Fuzzy Logic based Application Layer DDoS Mitigation System in Cloud

Neeraj Balaji

Submitted as part of the requirements for the degree
of MSc in Cloud Computing
at the School of Computing,
National College of Ireland
Dublin, Ireland.

January 2016

Supervisor Dr. Dominic Carr

Abstract

Cloud computing is at the forefront of Information Technology and has revolutionized computing in many ways. As numerous enterprises move into the cloud, the chances of being targeted by attacks especially Application Layer Distributed Denial of Service (DDoS) attacks will increase dramatically. Such attacks are capable of exhausting a victim's resources (such as servers, network, storage and applications), denying access and overloading with bogus requests, resulting in significant economic loss. Unlike the traditional DDoS attacks which occur at the network layer, these attacks occur at the application layer where the detection is comparatively difficult, since the attacker already has a valid connection to the victim server. Fuzzy logic is a precise algorithm for imprecise system that plays an important role in decision making process of the incoming HTTP requests to decide whether the request is malicious or genuine. Hence, a Fuzzy Logic based Application Layer Mitigation is necessary to handle such threats for maintaining cloud based services and ensuring availability of enterprise systems. Hence, this thesis is focused on combining evidences of the existing architectures (used for protecting and tracing DDoS attacks) as well as filtering malicious requests using DDoS mitigation system and Fuzzy Logic.

Keywords: DDoS, Application Layer, Cloud, Fuzzy Logic, Mitigation

Submission of Thesis to Norma Smurfit Library, National College of Ireland

Student name: Neeraj Balaji Student number: 14109069

School: School of Computing Course: MSc in Cloud Computing

Degree to be awarded: MSc in Cloud Computing

Title of Thesis: Fuzzy Logic based Application Layer DDoS Mitigation Sysrtem in Cloud

Signature of Candidate: ____________________

For completion by the School:
The aforementioned thesis was received by____________________ Date:______________

Submission of Thesis and Dissertation

National College of Ireland

Research Students Declaration Form

(Thesis/Author Declaration Form)

Name: Neeraj Balaji

Student Number: 14109069

Degree for which thesis is submitted: MSc in Cloud Computing

Material submitted for award

(a) I declare that the work has been composed by myself.

(b) I declare that all verbatim extracts contained in the thesis have been distinguished by quotation marks and the sources of information specifically acknowledged.

(c) My thesis will be included in electronic format in the College Institutional Repository TRAP (thesis reports and projects)

(d) ***Either*** *I declare that no material contained in the thesis has been used in any other submission for an academic award.

Or *I declare that the following material contained in the thesis formed part of a submission for the award of

Master of Science in Cloud Computing awarded by QQI at level 9 of the NFQ.

(State the award and the awarding body and list the material below)

Signature of research student: ______________________

Date: 26 January, 2016

Acknowledgement

This thesis ‘Fuzzy Logic Based Application Layer DDoS Mitigation System in Cloud’ for Masters in Cloud Computing has been accomplished at Cloud Competency Center, National College of Ireland.

I am highly indebted to my Supervisor Dominic Carr for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

I would like to express my gratitude towards Dr. Keith Brittle and Dr. Adriana Chis for their help in completion of the dissertation. I also wish to thank all the people who helped directly and indirectly to complete the research work.

I am grateful to my family for their blessings, my friends/classmates for their help and wishes for the successful completion of this project.

Declaration

I confirm that the work contained in this MSc Dissertation report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed:...................... Date: ....

Neeraj Balaji

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Chapter 1

Introduction

1.1 Preface

The recent history of enterprise computing is characterized by near-constant change. There are two changes that are prominent i.e. there is an evident shift from monolithic applications to distributed applications, as a result the enterprise data is moving to different locations. Secondly, the sensitive data of the enterprises is no longer in the datacenters, it is moving out from the company to the cloud.

Undoubtedly, cloud computing is a promising technology but as the technology emerges and more enterprises move to cloud, there are going to be voids and weaknesses which can be easily penetrated to implode a whole organization from within.

Hence, the attacks on the cloud are going to be frequent (Sabahi [2011]). Among these attacks, Distributed Denial of Service (DDoS) attacks are the most conspicuous in the cloud environment as they are capable of destroying organization by leeching resources and stealing bandwidth. Layer 7 (Application layer) based DDoS attacks are becoming the most prominent DDoS attacks in cloud which are devastating to organizations and are hard to identify and mitigate as compared to the network layer DDoS attacks.

