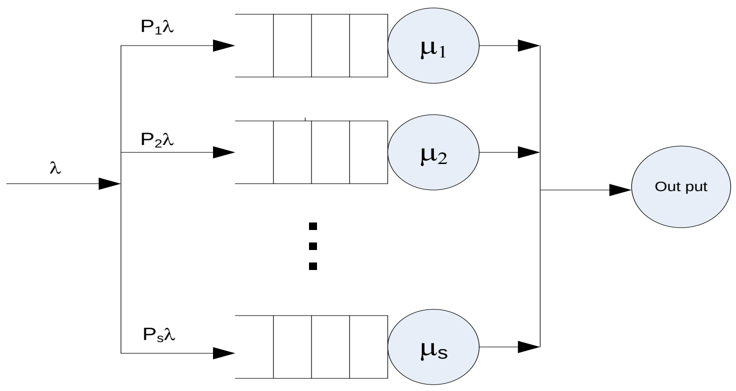


Figure 3.3: Queuing model for the computing resources

Since the LB is presumed to evenly distribute the requests among the *S* running instances, the routing matrix will have probabilities, *Pi*, that are equal to $1/S$, where *Pi* is the routing probability to the i^{th} instance. As a result, the total input into each instance is:

$$\Lambda_i = \frac{\lambda}{S}$$

The average delay of a request in instance *i* can be calculated based on *M/M/1* queuing theory to be:

$$\overline{T_i} = \frac{1}{\mu - \Lambda_i}$$

By little's formula, the average number of requests in instance *i* is $\Lambda_i \overline{T_i}$. Thus, the total number of requests in the network is:

$$\overline{N} = \sum_{i=1}^{S} \Lambda_i \overline{T_i} \quad = \sum_{i=1}^{S} \frac{\Lambda_i}{\mu - \Lambda_i}$$

Considering that the total rate of request flow into the network is $\sum_{i=1}^{S} \Lambda_i = \lambda$. Thus, by little's formula, the average delay of the network is:

$$\overline{T} = \frac{1}{\lambda} \sum_{i=1}^{S} \frac{\Lambda_i}{\mu - \Lambda_i}$$

Assuming that auto scaling of cloud instances is enabled and there is no delay in binding new instances to the group, the average response time of a request in the network, *Rt* will be:

$$Rt = \frac{S}{S\mu - \lambda} \tag{3.2}$$

Noting that when comparing the obtained equations for the given queuing model to *M/M/1* model with service rate of $S\mu$, both have the same mean computing utilization, *U*.

The mean response time, considering a single queuing model [79], is as follows:

$$Rt_{mm1} = \frac{1}{S\mu - \lambda}$$

Thus, the obtained average response time in Eq. (3.2) is *S* times the resulted one from *M/M/1* with service rate of $S\mu$. Such comparison will help in the simulation work by using only *M/M/1* queuing model while considering the mean response time to be Rt_{mm1} times *S*. However, in this work Eq. (3.2) was applied considering multi *M/M/1* queues.

As a summary, since the response time is an important requirement in most of the SLAs, Table 3.1 shows the measured response time considering *M/M/s*, *M/M/1*, and parallel *M/M/1* queues. It shows different response time measurements based on the input load.

Table 3.1: Response time equations for different queuing model

	Approximated M/M/s	M/M/1 with $S\mu$	S M/M/1 each with μ
Low load	$\frac{S}{S\mu - \lambda}$	$\frac{1}{S\mu - \lambda}$	$\frac{S}{S\mu - \lambda}$
High load	$\frac{1}{S\mu - \lambda}$	$\frac{1}{S\mu - \lambda}$	$\frac{S}{S\mu - \lambda}$

3.5 Analytical Modeling of EDoS Attack

Figure 3.4 shows the proposed queuing model for capturing the cloud service considering EDoS attack with rate of λ_m, and legitimate traffic with rate of λ_l requests per second.

The assumption of Poisson arrival for the DDoS attack has been discussed in [52, 76, 77, 78]. Since the EDoS behaves similarly to DDoS in generating malicious flooding traffic, we have assumed Poisson traffic for the EDoS attack. Although, attackers can choose any distribution to generate the traffic, the more attractive one is the distribution that is closer to the behavior of the legitimate traffic.

According to Poisson composition property [79], the aggregated traffic from multi sources each having an arrival of Poisson process follows a Poisson Process with an average arrival rate of $\lambda = \lambda_l + \lambda_m$. Thus, each node in the proposed queuing model has a Poisson arrival.

The intended metrics for the proposed model can be calculated as follows.

The mean utilization of the computing resources of the running instances can be calculated as:

$$U = \frac{\lambda_l + \lambda_m}{S\mu} \tag{3.3}$$

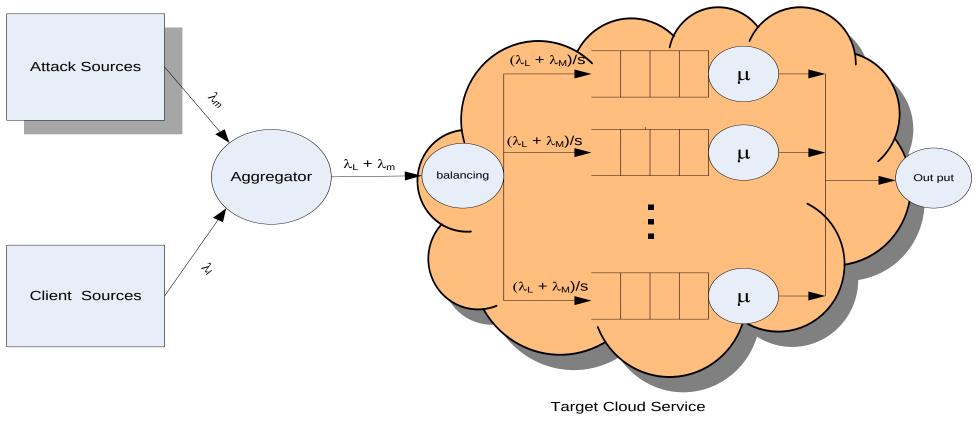


Figure 3.4: Queuing model for EDoS attack against a cloud service

The utilization incurred by the attack is:

$$U_m = \frac{\lambda_m}{S\mu}$$

which has an impact on both cost and performance metrics of the offered cloud services.

The average response time *Rt*, can be calculated based on Eq. (3.2) to be:

$$Rt = \frac{S}{S\mu - (\lambda_l + \lambda_m)} \tag{3.4}$$

One of the measurements that we have studied, and which is the target of an EDoS attack, is the cost associated with both the computing resources and the bandwidth on the cloud service side.

There are several pricing models that could be adapted to the Cloud computing system. Currently, Amazon mainly offers computing instances with three pricing models including on-demand, spot pricing, and fixed pricing models [81]. The on-demand model allows the cloud user to pay for the used resources by the hour with no long-term commitments. In the fixed pricing model, a Cloud user reserves Cloud instances with one-time payment for each instance, e.g., for a one or three year period; and in turn receives a significant discounted hourly pricing on usage. The spot pricing model, offered by Amazon EC2, allows a Cloud user to bid for available EC2 capacity and grants the user the requested resource only if the user's bid price is above the current spot instance price, which fluctuates periodically based on supply and demand.

A Cloud user has to pay for the computing resources, the network traffic volume, and for the storage service, if required. In our work, we consider the cost related to both the computing usage and bandwidth usage.

The cost with regard to the computing resources has been calculated based on CPU utilization as follows:

$$COST_{com} = \sum_{i}^{n} Price_{com} \times U_i \times t_i \times S_i \tag{3.5}$$

where U_i is the average CPU Utilization of all instances during the period t_i expressed in hours, $Price_{com}$ is the base price charged for the smallest amount of computing resources per hour per instance, and S_i is the number of running instances during time t_i.

Since we are interested in calculating the cost incurred by the attack assuming it lasts for T hours, the total cost during the attack can be expressed as follows:

$$COST_{com} = Price_{com} \times U \times T \times S \tag{3.6}$$

where U is the average utilization of the computing resources calculated in Eq. (3.3), and T is the total running time in hours.

By substituting U_m in Eq. (3.6), the extra cost incurred by the attack will be as follows:

$$COST_{com-attack} = Price_{com} \times T \times \frac{\lambda_m}{\mu}$$

The cost related to the bandwidth can be calculated as follows:

$$COST_{bw} = Price_{bw} \times \overline{\lambda} \times T$$

where $\overline{\lambda}$ is the effective arrival rate measured in GB/s, and $Price_{bw}$ is the price per GB. When assuming that the queuing–based loss probability of cloud service is zero, the effective arrival rate $\overline{\lambda}$, equals the arrival rate, λ . Thus, the total cost can be expressed as follows:

$$COST = \left(Price_{bw} \times \lambda_{GB/s} + Price_{com} \times U \times S\right) \times T \tag{3.7}$$

Where $\lambda_{GB/s}$ is the effective arrival rate in GB/s, and equals $\lambda_l + \lambda_m$.

The number of instances committed to the cloud application service could be calculated based on Eq. (3.3) when assuming 100% as the upper threshold utilization for provisioning to be as follows:

$$S_{required} = \left\lfloor \frac{\lambda_l + \lambda_m}{\mu} + 1 \right\rfloor$$

Thus, the number of instances is changed based on the arrival rate of requests and the capacity of the Cloud instance.

According to the auto scaling service offered by Amazon [70], thresholds are used to trigger the provision mechanism to increase or decrease the number of running instances committed to an Auto-scaling Group. When the mean of CPU utilization, U , is above the upper threshold, *Upper*, the number of instances should be increased either by a specific number of instances or by a percentage of the current used resources to handle an increase in traffic. Similarly, when U is below the lower threshold, *Lower*, the number of instances should be decreased to more efficiently use computing resources. For example, when we consider adding or removing instances by 10% of the running instances at the time of the threshold triggering, the provisioning can be expressed as follows:

$$\text{When } U > Upper, \quad S_{new} = [\, S + \lceil 0.1 \times S \rceil \,]$$

$$U < Lower, \quad S_{new} = [\, S - \lceil 0.1 \times S \rceil \,]$$

Where S is the current number of running instances, and S_{new} is the updated number of instances after the occurrence of the trigger.

3.6 Tuning the provisioning parameters

One of the main characteristics of the cloud computing is its elasticity at which the resources can be scaled up or down based on some monitored metrics of the cloud computing services. Statistics about these metrics are collected during a window time called the monitoring window. When a particular metric indicates a value above a given upper threshold, a policy is triggered to provision more resources so as to enhance its performance with respect to this metric. On the other hand, when a metric has a value below a given lower threshold, a policy is triggered to terminate some resources. This will enhance the usage of resources and keep the monitored metric value above the given lower threshold.

According to the auto scaling service offered by Amazon [70], VM instances are clustered in elastic groups to which users associate triggers that will automatically scale VM resources based on metrics measured by a monitoring system such as Amazon CloudWatch web service.

However, there are many factors related to the resource provisioning in cloud computing that might affect the performance of the cloud services. These factors include the overhead when allocating an instance to the cloud service, thresholds used to control the provisioning event such as the utilization threshold, the number of instances added every time the provisioning takes place, and the monitoring window time used for collecting the statistics.

3.6.1 Upper Threshold Utilization

In this section, we focus on tuning the upper utilization threshold factor which affects the performance of the cloud services. In this work, we only consider the usage of the upper threshold of the CPU utilization, which is used in the provisioning process to add more instances so as to cope with the load spikes.

We have conducted several simulation experiments that mimic the cloud computing service characteristics to show and evaluate the impact of varying the upper threshold of the CPU utilization on the cloud service performance metrics. These metrics include response time, average CPU utilization, and number of allocated instances as an indication about the cost.

In our case study, we assume a high load spike such as that caused by DDoS attacks. Thus, we have only considered the upper threshold for adding more instances so as to cope with the load spikes. Such upper threshold has been discussed in previous works [65, 69, 84, 89], and different values ranging between 50% and 90% have been used. In this work, and for simplicity, we considered studying four different thresholds, i.e., 50%, 70%, 80%, and 90%. For each one, we considered the scaling size to be two instances per provisioning occurrence. The time window that we have used to monitor the resources utilization is 5 minutes, as it is the default period used in the Amazon auto-scaling mechanism [70].

We assumed using a small instance that has a capacity of 100 requests per second as it was discussed by Catteddu and Hogben [3]. The arrival requests rate was assumed to be 200 Req/sec during the early periods of the simulation and then it increases to be 2400 Req/sec after 25 minutes. The number of initial running instances is 5 instances which can handle 200 Req/sec under the 50% of the utilization. This means that the load was increased 12 times indicating a high load peak.

Furthermore, we considered the overhead caused by committing instances to the cloud service to be 55.4 sec for provisioning one VM instance, as was measured by Islam et al. [93].

Figure 3.5 shows the output of the simulation experiments conducted for evaluating the impact of the utilization thresholds used with the provisioning process. Results show that all cases have the same trend during the early periods of the simulation time. For instance, all the evaluated performance metrics are identical for the first 95 minutes. This is due to the high load and the shortage of the instances processing this load, which leads to a high utilization values, i.e., greater than 90%.

When there are enough instances to process the load of the input requests, the trend for each case starts to improve until reaching a steady state. The response time graph shows that, starting from 100 minutes up to the end of simulation, the response time is low for all the cases. However, during such period, the trend can be recognized as follows. First, the case of 50% as a utilization threshold has the lowest response time, due to having more running instances, i.e., about 50 instances as shown in the Instances Graph. Second, the case of 90% has the highest response time as it has the lowest number of running instances, i.e., about 31 instances as shown in the Instances Graph. In this case, the provisioning has stopped at 100 minutes of the simulation time as the resources utilization value became below the threshold of 90%. Finally, the cases of 70% and 80% have identical trends in terms of the response time, with values ranging between those reported in the other two cases, i.e., 50% and 90%.

As a verification of the simulation results, we calculated the minimum number of instances required to insure that the average utilization is below the given upper threshold. In the case of 80% as the upper utilization threshold, the number of required instances can be calculated as follows:

$$\frac{\lambda}{S\mu} \leq 0.8 \text{. Thus, } S = \lceil 1.25 \times \lambda / \mu + 1 \rceil \qquad (3.8)$$

Where λ and μ represent the arrival rate into the auto-scaling group and the service rate of one instance, respectively.

When considering the case we studied, the minimum number of required instances is 31 instances according to Eq. (3.8), while it is found to be 33 instances in the simulation results. The two additional instances obtained in the simulation results are due to the queuing overhead that triggered the initial provisioning for adding two more instances so as to flush out the queued requests. Such requests have accumulated when the number of instances was insufficient to process the input load.

It is obvious from the reported results that using an upper threshold of low values such as 50% leads to using more instances while getting a lower response time. Whereas, using an upper threshold of high values such as 90% leads to using fewer instances while getting a higher response time. In other words, the higher the upper threshold value is, the less usage of instances and the higher response time we have. Similarly, the smaller the upper threshold value is, the more usage of instances and the lower response time we have.

To obtain the optimal upper utilization threshold that could be used with the provisioning process when considering such a high increase in the load, one has to solve an optimization problem with two objectives. These two objectives are minimizing the usage of instances and minimizing the response time.

There are several ways for solving such optimization problem [85, 101, 102]. However, we solve it for “our case study” using a graphical method by plotting the response time vs. the number of used instances [102]. This is done for different values of the threshold ranging between 50% and 90%.

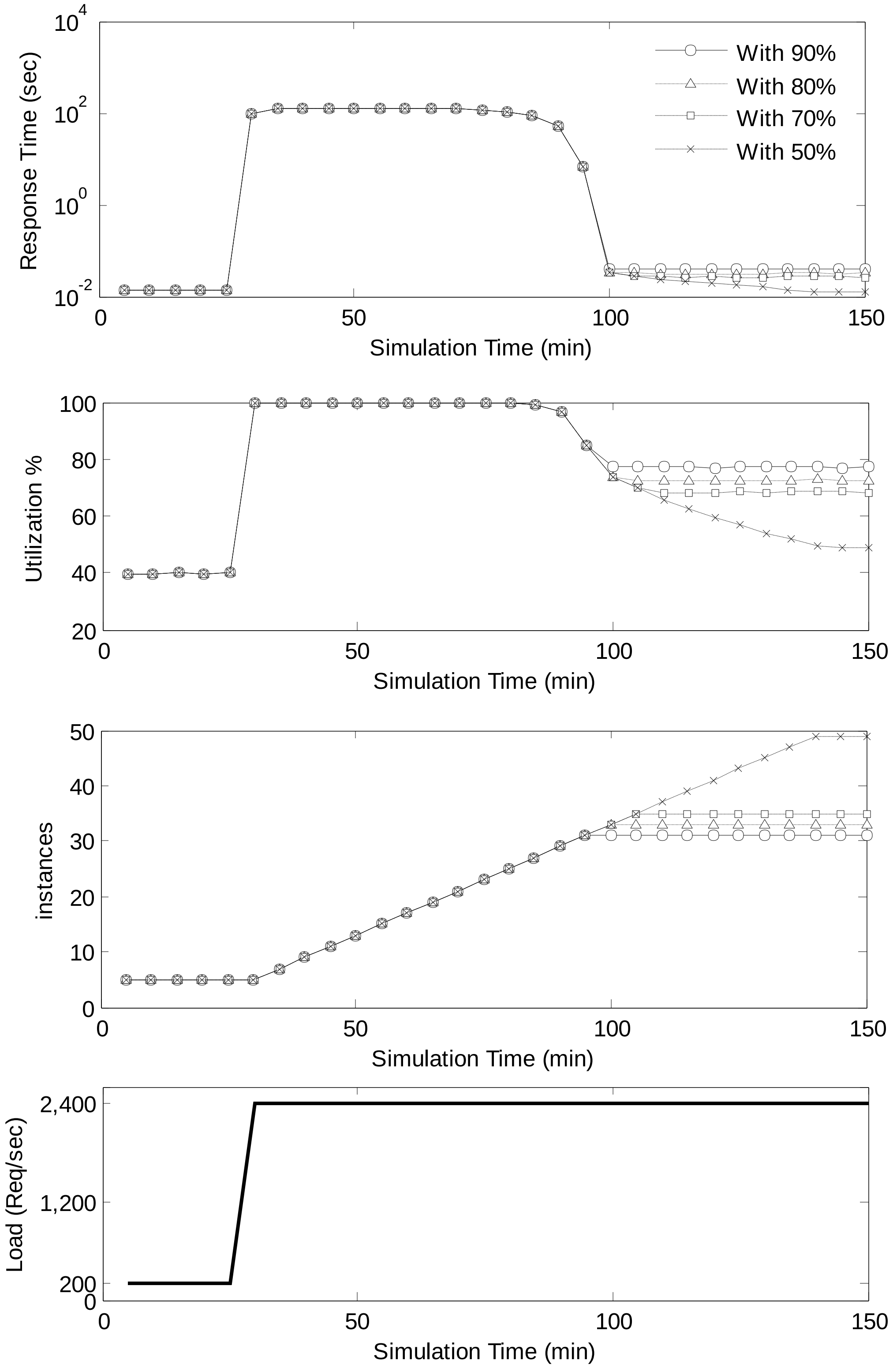


Figure 3.5: Results of using different utilization thresholds

Figure 3.6 shows the optimal value of the upper utilization threshold, in our case study, by assuming that both objectives have the same relative importance in the optimization problem. Since these two objectives conflict, by using the mediums, we guarantee that the selected interior points will not lead to a cost higher than the medium. By repeating such process recursively, we get the optimal value that satisfies the equilibrium between the two objectives. For example, the first step shown in Figure 3.6, guarantees that the interior points will lead to a number of instances (first objective) less than 45 instances and an average response time (second objective) less than 0.13 sec. the rest of steps will narrow the periods toward the optimal value that satisfy both objectives. Thus, for our case study, we draw several mediums to determine the optimal value that satisfies both objectives and we found it to be 80%.

In addition, we apply the above graphical method to determine the optimal thresholds corresponding to different input loads. Figure 3.7 shows the obtained results, and which indicate that the optimal upper thresholds for the given loads are between 70% and 90%.

Thus, in our study about the impact of the EDoS attack on the cloud computing services, we choose the upper utilization threshold to be 80% as the mean of both 70% and 90%.

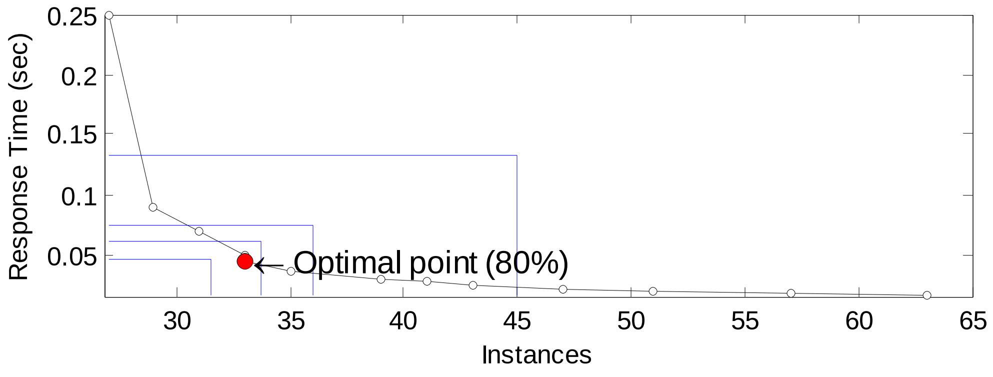


Figure 3.6: Optimal point leading to the minimum delay and resources

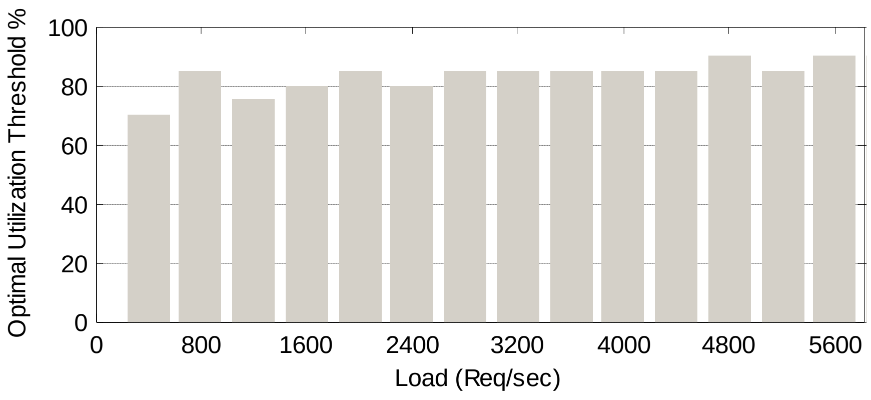


Figure 3.7: Optimal upper thresholds for different loads

3.6.2 Scaling size parameter

There is also another parameter related to the provisioning process that affects the performance of the cloud service when adding resources to cope with the spikes of the incoming load, and that is the number of instances that will be added every time such provisioning process takes place.

We conducted an investigation through simulation to show visually the impact of the scaling size parameter on the response time and the number of allocated instances; by considering five cases including allocating additional instances of 2, 3, 4, 26, or all the required instances during each provisioning period.

We have excluded the investigation of scaling size of one instance since we are considering high load spikes at which the scaling by one instance is not convenient. In addition, the trends for scaling sizes above 4 can be judged from the trends of the used scaling sizes, i.e., 2, 3, and 4. The choice of scaling size of 26, the fourth case, is based on the Eq. (3.8) given that the arrival rate is 2400 Req/sec, where the number of instances is 31, since the already allocated instances are 5, and the utilization threshold is 80%. For the last case, we calculated the required instances, by applying Eq. (3.8), where the utilization threshold is 80%. When the provisioning takes place at the first time, the scaling size is set to the calculated number of the required instances. Then, if the utilization is still above 80%, the provisioning takes place to add only two more instances.

For all the cases, we used 80% of the utilization as a threshold and a constant input load of 2400 Req/sec.

In Figure 3.8, Response time Graph, results show, as it is expected, that when adding all required resources based on the offered load to the cloud service, the delay considerably converged earlier to the stable value of the response time with using less instances. In addition, the results show that in the first, second, and third cases, it takes a long time to converge to the stable low value of the response time.

The results for the first case, with scaling size of 2, show that the performance metrics reached the steady state later compared to the second and third cases with scaling sizes of 3 and 4, respectively. For example, the first case reached the steady state at 75 minutes of the simulation time whereas the second case reached it at 55 minutes and the third case at 45 minutes. Nevertheless, after reaching the steady state, the response time for the three cases are comparable as it is shown in the response time Log Graph that presented the results using the log function for the purpose of obtaining a better view.

The results obtained for the fourth case, show that it has a close trend to the last case during the time periods prior to the steady state of the performance metrics. However, it consumed more instances compared to the other cases. For example, such case required 57 instances whereas the last case required only 33 instances to process the input load of 2400 Req/sec. Thus, we focus in this investigation on adding specific low number of resources as it is in the first three cases discussed above.

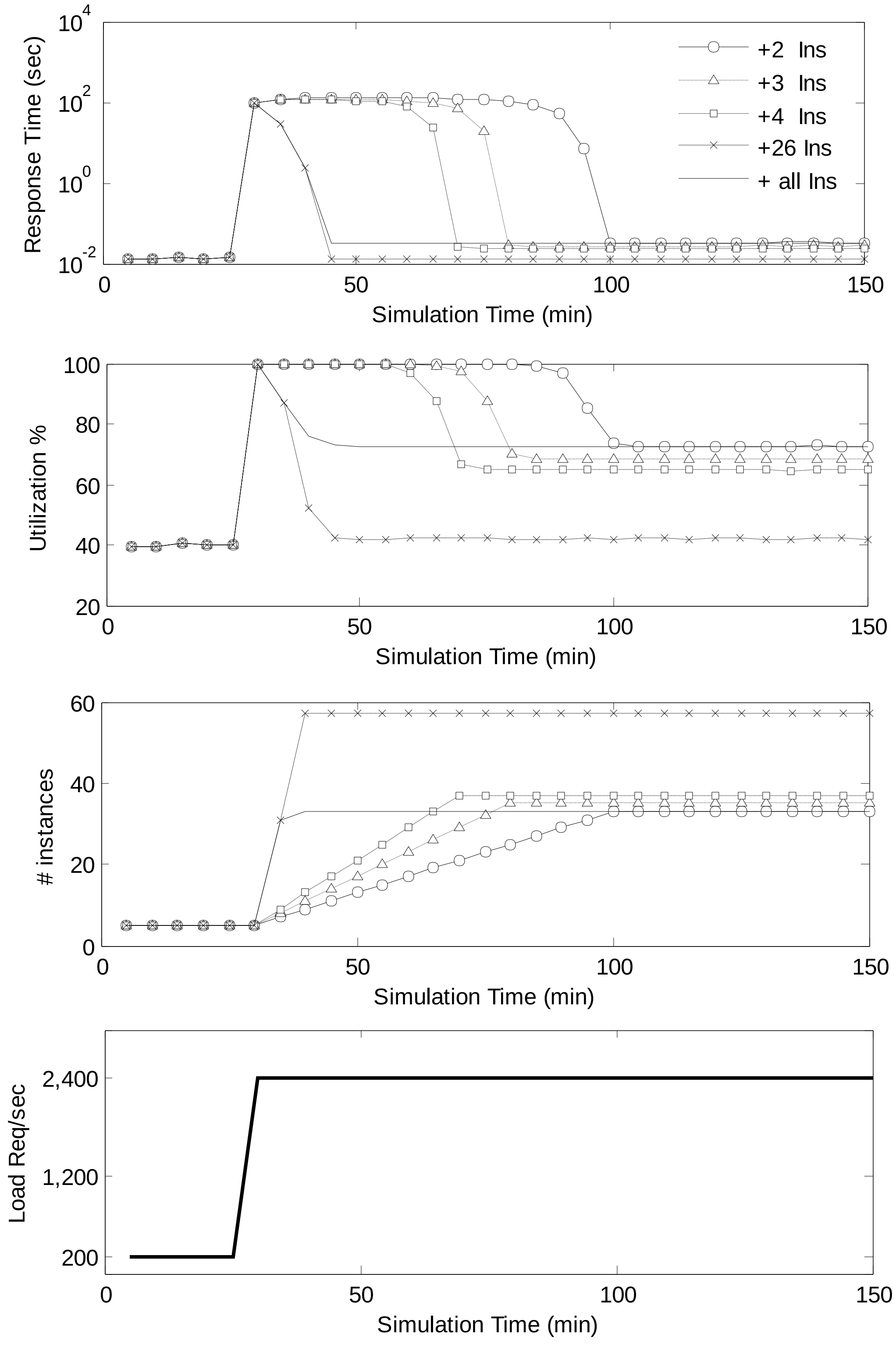


Figure 3.8: The impact of using different number of instances in the provisioning.