In an application layer DDoS attack, the attacker first establishes a genuine connection to the server and starts flooding the server with bogus HTTP requests which depletes resources in the application layer and makes the services unavailable. Identifying and mitigating such attacks is essential to protect sensitive data of an enterprise.

Fuzzy logic is a precise technique to analyze datasets. According to Novak et al. [1999], fuzzy logic is more accurate in determining the nature of incoming requests when compared to other methods . In this context, Fuzzy Logic will aid in determining the nature of incoming HTTP requests with the help of pre-defined rules and datasets.

So, the research question that is going to be discussed in this thesis is **'Can we mitigate Application-Layer DDoS attacks with the help of Fuzzy Logic techniques?'**

1.2 Motivation

There is a lack of substantial mitigation against application layer DDoS attacks. Unlike other malware and network based attacks, these are mostly invulnerable as they occur in the application layer. (Imperva [2012]) Also, when Botnets (nodes or terminals that are unaware of being involved in an attack) are used for a DDoS attacks, the origin is not limited to a single source, in fact the attack can have multiple sources and finding the primary source poses a great challenge for research.

Hence, the hypothesis is that by mitigating Application Layer DDoS attacks with the help of Fuzzy Logic will make defense against DDoS attacks more feasible.

1.3 Contributions

This thesis provides the following research contributions:

1. A comprehensive literature review that introduces the cloud and its architecture, Denial of Service attacks and its impact in cloud.

2. A taxonomy of the recent surviving DDoS defense mechanism for network and application layer.

3. Proposal of a novel cloud-based Application Layer DDoS mitigation system which incorporates effective algorithms that help in distinguishing between legitimate and non-legitimate HTTP POST/GET requests.

4. A cloud based application that will detect and mitigate HTTP based DDoS attacks pro-actively where Fuzzy Logic will act as a backbone.

5. The implementation of the proposed mitigation system that describes the workflow and the development involved in creating this framework.

6. The evaluation of the proposed system based on Open Web Application Security Project (OWASP)[1] testing framework and a set of metrics.

1.4 Outline

Chapter 2 introduces to cloud architecture and its challenges and provides an analysis of the work related to Distributed Denial of Service attacks and its impact in cloud. A taxonomy of current surviving DDoS mitigation system is also presented and analyzed. Chapter 3 provides the specification and outlines the design of the proposed system. Chapter 4 discusses the implementation of the proposed system. Chapter 5 presents the evaluation of the mitigation system based on testing frameworks and a set of metrics. Chapter 6 provides a corollary of inconsistencies, gaps and challenges and finally the conclusion of thesis

[1]https://www.owasp.org/index.php/Main_Page

Chapter 2

Background Research

2.1 Preface

Distributed Denial of Service attacks have become very common in the area of cloud. The availability of tools to launch such attacks attracts amateur hackers who have limited knowledge about them. Fortunately, there has been a lot of advancements in DDoS mitigation and protection which are capable to withstand attacks even by professional attackers.

The Literature Review will be classified under different headings and the corollary of inconsistencies, gaps and challenges followed by a hypothesis, expected contributions and bibliography in the end.

Section 2.2 introduces to the cloud and its architecture. Section 2.3 talks about application arcitecture that includes layers of the OSI model that are relevant for the thesis. Section 2.4 gives an overview of different DDoS attacks and how they affect cloud services. Section 2.5 classifies the mechanisms based on different parameters. Section 2.6 gives an outline of existing DDoS protection for Application Layer. and the fact that Traceback is equally significant. Section 2.7 gives an outline of surviving DDoS protection for Network and Transport layers. Section 2.8 discusses unsupervised and supervised machine learning and how fuzzy logic is helpful in determining the nature of HTTP requests and mitigation of a DDoS attack. The chapter concludes in Section 2.9.

2.2 Cloud Computing Architecture

Cloud Computing is emerging as a new form of computing which is nothing but accessing and synchronizing data over the internet. Armbrust et al. [2010] mention that cloud computing can be seen as an assembly of application, hardware and operating system provided as a service to the customers. Such services as referred as IaaS (Infrastructure as a Service), Paas (platform as a Service) and SaaS(Software as a Service).