However, regarding the last case, it is not practical in the market. For example in Amazon auto scaling technique, the customer can only configure the provisioned instances either by specifying the number of instances or the percentage of the instances to be provisioned when the provisioning takes place.

Based on the simulation results shown in Figure 3.8, we can conclude that the less the scaling size value is, the less usage of instances we have and the more time is required to reach the steady state of the performance metrics. However, adding more instances such as in the fourth case, leads to the worst over-provisioning of the resources but it is the earliest case to converge to the steady state.

Thus, we decided to tune such parameter by considering an optimization problem having two objectives including the early convergence to the steady state of the performance metrics and the efficient usage of the instances. The optimization problem is discussed in the following subsection.

The first objective regarding the convergence to the steady state of the performance metrics can be expressed by the number of time slots (windows) needed to reach the steady state, which can be modeled as follows.

The optimal number of instances required to process an observed load of $\bar{\lambda}$ can be calculated using Eq. (3.8). Thus, the optimal number of instances required to be added to cope with the spike of $\bar{\lambda}$ is:

$$needed\_res = 1.25 \times \bar{\lambda} / \mu + 1 - run\_res\,,$$

where run_res is the number of already running instances.

The required instances will gradually be committed to the service by adding *X* instances every time the provisioning takes place. In addition, after allocating all the required instances, it needs one window time to converge to the steady state as it was observed in the simulation experiments. As a result, the first objective can be expressed as follows:

$$Min \;\; F_1(X) = needed\_w = \left\lceil \frac{needed\_res}{X} \right\rceil + 1 \quad , \; X >= 1. \qquad \textbf{(First Objective)}$$

The second objective regarding the efficient usage of the instances can be presented by the total number of instances assigned to the cloud customer, which in turn can be expressed as follows:

$$Min \;\; F_2(X) = total\_res = needed\_w \times X + run\_res \, , \; X >= 1. \; \textbf{(Second Objective)}$$

The goal is to find *X* that minimizes both the convergence time expressed in the first objective and the total used instances expressed in the second objective.

There are several optimization methods for solving the multi-objective optimization problems. Marler et al. [101] presented a survey of multi objective optimization (MOO) methods which have been categorized as methods with a priori articulation of preferences, methods with a posteriori articulation of preferences, and methods with no articulation of preferences.

The first category includes methods that allow the user to specify preferences based on the structure of the optimization problem. Such preferences may be expressed in terms of the relative importance of different objectives. Examples of this category are Weighted

Sum (WS), weighted global criterion method, lexicographic method, weighted min-max method, exponential weighted criterion, and weighted product method.

The second category includes methods to solve optimization problems that are difficult for a decision-maker to express their input preferences on the objective functions. Thus, these methods consider only the preferences on the final solution. Examples of this category are physical programming method, normal boundary intersection (NBI) method, and normal constraint (NC) method.

The third category describes methods that do not require any articulation of preferences. Most of the methods in this category are simplifications of the methods in the first category considering that all the preferences are equal to 1. For example, the weighted sum method can be used to solve such problems, considering the weights for each objective function to be 1.

Among the three categories of the methods, our optimization problem can be solved within the first category, as it is easy to give an explicit approximation of the preferences for our optimization problem. For example, the relative importance of the objectives function in our optimization problem are known so that we decided to give the cost objective function more weights since the focusing in our work is on the cost impact of EDoS attacks.

We prefer using WS for the following reasons. First, it is the most common and simple approach to solve multi-objective optimization problems [85, 101, 103]. Second, it required setting only one parameter which is the weights, compared to other method like exponential method that required setting both the weights and P parameters, making it more complicated to specify the preferences. Third, our problem is not so complicated, as it has a well-known structure of two objective functions and the preferences for those objectives can be set based on the relative importance of the objectives. Thus, there is no need to use methods that require more parameters setting, or have more computational complexity.

The idea of the WS method is to convert the multi objective optimization problem into a single one using weights and summation for the objectives. The general formula of the WS method is as follows:

$$WS(w) = \min_{x \in X} \sum_{k=1}^{n} w_k f_k(x),$$

$$\sum_{k=1}^{n} w_k = 1, \quad w_k \geq 0 .$$

where $f_k(x)$ and w_k are the function of the k^{th} objective and its weight, respectively.

Thus, our two objectives optimization problem can be solved as a scalar optimization problem using the WS method as follows:

$$WS(\alpha) = \min_{x \in X} \left(\alpha \times total\_res + (1-\alpha) \times needed\_w\right),$$

where $X \geq 1$, and $1 > \alpha > 0$.

Practically, X has also an upper boundary as the number of instances that could be allocated is limited. For example, an Amazon EC2 account by default is limited to a maximum of 20 instances per EC2 region and can only be higher after getting an approval from Amazon. In other words, X is bounded which has been assumed in our case to be $1 \leq X \leq 100$.

We solved the above minimization problem using Matlab utilizing the “fminbnd” function [104] that solves such type of the bounded problems. The function searches for a proper value of X that minimizes *WS*. The solution to the above optimization problem depends on the weight, α, being used.

Figure 3.9 shows the impact of weighting on the solution of the problem. Results show different behaviors of the optimization function based on the weights being used in the WS method.

The weighing factor can be determined by several methods discussed in a survey done by Marler [101]. However, the weights might be chosen manually when the structure of the problem is well known considering that the relative value of the weights reflects the relative importance of the objectives. In our case, since we are focusing more on the impact of the EDoS attack on the cloud computing services, we decided to give the second objective, i.e., total number of instances, more weight than the first objective. In addition, we avoided using the weight of 0.9 and 0.8 as they imply a very high importance towards the second objective and a very low importance towards the first objective. This is not appropriate as the response time will have a very low importance. The weights of 0.7 and 0.6 gave comparable results of the WS function. Thus, we have chosen to use the weight of 0.6 with the second objective and 0.4 with the first objective.

We solved the above optimization problem considering different rates as it is shown in Figure 3.10. For example, if the observed input rate is 2400 Req/sec, such as the one used in the previous scenarios of the simulation, the optimal number of instances required to be added per provisioning event is 3 instances.

For verification purposes, Figure 3.11 shows the total number of required instance verses the convergence time of the delay when the load is 2400 Req/sec. The optimal point considering both objectives is the third point which is corresponding to using 3 instances in the provisioning process.

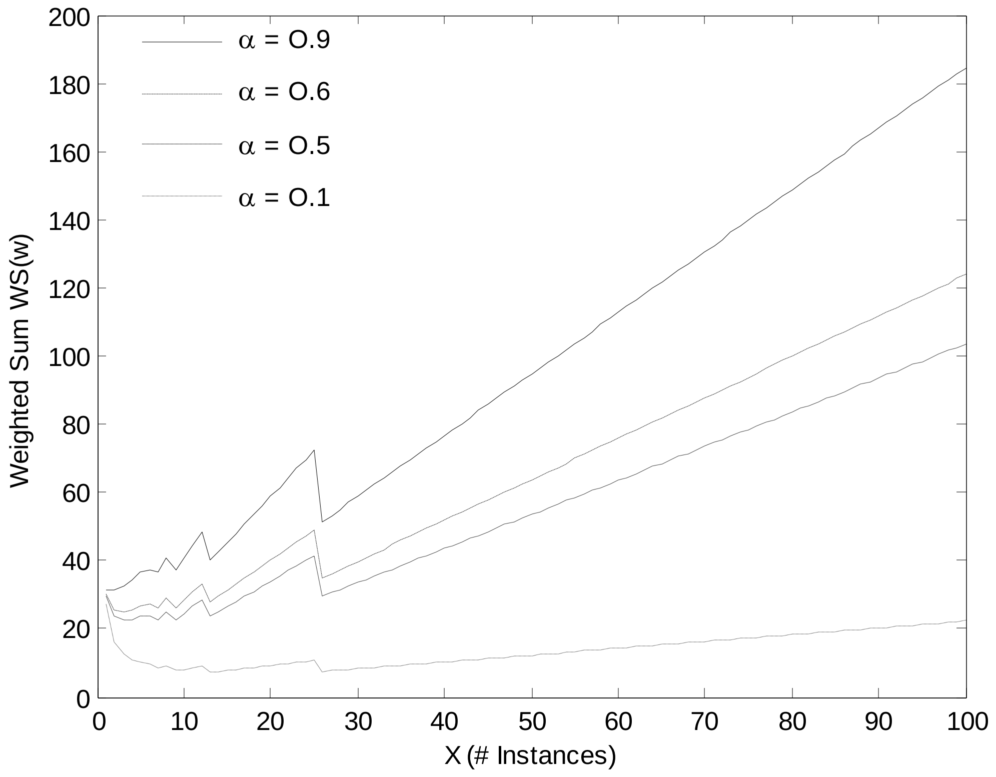


Figure 3.9: The impact of weighting on the solution of the optimization problem

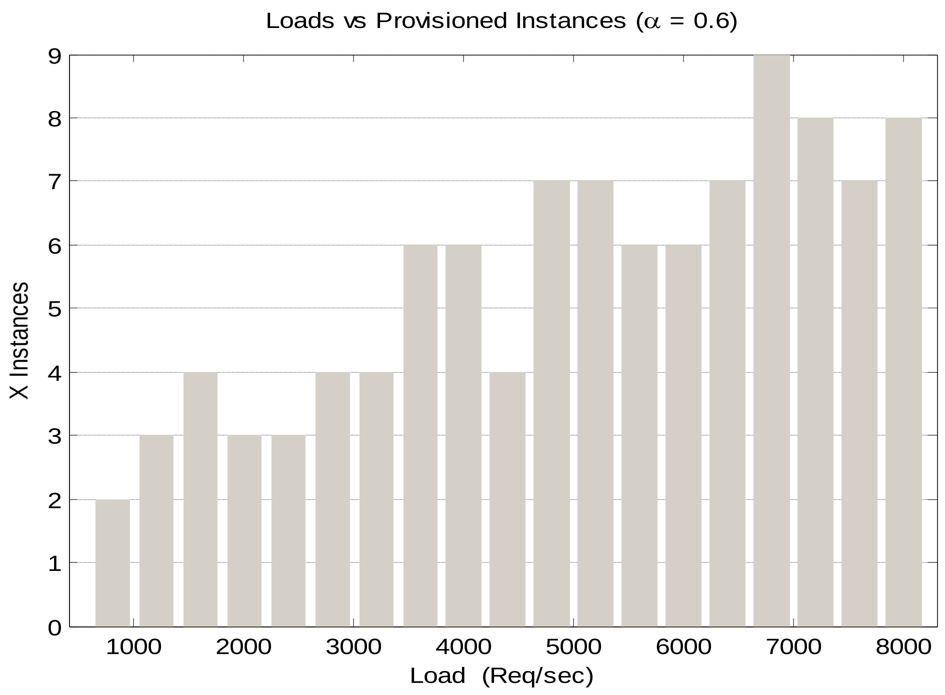


Figure 3.10: Loads vs. optimal number of provisioned instances.

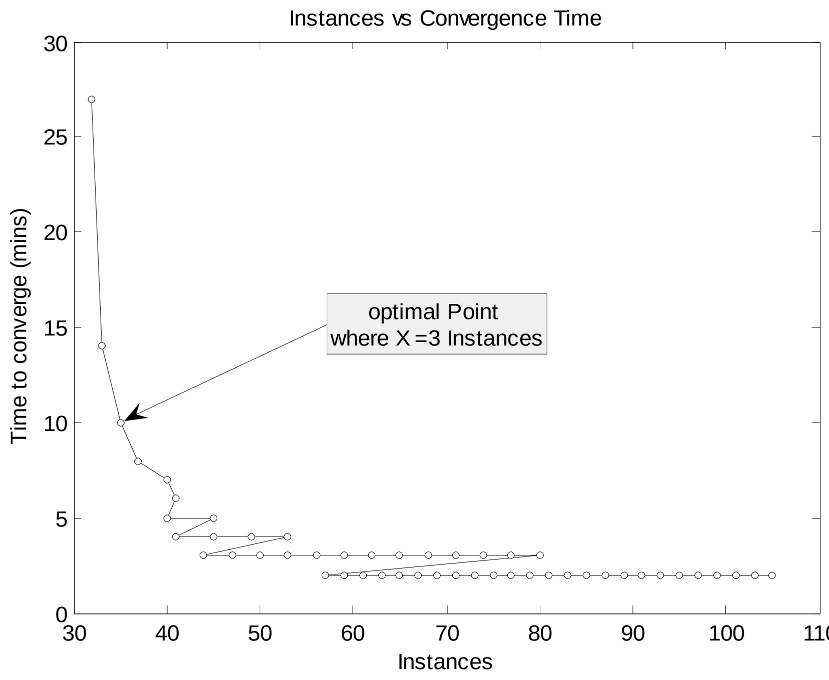


Figure 3.11: Optimal number of instances vs. convergence time

3.7 Simulation Setup

The simulation model used through this dissertation is a discrete-event simulation. The simulation followed closely the guidelines given by Law and Kelton [82], including the use of initial seeds that were ten million apart, and avoiding any overlapping in the random number streams during the simulation.

Proper seed selections have to be made in order to avoid wrong combinations of seeds and random number generators that may lead to erroneous results. A different stream is generated for each simulation variable. Here are briefly some of the guidelines that are followed in selecting seeds [82]:

- Arbitrary values for seeds were not used. Also, the values of zero and even values were not used.
- Every simulation variable has its own stream, and streams were not subdivided.
- Overlapping of streams, to prevent correlation, was avoided by choosing seeds spaced 100,000 apart. In our case, the seeds were spaced 800,000 apart.

As it is recommended by Law and Kelton [82], PMMLCG (prime modulus multiplicative linear congruential generator) is used in our simulation for generating random numbers. The PMMLCG is an efficient generator and one of the most popular methods for generating random numbers [82].

To determine the simulation run length, first we applied the Welch technique to eliminate the warm-up period [82]. Welch's technique is based on plotting of moving averages calculated for the means of the observations made in replications. The warm-up period is selected at the point at which the plot becomes smooth. Then, the length of the simulation is set to five times the length of the period. The number of arrival events was about 5 millions to reach the steady state.

Figure 3.12 depicts the general flowchart of the simulation model that is applied for each queue in our simulation. Our simulation model has two types of events including the ARRIVAL and DEPARTURE events. The ARRIVAL event occurs when a new request arrives to the system. The DEPARTURE event occurs when a request is completely processed by the system. The two events are generated independently such as each event has its own seed and random-number stream.

We have conducted a discrete-event simulation experiment to evaluate the performance of the cloud service under the EDoS attack in terms of key performance indicators including end-to-end response time, computing resources utilization, and throughput. Since the EDoS attack is mainly targeting the cloud adopter, we have also evaluated the cost associated with the computing resources and bandwidth allocations at the cloud service side.

In the simulation experiment, we have considered the same setup as that of the queuing model presented in Figure 3.4. The input to the simulation is an aggregated traffic from different sources including attackers' traffic. We have considered the Poisson nature of the incoming traffic as was clarified in section 3.5. We have assumed a fixed input rate of 400 Req/sec (request per second) representing the rate of the legitimate requests coming from clients and a variable input rate ranging from 400 Req/sec to 8000 Req/sec representing the rate of the attack traffic.

As it was discussed in the previous section about tuning the parameters, we have assumed the parameters in the simulation experiments as follows.

The capacity of an instance is of 100 requests per second

The number of initial running instances is 5 instances

The provisioning overhead is of 55.4 sec

The upper utilization threshold is of 80%

The scaling size is variable based on the optimization mechanism discussed earlier regarding tuning the scaling size parameter.

The cost has been calculated based on Eq. (3.7). The $Price_{com}$ has been set to \$0.115 as it is recently reported in Amazon for small on-demand instances running on the Windows operating system [81]. Regarding the cost associated with the bandwidth allocation, we have used a base price of \$0.01 per GB in/out data transferred based on the reported prices of Internet data transfer "in" and "out" of Amazon EC2 [81].

Finally, the relative error % is used to measure the accuracy of the queuing model results compared to the simulation model results. The percentage of the Relative error is defined as follows:

$$\mathrm{Re}\,lativeError = \left| \frac{Queueing\mathrm{Re}sults - Simulation\mathrm{Re}sults}{Simulation\mathrm{Re}\,sults} \right| \times 100 \tag{3.9}$$

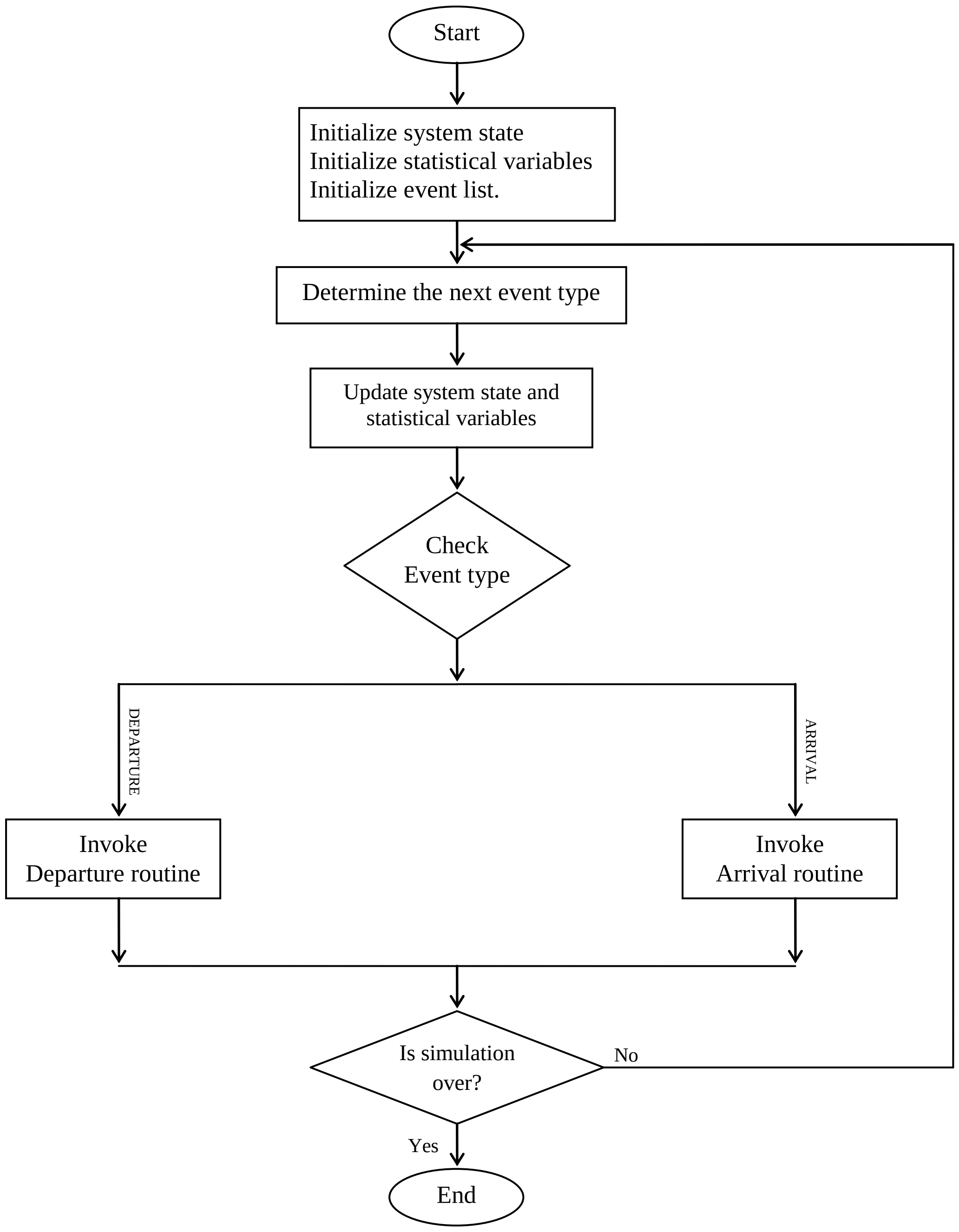


Figure 3.12 : Flowchart of the Simulation Model

3.8 Results and discussion

We have conducted two experimental scenarios using the simulation model discussed in the previous section. In the first scenario, we have considered different attack rates to show the impact of the attack on the targeted cloud service. The second scenario is for the optimal case where there is no attack targeting the cloud service. In addition, the output of the proposed analytical model has been compared to the simulation results.

Figure 3.13 shows the obtained results regarding the end-to-end response time of the legitimate requests. The results show that when the load increases, the corresponding response time also increases. It is obvious that the response time does not go up considerably when the attack traffic highly increases. This is due to the auto scaling mechanism that allocates more instances to process the high load caused by the attack traffic. However, the results in general show that the attack makes the legitimate clients suffer more response time compared to the optimal case.

Figure 3.14 shows the evaluation of the computing resources utilization. Results show a trend similar to the one of the end-to-end response time in Figure 3.13 such that as the attack rate increases, the utilization increases. It is obvious from the results that the average utilization does not exceed the upper threshold, 80%, used in both simulation and analytical models. Moreover, the results show that the EDoS attack consumes more computing resources when compared to the optimal case where there is no attack. For instance, based on the numerical results obtained from the simulation, at an attack rate of 6 KReq/sec, the mean utilization for 79 running instances is about 79%, whereas for the normal case the mean utilization is only about 67% while using only 6 instances.

Regarding the cost evaluation, Figure 3.15 shows an increase of the cost when the attack rate increases. In fact, the extra cost added because of the EDoS attack is very high when compared to the optimal cost where no attack takes place. For instance, at an attack rate of 6 KReq/sec, the total cost is about15 times the normal one.

Figure 3.16 shows the resulted relative error percentage for the response time, utilization, throughput, and cost results when comparing the queuing model to the simulation model. It shows a good accuracy for all the studied metrics with maximum error of about 0.1%.

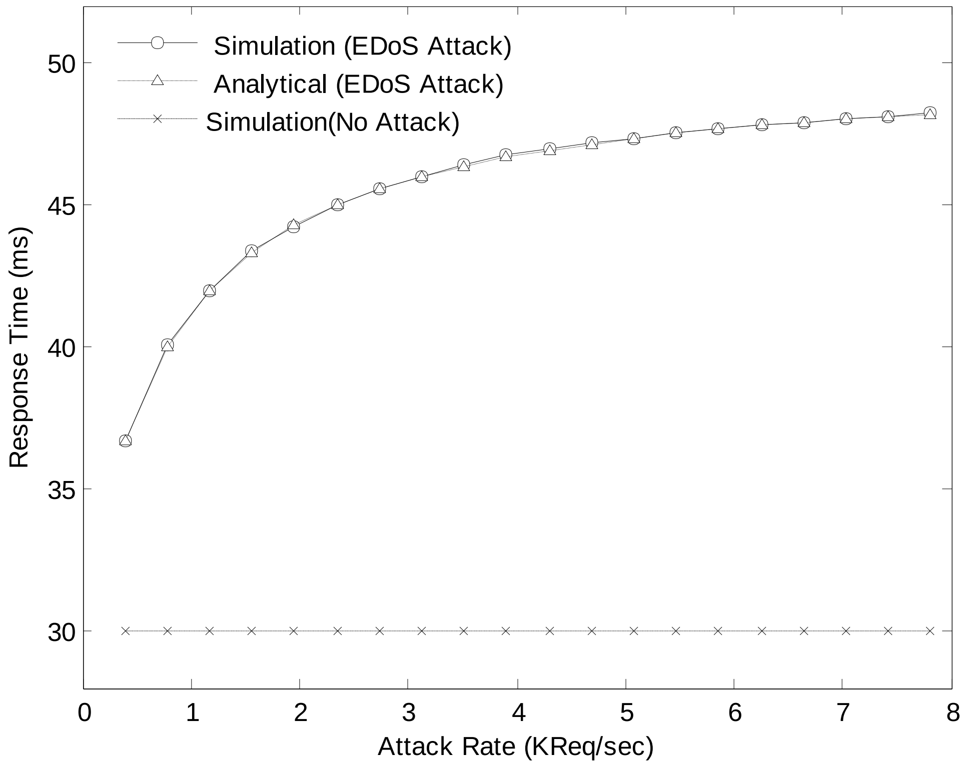


Figure 3.13: Response time evaluation, provision triggered at 80%

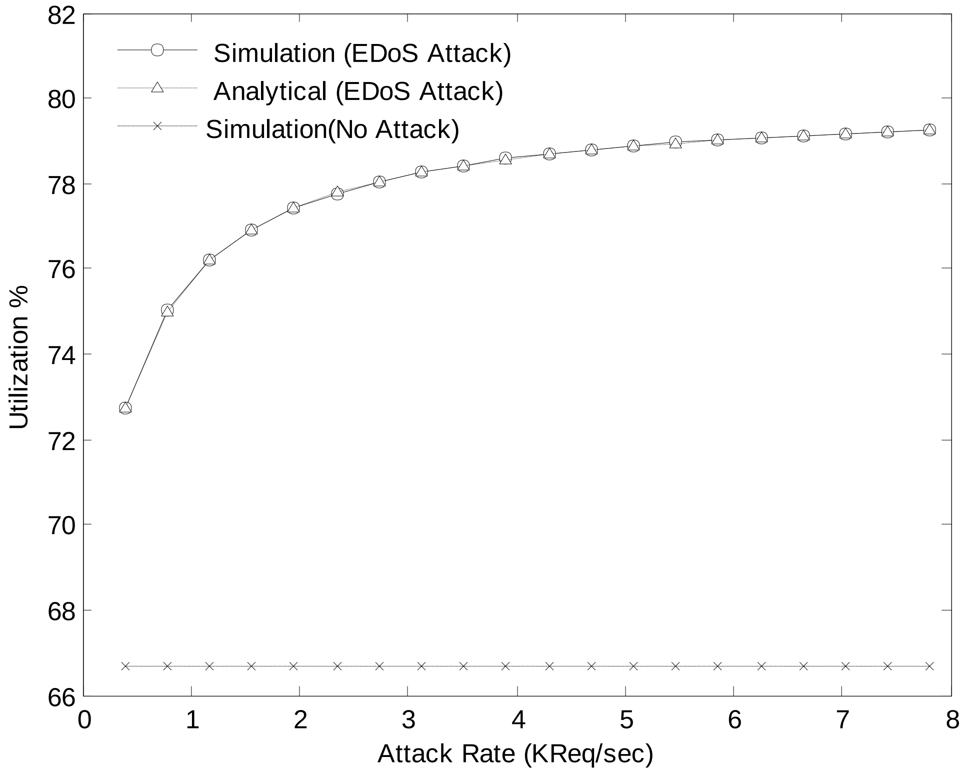


Figure 3.14: Computing resources utilization, provision triggered at 80%

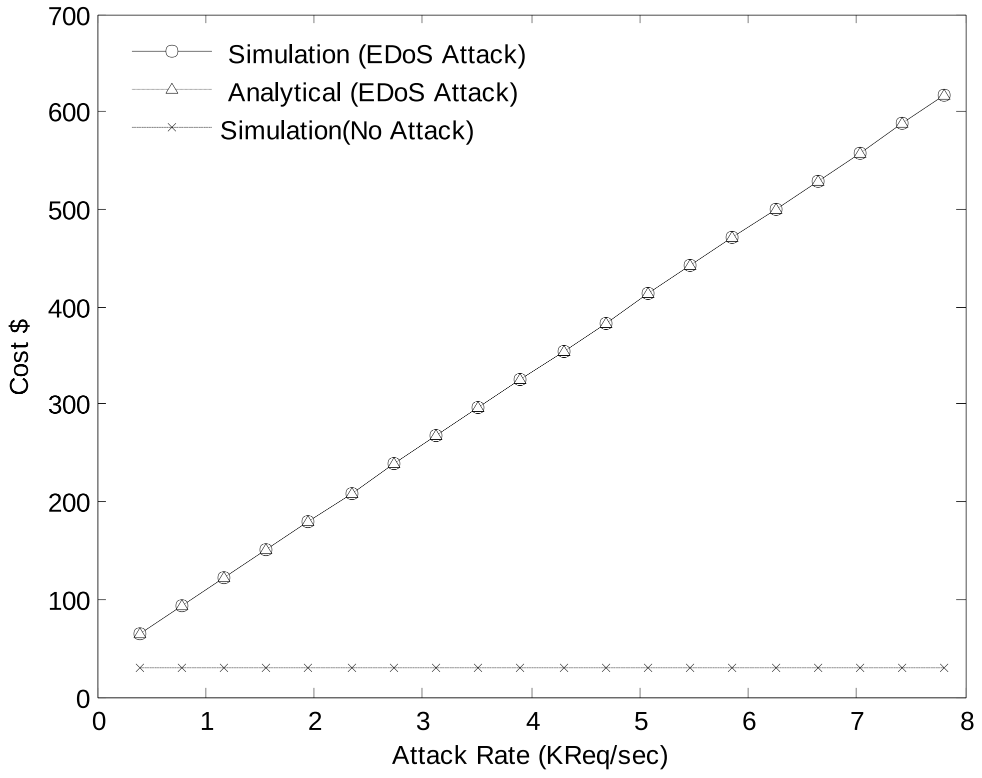


Figure 3.15: Cost evaluation, provision triggered at 80%

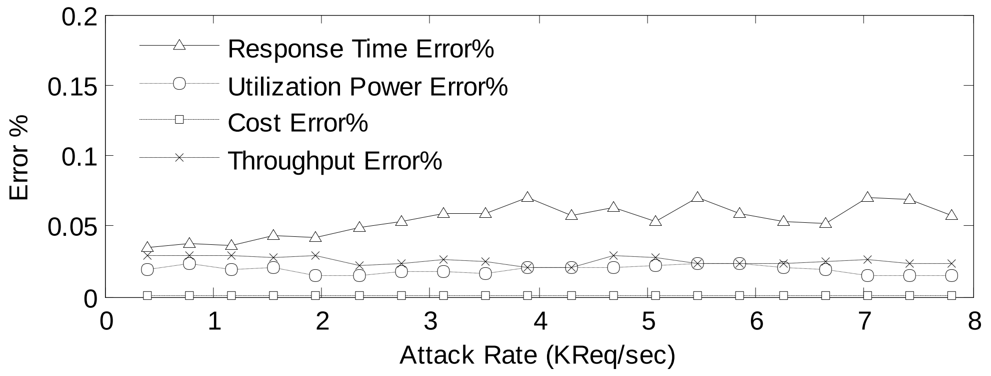


Figure 3.16: Relative Error percentage of the analytical model, provision triggered at 80%

CHAPTER 4

EDOS-SHIELD MITIGATION TECHNIQUE

In this chapter, we propose a novel solution, namely EDoS-Shield, to mitigate the Economic Denial of Sustainability (EDoS) attack in the cloud computing systems. We design a discrete simulation experiment to evaluate its performance and cost metrics. In addition, an analytical model has been developed to validate the results obtained from the simulation model.

This chapter is organized as follows. First, Section 4.2 presents an introduction about the problem statement covered in this chapter. In Section 4.2, we discuss the related works and in Section 4.3 we present the proposed architecture. The experimental setup including the simulation model and results are discussed in Sections 4.4 and 4.5, respectively.

4.1 Introduction

Cloud computing allows us to scale up our servers and to serve a large number of requests for a service. The introduction of resource-rich cloud computing platforms, where adopters are charged based on the usage of the cloud's resources, known as "pay-as-you-use” or utility computing, has transformed the Distributed Denial of Service (DDoS) attack problem in the cloud to a financial one. This new type of attack targets the cloud adopter’s economic resources, and is referred to as Economic Denial of Sustainability (EDoS) attack [48].

A well-known tactic taken by EDoS attacks is to remotely control zombies to smoothly (with low rate to avoid triggering security alarms) flood a targeted cloud service by undesired requests. As a result of such undesired requests, and because of the cloud elasticity notion, the service usage will be scaled up to satisfy the on-demand requests. And because of the “pay per use” notion, a cloud adopter’s bill will be charged for those undesired requests, leading to service withdrawal or bankruptcy.

What makes this more disastrous is that it is extremely difficult to selectively filter the malicious traffic without impacting the service as a whole. This also means that any proposed mitigating technique must be highly intelligent; otherwise, the technique itself could be utilized by the attackers as a source of EDoS attack.

In this chapter, we propose a novel mitigation technique against EDoS attack in Cloud Computing, namely EDoS-Shield. The main idea is to verify whether the requests coming from the users are from a legitimate person or generated by bots. This is achieved by forwarding the first request to a verifier node in our proposed architecture. This verifier node is responsible for the verification process and for updating the whitelist and blacklist based on the results of this verification process. The subsequent requests coming from the bots will be blocked by a virtual firewall since their IP addresses will be found in the black list. On the other hand, the subsequent requests coming from legitimate clients will be forwarded directly to the target cloud service since their IP addresses will be found in the whitelist. As a result, only the requests from legitimate clients will reach the target cloud service and thus mitigating the EDoS attack.

4.2 Related Work

Techniques for mitigating EDoS attacks are much needed for protecting the cloud infrastructure against the rippling effect of cost incurred on legitimate users through EDoS attacks. In this section, we discuss the related work including sPoW [56] which is a capabilities-based approach dedicated to mitigate EDoS, CloudWatch [107] which is an auto scaling technique, enabled by Amazon, and some overlay-based approaches that could be adopted to mitigate EDoS attacks in cloud computing.

CloudWatch [107] is an auto scaling technique enabled by Amazon as a control technology that will reduce the effects of the EDoS attacks. CloudWatch is a web service that provides monitoring for cloud resources by which clients will be able to define boundaries that would limit the elasticity of their cloud platforms and thus reducing the effect of the EDoS attacks. However, this cannot be considered as an efficient practical solution for the EDoS attack since the user account could still be charged to some extent defined by quota due to such attack. In addition, when elasticity reaches the upper bound, the cloud service freezes and thus legitimate clients will be denied service till refreshing the quota again leading to a behavior similar to a DoS attack.

HinKhor and Nakao [56] have described a self-verifying proof of work, sPoW, as a cloud-based EDoS mitigation mechanism by introducing an asymmetric step before committing the server's resources. The server requires a proof of work from the client, before committing its resources to the client. Clients expend their resources to solve a "crypto-puzzle" and submit a proof of the solution as an embedded signal (capability) within the packets. A server has to generate a "crypto-puzzle" to protect the connection server channel. A crypto-puzzle consists of both the encryption of channel information (such as IP address and port number) and the concealed encryption key with k bits representing the puzzle difficulty. A puzzle requester running on the client-side expands the client resources by brute-forcing these k bits to discover the server channel information where it can submit an initial connection request. sPoW, which mediates the communication between clients and the protected server, has to verify the puzzle correctness and prioritize the requests based on the puzzle difficulty thereby reducing the number of application-level EDoS connection requests.

To our knowledge, the only dedicated approach proposed to mitigate the EDoS in the cloud computing is sPoW approach. However, sPoW has several limitations described as follows. The first limitation is with regard to the asymmetric computation power for the end users. Since computational puzzles give advantage to end-users with faster CPUs, the mobile devices which have less power cannot receive services. Green et al. [108] have identified a problem of computational disparity when attackers use a Graphics Processing Unit (GPU) to solve the puzzle. Green et al. claimed that the computational disparity is so great that a legitimate client cannot receive services. The second limitation appears when attackers send huge number of requests for puzzles without solving them. In this case, the server has to generate a channel for each request which leads to a puzzle accumulation attack. One solution to this problem is to use shared channels; and in this case, there is a need to investigate the algorithm used to manage these channels. The third limitation is concerned with the fact that since the server is involved in the puzzle generation process, the cost of such generation at the server side has to be investigated considering that the attacker could send a large number of requests. The fourth limitation is related to the case when an attacker requests high difficulty puzzles without solving them. In this case, the server has to generate a channel with a high difficultly puzzle leading to a problem of difficulty inflation where the legitimate clients also have to solve such high difficulty puzzles.

Overlay networks are normally used to hide the location of a target server and thus prevent DoS attacks. Several studies have proposed using overlay networks to proactively defend against DoS and DDoS attacks [109, 110, 111, 112].

Stavrou et al. [110] have proposed an architecture based on Secure Overlay Services (SOS) [109], WebSOS, to protect a web service against DDoS attacks using a filtering mechanism around the protected web server that admits HTTP traffic from only trusted sources known to overlay nodes. Legitimate clients have to first pass the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [113]. This graphical Turing test is posed by an overlay node to decide on whether a client is allowed to get connected to a trusted approved location so as to access the web server. However, Karwaczynski and Kwiatkowski [114] evaluated the performance of WebSOS on the Planet Lab testbed, and reported 2 to 10 times performance degradation due to non-direct routing.

Fosel [111] is another overlay based system for DDoS defense. Fosel is a filtering mechanism with the help of an overlay security layer including an overlay network, secret green nodes, authentication techniques, and pre-defined rules for the purpose of filtering illegitimate packets. The green nodes are overlay nodes which are kept secret from the public. The application site has to randomly select a green node, or possibly a few green nodes, by sending a message to an overlay node to be selected as a green node related to that site. The idea is that the legitimate messages have to be verified by the overlay nodes, and then be forwarded through the overlay nodes to the secret green nodes related to the target. The Fosel filter, installed on the target, admits only those packets coming from the specific green nodes related to the application site.

Ping and Nakao [112] have proposed a cloud-based overlay network, named CLAD, as an on-demand network service to protect the web servers against the DDoS. Implementing the overlay network in the cloud infrastructure is a promising idea since the cloud has an aggregated capacity exceeding that of most of the botnets. A CLAD node, at least one, mediates the traffic targeting the protected server to verify the clients using graphical Turing tests. When the clients pass the tests, the CLAD relays the request to the protected server. This is the closest work to our proposed approach as it uses the overlay network implemented on the cloud computing utilizing the availability and the scalability of the cloud resources.

However, our proposed approach differs from the previously discussed overlay-based approaches in several important ways. First, the overlay routing could increase the end-to-end latency [114, 115] due to indirection where the overlay network mediates all the traffic between the clients and the target server. Whereas, our proposed approach significantly reduces such delay since subsequent packets after the first successful request would be forwarded directly to the protected cloud service. In other words, the indirect routing in our architecture would only happen for the first request coming from the client and the rest of the requests will use direct routing. Second, in most of the overlay-based approaches, the location of overlay nodes, i.e., IP addresses are clear to the public including attackers, and thus these nodes could be exposed to different attacks. However, our proposed technique does not require that the IP addresses of the verifier nodes be clear to the public. Third, most of the overlay-based approaches are using an overlay network as a security layer to hide the location of the protected server or application site. Such techniques raise a problem of feasibility of location-hiding [116]. However, our approach does not suffer at all from this problem as it is not required in our proposed approach to hide the location of the protected cloud service. Fourth, our focus is on mitigating the economic effect of the DDoS targeting the cloud service which was reported in [48] as an EDoS attack.

4.3 Proposed Mitigation Architecture and Approach

Figure 4.1 shows the proposed architecture of the EDoS-Shield for mitigating the EDoS in a cloud computing environment. The main components of the architecture are virtual firewalls (VF) and verifier cloud nodes (V-Nodes). The virtual firewalls work as filter mechanisms based on whitelist and blacklist that hold IP addresses of the originating nodes. And, the verifier cloud nodes update the lists based on the results of the verification process.

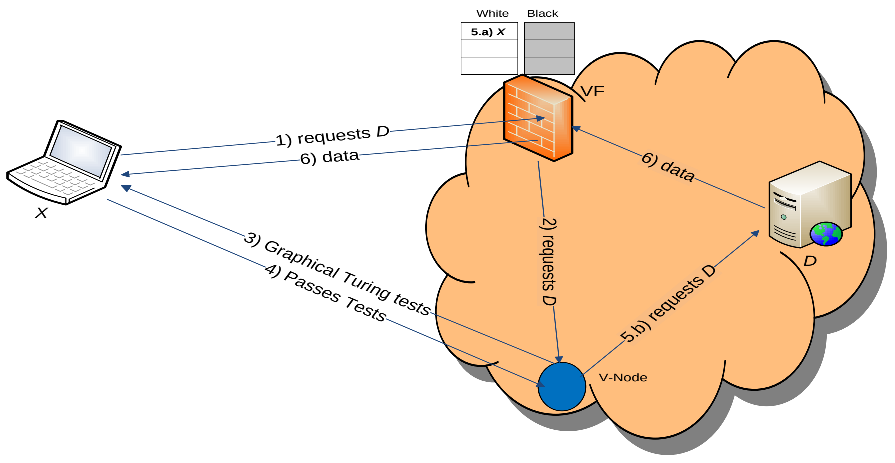


Figure 4.1: The proposed EDoS-Shield architecture for mitigating EDoS

The virtual firewall can be implemented in the cloud as a virtual machine that has the capabilities of filtering and routing. The VF uses two lists, a whitelist and a blacklist, to make a decision regarding the incoming packets from outside the cloud and destined to some services hosted in the cloud. The whitelist is used to track the authenticated source IP addresses so that the incoming traffic originating from these addresses will be allowed to pass the firewall towards the destined services. The blacklist is used to hold those unauthenticated source IP addresses so that the firewall will drop the incoming packets originating from these IP addresses. These two lists have to be updated periodically.

Another component in our proposed architecture is the verifier nodes (V-Nodes) which are represented by a pool of virtual machine nodes implemented based on the cloud infrastructure. The V-Nodes constitute a cloud-based overlay network. A V-Node has the capabilities to verify legitimate requests at the application level using graphic Turing tests [117], such as CAPTCHA [118, 119, 120]. Another role of the V-Node is to update the lists used by the VF as was explained earlier. If the application request gets verified successfully, then the source IP address of that request will be added to the whitelist and the request will be forwarded to the destined service in the cloud. All the subsequent packets passing through the VF and having this IP address as a source address will be forwarded to the destined service. If the application request fails, then the source IP address of that request will be added to the blacklist, and subsequent packets originating from that source IP address will be dropped.

Figure 4.2 shows a case of a legitimate request from a client, *X*, where the first request gets verified by a V-Node and passes the Turing test. Thus, its source IP address, *X*, has been added to the whitelist and the subsequent requests from *X* to the destination *D* have been forwarded directly to *D*.

Figure 4.3 shows a case of a request coming from an attacker (a bot), *Y*, where the first request gets verified by a V-Node and fails the Turing test. Thus, its source IP address, *Y*, has been added to the blacklist and the subsequent requests from *Y* to the destination *D* have been blocked by the VF.

Since the requests originating from the bots, i.e., compromised machines, will fail at the verification stage, all the automated malicious requests will not reach the victim in the cloud. Therefore, the customer will not be charged for such attacker activities.

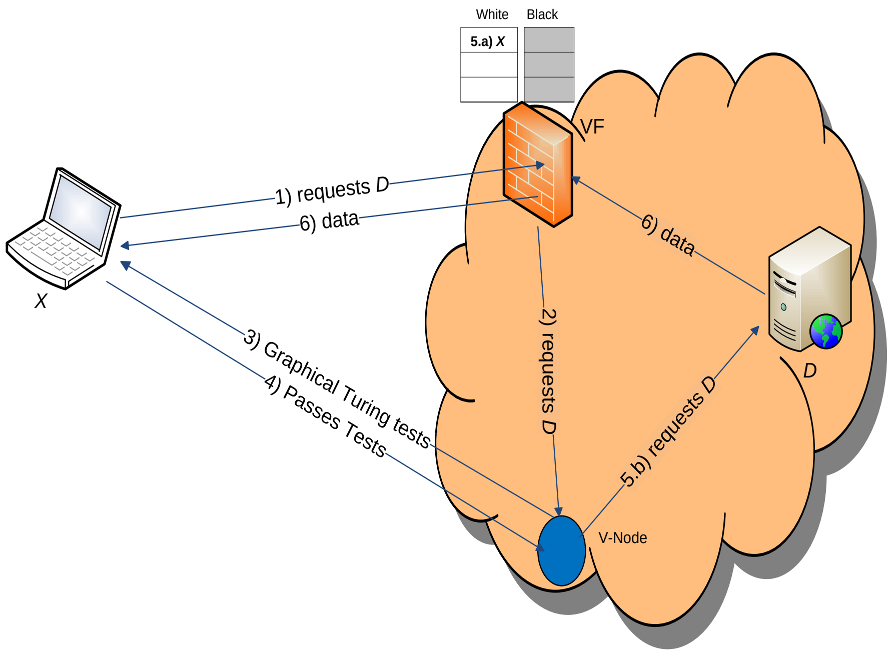


Figure 4.2: Normal request scenario

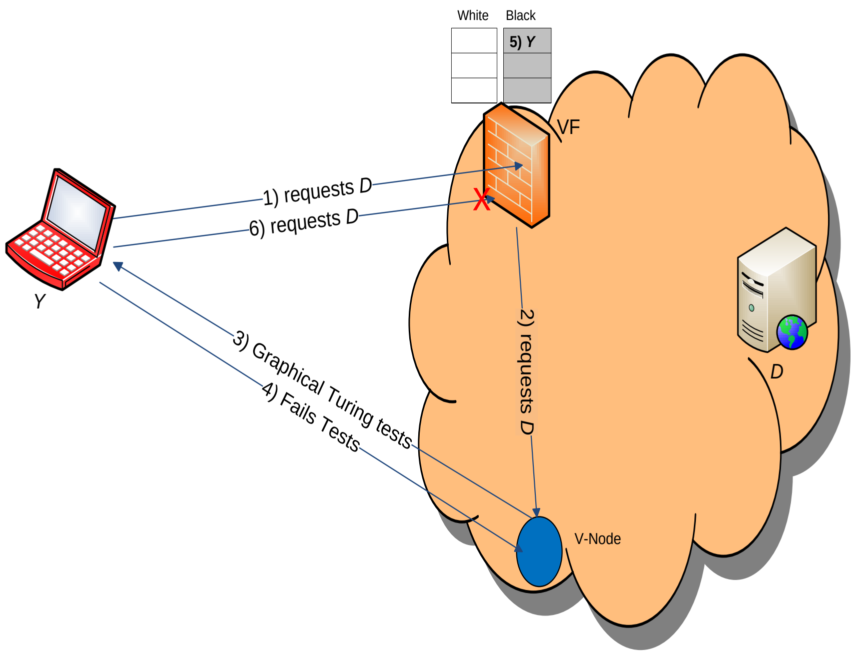


Figure 4.3: Bot requests scenario

4.3.1 Security Issues

The goal of such proposed architecture is to mitigate the risk of the EDoS attacks against the cloud services. The main idea is to verify whether a request coming from a user is originated by a human or it is an automated one. The objective of such verification is to distinguish between legitimate and malicious users. This is achieved by directing the first request to a V-Node that is responsible for the verification process using CAPTCHA. The subsequent requests coming from the bots will be blocked by the VF (because they will fail the verification phase) and will not reach the victim (i.e., customer) and thus the customer will not be charged for these requests.

Such proposed architecture is mainly used for protecting the cloud application services from the impact of application EDoS attacks. The non-HTTP traffic such as network layer attacks which targets the protected cloud service will be dropped by the VF. This is due to the fact that we are mainly protecting cloud application services, and the cloud infrastructure only allows Web traffic to pass through it.

One challenge related to security is the IP spoofing attacks. For our proposed architecture, it is a must to protect the architecture from the IP spoofing attacks since the decision to forward a packet or to drop it is mainly based on the source IP address present in the whitelist and blacklist. To overcome such problem, techniques like [121, 122, 123, 124, 125] could be used to detect and prevent the IP spoofing attacks. Algorithm 4.1 and Algorithm 4.2 show the actions taken by the VF and the V-Node when considering that the architecture is protected against the IP spoofing attacks.

Algorithm 4.1: VF Actions

```
Input:
     P← Packet
     S← Packet source IP address
     D← Packet destination IP address
     B← Blacklist
     W←Whitelist
Begin:
If (S ∊ W  && S∉B  )
      Forward P to D
ElseIf (S ∊B )
      Drop P
    Else
        Forward P to a V-Node
End
```

Algorithm 4.2: V-Node Actions

```
Input:
     P←Packet
     S← Packet source IP address
     D← Packet destination IP address
     B← Blacklist
     W←Whitelist
Begin:
If (S ∉B  && S∉W) {
     Send to S a graphic Turing test
     If (Turing test passes) {
            W←W+S
            Forward P to D.
       }
     Else
            B←B+S
END
```

4.3.2 Performance

The V-Nodes constitute an overlay network implemented in the cloud computing system. Using an overlay network to mediate the traffic between clients and target servers could increase the end-to-end latency due to indirection even in the normal mode where there is no attack as it was reported in [114, 115]. However, one contribution of our proposed architecture in terms of performance is to reduce the overhead caused by the indirect routing when there is no attack since the indirect routing in our architecture would only happen for the first request coming from the clients while the rest of requests would use a direct routing.

To overcome the bottleneck problem due to a centralized VF deployment, it is desirable to deploy VFs in the cloud considering load balancing issues. Firewall load balancing distributes traffic flows to one or more firewall farms which are a group of firewalls connected in parallel. The deployment of such firewall technology has been discussed in the literature [126, 127, 128]. O'rourke et al. [129] have presented a method to protect a firewall load balancer from a DoS attack and that work got a patent from Google Inc.

4.3.3 Deployment

Regarding the deployment of our proposed technique, the proposed architecture requires no modifications in the client side, the protected cloud service side, or the Internet network protocols. It requires only deploying a VF in the cloud computing system infrastructure and implementing V-Nodes as a pool of virtual machines which can grow in numbers to defeat the DDoS attack based on the scalability property of the cloud computing system.

The deployment of virtual firewalls in the cloud computing system has been discussed in the literature [130, 131, 132, 133, 134]. For example, Amazon EC2 provides a complete firewall solution as it is discussed in [130]. Virtual firewalls can be implemented based on VMware vShield Edge, and vShield App technology. vShield Edge is a virtual firewall appliance that can be provisioned on-demand and its services enabled on the fly to meet the flexibility requirement of cloud deployments [131]. Cisco Virtual Security Gateway (VSG) is a virtual firewall service that provides security policies on the virtualization layer of the data center or cloud environment [132].

The cloud provider has to offer the proposed EDoS-Shield technique as an on-demand network service. The customers who are willing to protect their cloud services have to request such offered service. The pricing model for such services should be attractive and the customer has to pay for such services as a package for a utility, i.e., the payment should be based on the period of use, e.g., per month or per year, rather than consumption based. When the customer needs to be unprovisioned from such service, the firewall rules have to be updated to forward the traffic related to the customer to the target service without any inspection.

There are several existing Cloud firewalls that provide a general security layer for protecting the cloud service, such as [23, 135, 136]. However, EDoS Shield mitigation technique requires special firewall capabilities including dynamic rules updating, dealing with blacklist and whitelist, and routing capabilities. Thus, we suggest deploying VF as a large EC2 instance that has a capacity of four EC2 computing units [137].

4.4 Analytical Model

In this section, we present the analytical model of EDoS-Shield mitigation technique. Figure 4.4 shows the queuing network that represents the architecture of EDoS-Shield shown in Figure 4.1. The input to the model is an aggregated traffic from different sources including attackers' traffic. We have considered the Poisson nature of the incoming traffic.

The results out of the analytical model include the performance and cost metrics at the cloud service side. The performance metrics include the end-to-end response time, computing resources utilization, and the throughput rate.

Since we are studying the performance at the cloud service side, metrics include computing resources utilization, throughput, and the cost can be obtained from the sub network represents the cloud service. The end-to-end response time involves considering the whole queuing network as requests encounter delay while traveling through the mitigation devices to reach the target cloud service.

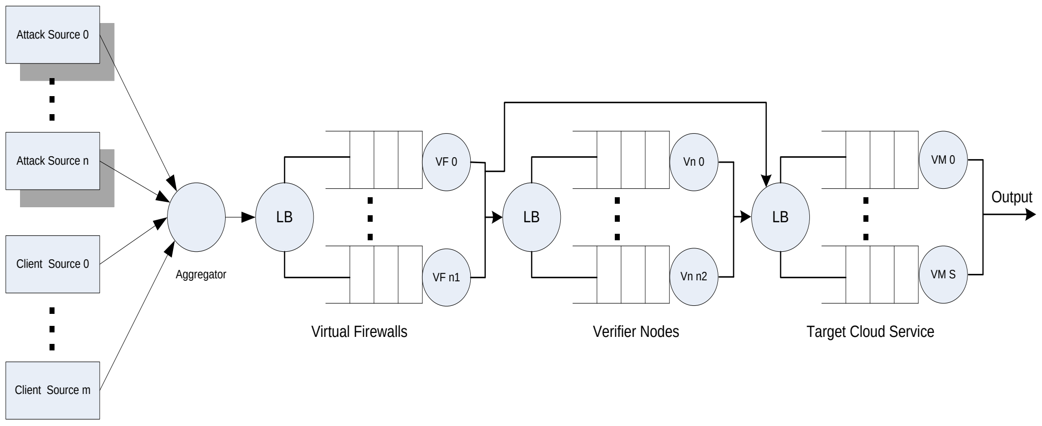


Figure 4.4: EDoS-Shield queuing model

For analyzing the queuing networks considered for the purpose of calculating the end-to-end response time, we use the decomposition method discussed by Chandy and Sauer [138]. First, the network is broken up into subsystems including the VFs, V-Nodes, and the cloud service subsystems. Then, for each individual queuing subsystem, we get the average delay considering their related input and capacity. Finally, the end-to-end response time is calculated by aggregating all the delays along paths from the source to the cloud service as a destination. For further explanation, Figure 4.5 shows the considered subsystems and paths from the sources up to the target cloud service. In fact, most of the requests arriving at the cloud service go through path P1, as only few requests go through path P2 for the verification purpose and the subsequent requests will be forwarded to the cloud service via P1. Thus, for calculating the end-to-end response time, we only considered the subsystems along P1 which includes the outgoing link of the sender (Link1), the firewalls group (VFs), the incoming link of the cloud service (link2), and the cloud instances group (clouds).

Figure 4.6 shows the corresponding network queuing model for the selected path P1. Links have been modeled as *M/D/1* queuing system with exponential arrival time and a deterministic service time representing the link capacity.

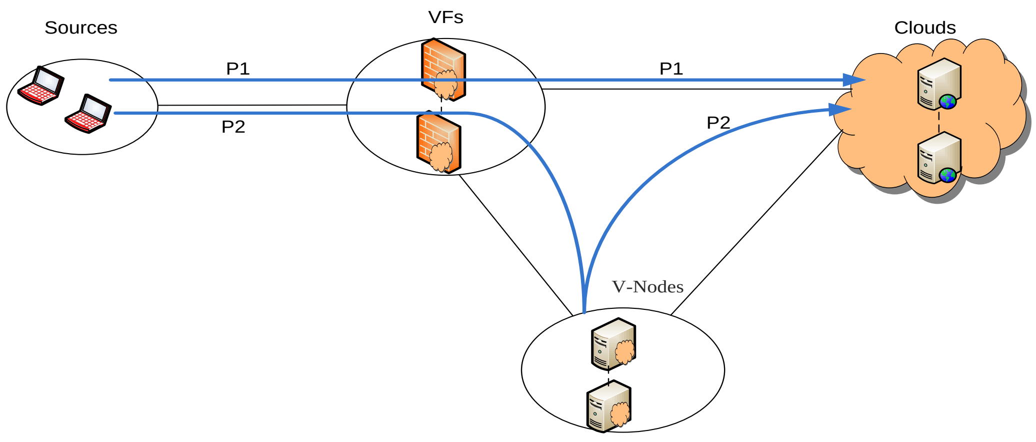


Figure 4.5: Traffic flow paths from sources to a destination

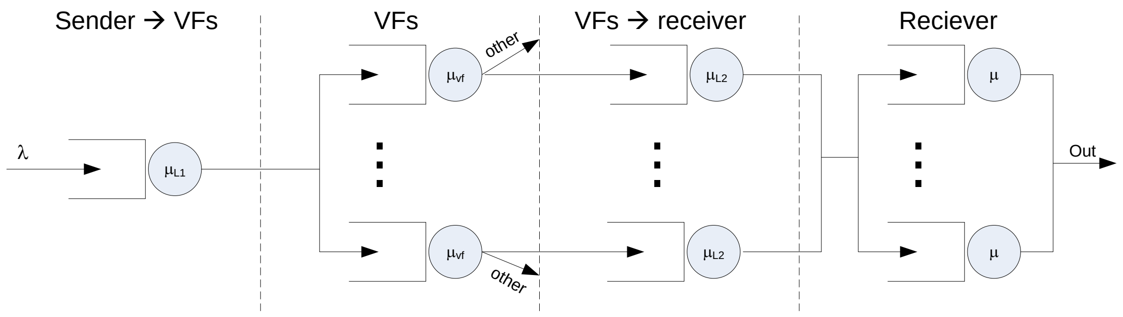


Figure 4.6: Network queuing model for path P1

The average delay for a request passing through an *M/D/1* queue can be computed as follows [139].

$$link\_delay = \left(1 - \frac{\lambda_{in}}{\mu_{link}}\right) \Big/ \left(\mu_{link} - \lambda_{in}\right) \tag{4.1}$$

where λ_{in} is the mean request arrival rate and μ_{link} is the mean link request service rate, i.e., its capacity.