The Figure 2.1 depicts the cloud services offered to the customers. Starting off from the hardware that includes the mainframe and the infrastructure of the data center, each layer from the bottom represents a cloud service model. Infrastructure as a service is provided in the bottom layer that has a pool of networking, storage and computational services that is partitioned using virtualization. Platform as a service is the middle layer that includes programming environments and application frameworks such as Azure, Google App Engine and Amazon EC2. Finally, the top layer, software as a service includes software applications such as google docs that are provided to the clients as a service.

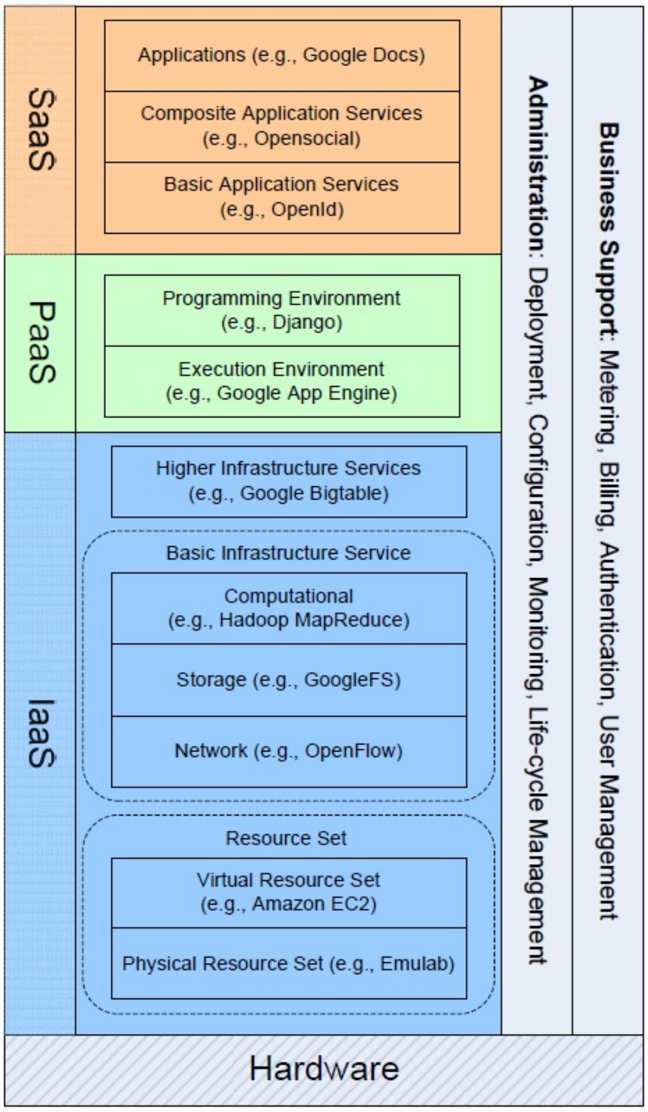

SaaS
Applications (e.g., Google Docs)
Composite Application Services (e.g., Opensocial)
Basic Application Services (e.g., OpenId)
PaaS
Programming Environment (e.g., Django)
Execution Environment (e.g., Google App Engine)
IaaS
Higher Infrastructure Services (e.g., Google Bigtable)
Basic Infrastructure Service
Computational (e.g., Hadoop MapReduce)
Storage (e.g., GoogleFS)
Network (e.g., OpenFlow)
Resource Set
Virtual Resource Set (e.g., Amazon EC2)
Physical Resource Set (e.g., Emulab)
Hardware
Administration: Deployment, Configuration, Monitoring, Life-cycle Management
Business Support: Metering, Billing, Authentication, User Management

Figure 2.1: Cloud Stack (Lenk et al. [2009])

When the deployment of cloud is considered, there are four cloud deployment models namely private, public, community and hybrid. According to Mell and Grance [2011], *private cloud* refers to the cloud infrastructure owned and managed by a single organization regardless of the location. The *community cloud* is exclusively intended for a group that have mutual interests and can be managed/owned