It is worth noting that using a deterministic service time for the involved links violates the analysis by decomposition of queuing networks in [138], which assumes exponential service times for all network elements including links. However, when changing the models for links from *M/D/1* to *M/M/1*, a negligible difference was observed which comes in line with the claim by [140] regarding the insensitivity of the main system performance to violations of the homogeneity of service times.

Clouds and VFs can be modeled as an open queuing network model of parallel single queues. Such a model has been discussed in Chapter 3. Thus, the average delay encountered for a request at VFs or Clouds subsystems can be computed as:

$$Rt = \frac{S}{S\mu - \lambda} \tag{4.2}$$

Where S is the number of instances in the group, λ is the total arrival rate to the group, and μ is the service rate of the instance in the group assuming that all the instances in each group have identical capacity of μ.

Finally, the total delay for path P1 can be determined as follows:

Delay=Delay_VFs+ Dealy_link2+Delay_Clouds,

where we have ignored the delay encounter because of the link from the source to the VF since we focus on the performance of mitigation technique which starts at VFs.

Thus, the end-to-end response time, *Rt*, can be computed based on Eq. (4.1) and Eq. (4.2) to be as follows:

$$Rt = \frac{S_1}{S_1\mu_1 - \lambda} + \left(1 - \frac{\lambda_1}{\mu_{link2}}\right) \Big/ (\mu_{link2} - \lambda_1) + \frac{S}{S\mu_2 - \lambda_2} \tag{4.3}$$

Where μ_{link2} is the capacity of the link from the VF to the cloud service, S_1 is the number of the instances representing the *VFs*, S is the number of instances representing the cloud service, μ_1 is the processing rate of a *VF*, μ_2 is the processing rate of a cloud instance, λ_1 is the rate of the requests forwarded by a VF toward the target cloud service (receiver), and λ_2 is the rate of the total requests arrived at the cloud service. Since we have assumed using Elastic Load Balancer (ELB) in front of the cloud service, the rate will be evenly distributed among the available instances.

The computing resources *U* at the cloud service can be calculated as:

$$U = \frac{\lambda_2}{S\mu} \tag{4.4}$$

One of the measurements that we have studied, and which is the target of an EDoS attack, is the cost associated with both the computing resources and the bandwidth on the cloud service side. A cloud adaptor has to pay for the computing resources, the network traffic volume, and for the storage service, if required. In our work, we consider the cost related to both the computing usage and bandwidth usage.

The total cost can be calculated as follows:

$$COST = \left(\Pr ice_{bw} \times \lambda_2 + \Pr ice_{com} \times U \times S\right) \times T \tag{4.5}$$

Where $\Pr ice_{bw}$ is the price per GB, λ_2 is the arrival rate at the cloud service side measured in GB/s, $\Pr ice_{com}$ is the base price charged for the smallest amount of computing resources per hour per instance, *U* is the average utilization of the computing resources calculated in Eq. (4.4), *T* is the total running time in hours, and S is the number of running instances during time *T*.

Regarding the cost of the mitigation technique, the cloud provider has to offer the proposed EDoS-Shield technique as an on-demand service. The pricing model for such services should be attractive and the customer has to pay for such service as a package for a utility, i.e., the payment should be based on the period of use, e.g., per month or per year, rather than consumption based. Thus, we are assuming that the cost of mitigation is based on the cloud provider who is offering it as a service. For example, The Cloud Leverage Cloud IPS/Firewall delivers multi-ten gigabit security performance for $150 per month, to include one terabyte of traffic [135].

For the throughput, it can be calculated directly from the little's formula. When the number of running instances at the cloud service side is S, the arrival rate λ_2 will be distributed among *S* instances due to load balancing. Thus, each will have λ_2 / S as an arrival rate. According to the little's formula, the throughput of *M/M/1* queuing system, with arrival rate of λ_2/S and a service rate of μ, is $\mu \times \rho = \mu \times (\lambda_2/S\mu) = \lambda_2/S$. As a result, the average throughput at the cloud computing service having *S* running instances will be λ_2.

4.5 Selecting System Parameters

For the target cloud service as well as the V-Node, we assumed that each instance to be a small instance that has a capacity of 100 Req/sec as it was discussed by Catteddu and Hogben [3].

The average service time of VF can be estimated by the sum of the average processing time for a request in the device driver and the average processing time for the rule set [141].

Liu and Wee [142] reported that an Amazon EC2 instance can handle at most 400 Mbps of combined ingress and egress traffic. We assume requests with an average size of 580 bytes [83], by simple calculations, $(580 \times 8)/(400 \times 10^6)$, the average processing time for a request in the device driver is about 11.6 μs.

Regarding the average processing time for the rule set, Salah et al. [141] measured the average processing time per rule to be 0.05 μs using Intel Pentium 4 processors running at 3.2 GHz. However, the capacity of the Amazon large EC2 instance is 4 EC2 Computing Units (ECU) where “One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0 - 1.2 GHz 2007 Opteron or 2007 Xeon processor” [144]. With the 3.2 GHz machine, 0.05 μs is equivalent to 0.16 CPU cycles, which in turn is equivalent to approximately

$0.16 \times (1/4.4) \times 10^{-6}$ or 0.036 μs for the large EC2 instance assuming that each ECU has an average capacity of 1.1 GHz.

According to the statistical measurements performed by Wool [145] about the size of the firewall rule set from operational firewalls, the maximum size of the rule set is 2671 rules. For the worst case, when assuming that each request hits the bottom rule, the average processing time for the rule set is about $2671 \times 0.036 \mu s$ or 0.0962 *ms*. Thus, the average service time of the VF, $1/\mu_1$ in Eq. (4.3), is as follows:

$$\frac{1}{\mu_1} = 11.6 \text{ µs} + 0.0962 \text{ ms} = 108 \text{ µs} .$$

For the analytical model, Eq. (4.3) has been used to compute the average response time considering the following settings. The capacity of the links to the cloud service μ_{link2} is assumed to be 10 Gbps. The number of the running virtual firewalls, S_1, and the number of instances of the cloud service, *S*, are assumed to be variable based on the provision technique parameters that have been set as follows.

1- The number of initial running instances is 5.

2- The provisioning overhead is of 55.4 sec.

3- The upper utilization threshold is of 80%.

4- The scaling size is variable based on the optimization technique discussed in chapter 3.

Since the attack rate increases with the time, we have considered using only the upper threshold to add more instances to the scaling group.

The cost has been calculated based on Eq. (4.5). The $\Pr ice_{com}$ has been set to \$0.46 for the large EC2 instance uses for the *VF*, and \$0.115 for the small EC2 instance uses for the cloud service instances. Such prices recently have been reported in Amazon for on-demand instances running on the Windows operating system [81]. For the cost associated with the bandwidth allocation, $\Pr ice_{bw}$, we have used a base price of \$0.01 per GB in/out data transferred based on the reported prices of Internet data transfer "in" and "out" of Amazon EC2 [81].

4.6 Simulation Setup

We have conducted a discrete-event simulation to evaluate the performance of the EDoS-Shield mitigation technique in terms of key performance indicators including end-to-end response time, computing resources utilization, and throughput. Since the EDoS attack is mainly targeting the cloud adopter, we have also evaluated the cost associated with the computing resources and bandwidth allocations at the cloud service side.

In the simulation work, we have conducted three simulation-based scenarios. In the first scenario, we have considered different attack rates while using the EDoS-Shield to mitigate the attack. The second scenario is for the optimal case where there is no attack targeting the cloud service. The last scenario is for the worst case where the attack rate gradually increases while not using the EDoS-Shield to mitigate the attack.

The simulation followed closely the guidelines given by Law and Kelton [82] including the use of independent replications with different initial seeds that were ten million apart, and avoiding any overlapping in the random number streams during the simulation. We have considered the Poisson nature of the incoming traffic.

Regarding the LB which is a part of the model shown in Figure 4.4, we have used a randomized algorithm based on uniform distribution to implement the load balancing process [72].

We have assumed a fixed input rate of 400 Req/sec representing the rate of the legitimate requests coming from clients and a variable input rate ranging from 400 Req/sec to 8000 Req/sec representing the rate of the attack traffic.

Finally, the output of the proposed analytical model has been compared to the simulation results. The relative error percentage is used to measure the accuracy of the queuing model results compared to the simulation model results. The percentage of the Relative error has been determined based on Eq. (3.9).

4.7 Results and Discussions

Figure 4.7 shows the obtained results regarding the end-to-end response time. The results show that when using the proposed EDoS-Shield, the corresponding response time is approximately constant and very close to the optimal case where there is no attack. This is due to blocking the attack's requests for reaching the protected cloud service. However, the little increase in the end-to-end response time for the EDoS-Shield when compared to the optimal scenario is due to the delay that requests encounter at the VF. In addition, the results show considerable degradation effects on the cloud service performance in terms of the response time when the attack rate increases and not implementing the EDoS-Shield mitigation technique.

Regarding the results of the computing resources utilization, Figure 4.8 shows that with the proposed EDoS-Shield mitigation technique, the computing resources utilization is not affected due to the attack rate since the attack requests will not reach the target cloud service. As a result, the utilization will remain constant and identical to the one of the optimal case with no attack.

Figure 4.9 shows the results of the cost associated with the computing resources and the bandwidth allocations. The results show that with the proposed EDoS-Shield technique, there is a little extra cost compared to the base case where there is no attack. Such extra cost of EDoS Shield is due to cost incurred at the VF and the verifier instances. However, the cost when not applying the mitigation technique increases significantly with the increase of the attack rate which consumes more resources as was also indicated clearly by the utilization results presented in Figure 4.8. For example, when the attack rate is 8000 Req/sec, the results show a cost of about \$654, \$24, and \$10 for the no mitigation case, EDoS-Shield case, and base case respectively.

Regarding the throughput rate, it is expected that the throughput of the legitimate requests will not be affected by the attack rate even without applying the mitigation technique. This is due to the fact that the targeted cloud service is an on-demand cloud–based. According to the nature of the cloud computing system, i.e., scalability nature, we are assuming that there are enough on-demand cloud resources to be provisioned to the cloud instances executing the service. As a result, there is no degradation of the throughput rate of the legitimate requests in both scenarios. Figure 4.10 shows the same expected trend of the throughput of the legitimate requests.

Figure 4.11 shows the results obtained from the analytical model regarding the EDoS-Shield compared to its simulation results. In general, the two models are very close to each other for all the studied metrics indicating a validation of the analytical model for the EDoS-Shield mitigation technique.

When comparing the queuing model results to the simulation model results, the obtained relative errors percentage presented in Figure 4.12 show a good accuracy for all the studied metrics. For the response time, results show a maximum relative error percentage of about 0.2%. For the utilization, the cost, and the throughput metrics, results show error percentages less than 0.1% which indicates that the two models are almost identical for all the metrics.

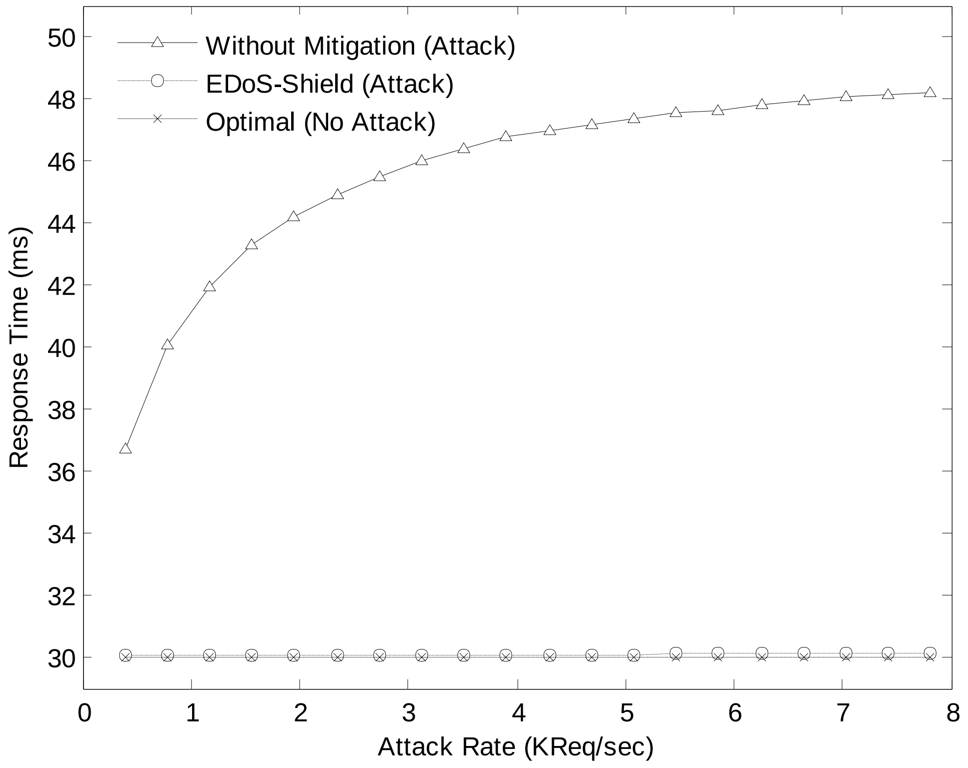


Figure 4.7: Simulation results for the end-to-end response time.

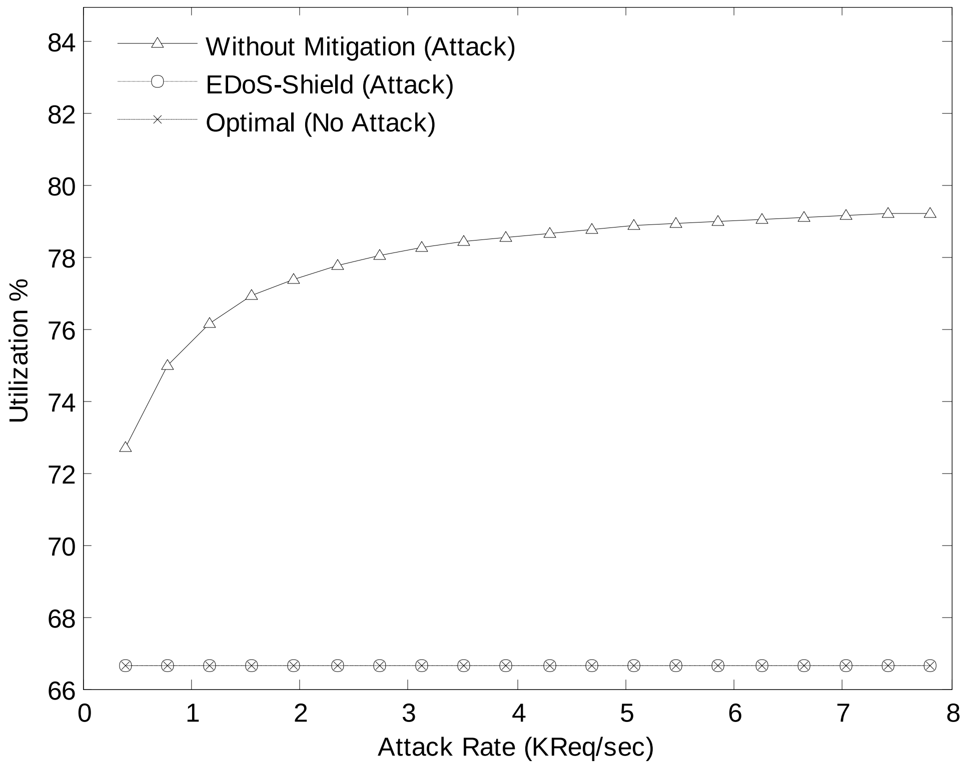


Figure 4.8: Simulation results for the computing resource utilization.

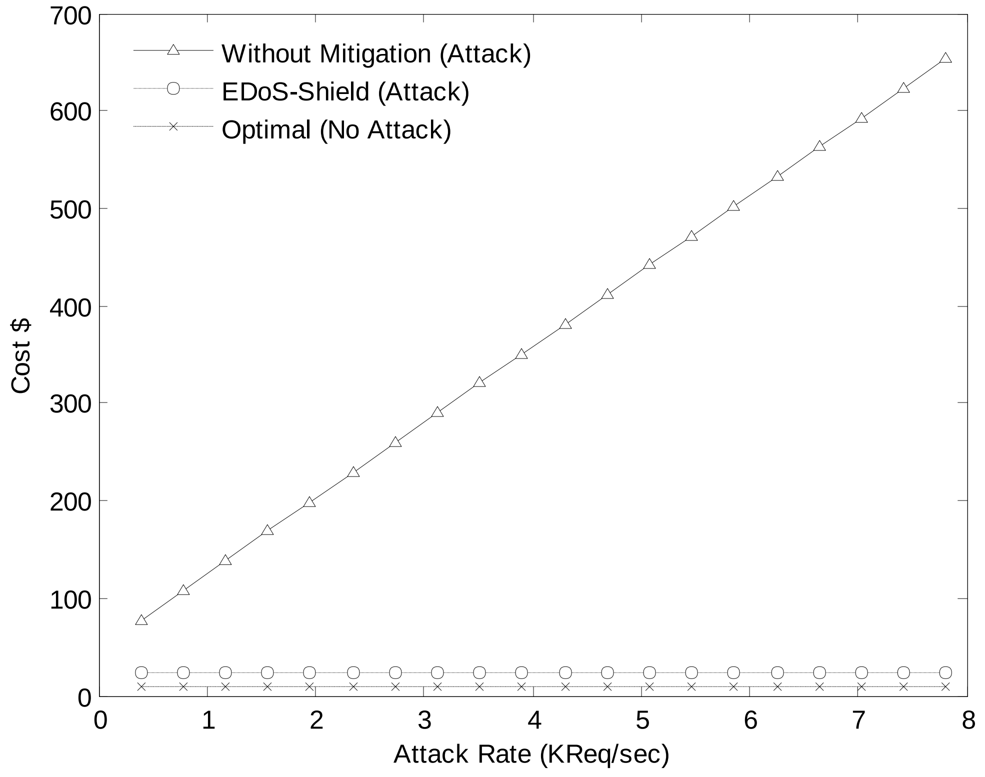


Figure 4.9: Simulation results for the cost.

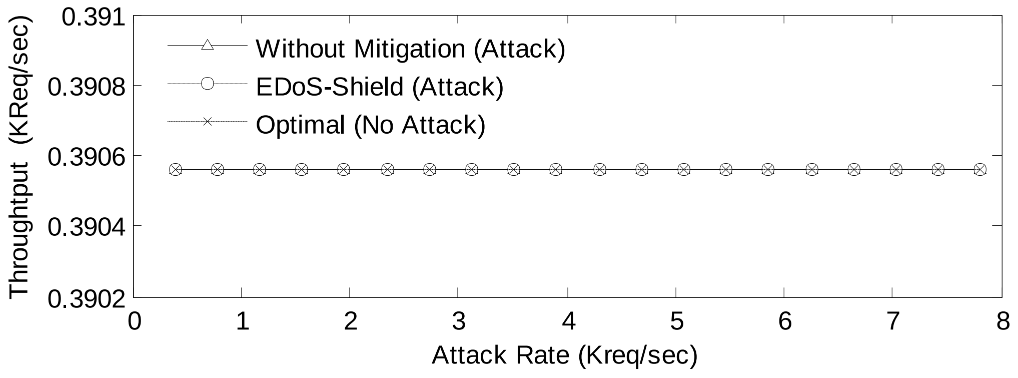


Figure 4.10: Simulation results for the throughput rate.

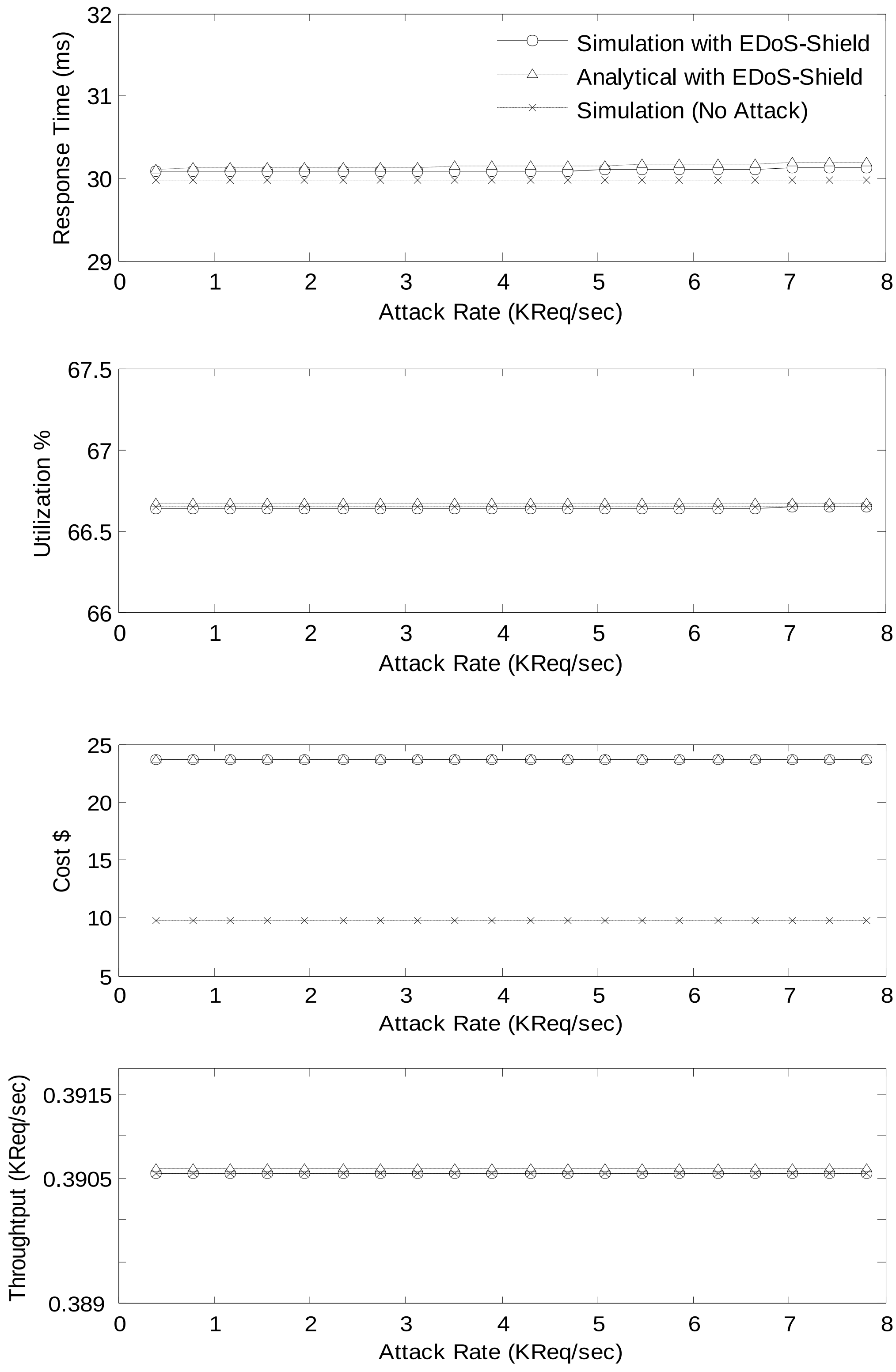


Figure 4.11: Analytical vs. simulation results with respect to the evaluated metrics

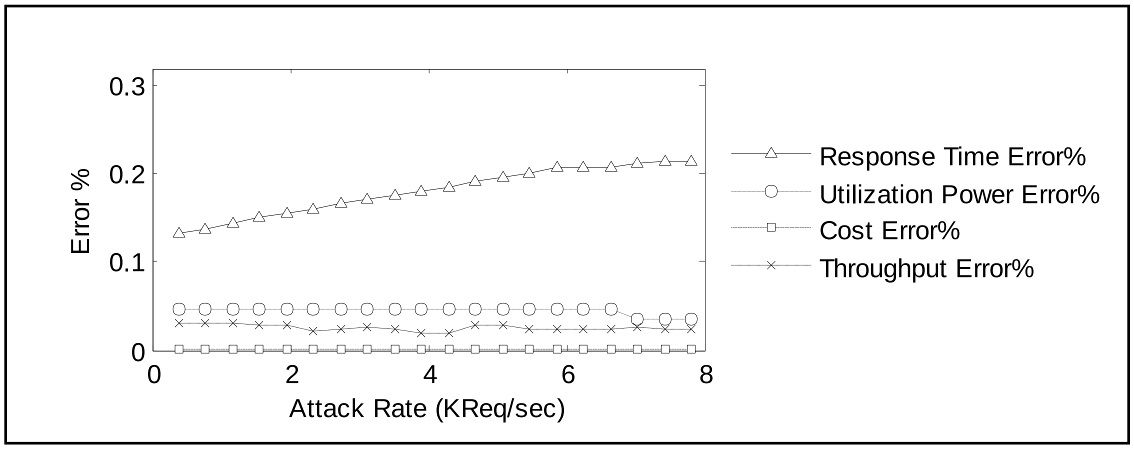


Figure 4.12: Relative error percentage of the analytical model

CHAPTER 5

ENHANCED EDOS-SHIELD FOR MITIGATING EDOS ATTACKS ORIGINATING FROM SPOOFED IP ADDRESSES

In this chapter, we advocate a novel solution as an enhancement to our work presented in chapter 4, namely EDoS-Shield [146], to mitigate the EDoS attacks originating from spoofed IP addresses. We design a discrete event simulation experiment to evaluate its performance. The simulation model has been verified by an analytical model. The results show that it is a promising solution to mitigate the EDoS attacks originating from spoofed IP addresses. The enhanced EDoS-Shield technique also outperforms the original EDoS-Shield in terms of performance and cost metrics.

This Chapter is organized as follows. Section 5.1 presents an introduction about the problem. In section 5.2 we discuss the background related to using TTL in mitigating IP spoofing. In section 5.3, we discuss the related works. Section 5.4 presents the proposed architecture and section 5.5 discusses its performances issues. The analytical model for the proposed mitigation technique was discussed in section 5.6. Section 5.7 and section

5.8 present the parameters setting and the simulation setup respectively. Finally, the results out of both the simulation and analytical model are presented and discussed in sections 5.9.

5.1 Introduction

The main idea of EDoS-Shield technique, presented in chapter 4, is to verify whether the requests coming from the users are from a legitimate person or generated by zombies. Based on the verification performed at a verifier node (V-Node), the source IP address will be added either to a blacklist in case of failure, or to a whitelist in case of success. The subsequent requests coming from the zombies will be blocked by a virtual firewall, and the subsequent requests coming from legitimate clients will be forwarded directly to the target cloud service.

For the original EDoS-Shield technique to achieve its goal of mitigating EDoS attacks, it is also as important to protect the cloud from IP spoofing attacks. This is due to the fact that the decision to forward or drop a packet is mainly based on the source IP address present in the whitelist and/or blacklist. To overcome this problem, many techniques [112, 122, 123, 124, 125] could be used to detect and prevent IP spoofing attacks. However, these techniques could have a negative impact on the performance of the original EDoS-Shield. In addition these techniques may not be easily deployed and incorporated into the infrastructure of the cloud. For instance, some techniques required involving routers along the path between the source and the destination [124, 147]. Other mitigation techniques incur more overhead at the destination side, which affects the performance of the protected server [56, 148].

In this chapter, we address the issue of IP spoofing and show how a novel EDoS-Shield architecture can incorporate a mitigating mechanism for EDoS attacks. In the enhanced EDoS-Shield architecture, we make use of the time-to-live (TTL) value found in the IP header for the purpose of detecting the IP spoofed packets. These spoofed packets will then be dropped before reaching the protected server and hence mitigating the EDoS attacks that originate from spoofed IP addresses. In this architecture, when a V-Node performs a verification of a request, it will record the corresponding TTL value related to the source IP address. Then both values, i.e., IP address and TTL value, will be placed in the whitelist or blacklist, based on the results of the verification phase. Such information will be used later to distinguish the packets having spoofed IP addresses, and then can selectively filter out these packets by a virtual firewall (VF).

5.2 Background

The TTL value is defined to indicate the maximum lifetime of an IP packet, for the purpose of preventing it from circling on the network forever when a routing loop is present. When a packet passes through a router, the packet gets discarded if the TTL value is equal to zero; otherwise, the TTL field gets decremented by one. The difference between the final value and the initial value of the TTL represents the number of hops taken by the packet. And, it can be considered as the Hop-count value of a source and destination pair. In fact, calculating the Hop-count value is not a direct operation, since the initial values used in the TTL field are not fixed and vary based on the Operating Systems. However, the Swiss Academic Research Network (SWITCH) stated that the values of 32, 64, 128, and 255 are the default initial TTL values being used by modern operating systems including Microsoft Windows, Linux variants of BSD, and UNIX systems. As was stated by Cheswick et al. [149], a Hop-count normally has a value between 1 and 30, and thus the initial TTL value can be deduced from the final TTL value. For instance, a packet having a final TTL value of 119 cannot have an initial TTL value of 255 and it must have128 as its initial value [150].

One of the possible problems in using the Hop-count for the purpose of detecting the IP spoofed packets is that multiple IP addresses may have the same Hop-count values. For example, if a zombie machine is spoofing an IP address IP1, and this machine has the same Hop-count value to the victim as that of the legitimate machine having the IP address IP1, then this will be undetected. In such a case, the filtering schema cannot recognize those packets coming from the zombie machine as spoofed packets. Reasonable diversity of the Hop-count values is required to ensure that different sources will not have the same Hop-count values to a destination. The stability of Hop-count values is also required to ensure the absence of frequent changes in the Hop-count values between a server and each of its clients. In other words, for the Hop-count based filters to be more accurate, the Hop-count values must be unique and stable for each source.

Wang et al. [151] have assessed the diversity of the Hop-count values, in which they claimed that the Hop-count distribution in the observed data set satisfies the diversity property of the Hop-count values. This means that the Hop-count distribution in the observed data set has a reasonable diversity over the entire range of Hop-count values. This enables a very effective filtering schema since matching the Hop-count with the source IP address of each packet suffices to recognize spoofed packets with a high probability.

The stability in Hop-count values between a source and a destination node is crucial to ensure the correctness and effectiveness of the Hop-count based filtering schemas. This is due to the fact that the decision about the spoofed packets is made based on the mapping between the source IP address and the corresponding Hop-count value to the destination. In fact, the Hop-count stability is dictated by the end-to-end routing behaviors in the Internet. The more stable the end-to-end routes are in the Internet, the more stable the Hop-counts are.

Several studies have been conducted to confirm the stability of routes during the packet delivery from a source to a destination [152, 153, 154, 155, 156]. Paxson [152] has conducted a study of end-to-end routing stability, and has shown that Internet paths are strongly dominated by a single route, and about two thirds of the studied Internet paths were persisting for either days or weeks. Rexford et al. [153] have claimed the stability of the BGP routes of popular prefixes for days or weeks at times, regardless of the large number of BGP update messages. Similarly, Shaikh et al. [154] studied the intra-domain routing behavior, and showed that the intra-domain topology changes are due mainly to external changes and no network-wide meltdown or network-wide instability is observed.

Furthermore, a recent measurement study of the diversity of end-to-end paths in the Internet [155] has shown that end-to-end route properties did not significantly change in recent years, including the stability of the routes between pairs in the Internet. Cunha et al. [156] have also shown that despite the growth of the Internet and the introduction of new traffic engineering practices, route persistence and prevalence presented by Paxson [152] have not changed significantly.

5.3 Related Work

Several studies have used the TTL values in different ways to detect IP spoofing attacks [147, 150, 151, 158, 161, 159, 160]. In addition, there are several proposed network security techniques based on whitelist and/or blacklist approaches [169,170, 171, 172].

Beverly et al. [161] have presented a forensic carving technique of network packets and associated data structures. They have utilized the IP TTL parameter of packets to determine if they were originated by a system in the local network, or received from a remote system.

Mopari et al. [150] have used Hop-count for each IP address in an IP2HC (IP to Hop-Count) mapping table to detect and drop IP spoofing packets. Their technique computes the Hop-count for each packet that reaches the terminal. The source IP address for each packet is considered as an index to the mapping table to retrieve the corresponding Hop-count from the table. The packet is classified as spoofed when the calculated Hop-count differs from the retrieved Hop-count, otherwise it is legitimate. Mopari et al. have claimed that it is a simple and effective solution in protecting Internet servers against spoofed IP packets.

Wang et al. [151] have presented a Hop-count based scheme that detects and discards spoofed IP packets. The Hop-count of the incoming packets is used to validate the packets' legitimacy. They have studied some of the Hop-count properties that are crucial for the effectiveness of the Hop-count based approach in detecting spoofed IP packets. Among those properties are the diversity and stability of Hop-count values. Wang et al. have claimed the appearance of diversity and stability of the Hop-count values, and thus such properties were the central assumptions behind the correctness and effectiveness of their proposed approach.

Ohta et al. [158] have proposed a two-step approach for detecting spoofed packets by examining both TTL and Identification fields (IPID) of the IP header. Through the study, they presented an analysis to validate the uses of the TTL and IPID values to distinguish the spoofed packets from the other packets.

KrishnaKumar et al. [159] have proposed a Hop-count based packet processing approach to counter DDoS attacks. Their proposed approach works as follows. First, each packet traveling along the router path is marked with a Path Identification (PID) which is derived from the concatenation of the hash value of the source IP address and the encrypted value of the Hop-count. Then, the router will maintain an accurate mapping table that contains both the IP2HC and the calculated PID. The filtering phase is executed for each packet with the help of the mapping table at the router by examining both the Hop-count value and the PID number marked in the packet header. A packet is considered to be legitimate only if its Hop-count and PID values match the ones placed in the mapping table.

Hwang et al. [169] have proposed a three-tier intrusion detection system including blacklist, whitelist, and multi-class classifier. They have employed data mining and machine learning techniques to create both the whitelist and blacklist and to classify the attacks. The blacklist is used to filter out the known attacks from the traffic, and the whitelist to identify the normal traffics.

Soldo et al. [170] have introduced a framework for studying filter selection focusing on the optimal construction of filtering rules based on the source address/prefix. However, through their study, they assumed using an intrusion detection system to create the blacklist, while the sources of legitimate traffic are also assumed to be known. In addition, they considered addresses in the blacklist to be true and not spoofed.

Simpson et al. [171] have proposed a whitelist-based approach at intermediate routers to defeat the DDoS attacks. The whitelist is created at the routers to hold the IP addresses of likely legitimate clients by observing outgoing traffic, presenting a challenge though proof-of-work, and providing flow cookies. For example, if an intermediate router observes TCP data from the server to the client, it whitelist the client IP address.

Park et al. [172] have proposed a unified rate limiting algorithm to work against both Internet worms and DDoS attacks. The idea of their work is to use five execution modules to decide whether a packet should be dropped or passed to the destination. The modules are put in an order as follows. First, the scheme examines credit value, based on the number of failed connections, to filter out packets with invalid credit value. Second, the IP spoofing check module is used to decide about the legitimacy of the source based on the observed local address and the outgoing IP address. Third, the whitelist approach is used to reduce the execution time of the rate limiter by checking the packet transmission rate for only the whitelist connections. Fourth, the blacklist approach is used to drop packets if their source IP addresses are present in the blacklist. Finally, if a source IP address is not listed in either the whitelist or the blacklist, the packet will be validated to add its source IP address in either list. The validation is done by monitoring the ACKs within a specific time.

However, our proposed approach differs from the previously discussed approaches in a couple of important respects. First, we have combined using the TTL approach with the Whitelist and Blacklist approaches to detect and prevent EDoS attacks. Second, regarding the overhead introduced due to the used technique to detect the attack, our approach has less overhead since it uses only the IP address and the corresponding TTL value for the purpose of detection. Whereas, approaches presented in [169], [171], and [172] introduce more overhead as follows. In [169], the detection approach is based on the data mining and machine learning, which required more overhead to update the lists. In [172], all the packets have to be checked regarding their credit values and their legitimacy, even they were already checked and present in the blacklist or whitelist. In [171], the absence of the blacklist allows the attacks to make an overhead problem at the routers since the approach requires proofing the legitimacy of the sources even they already have been identified as malicious. On the contrary, our approach uses the whitelist and the blacklist as a first stage to decide about passing or blocking the packets. Thus, the subsequent packets will not need a further investigation at all. Third, our approach is more effective in terms of false alarm rates due to the thresholds used in our proposed algorithm. Park et al. [172] did not consider cases of the IP spoofing attacks where attackers attempt to block some of the legitimate clients by adding their source IP addresses to the blacklist. On the other hand, in our approach, we used thresholds to handle such case. For example, if the source IP address of a client is present in the blacklist due to an IP spoofing attack or due to a failure in the validation scheme, then, according to the approach presented in [172], the client is no longer able to communicate with the destination unless the blacklist is reset. However, in our approach, the client will have chances to verify its legitimacy based on the used thresholds such as the number of failures of the source IP address, and the time period of the attack. Moreover, Soldo et al. [170] did not consider the case of spoofed IP address in their approach.

5.4 Proposed Architecture and Algorithms

As an extension to prior work on EDoS-Shield, we present in this chapter a technique to detect EDoS attacks originating from IP spoofed addresses. The same architecture of the original EDoS-Shield is used, but with extra fields utilized in the whitelist and blacklist to mitigate the IP spoofing attacks that could affect the original EDoS-Shield technique. These fields are the TTL values, and a counter of unmatched TTL values in both whitelist and blacklist, in addition to the attack start time in the blacklist.

The TTL values used with both whitelist and blacklist are based on the verification phase done at the V-Node. A V-Node has the capabilities to verify legitimate requests at the application level using graphic Turing tests [117], such as CAPTCHA or reCAPTCHA [119, 162]. If the source IP address passes the test then the final value of the TTL will be placed in the whitelist along with the source IP address. If the test fails, the TTL value will be placed in the blacklist along with the source IP address.

In general, the TTL values in the whitelist will be used to filter out the spoofed packets that have a spoofed IP address present in the whitelist. This case occurs when the attacker is spoofing an IP address which is currently in the whitelist. Thus, the TTL value will be used to filter out those packets coming from the attacker, since their TTL value more probably do not match the TTL value of the legitimate source address. The TTL values in the blacklist will be used as a further filtering phase to reduce the false positives that may occur when the IP address of a legitimate client is spoofed by the attacker and has been wrongly placed in the blacklist. For example, when receiving a legitimate packet while its source IP address is already in the blacklist, its TTL value will be compared to the one placed in the blacklist to decide about its legitimacy.

The unmatched TTL counter field will be used to reduce the false positives based on the exact TTL matching filtering criterion. Instead of dropping packets because of not matching the TTL value, a verification phase will be performed at the V-Node as long as the “unmatched TTL” counter does not exceed a given threshold. This will reduce the false positive results since packets having different TTL values still have a chance to verify their legitimacy at a V-Node.

Changes of the TTL value between a source and a destination are limited over a period of time [151, 152, 155, 156]. Based on this assumption, if the number of changes exceeds a defined threshold, then these changes could be considered as abnormal and are due to an attacker forging the TTL values. Thus, when the threshold is exceeded, packets coming from the corresponding IP address, which is present in the blacklist, will be dropped without any further verification.

The attack timestamp field in the blacklist is used to record the start time of the attack which is the time at which the source IP address is placed in the blacklist. The timestamp field will be utilized to make the verification phase at the V-Node more restricted during the attack. For instance, if a packet arrives during the lifetime of the attack with a source IP address that is present in the blacklist, it will be dropped without performing a further verification phase. On the other hand, if the packet arrives after the attack’s lifetime elapses, then a verification phase will be performed since there is a probability that it is a non-spoofed packet.

To avoid accumulating spoofed IP addresses in the blacklist which could be done maliciously by the attacker, both the start time and the common lifetime of the attack [163] can be used to estimate the end of the spoofing. In this case, the source IP address can be removed from the blacklist.

Algorithm 5.1 describes the actions at the VF node when receiving a packet. At the VF, the filtering phase is kept simple with negligible overhead. A packet will be forwarded directly to the destination only if its source IP address is found in the whitelist and its TTL value matches the TTL mapped to the IP address present in the whitelist. Otherwise, packets will be forwarded to V-Nodes for further investigation.

Algorithm 5.1: VF Actions

```
Input:
     P← Packet
     S← Packet source IP address
     D← Packet destination IP address
     B← Blacklist
     W← Whitelist
     TTL← Packet TTL value
Begin:
If (S ϵ W  and  W[S].TTL==TTL)
         Forward P to D
Else
         Forward P to a V-Node
End
```

Algorithm 5.2 describes the actions taken by a V-Node when receiving a packet. Figure 5.1 shows an abstracted view of Algorithm 5.2. At the V-Node, when receiving a packet from the VF, there are four cases based on the presence of its source IP address in the whitelist and/or blacklist, as shown in Algorithm 5.2. These cases are: The source IP address is new (first come), it is already present only in the whitelist, it is already present only in the blacklist, and it is present in both lists.

For the first case, where the source IP address is neither in the whitelist nor in the blacklist, the V-Node will perform a verification phase using Graphical Turing test. If the test passes, the IP address along with the TTL value will be placed in the whitelist and the unmatched counter will be initialized to zero. If the test fails, the IP address along with the TTL value will be placed in the blacklist; and the timestamp and unmatched counter will be initialized to the current time and zero, respectively.

For the second case, where the source IP address appears only in the whitelist, the V-Node will perform a verification phase. If the test passes, the corresponding TTL value in the whitelist will be updated to the new value obtained from the last verified request. If the test fails, the unmatched TTL counter in the whitelist will be incremented and the source IP address will be added to the blacklist with its TTL value and timestamp.

Algorithm 5.2: V-Node Actions

```
Input:
     P←Packet
     S←Packet source IP address
     TTL←Packet TTL
     D←Packet destination IP address
     B←Blacklist
     W←Whitelist
     Threshold←Constant representing maximum unmatched TTL
     LifeTim←Constant representing the common Attack's duration
Begin:
If (S ∉B) {
  Send to S a graphic Turing test
  If (passes) {
    If (S ∉ W) {
      W←W+W[S]; W[S].TTL=TTL; W[S].unmatched=0;}
    Else
      W[S].TTL=TTL
    Forward P  to D }
  Else {
    B←B+B[S]; B[S].TTL=TTL; B[S].unmatched=0 ;
    B[S].timestamp=CurrentTime
    If (S є W) W[S].unmatched++
    Drop P   }  }
Else {    // (S є B )
  During =  ((CurrentTime - B[S].timestamp) <LifeTime)
  ExceedThr = (B[S].unmatched  > Threshold)
  MatchesTTL= (B[S].TTL==TTL)
  If (  During  and  (MatchesTTL  or  ExceedThr))
    Drop P
  Else {
    Send to S a graphic Turing test
    If (passes) {
      If (S ∉ W) {
        W←W+W[S]; W[S].TTL=TTL; W[S].unmatched =0; }
      Else{
        W[S].TTL=TTL; W[S].unmatched =0 }
      If (Not During)
        B←B - B[S]
      Forward P to D }
    Else {
      Drop P
      If (Not During) {
        B[S].TTL=TTL; B[S].unmatched =0 ;
        B[S].timestamp=CurrentTime  }
      Else {
        B[S].unmatched ++
        If (S є W)
          W[S].unmatched ++
      }
  }      }
}
End
```

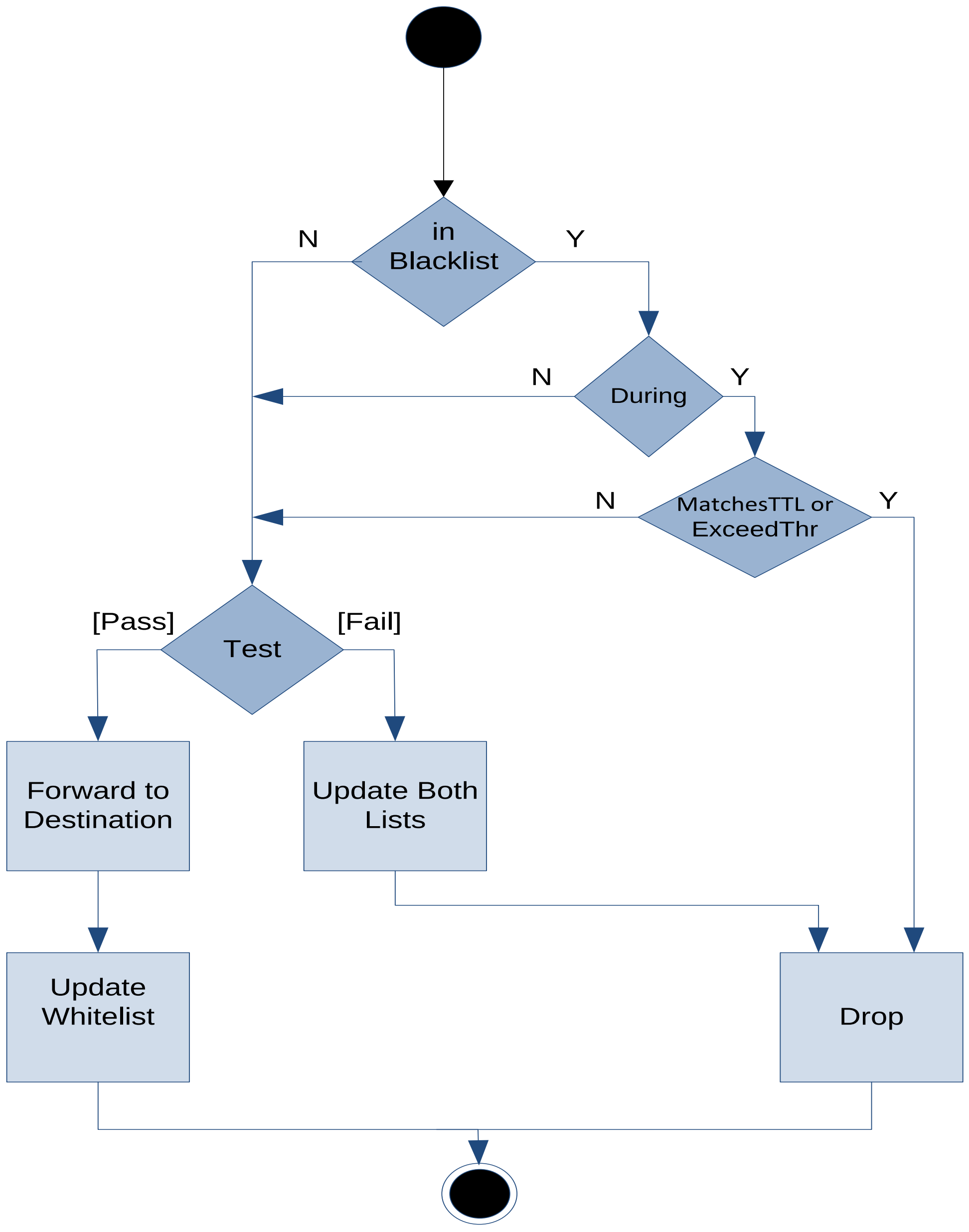


Figure 5.1: Main activities at V-node side

For the third case, where the source IP address appears only in the blacklist, the packet will be dropped when its TTL value matches the corresponding TTL value in the blacklist or when the unmatched TTL counter in the blacklist reaches the threshold during the attack's lifetime. We are assuming that two different sources will not have the same TTL value, since the probability of having the same TTL value for two different sources is very low. Otherwise, the V-Node will perform the verification phase. If the test passes, the packet will be forwarded to the destination and its source IP address will be placed in the whitelist along with the packet TTL value. If the test fails, the packet will be dropped and the corresponding entry in the blacklist will be updated as follows. If the packet is received within the attack's lifetime, the unmatched TTL counter will be incremented. If it is received after the attack's lifetime elapses, the corresponding fields, i.e., TTL, timestamp, and unmatched TTL counter, in the blacklist will be reset to packet TTL, current time, and zero, respectively.

For the fourth case, the source IP address appears in both lists, which means that the incoming traffic at the V-Node side might have some packets with spoofed IP addresses and some other legitimate packets. In such a case, as shown in Algorithm 5.2, the packet will be dropped when its TTL value matches the TTL mapped to its source address in the blacklist, or when the unmatched TTL counter in the whitelist reaches the specified threshold within the attack's lifetime. Otherwise, the V-Node will perform a verification phase. If the test passes, the packet will be forwarded to the destination and its corresponding entry in the whitelist will be updated by the new TTL value and by resetting the unmatched TTL counter to zero. If the test fails, the packet will be dropped and the unmatched TTL counters in both whitelist and blacklist will be incremented if the packet is received within the attack's lifetime. If the packet arrives after the attack's lifetime elapses, then the corresponding entry in the blacklist is updated so that its fields, i.e., TTL, timestamp, and unmatched TTL counter, will hold the new values, i.e., packet TTL, current time, and zero, respectively.

5.5 Security and performance Issues

Considering a legitimate client with IP address *S* targeting a cloud service with IP address *D*, the normal behavior for a client is when no attacker is spoofing the IP address *S* that communicates with *D*. In such a case, the first request from *S* will be verified at the V-Node and added to the whitelist; and as long as the subsequent packets have the same Hop-count, they will be forwarded directly to the target address *D* without being routed through the V-Nodes. If the TTL value changes while *S* is in the whitelist, the packet will not be dropped as long as the number of changes indicated by the unmatched TTL counter does not exceed the threshold.

A simple attack could occur when the attacker is using a fixed spoofed source IP address without altering the initial TTL value. In such a case, the first request will be forwarded to a V-Node for the verification phase. Since it will fail the test, the spoofed IP address will be added to the blacklist along with its TTL value and the start time of the attack. The subsequent packets will be dropped in case of matching the corresponding TTL present in the blacklist. Otherwise, a request will go through the verification phase as long as the unmatched TTL counter does not exceed the given threshold or if the packet arrives after the attack's lifetime elapses.

A more sophisticated attack could happen when the attacker changes the initial value of the TTL for each request, leading to a verification phase for each request, since dropping only occurs when matching the TTL in the blacklist. However, to reduce the overhead that occurs due to the verification phase at the V-Nodes, the unmatched TTL counter and the timestamp values in the blacklist are used to limit the number of verification steps. Therefore, the verification phase will only take place if the unmatched counter does not reach the threshold or when the request arrives after the attack's lifetime elapses.

However, there are two cases, namely the whitelist case and the blacklist case in which an attacker could utilize IP spoofing to bypass the filtering mechanism proposed in the original EDoS-Shield technique, presented in chapter 4.

Whitelist Case. In this case, a V-Node has identified the source IP address as a non-spoofed address, and therefore this IP address has been placed in the whitelist. A problem might occur when an attacker sends requests with a spoofed IP address which is already in the whitelist targeting the same protected cloud service. A scenario of such a case is that the attacker initially sends a legitimate request so that its IP address can be added to the whitelist as a legitimate source. Then, the attacker could control a number of zombies to generate an extensive number of requests having his whitelisted IP address as a source IP address, which represents a spoofed address in this case. In such a scenario, all packets will bypass the filtering scheme, and thus will lead to flooding the victim's targeted service.

Figure 5.2 explains the scenario in which a fixed spoofing method is used to spoof the source IP address *S* that has already been approved and is present in the whitelist. Then, zombies will use *S* as a source IP address to flood the destination *D*. Thus, only those packets having the final TTL values matching the TTL value in the whitelist will pass through the VF to the destination *D*. This could occur if the flooding sources have the same Hop-count to the destination, i.e., same as the Hop-count between *S* and *D*. But, such a case has a low probability to occur as was discussed in the previous section.

However, since zombies use the same IP address *S* present in the whitelist, it is easy to detect such attack based on the spike of the traffic rate, because all the incoming packets in this case have the same source IP address *S*. Once this gets detected, the source IP address will be placed in the blacklist along with a timestamp, and it will be removed from the whitelist. Another way, presented in Algorithm 5.1, to mitigate the attack in such a case is to use TTL values to filter the packets at the VF before forwarding them to the destination even if their source IP address is present in the whitelist. In fact, the accuracy of the filtering phase at the VF in such a case is guaranteed when the final TTL value of a spoofed packet does not match the TTL value present in the whitelist. This is valid when assuming that the attacker will not forge the initial values of the TTL field in the IP header to make the final TTL values match the one present in the whitelist. The validity of such assumption is obtained from observing that most of the available DDoS attacking tools including TFN2k [164] and Stacheldraht [165] do not modify the initial TTL values [166, 167]. Moreover, for a sophisticated attack that could change the initial value of the TTL, still it is difficult for the attacker to know the actual Hop-count between the flooding zombie source and the victim [159].

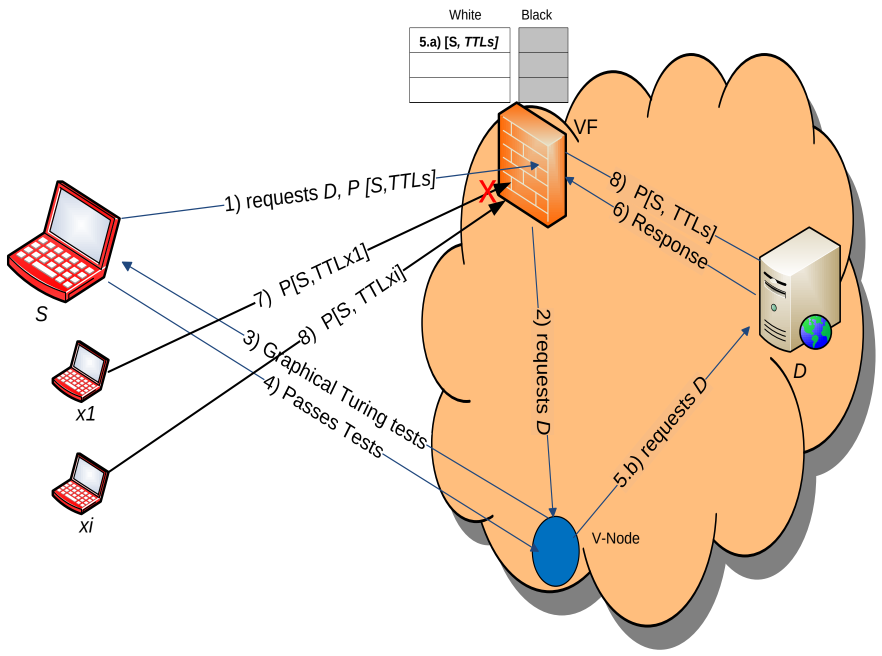


Figure 5.2: Whitelist case scenario

Blacklist Case. In this case, a V-Node has identified an IP address as a spoofed address and that IP address has been placed in the blacklist. A problem might occur when a legitimate client, having the same IP address as the one that has already been placed in the blacklist, is sending a request targeting the protected cloud service. If we decide to block all requests having the blacklisted IP address targeting the same protected cloud service, then also the legitimate client's requests will be blocked leading to a behavior similar to that of a denial of service attack. Figure 5.3 describes such a problem along with the mitigation scenario.