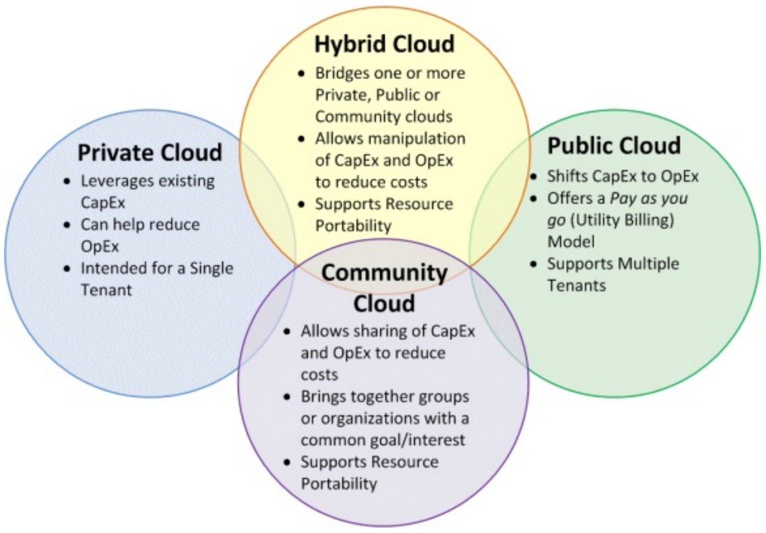

Hybrid Cloud
Bridges one or more Private, Public or Community clouds
Allows manipulation of CapEx and OpEx to reduce costs
Supports Resource Portability
Private Cloud
Leverages existing CapEx
Can help reduce OpEx
Intended for a Single Tenant
Public Cloud
Shifts CapEx to OpEx
Offers a Pay as you go (Utility Billing) Model
Supports Multiple Tenants
Community Cloud
Allows sharing of CapEx and OpEx to reduce costs
Brings together groups or organizations with a common goal/interest
Supports Resource Portability

Figure 2.2: Cloud deployment models (Jess [2012])

by one or more individuals of that group. *public cloud* as the name suggests is intended for general public, the cloud is owned and managed by the cloud service provider.Dillon et al. [2010] says *hybrid cloud* is a combination of one or more public, community or private clouds that are bound by standardized technology and regulations.

Takabi et al. [2010] entails some of the essential features that include ubiquitous computing, on-demand self service, rapid elasticity, location independent resource pooling and measured service. The author also highlights multi-tenancy aspect of public cloud that involves sharing of services between different customers

Although, cloud computing is becoming a major success, there are significant obstacles and challenges that affects the growth of cloud computing that have been discussed by Armbrust et al. [2010] and Zhang et al. [2010]. Some of the challenges include data security and confidentiality, standardization, software licensing, interoperability, service availability and reliability. Hence, Cloud computing is not mature enough to be considered as a competent computing model yet. There are still some areas where researchers can contribute to strengthen the core features provided by cloud and help in delivering them to the customers without any limitation of standardization and interoperability.

2.3 Cloud Application Architecture

It is important to understand the web application architecture before analyzing DDoS attacks. Ollman [2014] illustrate that applications have different architectures in order to meet business requirements that affect the strategies that are deployed to detect DDoS attacks in a cloud environment.

The OSI (Open Systems Interconnection) model, originally defined by ISO [1989] (International Standards Organization) depicts seven layers which are vital for computer networking and the cloud. The figure 2.3 below represents the detailed cloud model of OSI model presented by Ollman [2014].

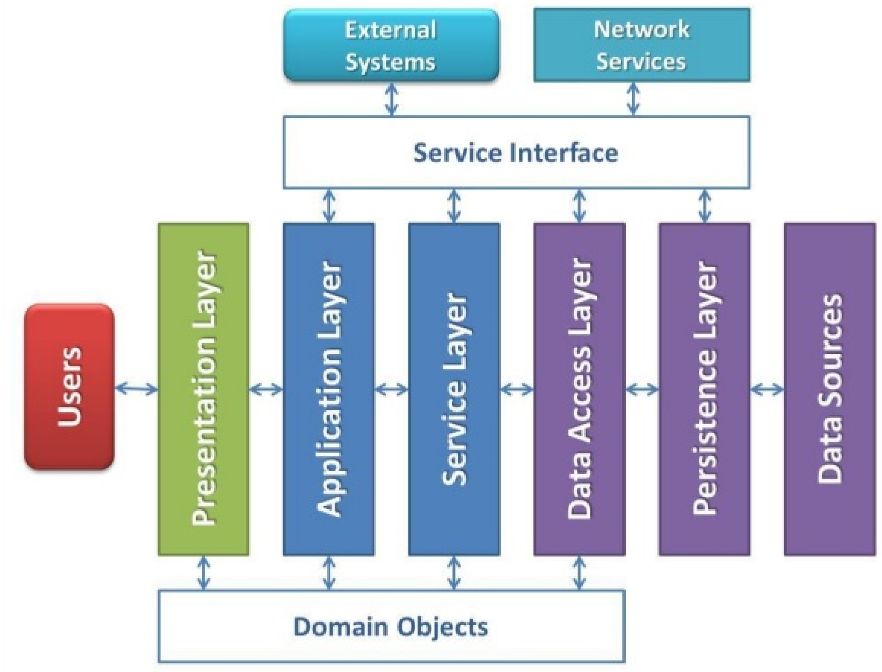


Figure 2.3: OSI seven layer representation (Ollman [2014])