To mitigate such type of attacks, according to our proposed filtering technique, instead of blocking the subsequent packets coming from a blacklisted IP address, these packets have to be forwarded to the V-Nodes for further investigation as it is shown in Figure 5.3. If a legitimate client, having S as a source IP address, sends a request during the attack lifetime, then it will go through the verification phase as long as the unmatched counter does not reach the threshold. Upon the success of the test, S will be added to the whitelist leading to a mixed state where S is present in both the whitelist and the blacklist. This means that the subsequent packets having S as a source IP address will be filtered based on the information given in both the whitelist and the blacklist, i.e., TTL, timestamp, and unmatched TTL counter, as it is described in Algorithm 5.2.

If a legitimate request arrives after the attack's lifetime elapses, then it will go through the verification phase. Upon a successful verification, it will be added to the whitelist in addition to removing S from the blacklist. After removing the corresponding IP address from the blacklist, the subsequent packets will be filtered at the VF based on both the source IP address and the TTL value.

One case that may lead to a false positive decision is when a legitimate request from *S* arrives during the attack's lifetime and has a TTL value equals to the TTL value corresponding to *S* present in the blacklist. However, the probability of having such a case is very low as was discussed earlier.

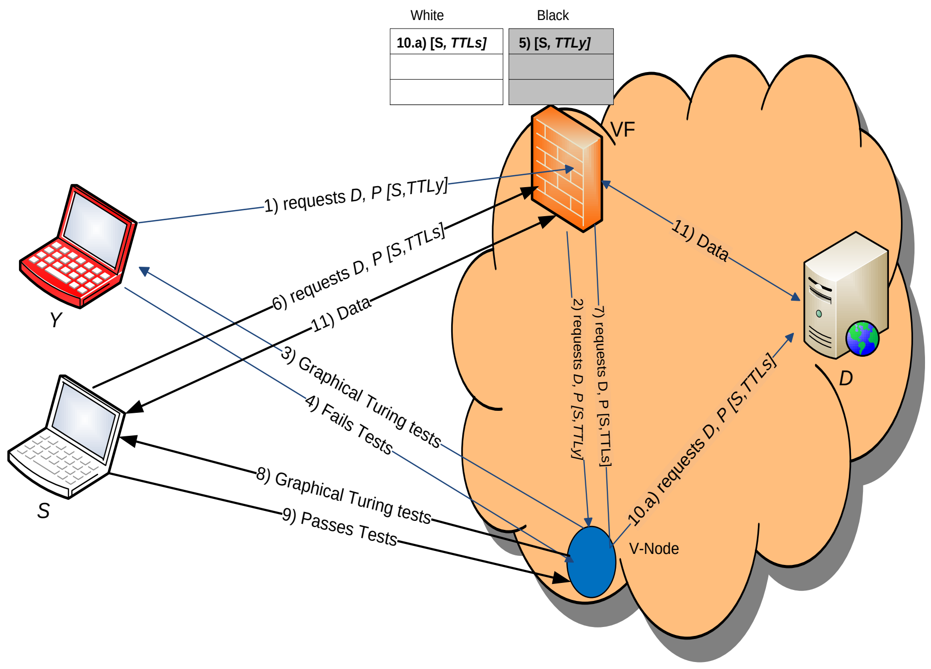

White
Black
10.a) [S, TTLs]
5) [S, TTLy]
VF
1) requests D, P [S,TTLy]
6) requests D, P [S,TTLs]
11) Data
11) Data
Y
3) Graphical Turing tests
4) Fails Tests
2) requests D, P [S,TTLy]
7) requests D, P [S,TTLs]
D
10.a) requests D, P [S,TTLs]
S
8) Graphical Turing tests
9) Passes Tests
V-Node

Figure 5.3: Blacklist case scenario

5.6 Analytical Model

In this section, we present the analytical model of the enhanced EDoS-Shield mitigation techniques. Figure 5.4 shows the queuing network that represents the architecture of the mitigation technique which is similar to the architecture of the original one but with different algorithms implemented at the VF and V-Node. The input to the model

In this section, we present the analytical model of the enhanced EDoS-Shield mitigation techniques. Figure 5.4 shows the queuing network that represents the architecture of the mitigation technique which is similar to the architecture of the original one but with different algorithms implemented at the VF and V-Node. The input to the model is an aggregated traffic from different sources including attackers' requests. The aggregator can be implemented within the LB. We have considered the Poisson nature of the incoming requests as it is discussed in chapter 3.

In the analytical model as well as in the simulation model, we evaluated the performance metrics including the end-to-end response time, computing resources utilization, and the throughput rate. As it is the main concerns with the EDoS attacks, we studied the cost evolution when applying the mitigation technique.

The analytical model for the enhanced mitigation technique is similar to the one of the original EDoS-Shield, discussed in Chapter 4, except of some parameters that should be set to values corresponding to the studied cases with the enhanced EDoS-Shield technique. Mainly, the percentage of the attack requests that may pass the filters toward the destination, i.e. the false positive rate due to the filtering mechanism in the enhanced technique, will differ than the one uses with the original EDoS-Shield.

In the original EDoS-Shield, the false positive rate is zero as we assumed that it is protected against IP spoofing attacks. However, in the enhanced technique, the false positive rate is subjected to the attack cases. In the other words, the equations of the analytical model used for analyzing the original EDoS-Shield have been utilized to analyze the enhanced mitigation technique while involving the false positive rate in the arrival rates to the subsystems of the model.

The studied metrics have been evaluated based on the corresponding equations discussed in Chapter 4. The queuing model has been summarized in Table 5.1.

All the metrics listed in Table 5.1 have been evaluated based on the corresponding equations while setting λ_1 and λ_2 to be the legitimate rate plus the false positive rate. We have not considered the false negative rate since all the studied cases have no false negative rates according to our proposed technique.

We studied two different cases of the EDoS attacks to evaluate the enhanced mitigation technique, which are the whitelist case, and the blacklist case discussed above in Section 5. Each case has a different false positive rate based on the behavior of the attack and the used mitigation technique. The false positive rate plays the main role on the results of the analytical model.

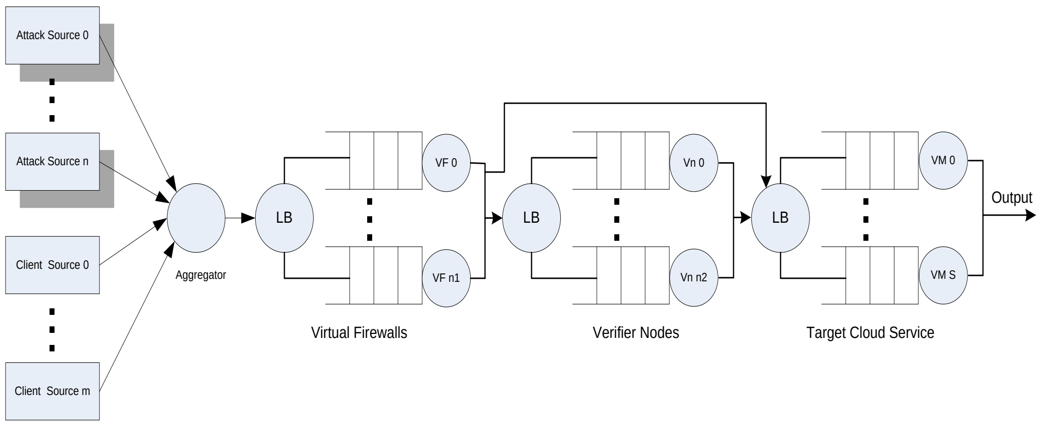


Figure 5.4: Mitigation techniques' queuing model

Table 5.1: A summary of the equations of the analytical model

Metric	Equation
Response Time	$\frac{S_1}{S_1\mu_1 - \lambda} + \left(1 - \frac{\lambda_1}{\mu_{link2}}\right) \Big/ (\mu_{link2} - \lambda_1) + \frac{S}{S\mu_2 - \lambda_2}$
Average Utilization	$\frac{\lambda_2}{S\mu}$
Cost	$(\Pr ice_{bw} \times \lambda_2 + \Pr ice_{com} \times U \times S) \times T$
Total Throughput	λ_2

Description:

μ_{link2} is the capacity of the link from a VF to the cloud service.

S_1 is the number of the instances representing the VFs.

S is the number of instances representing the cloud service.

μ_1 is the processing rate of a VF.

μ_2 is the processing rate of a cloud instance.

λ_1 is the rate of the requests forwarded by a VF towards the target cloud service (receiver).

λ_2 is the rate of the total requests that arrived at the cloud service.

For the whitelist case, we assumed that an attacker is generating normal requests toward the victim just to place his source IP address in the whitelist. Then, later, the attacker will command the compromised machines, zombies, to flood the victim by automated requests having the source IP already present in the whitelist. Thus, the requests coming from the attacker master node will pass the VF to reach the cloud service since requests have IP address present in the whitelist and their TTL values match the one in the whitelist. Moreover, some of the automated requests from the zombies will pass the VF when their TTL values match the one present in the whitelist. We assumed that the zombies are distributed randomly around the master node of the attacker so that their TTL values will have random values out of 255 as the maximum value of the TTL field. As a result, the percentage of the false positive rate for such attack case can be calculated as follows:

$$False\_p = \frac{1}{num\_attack\_sources} + \frac{1}{255},$$

where $num\_attack\_sources$ is the number of sources generating the attack requests. For example, when the number of the attack sources is 10, the $False\_p$ will be about 0.1039.

For the blacklist case, we assumed that an attacker is spoofing an IP address related to a user to send a malicious request toward the protected cloud service just to place that legitimate source IP address in the blacklist. However, according to the enhanced mitigation technique's algorithms, such case will not prevent the legitimate requests from reaching the cloud service as the legitimate user will have a chance to verify its legitimacy even his IP address is present in the blacklist. The problem is that the IP address will be present in both the blacklist and the whitelist at the same time. As a result of such mixed case, some of zombies' requests may reach the protected cloud service as long as their TTL values match the one present in the whitelist corresponding to the legitimate source IP address. For such case, we assumed that the zombies are also distributed randomly around the legitimate user's machine so that their TTL values will have random values out of 255. Thus, the percentage of the false positive rate for the blacklist case is about 1/255, which is about 0.0039.

5.7 Selecting System Parameters

We have set the parameters required for the simulation model as well as for the analytical model based on the discussions presented in Chapter 4, which can be summarized as follows:

1- The capacity of the target cloud instance and the V-Node, as small instances, is set to 100 Req/sec.

2- The average service time of VF, as a large EC2 instance, is set to 108 μs which is equivalent to a rate of 9260 Req/sec.

3- The capacity of the links in the cloud infrastructure is set to 10 Gbps.

4- Two running virtual firewalls.

5- V-Nodes and Cloud instances are assumed to be horizontally scalable.

6- The number of initial running cloud instances is 5.

7- The provisioning overhead is of 55.4 sec.

8- The upper utilization threshold is of 80%.

9- The scaling size is variable based on the optimization technique discussed in Chapter 3.

10- The price of the small instance and the large instance is \$0.115 and \$0.46 per hour, respectively.

11- The bandwidth price is of \$0.01 per GB in/out data transferred.

Regarding the threshold that has been used at a V-Node to represent the maximum allowable changes made to a TTL value in the proposed algorithm of the enhanced mitigation, it has been set to a value of 5. This is based on results which reported that about 95% of the studied paths had fewer than 5 observable daily changes [151]. The lifetime of the attack used in the filtering and verifying steps at the V-Node has been set to a value of one hour [163]. The TTL values have been generated randomly between 1 and 255.

5.8 Simulation Setup

We have conducted a discrete event simulation experiment to evaluate our proposed mitigation technique considering the performance of the protected cloud service when deploying the proposed architecture. The performance has been studied in terms of key performance indicators which include end-to-end response time, computing resources utilization, and throughput. Since the main goal of the proposed architecture is to protect the cloud service against the EDoS attacks, we have also evaluated the cost associated with the computing resources and bandwidth allocation at the protected cloud service side.

We have studied and evaluated four different scenarios using the discrete event simulation model discussed in the previous section. In the first scenario, we have assumed that the cloud service was not protected at all against the EDoS attacks which represents the worst case scenario, i.e., without any mitigation technique. In the second scenario, we have assumed the original EDoS-Shield mitigation technique has been used to mitigate the attack. In the third scenario, our proposed enhanced EDoS-Shield mitigation technique is used to protect the cloud service. The last scenario is considered as the base scenario in which we are assuming there is no attack at all, and therefore it represents the optimal case.

The simulation followed closely and carefully the guidelines given in [82] including the use of independent replications with different initial seeds that were ten million apart, and avoiding any overlapping in the random number streams during the simulation.

We have assumed a fixed input rate of 400 Req/sec representing the rate of the legitimate requests coming from clients and a variable input rate ranging from 400 Req/sec to 8000 Req/sec representing the rate of the attack traffic generated from 200 sources. The arrival rate has been assumed to follow a Poisson distribution.

One of the measurements that we have studied using the simulation is the cost associated with the computing resources and bandwidth allocations at the protected cloud service side. The cost with regard to the computing resources has been calculated based on the equation given in [168] as follows:

$$CPU_{\cos t} = \sum_{i}^{m} CPU_{base} \times R_i^{cpu} \times t_i + (m-1) \times CPU_{trans}$$

where R_i^{cpu} is the CPU Utilization at the i^{th} allocation, t_i is the duration of i^{th} allocation, CPU_{base} is the base price charged for the smallest amount of CPU time allocation, CPU_{trans} is the transfer fee charged each time we change the CPU allocation, and m represents the total number of changes. For simplicity, we have assumed that the CPU_{trans} is zero, by assuming one billing cycle during the attack period.

Finally, the output of the proposed analytical model has been compared to the simulation results. The relative error % is used to measure the accuracy of the queuing model results compared to the simulation model results. The percentage of the Relative error has been determined based on Eq. (3.9).

5.9 Results and Discussion

We have considered two cases which are the whitelist case, and the blacklist case. Both cases have been evaluated by the simulation as well as by the analytical model.

5.9.1 Whitelist Case Results

Figure 5.5 presents the response time evaluation, and it is obvious from the results that the original EDoS-Shield scenario behaves similarly to the scenario with no mitigation technique. This is expected since the original EDoS-Shield will not filter out any packet, since all the packets coming from zombies have the source IP address that is already approved and placed in the whitelist. Moreover, the original EDoS-Shield causes more delay as requests will encounter extra delay at the VF and also through the link from the VF to the cloud service. However, the enhanced EDoS-Shield technique will filter out most of malicious requests based on the TTL values.

For the attack rates from 400 to 4800 Req/sec, the enhanced EDoS-Shield technique shows response time results less than the results obtained for the base case due to using more instances added by the provisioning technique to cope with the spike caused by the false positive rates reaching the destination. Since the false positive rate increases along with the attack rate increases, the attack traffic reaching the destination makes an overhead at the destination so that the average utilization exceeds the threshold and hence the provisioning occurs to add more instances to the cloud service. However, for the attack rates below 4800 Req/sec, the attack rate reaching the destination consumed a little computing power of the added instances leading to less average response time compared to the base case. Whereas, for the attack rates above 4800 Req/sec, the attack rate reaching the destination consumed most of the computing power of the added instances leading to an average response time greater than the one observed for the base case.

In fact, some requests arriving from the attacker node will behave as legitimate requests as long as their TTL values match the one in the whitelist, and therefore, such requests will not be filtered out and will reach the destination. Since we design the simulation with 200 attacker's sources, the amount of the attack traffic that will reach the destination is about 0.5% of the total attack traffic. Furthermore, since TTL value have been set randomly, about (1/255) or 0.39% of the traffic arriving from the zombies, having a TTL value matching the one initiated by the attacker, will also not be filtered out and will reach the destination. Thus, when comparing the enhanced EDoS-Shield to the optimal case, results show an increase of the response time, the computing resources utilization, and the cost due to such false positive rates.

Figure 5.6 shows the performance of the evaluated scenarios in terms of the computing resources utilization. It is obvious from the results that the average utilization does not exceed the upper threshold, 80%, used in simulation. However, the proposed enhanced EDoS-Shield mitigation technique outperforms the original EDoS-Shield in terms of the computing resources utilization as it uses fewer resources than the original one. Nonetheless, there is still an increase in the computing resources utilization with the enhanced proposed technique compared to the optimal case. This is due to processing about 1% of the attack traffic, which is the false positive rate that passes the filter.

Figure 5.7 shows the results of the cost associated with the computing resources and bandwidth allocations, assuming that the attack lasts for 10 hours. The results show that when not applying the mitigation technique or when using the original EDoS-Shield, the cost increases significantly with the increase of the attack rate. This is due to consuming more computing resources for processing the attack traffic which was also indicated clearly by the utilization results presented in Figure 5.6. However, the little increases in the cost of the enhanced EDoS-Shield compared to the base case is due to the cost of the mitigation technique itself, i.e., the cost incurred at VFs and V-Nodes.

Regarding the throughput rate, it is expected that the throughput of the legitimate requests will not be affected by the attack rate even without applying the mitigation technique as it is shown in Figure 5.8. This is due to the fact that the targeted cloud service is an on-demand cloud–based service. According to the nature of the cloud computing system, i.e., scalability nature, we assumed that there are enough on-demand cloud resources to be provisioned to the cloud instances executing the service. As a result, there is no degradation of the throughput rate of the legitimate requests in all of the evaluated scenarios except in case 2 where the attacks caused blocking of all the legitimate traffic, when the original EDoS-Shield technique is used.

Figure 5.9 shows the corresponding evaluation results for the analytical model when considering the whitelist case scenario. In general, the analytical model and the simulation model are very close to each other for all the studied metrics indicating a validation of the analytical model for the enhanced EDoS-Shield mitigation technique.

The resulted relative error percentage presented in Figure 5.10 for the whitelist case scenario shows a good accuracy in terms of the response time with maximum error of about 0.5%. Also, the accuracy of the utilization and the cost results shows a maximum error percentage of about 0.1%.

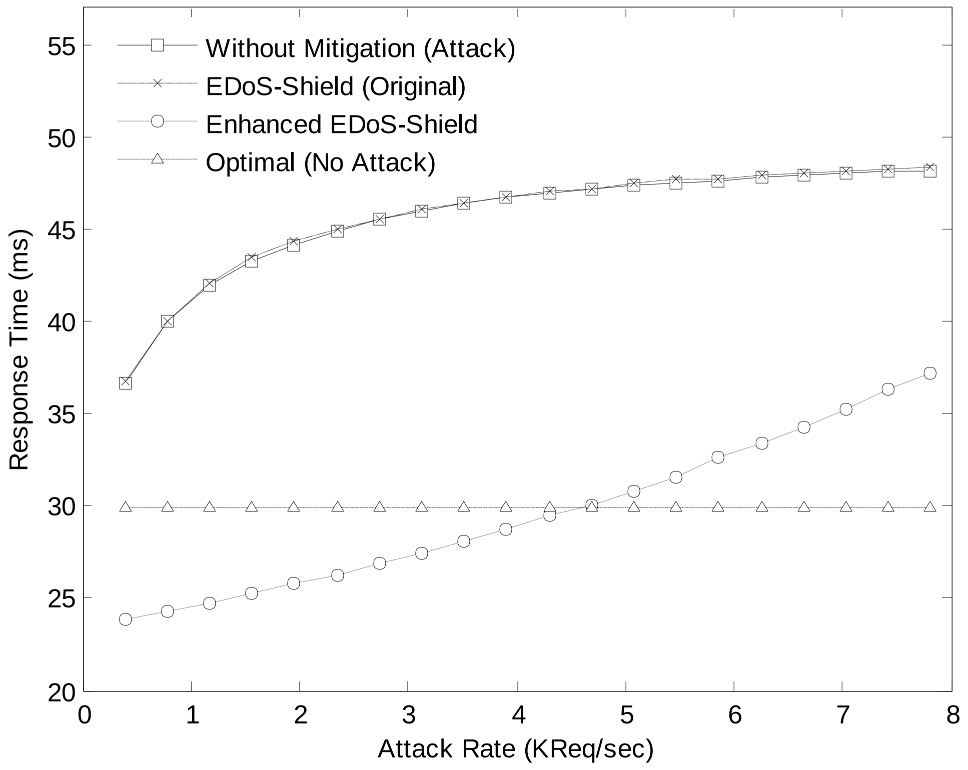


Figure 5.5: Whitelist case, simulation results for the end-to-end response time

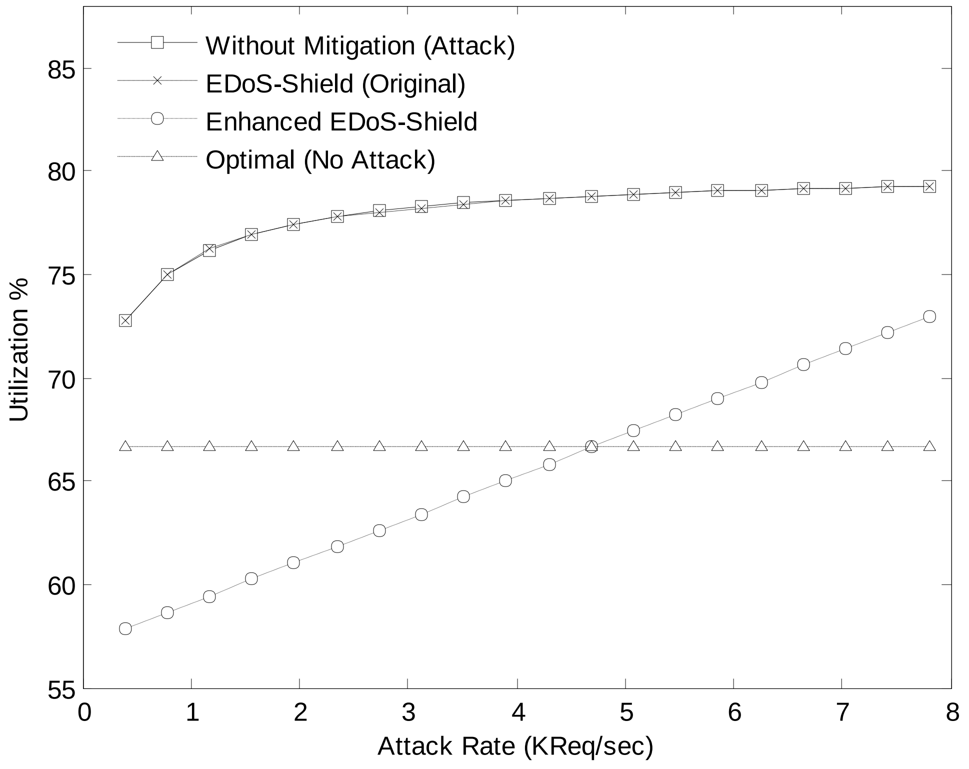


Figure 5.6: Whitelist case, simulation results for the computing resource utilization

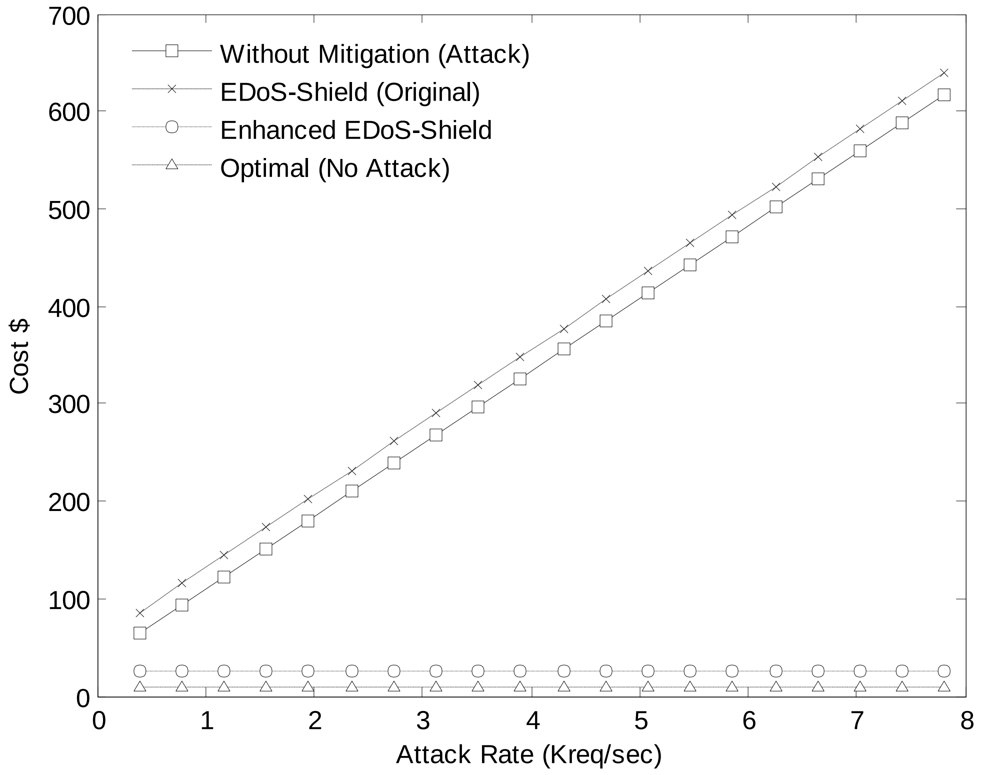


Figure 5.7: Whitelist case, simulation results for the cost evaluation

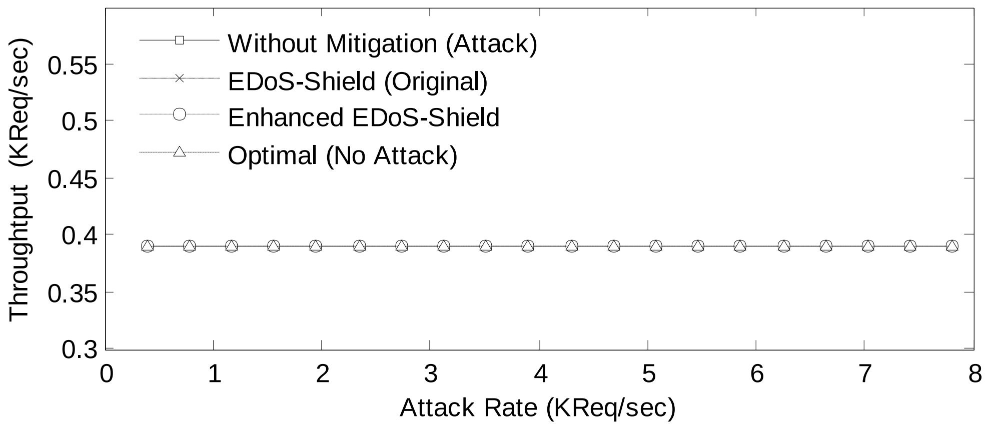


Figure 5.8: Whitelist case, simulation results for the legitimate client throughput rate