This thesis is only concerned about the Network Layer, Transport Layer and the Application Layer. Network Layer is responsible for the end-to-end logical connections over a wide network. According to Li et al. [2011], the primary function of network layer is to complete packet transmissions between two hosts with the help of data link layer.In addition, Transport Layer is one of the most important layers in the OSI model which is responsible for determining what services to provide for the end-users. This layer is very vital in a network as it involves data communication and multiplexing between different hosts. Whereas, Li et al. [2011] have described the application layer as the highest level for user-experience and software applications accessed through a graphical user interface. All the software and applications are deployed and installed in that layer. All the dependencies and configurations aid the application layer in providing the services required by the user.

2.4 DDoS attacks and their impact in cloud

This section provides an overview about DDoS attacks in general and how they pose a threat to cloud based services.

2.4.1 Distributed Denial of Service (DDoS)

Anitha and Malliga [2013] and Darwish, Ouda, and Capretz [2013] ascertain that DDoS is an attack made on a victim by multiple systems by flooding messages to an extent that the victims server cannot process any more requests and eventually it collapses. In addition, Darwish, Ouda, and Capretz [2013] also say DDoS attacks can also leech resources such as memory, bandwidth and processing power.

Darwish, Ouda, and Capretz [2013] establish the fact that DDoS attacks can affect any layer of the cloud computing architecture i.e. SaaS, PaaS and IaaS that are described in 2.2. Also the attacks can happen either internally or externally. SaaS layer is affected when the attack happens internally i.e. within the cloud, whereas PaaS and IaaS layers are affected when attacks happen outside the scope of cloud. Hence, DDoS is a prominent attack and defending against such attacks is a challenge for cloud based services.

Vissers et al. [2014] and Fan, Hassanein, and Martin [2003] outline that when a Denial of Service attack occurs, the resources are not available for the users as the system is compromised by the attacker resulting in a denial of service, whereas in the Distributed case of DoS, the attacker takes control over computers which Vissers et al. [2014] refer to as zombies. They have the potential to destroy an organization by ruining the services, consuming the resources and making them unavailable to the users. In addition, Fan, Hassanein, and Martin [2003] also say that it is difficult for the router to distinguish between legitimate and non-legitimate packets.

Chonka et al. [2011], Vissers et al. [2014] and Anitha and Malliga [2013] discuss the emerging application layer attacks on cloud based web services i.e. X-DoS (XML based Denial of Service) and H-DoS (HTTP based Denial of Service) attacks collectively known as HX-DoS attacks. These attacks consume web resources by sending malicious packets to the server. Vissers et al. [2014] say that these malicious packets are considered as legitimate packets at the TCP/IP layer; so they suggest that a defense mechanism should be designed at the application layer rather than the network layer.

Vissers et al. [2014] and Chonka et al. [2011] share the same thoughts on X-DoS and H-DoS attacks. X-DoS attacks are targeted on web servers that are based on XML web-pages. One of the examples is the coercive parsing attack which uses SOAP (used to communicate between web servers via XML) to create an overhead on web servers.Whereas, H-DoS attacks are very basic, they flood the web server with HTTP requests. Since such attacks are easy to implement, denial of service achieved effortlessly as these requests consume minimum amount of resources.

According to Imperva [2012] the DDoS attacks can be classified as per the layers of OSI model i.e. Layer 3 that comprise of IP attacks affecting network bandwidth. Layer 4 consist of TCP/UDP attacks on servers and Layer 7 are affected by HTTP/XML based web application attacks. Network and Application Layer attacks are indifferent when considering the initiation and propagation. The differences have been listed and explained in the Table 2.1.

2.4.2 Impact of DDoS attacks on Cloud

As far as the impact of DDoS attacks are concerned, the destination can range from a small private server to huge enterprises like Amazon, Google, and IBM.

There have been several attacks but a noticeable attack was on Bitbucket.com which runs on Amazons Elastic Compute Cloud (EC2); it was collapsed by a DDoS attack in 2009. The problem was that the storage system of that website was situated on a network channel outside the scope of cloud. (Metz [2009])

Perhaps these frequent attacks would require some tools or applications wherein Chonka et al. [2011] mention that the most popular DDoS tools such as Agobot, Mstream and Trinoo are comparatively more complex than the tools used for XML and HTTP based

DDoS Attacks	
Network Layer Attacks	Application Layer Attacks
A typical attack occurs at the TCP layer where three-way handshake of SYN-SYN ACK-ACK	AL-DDoS attacks are of a tricky sort. The attacker mimics the behavior of a genuine user by completing the three-way handshake.
Attacks ignore the SYN ACK sent by the server and keep flooding with SYN packets	Later on, the attacker floods the server with HTTP POST/GET requests which ultimately leads to denial of service.
Such attacks can be mitigated by traditional defense mechanisms	More difficult to mitigate as the server can be penetrated with HTTP requests after the three-way handshake.

Table 2.1: Difference between Layer 3/4 and Layer 7 attacks

DDoS attacks, hence attackers use the application layer attack tools more as the former are very simple to implement and there are no practical defenses against such tools.

Hence it is evident that defending against such attacks is a vital challenge for all enterprises and mitigating such attacks will help secure a companys sensitive data. Such mitigation techniques will be discussed in the next section below

2.5 DDoS mitigation techniques

Generally, mitigation systems deal with defending, identifying and sometimes tracking the source of the attack from where it originated. There are quite a lot of mitigation techniques for DDoS attacks but they are classified based on the layers of the OSI model they attack - Layer 3/4 attacks and Layer 7 attacks

2.5.1 Classification based on Layers

Layer 3/4 defense mechanisms involve tackling DDoS attacks at network/transport layers. Since, the attack tools for Layer 3/4 attacks are easy to set up, the DDoS attacks are very common in general. However, there are various defense mechanisms to tackle such attacks easily. Hence, there has been a major switch of the attacks being focused on Layer 7 rather than the former.

Layer 7 is the Application Layer which has been gaining popularity among DDoS attacks. Attackers have resorted to the Application Layer which is the most vulnerable in a fully developed stack. The attacker establishes a valid connection at the Network and Transport layers and then starts flooding HTTP requests at the Application Layer in Cloud. This makes mitigating DDoS attacks at Layer 7 a tedious job.

2.5.2 Classification based on the type of approach

Whereas, based on the type of approach, mitigation techniques can be classified into Reactive and Proactive approaches.

Reactive approaches are based on an accident or a failure that has occurred in the past. The parameters are based on the alerts that happened because of a failure, for example server node down, system alerts on application etc. There are only a few interesting reactive approaches for DDoS security. Kwon et al. [2012] put forward a forecasting architecture for DDoS attacks using Honeynet which is a reactive approach for assigning countermeasures and strengthening the security of devices after the attack has occurred.

Generally, reactive approaches are not favorable for all the enterprises, they want to protect their sensitive data before it is compromised. Proactive approaches are described in the Section below.

Proactive approach is based on alerts that occur before a failure. It is usually based on a pre-defined set of rules that help in evading the collateral damage done to the machine to an extent. Component failure in a server is one of the examples. Although a proactive approach wont be able to mitigate attacks completely but it minimizes the risk inflicted upon the servers.

2.5.3 Classification based on the location

DDoS defense techniques can be categorized based on the location : Victim-based, Source-based and intermediate.

Victim-based defense systems are set up within the vicinity of the victim. Eventually, the victim is the most vulnerable and protecting from DDoS attacks when the defense system is installed at the victim is better as the attacks can be responded quickly. This is the most common location for laying out the defense systems.

Source-based defense systems are set up at the predicted origin of attacks. This way, the victim is not harmed and there are no resource overheads at the targetted servers.

Intermediate defense system are third-party provisions which are preferred when the organization is not certain about the origin of attacks. Such third-party infrastructures can be contacted for providing an adequate compensation for protection.

Table 2.2 shows all the DDoS defense techniques that have been considered for the research and they also list the type of attacks and the layers that are affected.