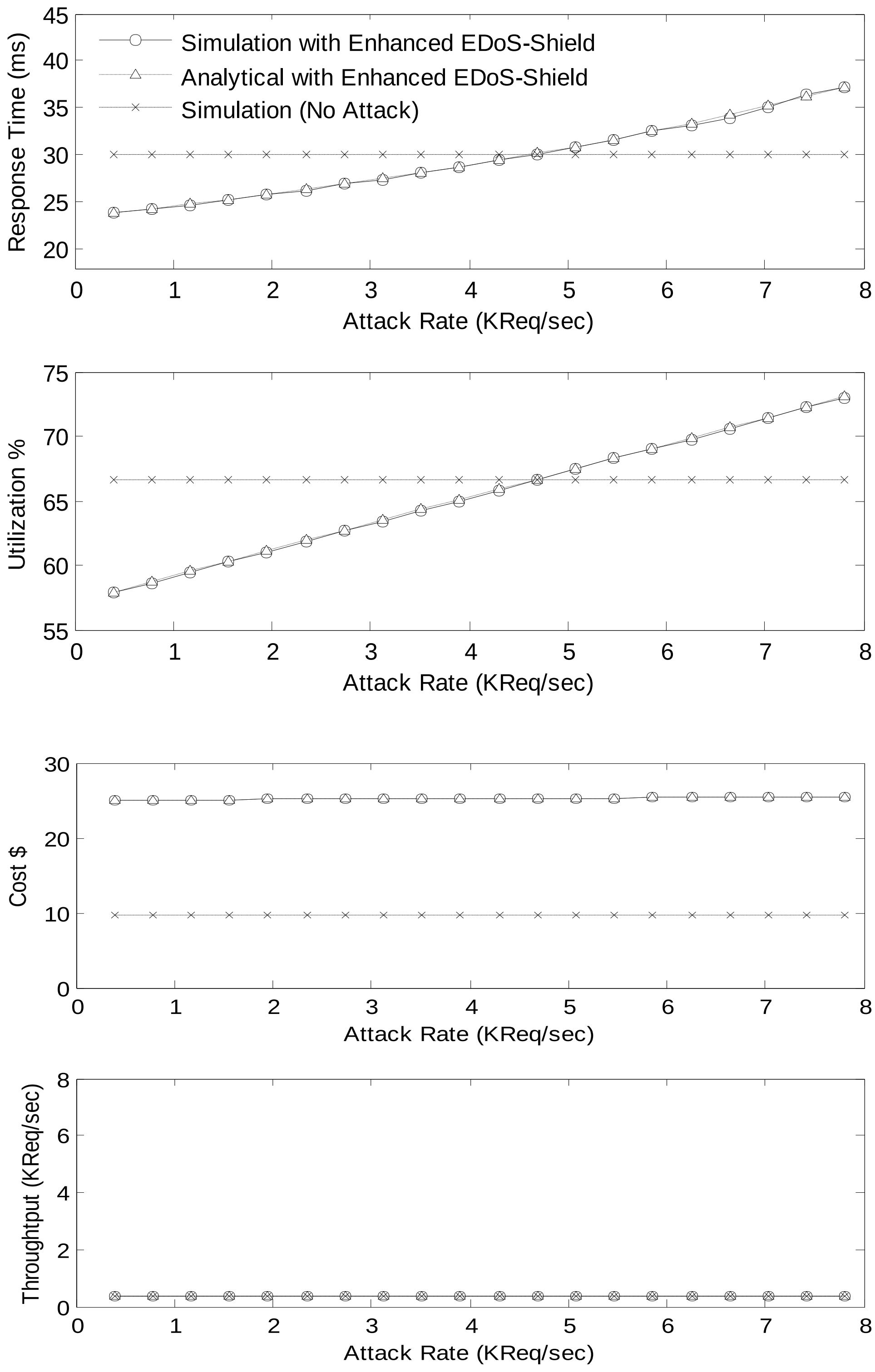


Figure 5.9: Whitelist case, analytical vs. simulation results with respect to the evaluated metrics

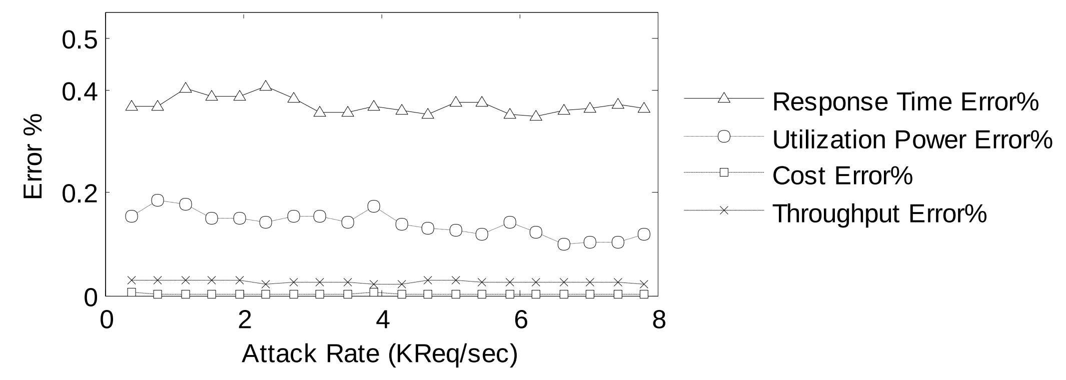


Figure 5.10: Whitelist case, relative error percentage of the analytical model

5.9.2 Blacklist Case Results

For the blacklist case, as it is obvious from all the figures related to the performance of the blacklist case, all the measurement metrics of the original EDoS-Shield technique have results of zero. This is due to blocking all the legitimate traffic targeting the cloud service, since their source IP address has already been verified to be a spoofed address and therefore has been added to the blacklist.

For the enhanced EDoS-Shield, results show that it has a performance that is close to the optimal base case. This is because the spoofed source IP address used by the attacker will be added to the blacklist. And, later that source IP address of the legitimate traffic will pass the verification phase to be added to the whitelist leading to a mixed case where that IP address will present in both the blacklist and the whitelist. As a result, all the legitimate traffic will pass the filter at VF and reach the destination. However, few of the attack traffic will be forwarded to the destination causing a little increase in the end-to-end response time, computing resources utilization, and the cost shown in Figures 5.11, 5.12, and 5.13 respectively.

Figure 5.15 shows the results obtained from the analytical model regarding the enhanced EDoS-Shield compared to its simulation results when considering the blacklist case scenario. In general, the analytical model and the simulation model are very close to each other for all the studied metrics with relative error percentage less that 0.3% as it is shown in Figure 5.16.

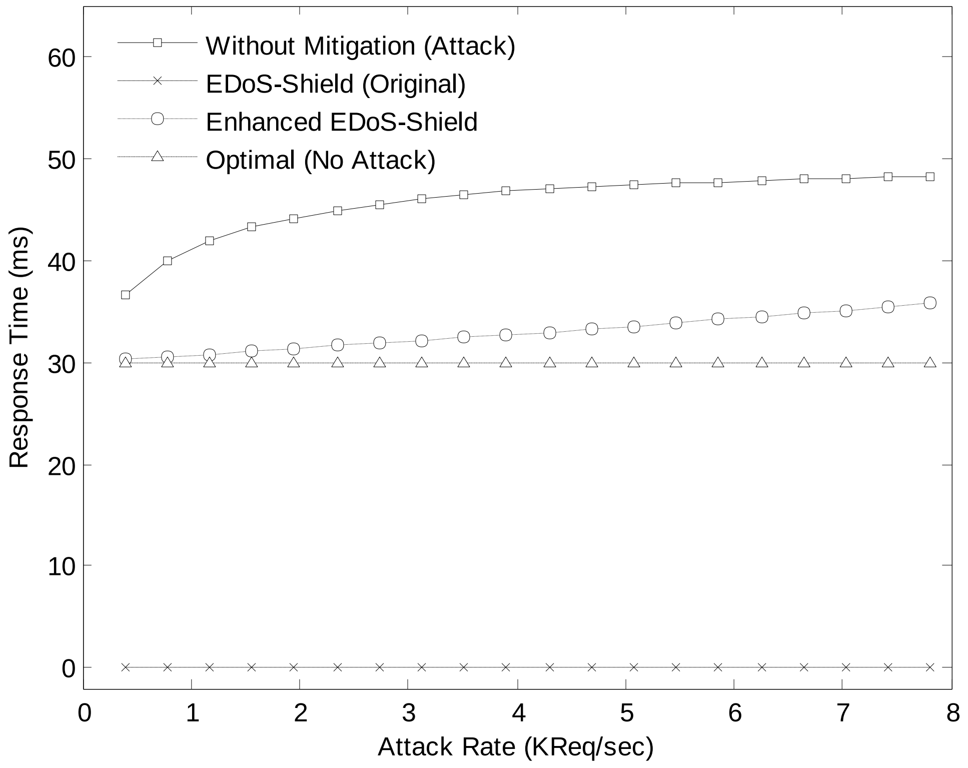


Figure 5.11: Blacklist case, simulation results for the end-to-end response time

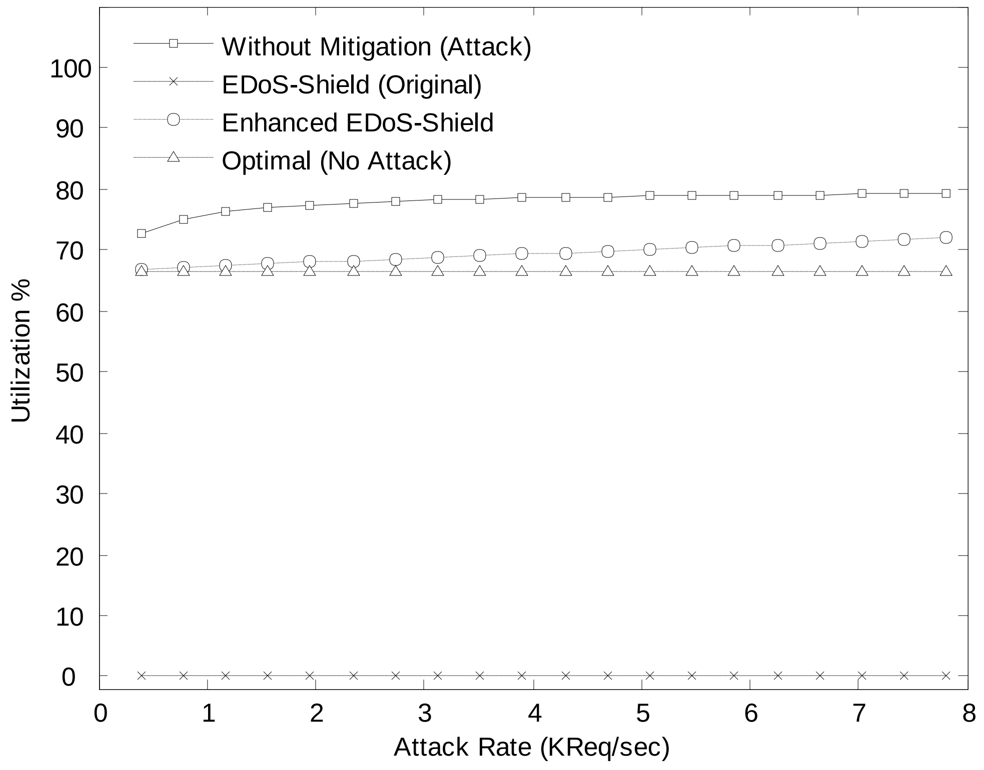


Figure 5.12: Blacklist case, simulation results for the computing resource utilization

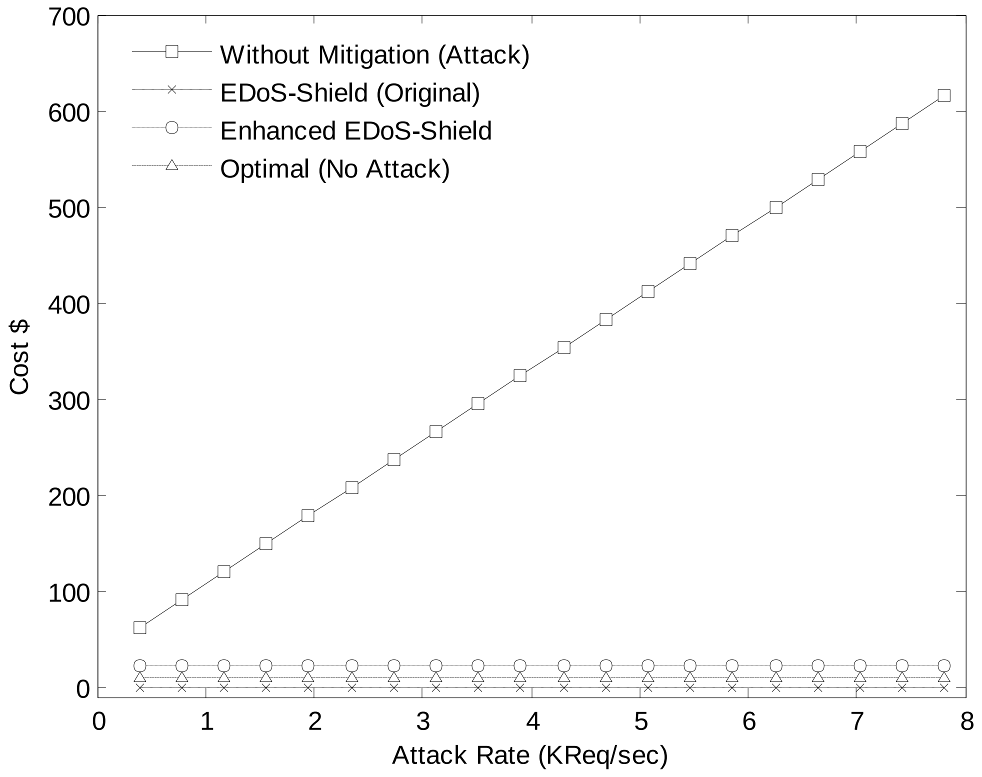


Figure 5.13: Blacklist case, simulation results for the cost evaluation

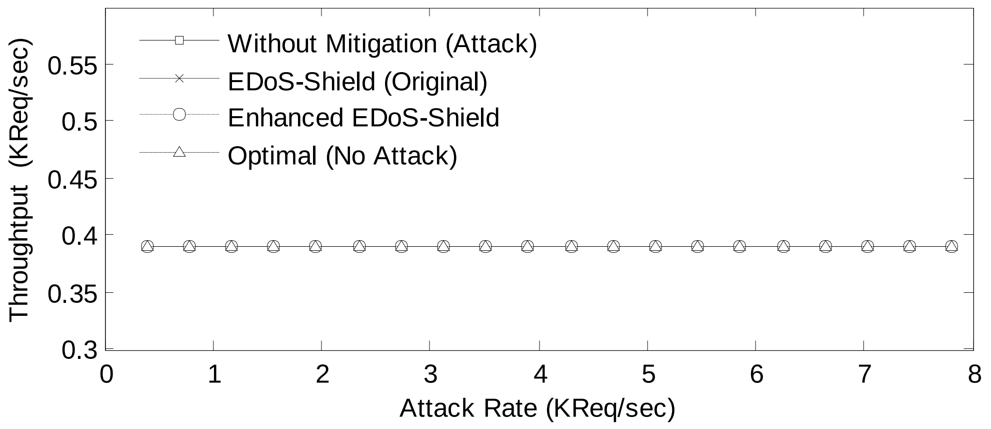


Figure 5.14: Blacklist case, simulation results for the legitimate client throughput rate

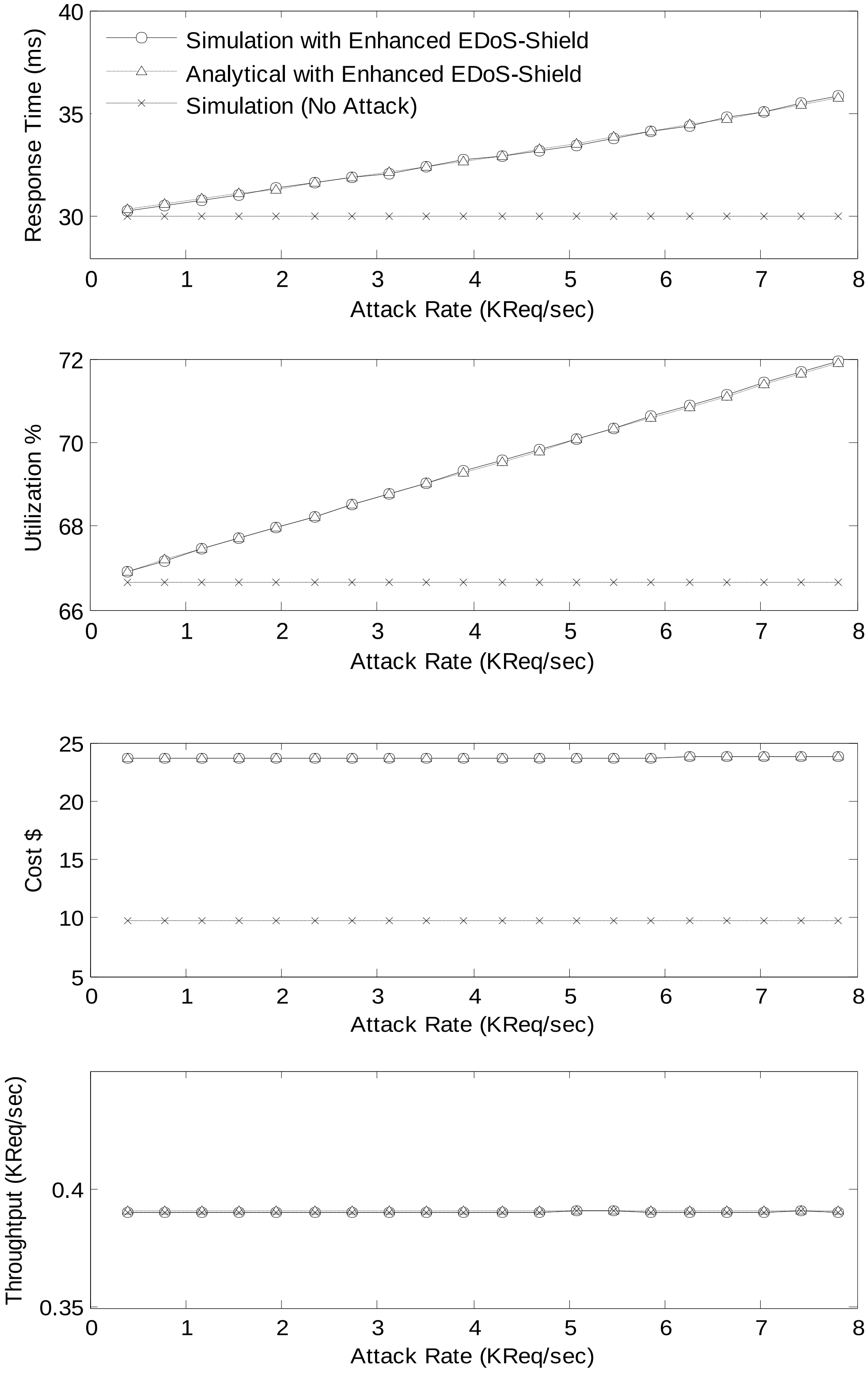


Figure 5.15: Blacklist case, analytical vs. simulation results with respect to the evaluated metrics

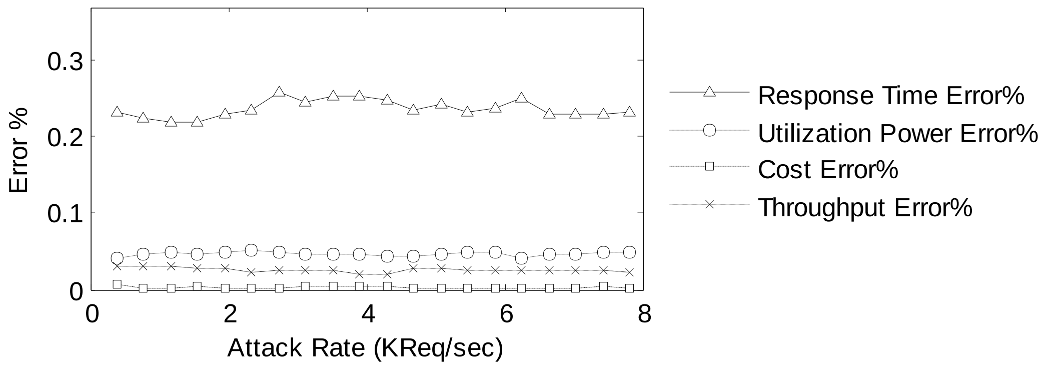


Figure 5.16: Blacklist case, relative error percentage of the analytical model

CHAPTER 6

CONCLUSION AND FUTRE WORK

This chapter presents a summary of our major contributions and findings in this dissertation work to model and mitigate the EDoS attacks targeting the cloud computing environment. It also gives indications of future research directions.

This dissertation first presented a literature review focusing on the taxonomy of the cloud computing attacks as well as the EDoS attacks targeting the cloud computing services, and then studied the impact of EDoS attacks on the performance and cost metrics of cloud computing services. Finally, this dissertation developed and modeled mitigation techniques to detect and prevent EDoS attacks. The major contributions and findings of this dissertation are summarized below.

Security of the cloud computing has been pointed as the main concern and challenge to the growth of this evolving paradigm. As a first contribution, in Chapter 3, this dissertation presented a study about the impact of the EDoS attacks on the cloud computing services. We developed a simulation model verified by analytical models to study such impact of EDoS attacks, considering a number of performance metrics. These metrics include end-to-end response time, utilization of computing resources being consumed, and the incurred cost resulting from the attack. The proposed analytical model is convenient to show the impact of EDoS attacks on both the performance and cost issues for the services offered by cloud computing. Although we concentrate on modeling the EDoS attack against cloud computing, the proposed model is also suitable for other similarly behaving attacks such as DDoS. Based on the obtained results from the evaluation models, we found that the EDoS attack has a considerable impact on both the performance metrics and the cost of the cloud services. For example, results showed that at an attack rate of 6,000 Req/sec, the total cost is about 15 times the normal case where no attack takes place. In addition, even though the main target of EDoS attacks is to charge the cloud adopters more cost due to the attack activities, we also found that it has an impact on the performance metrics of the cloud computing services such as the end-to-end response time.

As it is the case with DDoS attacks, an EDoS attack is more sophisticated and consequently more difficult to detect as its behavior is becoming closer to the legitimate clients. Thus, as a second contribution, this dissertation in Chapter 4, described a novel architecture, namely EDoS-Shield, which is deployable as an on-demand cloud-based EDoS mitigation technique. We developed discrete simulation and analytical models to verify the effectiveness of the EDoS-Shield mitigation technique. Based on the results obtained from both the simulation and analytical models, we found that EDoS-Shield is an effective approach in terms of performance and cost metrics, when considering that it is protected against the IP spoofing attacks. In addition, EDoS-Shield has more contributions as follows. First, our proposed approach reduces extremely the overhead due to indirect routing since subsequent packets after the first successful request would be forwarded directly to the protected cloud service. Second, our proposed technique does not require that the IP addresses of the verifier nodes be clear to the public; hence reducing the probability of exploiting these nodes. Finally, unlike most of the overlay-based approaches, our approach does not suffer from the problem of location-hiding as it is not required in our approach to hide the location of the protected cloud service.

In the original EDoS-Shield technique, the decision to forward or drop a packet is mainly based on the source IP address present in the whitelist and blacklist. Thus, it is as important to consider the case of IP spoofing when mitigating EDoS attacks. As a third contribution, this dissertation in Chapter 5, developed a novel and practical approach using a Graphical Turing test and TTL values from the IP header to update the whitelist and blacklist in order to mitigate EDoS attacks originating from spoofed IP addresses. The proposed approach is an enhancement to the EDoS-Shield where we considered different security issues regarding the proposed approach. In addition, we developed an analytical model for the enhanced EDoS-Shield to verify the simulation results. Based on the results obtained from both the simulation and analytical models, we found that it is an effective approach to mitigate EDoS attacks originating from spoofed IP addresses. Moreover, the enhanced EDoS-Shield technique also outperforms the original EDoS-Shield in all studied performance and cost metrics.

In the context of cloud computing, auto scaling is one of the main mechanisms used to assure QoS for cloud services as well as the efficient usage of resources. While modeling the cloud services, we found that the configuration parameters of the provisioning mechanism have a notable effect on the performance of cloud services. One of such important factors is the setting of thresholds that control the triggering of the auto scaling policies, for the purpose of adding or terminating resources from the auto-scaling group. Thus, in this dissertation, as a fourth contribution, we simulated and studied the impact of setting the upper utilization threshold on the performance of the cloud services. In addition, we formulated and solved an optimization problem for tuning the upper utilization threshold based on input loads. Even though we focus on a specific case study, such evaluation will help in understanding and deciding about the optimal threshold. The optimal threshold should lead to minimizing the cost in terms of the number of allocated instances and to provide an acceptable QoS for the cloud services.

Another important factor in the auto-scaling mechanism is the scaling size factor which has an impact on the performance of cloud services. The scaling size refers to the number of instances that will be added every time the provisioning process takes place to add more resources to cope with the spikes of the incoming load. The spikes in the incoming load might occur due to normal behavior of the cloud customers or due to malicious behavior such as EDoS attacks. Thus, in this dissertation, as a fifth contribution, we simulated and studied the impact of the scaling size during the provisioning process on the performance of the cloud services. In addition, we formulated and solved an optimization problem for tuning the scaling size of the provisioning mechanism considering both the cost and the response time. The study will help in understanding and deciding about the optimal scaling size that leads to minimizing the cost, and to provide an acceptable QoS for the cloud services.

Some directions for future work include the following. First, a suggested future work is to conduct an experimental implementation to validate the effectiveness of both the original EDoS-Shield and the enhanced one for mitigating EDoS attacks. Second, it would be interesting to extend the work done in this dissertation regarding the optimization problems for the purpose of tuning the provisioning mechanism parameters. For example, one could investigate tuning more parameters including monitoring window size, lower utilization threshold, and delay constraints. Third, another suggested future work is to evaluate the impact of EDoS attacks on the cloud computing services, while considering the spot pricing model instead of the on-demand pricing model that we have used through this dissertation. Finally, as a future work, we suggest investigating the impact of EDoS attacks using an experimental test bed to validate the findings obtained from both the simulation and analytical models that we have carried out to study the impact of EDoS attacks on the cloud computing services.

References

[1] Gartner, “Gartner Identifies the Top 10 Strategic Technologies for 2011”, Analysts Examine Latest Industry Trends During Gartner Symposium/ITxpo, October 18-22, in Orlando, 2010.

[2] P. Mell, and T. Grance, “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology. 2009.

[3] D. Catteddu, and G. Hogben, “Cloud Computing: benefits, risks and recommendations for information security”, Technical Report, European Network and Information Security Agency, 2009.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in cloud computing”, 17th IEEE International Workshop on Quality of Service (IWQoS 2009), Charleston, South Carolina, July 13-15, 2009

[5] Kaufman, L. M., “Data Security in the World of Cloud Computing”, Security & Privacy, IEEE 7, 4, pp. 61-64. 2009.

[6] J. Meiko and S. Joerg, “The accountability problem of flooding attacks in service-oriented architectures,” in Proceedings of the IEEE International Conference on Availability, Reliability and Security (ARES), 2009.

[7] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: vision, hype and reality for delivering computing as the 5th utility”, Future Generation Computer Systems 25 (6) (2009) 599616.

[8] N. Jesper, “On our extended downtime, Amazon and what’s coming”, Retrieved December 1, 2009, from http://blog.bitbucket.org/2009/10/04/on-our-extended-downtime-amazon-and-whats-coming, 2009.

[9] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared”, to appear at IEEE Grid Computing Environments (GCE08) 2008, co-located with IEEE/ACM Supercomputing 2008

[10] Google Apps, http://www.google.com/apps/intl/en/business/index.html, 2010

[11] K. Salah, A. Al-Dharrab, “Security in Cloud Computing: A Survey”, KFUPM, technical report, February 03, 2010.

[12] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2010.

[13] Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3, 2010.

[14] G. Briscoe, and A. Marinos, “Digital ecosystems in the clouds: Towards community cloud computing”, Digital Ecosystems and Technologies, DEST '09. 3rd IEEE International Conference on June 2009.

[15] C. A. Lee, “A perspective on scientific cloud computing”, Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10). ACM, New York, NY, USA, pp. 451-459, 2010.

[16] M. Jensen, J. Schwenk, N. Gruschka, and L. Iacono, “On Technical Security Issues in Cloud Computing”, in Proceedings of the IEEE International Conference on Cloud Computing, Bangalore, India, September 2009, pp.109-116.

[17] X. Li, Y. Li, T. Liu, J. Qiu, and F. Wang, “The Method and Tool of Cost Analysis for Cloud Computing”, cloud, 2009 IEEE International Conference on Cloud Computing, pp.93-100, 2009

[18] J. Heiser and M. Nicolett, “Assessing the security risks of cloud computing” Gartner, 3 June 2008, available at www.gartner. Com /DisplayDocument? id685308.

[19] International Data Corporation, “New IDC IT Cloud Services Survey: Top Benefits and Challenges”, IDC, December 2009. Available at http://blogs.idc.com/ie/wp-content.

[20] A. Bakshi, Yogesh B. Dujodwala, "Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machine," iccsn, pp.260-264, 2010 Second International Conference on Communication Software and Networks, 2010.