DDoS Mitigation Techniques			
Attacks/Vulnerabilities	Layer	Mitigation	Authors
Application and Network based DDoS attacks	Layer 7/4	DDoS attack forecasting system architecture using Honeynet	Kwon et al. [2012]
Policy Violations and Malicious activities	Layer 7	An architecture based on proactive model for security in cloud computing	Srivastava et al. [2011]
Application Layer DDoS attacks	Layer 7	Trust management framework for attenuation of application layer DDoS attack in cloud computing	Contractor and Patel [2012]
HTTP, XML based DDoS attacks	Layer 7	Defense System of Web Services in Cloud Environment	Vissers et al. [2014]
HTML, XML based DDoS attacks	Layer 7	ENDER(Pre decision, advance decision, learning system)	Chonka and Abawajy [2012]
Network and Transport Layer based DDoS attacks	Layer 3/4	Moving Target Defense Mechanism	Wang et al. [2014]
IP spoofing and other network attacks	Layer 3	FDPM (Flexible deterministic packet marking approach)	Xiang, Zhou, and Guo [2009]
Network Layer attacks	Layer 4	SBTA (SOA based Traceback approach)	Yang, Zhang, Song, Wang, and Chen [2012]
TCP, UDP based attacks	Layer 4	IDP (Intelligent Decision Prototype)	Chonka et al. [2008a]

Table 2.2: A taxonomy of surviving DDoS mitigation systems across layers.

2.6 Application Layer DDoS defense Mechanisms

This section talks about the defense techniques for DDoS attacks at the Application Layer. Before discussing the different techniques, let's consider the two main factors outlined by Surace [2013] that make an effective defense mechanism worse.

Firstly, the lack of knowledge about Application Layer attacks poses a threat to the existing defense systems. Secondly, while looking from a developer's point of view, developing applications in low budget as well as meeting deadlines make the application vulnerable to DDoS attacks at Layer 7.

Srivastava et al. [2011] propose a proactive model wherein a Cloud Policy has been constructed which acts as a security policy. Also a distinct cloud within a private cloud is assigned to proactively monitor the operations. Srivastava et al. [2011] claim that the system is not passive rather it actively monitors the ongoing processes. Quite similar to the suggested model by Srivastava et al. [2011], Contractor and Patel [2012] suggest that a trust management framework which uses trust to differentiate between legitimate and non-legitimate packets with an added license feature. There are no specific security measures taken beyond the authentication and filtering.

However, the Moving Target defense mechanism Against Internet DDoS attacks (MOTAG) architecture proposed by Wang et al. [2014] is really interesting. It involves proxy nodes (whose IP address is hidden) which not only provide security but the fact that they are moving which makes the possibility of inflicting damage even more problematic. Another key aspect of this method is that the proxy nodes are replaced when they are attacked. Although this architecture is capable of defending DDoS attacks it entails a great deal of overhead on the machines.

On the other hand, Vissers et al. [2014] implement a system which effectively mitigates attacks made through XML or HTTP requests, but anything other than that is not affected by the proposed filter. Nevertheless, it proves to be an efficient way of mitigating DDoS attacks while inducing minimal overheads. Chonka and Abawajy [2012] have

come up with ENDER (pre decision, advance decision, learning system) which is a cloud application based on the intelligent decision prototype which is proposed in previous research papers by the same authors. With the help of CLASSIE (A pre decision system based on IDP), RAD (Reconstruct and Drop) and ADMU (Added decision making and update); the ENDER is able to proactively detect and mitigate HX-DOS attacks on cloud. Basically, the ADMU decides the nature of the incoming packet, sends it to CLASSIE which marks the packet with a 1bit mark. These marked packets are sent to RAD where the marked messages are dropped while the others are reconstructed. Overall, this approach has minimal overheads and is an effective mitigation technique for DDoS attacks.

Some of the surviving Network and Transport Layer based DDoS Mitigation systems have been analyzed and discussed in the section below.

2.7 Network and Transport Layer DDoS defense Mechanisms

Network And Transport layer attacks are now considered as a primitive way to attack an organization, as there have been many developments and research in that field. Additionally, they are also easier to oppose compared to the Application level as discussed in Section 2.6 . This section accounts some of the techniques involved in mitigating against DDoS attacks in Layer 3 and 4. Some of the traceback algorithms have also been discussed in this section which are done by canonical methods of packet marking.

Traceback algorithms and techniques involve tracing the path to the source of an attack. This is generally done with the help of packet marking. Packet marking can be of two types- deterministic Packet Marking (DPM) and Probabilistic Packet Marking (PPM).