[21] J. Viega, “Cloud Computing and the Common Man,” published on the IEEE Journal ON Cloud Computing Security, McAffee, pp. 106-108, August 2009.

[22] S. Hanna, “Cloud Computing: Finding the Silver Lining”, Juniper Networks, pp. 2-30, 2009.

[23] “Amazon Web Services: Overview of Security Processes”, Amazon White Paper, available at http://aws.amazon.com/about-aws/whats-new/2009/06/08/new-aws-security-center-and-security-whitepaper, August 2010.

[24] J. Varia, “Cloud Architectures”, Whitepaper. Amazon Web Services, 2009.

[25] Cloud Security Alliance, “Security Guidance for Critical Areas of Focus in Cloud Computing”, Version 2.1, released December 17, 2009, Available at http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf

[26] A. Kieyzun, P. Guo, K. Jayaraman, M. Ernst, “Automatic creation of SQL Injection and cross-site scripting attacks,” icse, IEEE 31st International Conference on Software Engineering, pp.199-209, 2009

[27] G. Wassermann, and Z. Su, “Static detection of cross-site scripting vulnerabilities”, Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, May, 2008.

[28] X. Lin, P. Zavarsky, R. Ruhl, and D. Lindskog, “Threat Modeling for CSRF Attacks,” cse, vol. 3, , 2009 International Conference on Computational Science and Engineering, pp.486-491, 2009

[29] K. Wei, M. Muthuprasanna, and S. Kothari, “Preventing SQL Injection Attacks in Stored Procedures”, aswec, pp.191-198, Australian Software Engineering Conference (ASWEC'06), 2006

[30] M. McIntosh and P. Austel, “XML signature element wrapping attacks and countermeasures”, in SWS ’05: Proceedings of the 2005 workshop on Secure web services. ACM Press, pp. 20–27, 2005.

[31] N. Gruschka and L. Lo Iacono, “Vulnerable Cloud SOAP Message Security Validation Revisited”, in ICW ’09: Proceedings of the IEEE International Conference on Web Services. Los Angeles, USA: IEEE, 2009.

[32] “VMware KB- Critical VMware Security Alert for Windows-Hosted VMware Workstation, VMware Player, and VMware ACE”, KB Article: 1004034, Aug 14, 2009; Available at http://kb.vmware.com

[33] R. Wojtczuk, “Adventures with a certain Xen vulnerability”, version 1.0, October 14, 2008

[34] “MS09-033- Vulnerability in Virtual PC and Virtual Server could allow elevation of privilege”, Article ID: 969856, July 15, 2009; Available at http://support.microsoft.com

[35] S. King, P. Chen, “SubVirt: implementing malware with virtual machines”, Security and Privacy, IEEE Symposium on Digital Object, May 2006.

[36] J. Oberheide, E. Cooke, and F. Jahanian, “Empirical exploitation of live virtual machine migration”, In Proc. of BlackHat DC convention, 2008.

[37] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds”,ACM, November, 2009.

[38] B. Kandukuri, V. Paturi, A. Rakshit, “Cloud Security Issues”, Services Computing, 2009. SCC '09. IEEE International Conference, pp.517 - 520, Sept 2009.

[39] M. Xu, L. Cui, H. Wang, and Y. Bi, “A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing”, ispa, 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, pp.629-634, 2009

[40] L. Youseff, M. Butrico, and D. DaSilva, “Toward a Unified Ontology of Cloud Computing”, In Grid Computing Environments Workshop (GCE '08), Austin, Texas, USA, November 2008.

[41] J. Wei, X. Zhang, G. Ammons, et al., “Managing security of virtual machine images in a cloud environment”, CCSW '09: Proceedings of the 2009 ACM workshop on Cloud computing security, pp. 91-96, November 2009.

[42] N. Vax, “Securing Virtualized Environments and Accelerating Cloud Computing”, White Paper, CA Technologies, united state, May 2010.

[43] Cloud Security Alliance Group, “Top Threats to Cloud Computing V1.0”, Cloud Security Alliance, 2010. http://cloudsecurityalliance.org/topthreats.html.

[44] P. Baillie, “Security in a Public IaaS Cloud Part 3: Data Storage”, article on Cloud Computing, Des 2010. http://www.cloudsigma.com/en/blog/2010/09/13/10-security-in-a-public-iaas-cloud-networking.

[45] S. Castro, “Virtual machine Trojans: A new Type of threat?”, article in infosegura website, April 16th, 2009, http://www.infosegura.net/VMTthreat.html.

[46] Grove Group, “DDOS the biggest threat to cloud computing?”, iTWire's daily newsletter for telecommunications professionals, November 2010. http://www.itwire.com/it-industry-news/strategy/27498-ddos-the-biggest-threat-to-cloud-computing.

[47] R. Dobbins, “The Emperor’s New Cloud: An Analysis of the July 2009 RoK/ USA DDoS Attacks”, Arbor Networks, US, 2009, available at http://www.nanog.org/meetings/nanog47/presentations/Monday/Dobbins_ISPSecTrac_N47_Mond.pdf.

[48] C. Hoff, "Cloud Computing Security: From DDoS (Distributed Denial Of Service) to EDoS (Economic Denial of Sustainability)", Blog, Retrieved November 27, 2008, available at http://rationalsecurity.typepad.com/blog/2008/11/cloud-computing-security-from-ddos-distributed-denial-of-service-to-edos-economic-denial-of-sustaina.html

[49] K. Hosseini, I. Sommerville, and I. Sriram, "Research Challenges for Enterprise Cloud Computing", 1st ACM Symposium on Cloud Computing, SOCC, 2010.

[50] N. Dhanjani, B. Rios, and B. Hardin, "Hacking: The Next Generation", CA, O'Reilly Media Inc., 2009.

[51] A. Noureddine, and M. Damodaran, "Security in web 2.0 application development", Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, New York, ACM, 2008, pp. 681-685.

[52] H. Liu, "A New Form of DOS Attack in a Cloud and Its Avoidance Mechanism", in Proceedings of the 2010 ACM workshop on Cloud computing security workshop, Chicago, Oct. 2010, pp. 65-76.

[53] C. Metz, "DDoS attack rains down on Amazon cloud", Retrieved December 1, 2009, available at http://www.theregister.co.uk/2009/10/05/amazon_bitbucket_outage/

[54] J. Meiko, G. Nils, and L. Norbert, "The Impact of Flooding Attacks on Network-based Services", In Proceedings of the IEEE International Conference on Availability, Reliability and Security (ARES), 2008.

[55] M. Jensen and N. Gruschka, "Flooding Attack Issues of Web Services and Service-Oriented Architectures", In Proceedings of the Workshop on Security for Web Services and Service-Oriented Architectures (SWSOA, held at GI Jahrestagung 2008), pp. 117–122, 2008.

[56] S. HinKhor and A. Nakao, "sPoW: On-Demand Cloud-based eDDoS Mitigation Mechanism", HotDep (Fifth Workshop on Hot Topics in System Dependability), Lisbon, Portugal, 2009.

[57] L. Leong, "How to Select a Cloud Computing Infrastructure Provider", Gartner research, April 2009.

[58] T. Mather, S. Kumaraswamy, and S. Latif, "Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance", O'Reilly Media, Inc., 2009.

[59] Amazon Web Services Simple Monthly Calculator, available at http://calculator.s3.amazonaws.com/calc5.html.

[60] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", Technical Report, University of California, Berkeley, 2009.

[61] P. Singh and D. Kumar, "An Analytical Approach to Mitigate DDoS Attacks and improve Network Performance under Collaborative Software as a Service (SaaS) Cloud Computing Environment", DAV Institute of Engineering & Technology, oct 10, 2009.

[62] L. Rodero-Merino, et al., "From infrastructure delivery to service management in clouds", Future Generation Computer Systems,doi, 2010.

[63] H. Chen, and S. Li, "A queueing-based model for performance management on cloud", 6th International Conference on Advanced Information Management and Service (IMS), Seoul, pp. 83-88, Feb. ,2011.

[64] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model", in IEEE MMSP, pp. 1–6, Oct. 2010

[65] R. Calheiros, R. Ranjan, and R. Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments", International Conference on Parallel Processing (ICPP), Taipei City, pp. 295 – 304, Sept 2011.

[66] Amazon, "Amazon Load Balancer Service", available at http://aws.amazon.com/elasticloadbalancing/.

[67] AWS Documentation, AWS Web Application Hosting for Microsoft Windows, available at http://docs.amazonwebservices.com/gettingstarted/latest/wah/web-app-hosting-intro.html?r=1052

[68] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services", In The 10th International Conference on Algorithms and Architectures for Parallel Processing, Busan, Korea, 2010.

[69] D. Bellenger, J Bertram, A Budina, A Koschel, and et al., "Scaling in cloud environments", Proceedings of the 15th WSEAS international conference on Computers, Wisconsin, pp. 145-150, 2011.

[70] "Amazon Auto Scaling Developer Guide", Amazon Web Services LLC, 2012.

[71] "Web Application Hosting in the AWS Cloud: Best Practices", Amazon Web Services LLC, May 2010.

[72] J. Idziorek, "Discrete Event Simulation Model for Analysis of Horizontal Scaling in the Cloud Computing Model", Proceedings of the 2010 Winter Simulation Conference, pp. 3004-3014, Dec 2010.

[73] M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", IEEE/ACM Transactions on Networking, Vol. 5, No. 5, pp. 815-826, October 1997.

[74] J. Walraevens, S. Wittevrongel, and H. Bruneel, "Performance analysis of a priority queue with session-based arrivals and its application to E-commerce web servers", International Journal On Advances in Internet Technology 2(1) (2009) 46–57, 2009.

[75] Z. Liu, N. Niclausse, and C. Jalpa, "Traffic model and performance evaluation of web servers", Performance Evaluation, vol. 46(2-3),pp. 77–100, 2001.

[76] N. Singh, S.P. Ghrera, and P. Chaudhuri, "Denial of Service Attack: Analysis of Network Traffic Anormaly using Queuing Theory", Journal of Computer Science and Engineering, Vol. 1, Issue 1, pp. 48-54, May 2010.

[77] Y. Wang, C. Lin, Q. Li, Y. Fang, " A Queueing Analysis for the Denial of Service (DoS) Attacks", Computer Networks 51, 3564–3573 (2007)

[78] D. Boteanu, J.M. Fernandez, J. McHugh,and J. Mullins, "Queue Management as a DoScounter-measure?",In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 263–280. Springer, Heidelberg (2007)

[79] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, "Fundamentals of Queuing Theory", John Wiley and Sons Inc., 2008.

[80] H. Liu and S. Wee, "Web Server Farm in the Cloud: Performance Evaluation and Dynamic Architecture", In Proc. of the 1st International Conference on Cloud Computing (CloudCom 2009), Dec 2009.

[81] Amazon EC2 Pricing, available from http://aws.amazon.com/ec2/pricing/

[82] A. Law and W. Kelton, "Simulation Modeling and Analysis", McGraw-Hill, Third Edition, 2000.

[83] K. C. Claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone", In the Proceedings of INET 1998, Geneva, Switzerland, July 1998.

[84] K. Xiong and H. Perros, “Service performance and analysis in cloud computing”, In SERVICES ’09: Proceedings of the2009 Congress on Services - I, 2009.

[85] S. Shan and G. Wang, “An efficient pareto set identification approach for multi objective optimization on black-box functions”, Journal of Mechanical Design 127 (5): 866, 2005.

[86] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, “Resource provisioning for cloud computing”, in Proceedings of the 2009 Conference of the Centerfor Advanced Studies on Collaborative Research (CASCON ’09). ACM,pp. 101–111, 2009.

[87] J. Bi, Z. Zhu, R. Tian, and Q. Wang, “Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in CloudData Center”, Proceedings of IEEE 3rd International Conference on Cloud Computing (CLOUD 2010), pp. 370-377, 2010.

[88] W. Dawoud, I. Takouna, and C. Meinel, “Elastic VM for rapid and optimum virtualized resources' allocation”, 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM), Paris, pp. 1-4, Oct. 2011.

[89] G. Juve and E. Deelman, “Automating Application Deployment in Infrastructure Clouds”, In 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011), 2011.

[90] D. Kossmann, T. Kraska, and S. Loesing, “An Evaluation of Alternative Architectures for Transaction Processing in the Cloud”, In Proc. of international conference on Management of data (SIGMOD), 2010.

[91] R. Pal and P. Hui, “Economic Models for Cloud Service Markets”, Lecture Notes in Computer Science, Distributed Computing and Networking, Springer, vol. 7129, pp. 382-396, 2012.

[92] Y. Shi, X. Jiang, and K. Ye “An Energy-Efficient Scheme for Cloud Resource Provisioning Based on CloudSim”, 2011 IEEE International Conference onCluster Computing (CLUSTER), Austin, TX, pp. 595 – 599, Sep. 2011.

[93] S. Islam, K. Lee, A. Fekete, and A. Liu, “How A Consumer Can Measure Elasticity for Cloud Platforms”, technical report, SCHOOL OF INFORMATION TECH, Univercity of Sydeny, 2011.

[94] R. Knode, “BP Fuels Cloud Computing Interest”, Retrieved January 2010, available from http://www.trustedcloudservices.com/Individual-Case-Studies/bp-fuels-cloud-computing-interest.

[95] M. Kihl, G. Cedersj¨o, A. Robertsson, and B. Aspern¨as, “Performance measurements and modeling of database servers”, in Sixth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID 2011), Jun. 2011.

[96] C. Sutton and M. I. Jordan, “Bayesian Inference for Queueing Networks and Modeling of Internet Services”, Institute of Mathematical Statistics in the Annals of Applied Statistics, vol. 5 (1), pp. 254–282, 2011.

[97] T. Do, U. R. Krieger, and R. Chakka, “Performance modeling of an apache web server with a dynamic pool of service processes”, Telecommunication Systems , vol. 39 (2), pp.117–129, 2008.

[98] R. Singh, et al., “Autonomic mix-aware provisioning for non-stationary data center workloads”, InProc. of the 7th Intl. Conf. on Autc. Comp., USA ,2010)

[99] “Intel 82599 10 gigabit Ethernet controller”, Intel, 2009. Available at http://download.intel.com/design/network/prodbrf/321731.pdf.

[100] Y. DONG, X. YANG, X. LI, K. TIAN, and H. GUAN, “High performance network virtualization with SR-IOV”, In IEEE International Symposium on High Performance Computer Architecture (HPCA) , 2010.

[101] R.T. Marler and J.S. Arora, “Survey of Multi-Objective Optimization Methods for Engineering”, Structural and Multidisciplinary Optimization, vol. 26, no. 6, pp. 369-395, 2004.

[102] A. Antoniou, and W.S. Lu, “Practical Optimization: Algorithms and Engineering Applications”, Springer, Berlin, 2007.

[103] J. Cohon, “Multi objective programming and planning”, New York: Academic Press, 1978.

[104] MathWorks, ”fminbnd”, R2012a Documentation , available at http://www.mathworks.com/help/techdoc/ref/fminbnd.html

[105] W. Scheinhardt, “Markov-modulated and feedback fluid queues”, Available at http://www.ub.utwente.nl/webdocs/tw/1/t0000008.pdf, Ph.D. Thesis, University of Twente, the Netherlands, 1998.

[106] X. Shen, H. Chen, J. Dai, and W. Dai, “The finite element method for computing the stationary distribution of an SRBM in a hypercube with applications to finite buffer queueing networks”, Queueing Systems, 42(1):33–62, 2002.

[107] "Amazon CloudWatch", Amazon Website, available from http://aws.amazon.com/cloudwatch/

[108] J. Green, J. Juen, O. Fatemieh, R. Shankesi, D. Jin, and C. Gunter, " Reconstructing hash reversal based proof of work schemes", in LEET'11 Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats, Boston, MA, March 2011.

[109] D. Keromytis, V. Misra, and D. Rubenstein. "Sos: Secure overlay services", In Proceedings of the Con-ference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIG-COMM'02), August 2002.

[110] A. Stavrou et al, "Websos: An overlay-based system for protecting web servers from denial of service attacks", the International Journal of Computer and Telecommunications Networking, vol. 48(5), pp.781–807, August 2005.

[111] H. Beitollahi and G. Deconinck, "Fosel: Filtering by helping an overlay secure layer to mitigate dos attacks", In Proceedings of the 7th IEEE Int. Symp. on Network Computing and Applications (NCA-2008), pages 19–28, Cambridge, MA (USA), July 2008.

[112] D. Ping, and A. Nakao, "DDoS defense as a network service", Network Operations and Management Symposium (NOMS), IEEE, Osaka, pp. 894 – 897, April 2010.

[113] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security", In Proceedings of Eurocrypt, 2003.

[114] P. Karwaczynski and J. Kwiatkowski, "Analysis of overlay network impact on dependability", In proceedings of the 38th Hawaii International conference on system science, 2005.

[115] H. Beitollahi and G. Deconinck, "An overlay protection layer against denial-of-service attacks", In Proceedings of the 22rd IEEE International Parallel and Distributed Processing Symposium, pp.1–8, April 2008.

[116] J. Wang and A. Chien, "Understanding when location-hiding using overlay networks is feasible", Journal of Computer Networks, vol. 50(6), pp.763 – 780, April 2006.

[117] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein, "Using graphic turing tests to counter automated ddos attacks against web servers", in ACM CCS, 2003.

[118] J. Yan,A.s. El Ahmad, “CAPTCHA Security: A Case Study”, Security & Privacy, IEEE, vol. 7, pp. 22 – 28, Aug. 2009.

[119] M. Mehra , M. Agarwal, R. Pawar, and D. Shah, “Mitigating denial of service attack using CAPTCHA mechanism”, ICWET Workshop on Emerging Trends in Technology, ACM New York, NY, USA, pp. 284-287, 2011.

[120] S. Raj, B. Edwin, D. Deepa; J. Jiji, “A new architecture for the generation of picture based CAPTCHA”, Electronics Computer Technology (ICECT) 2011, 2011 3rd International Conference, Kanyakumari, India , pp. 67-71, April 2011.

[121] J. Israr, M. Guennoun, and H. Mouftah, “Mitigating IP Spoofing by Validating BGP Routes Updates”, IJCSNS International Journal of Computer Science and Network Security, vol. .9, no. 5, May 2009.

[122] C. Schridde, M. Smith, and B. Freisleben, “TrueIP: prevention of IP spoofing attacks using identity-based cryptography”, SIN '09 Proceedings of the 2nd international conference on Security of information and networks, ACM, pp. 128-137, 2009.

[123] M. Ravi, S. Narasimman, G. K. Arun, and D. Karthikeyan, “A Cryptographic Approach to Defend against IP Spoofing”, Communications in Computer and Information Science, CCIS, vol. 70, pp. 290-296, 2010.

[124] M. Subramanian, and T. Angamuthu, “An Autonomous Framework for Early Detection of Spoofed Flooding Attacks”, International Journal of Network Security, vol. 10, no. 1, pp. 39-50, Jan. 2010.

[125] C.F. Sharp, and S. Dark, “System and method for detecting and eliminating IP spoofing in a data transmission network”, US Patent 7,865,945 B2, Google Patents, Jan 2011.

[126] N. Desai, S. Dillon, D. Stone, “Methods and Systems for Load Balancing Using Forecasting and Overbooking Techniques”, US Patent, Pub no. 2011/0078318 A1, Google Patents, March 2011.

[127] J. Leitão, R. Renesse, and L. Rodrigues, “Balancing gossip exchanges in networks with firewalls”, IPTPS'10 Proceedings of the 9th international conference on Peer-to-peer systems, USA, 2010.

[128] V. Ramsurrun, and K. Soyjaudah, “A stateful CSG-based distributed firewall architecture for robust distributed security”, Communication Systems and Networks and Workshops, First International COMSNETS, Bangalore, pp. 1 - 10, Jan 2009.

[129] C. O'rourke et al. , "Method for protecting a firewall load balancer from a denial of service attack", US Patent 7,770,215 B2, Google Patents, Aug 2010.

[130] S. Carlin and K. Curran, "Cloud Computing Security", International Journal of Ambient Computing and Intelligence, Vol. 3, No. 1, pp. 38-46, June 2011.

[131] VMware, Inc, White Paper, "vShield Edge Design Guide", Jun 8, 2011, Available from http://www.vmware.com/files/pdf/techpaper/vShield-Edge-Design-Guide-WP.pdf

[132] Cisco White Paper, "Secure Virtual Applications and Data Centers", 2011. Available at http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns1097/white_paper_181-652663.html

[133] Savvis, Inc, White paper, "Securing-Virtual-Compute-Infrastructure-in-the-Cloud", 2011, Available at http://www.savvis.com/en-US/Info_Center/Documents/HOS-WhitePaper-SecuringVirutalComputeInfrastructureintheCloud.pdf

[134] W. Huang, J. Yang, "new network security based on cloud computing", Education Technology and Computer Science (ETCS), Second International Workshop , Wuhan, March 2010, pp. 604 – 609.

[135] http://cloudleverage.com/cloud-ips-firewall/pricing/

[136] FlexiScale, "Utility computing on demand," http://flexiscale.com/, January 2009.

[137] Amazon Web Services, "Amazon EC2 Instance Types", Available from: http://aws.amazon.com/ec2/instance-types/

[138] K. Chandy and C. Sauer, "Approximate methods for analyzing queueing network models of computing systems", Journal of ACM Computing Surveys, vol. 10, no. 3, pp. 281-317, Sep. 1978.

[139] L. Kleinrock, "Queueing Systems: Theory", vol 1, New York, Wiley, 1975.

[140] R. Suri, "Robustness of Queueing Network Formulas," Journal of the ACM, vol. 30, no. 3, pp. 564-594, Jul. 1983.

[141] K. Salah, K. Elbadawi, and R. Boutaba, "Performance modeling and analysis of network firewalls", IEEE Transactions on Network and Service Management, vol. 9, no. 1, pp. 12–21, 2012.

[142] H. Liu and S. Wee, "Web Server Farm in the Cloud: Performance Evaluation and Dynamic Architecture", In Proc. of the 1st International Conference on Cloud Computing (CloudCom 2009), Dec 2009.

[143] K. C. Claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," In the Proceedings of INET 1998, Geneva, Switzerland, July 1998.

[144] Amazon Web Services, "Amazon EC2 FAQs", available from: http://aws.amazon.com/en/ec2/faqs/#What is an EC2 Compute Unit and why did you introduce it.

[145] A. Wool, "A quantitative study of firewall configuration errors", IEEE Computer, 37(6), 2004.

[146] M. Sqalli, F. Al-Haidari, and K. Salah, "EDoS-Shield - A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing", Fourth IEEE International Conference on Utility and Cloud Computing (UCC 2011), Victoria, NSW, pp. 49 – 56, 2012.

[147] S. Malliga, and A. Tamilarasi, "Collaborative Framework for Detection, Prevention, and Traceback of Flooding Attacks Using Marking and Filtering", Information Security Journal: A Global Perspective,USA , vol. 18,pp. 74-86 Jan. 2009.

[148] A. Bremler-Barr and H. Levy, "Spoofing Prevention Method", In Proceedings of IEEE INFOCOM, Miami, FL, March 2005.

[149] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the Internet", in Proc. USENIX Annu. Technical Conf., pp. 1-12, 2000.

[150] B. Mopari, S. G. Pukale and M. L. Dhore, "Detection and defense against DDoS attack with IP spoofing", International Conference on Computing, Communication and Networking, 2008, pp. 1-5, Dec. 2008.

[151] H. Wang, C. Jin and K. Shin, "Defense against Spoofed IP Traffic Using Hop-Count Filtering", IEEE/ACM Trans. Networking, vol.15 No. I, Feb. 2007.

[152] V. Paxson, "End-to-end routing behavior in the Internet", IEEE/ACM Trans. Networking 5, pp. 601–615, 1997.

[153] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations", in Proc. ACMSIGCOMM Internet Measurement Workshop, pp. 197–202, 2000.

[154] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb, “A case study of OSPF behavior in a large enterprise network”, in Proc. ACM SIGCOMM Internet Measurement Workshop, pp. 217–230, 2002.

[155] Y. Schwartz, Y. Shavitt, and U. Weinsberg, “On the Diversity, Stability and Symmetry of End-to-End Internet Routes”, in Global Internet, San Diego, CA, pp. 1-6, March 2010.

[156] I. Cunha, R. Teixeira, and C. Diot, “Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing”, in Passive and Active Measurement Conference, 2011.

[157] X. Wang, M. Li, and M. L I, “A scheme of distributed hop-count filtering of traffic”, In IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009), Shanghai, China, pp. 516 – 521, 2010.

[158] M. Ohta, Y. Kanda, K. Fukuda, and T. Sugawara, “Analysis of Spoofed IP Traffic Using Time-to-Live and Identification Fields in IP Headers”, in IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA), Biopolis , pp. 355-361, 2011.

[159] B. KrishnaKumar, P.K. Kumar, and R. Sukanesh, “Hop Count Based Packet Processing Approach to Counter DDoS Attacks”, in International Conference on Recent Trends in Information, Telecommunication and Computing (ITC), Kochi, Kerala, pp. 271 – 273, 2010.

[160] R. Beverly, A. Berger, Y. Hyun, and k. claffy, “Understanding the efficacy of deployed internet source address validation filtering”, Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, Chicago, Illinois, USA, pp. 356-369, Nov. 2009.

[161] R. Beverly, S. Garfinkel, and G. Cardwell, “Forensic carving of network packets and associated data structures”, The Proceedings of the Eleventh Annual DFRWS Conference, 11th Annual Digital Forensics Research Conference, Digital Investigation, 8(Supplement 1):S78 – S89, 2011.

[162] L. Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, “reCAPTCHA: Human-Based Character Recognition via Web Security Measures”, Science, pp. 1465-1468, Sep. 2008.

[163] D. Moore, G. Voelker, and S. Savage, “Inferring Internet Denial-of-Service Activity”, ACM Transactions on Computer Systems (TOCS), USA, vol. 24, Issue 2, pp. 115 – 139, May 2006.

[164] CERT Coordination Center, “Cert advisory ca-1999-17 denial-of-service tools”, 2004. Available at http://www.cert.org/advisories/CA-1999-17.html.

[165] D. Dittrich, “The 'stacheldraht' distributed denial of service attack tool”, 2004. Available at http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

[166] B. Xiao, W. Chen, and Y. He, “An Autonomous Defense against SYN Flooding Attacks: Detect and Throttle Attacks at the Victim Side”, Journal of Parallel and Distributed Computing (JPDC - Elsevier), Vol. 68, Iss. 4, pp. 456-470, April 2008.

[167] S. Dietrich, N. Long, and D. Dittrich, “Analyzing distributed denial of service tools: The shaft case”, in Proc. USENIX LISA 2000, New Orleans, LA, pp. 329–339, 2000.

[168] Q. Zhu and G. Agrawal, “Resource provisioning with budget constraints for adaptive applications in cloud environments”, in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC’10), 2010.

[169] T. Hwang, T. Lee, and Y. Lee, “A Three-tier IDS via Data Mining Approach”, In Proceedings of the 3rd annual ACM workshop on Mining network data, pp.1-6, ACM Press New York, NY, USA, 2007.

[170] F. Soldo, A. Markopoulou, and K. Argyraki, “Optimal Filtering of Source Address Prefixes: Models and Algorithms”, IEEE INFOCOM, Roide Janeiro, Brazil, 2009

[171] S. Simpson, A.T. Lindsay, and D. Hutchison, “Identifying Legitimate Clients under Distributed Denial-of-Service Attacks”, In Network and System Security (NSS), IEEE, Melbourne, VIC, pp.365-370, 2010.

[172] K. Park, D. Seo, J. Yoo, H. Lee, H. Kim, “Unified Rate Limiting in Broadband Access Networks for Defeating Internet Worms and DDoS Attacks”, 4th International Conference on Information Security Practice and Experience, ISPEC, Sydney, Australia, pp. 176-187, 2008.

Vita

Name	Fahd Abdulsalam Al-Haidari
Nationality	Yemen
Date of Birth	22/8/1975
Present Address	Dhahran-KSA
Permanent Address	Taiz-Yemen
Email	fahdhyd@yahoo.com
Mobile	0501371232
Academic Background	Computer Science