Earlier, Belenky and Ansari [2003] introduced the DPM (deterministic Packet Marking scheme) for IP Traceback. This technique involves marking the packets and storing additional information in the headers. The outbound packets are not marked as that will cause an overlap of marking. While reconstructing the path to the source, packets are then matched using the reconstruction table (RecTbl). This particular approach was set as a benchmark for further research.

Xiang, Zhou, and Guo [2009] propose a Flexible approach to DPM known as FDPM (Flexible deterministic packet marking approach). Based on the incoming load of the packets, the marking length is varied by using a flexible mark length strategy. Additionally, there is also an overload prevention mechanism i.e. if there is a high load of packets in the router then the tracing is reduced so that the router is not burdened.

This makes the approach equally efficient and stable, but the infrastructure is compromised when there is an activity of high load on the router. Based on DPM proposed by Belenky and Ansari [2003], Chonka et al. [2008b] have come up with the SOTA (service oriented Traceback architecture) framework which replaces the SOAP header with their own header so that the framework can track down the source of a DDoS attack.

Similarly, Yang, Zhang, Song, Wang, and Chen [2012] propose the SBTA (SOA based Traceback approach) which is quite similar to the approach proposed by Chonka, Zhou, and Xiang [2008b] but it is primarily in the cloud computing environment. Furthermore, Yang et al. [2012] designed a cloud filter which works alongside the SBTA. The cloud filter helps the victim to filter out the attack messages specifically to facilitate the reconstruction of the path to the source. On the whole, this framework is efficient and requires few messages to reconstruct the path.

Chonka et al. [2008a] propose a hybrid system known as IDP (Intelligent Decision Prototype) which is based on DPM, PPM (Probabilistic packet marking Scheme) and IP logging. It consists of two sections:

PMD (Pre-Marked Decision)decides if the incoming packet is legitimate or not with the help of DPM and PPM, if it is legitimate then it is forwarded for processing, and if it is malicious it is sent to for packet marking. The packet marker used here is known as Intelligent Decision Prototype Market (IDPM) which follows the DPM technique. RAD (Reconstruct and Drop)is a logging technique that is used to reconstruct the path to the source of the attack with the help of the marked packets forwarded by the previous section.

This Intelligent Decision Prototype (IDP) approach was modified into a cloud application called ENDER by the same authors. It is obvious that it is more efficient than DPM and FDPM proposed by Belenky and Ansari [2003] and Xiang et al. [2009] respectively as it does not mark each and every packet; hence reducing overheads and increasing efficiency. To trace the packets using Traceback algorithms, machine learning is a requirement and will be discussed in the next section below.

2.8 Machine Learning and Fuzzy Logic

This section focusses on the principles of machine learning and how they are utilized in tracing the origin of an attack. Machine Learning is helpful in analyzing datasets, predicting events and decision making process. There are various Machine Learning principles and have been discussed in the sections below.

2.8.1 Machine Learning and its application in DDoS Detection

Machine learning is a process of collecting useful data which can be used to enhance the understanding of an application or a program. Porter et al. [2013] have discussed about machine learning and state that:

> The goal of interactive machine learning is to help scientists and engineers exploit more specialized data from within their deployed environment in less time, with greater accuracy and fewer costs. (2013, p. 12)

Jin and Sendhoff [2008] have categorized machine learning in three categories. Supervised machine learning is a machine learning in which datasets are provided and the system or an application handles the input based on the data sets provided. In supervised learning, an error function is defined which judges the quality of packets. Unsupervised machine learning is a machine learning where no such datasets are given, instead the system or an application tries to find similar datasets and tries to filter the other anomalies. Data clustering is one such example where resemblance of datasets should be reduced. Reinforcement learning is designed such that the system or an application works towards gaining a cumulative reward in a given task. Here, the reward is predicted in a given environment.

Considering the application of Machine Learning, it adds a tier of protection from DDoS as it helps in analyzing historic data, creating pre-defined rules, decision making etc. There are a few methods that use machine learning for DDoS detection. Some of them have been discussed below.

Kwon et al. [2012] and Chonka and Abawajy [2012] implement the supervised machine learning in distinct ways. Kwon et al. [2012] uses machine learning in machine learning modeler in Intrusion forecasting module. The data providers give raw data (labelled data) which is analyzed by the modeler so that the attack can be predicted beforehand. The modeler consists of machine learning algorithms which helps in establishing a baseline of the architecture. Collectively, the machine learning has been utilized in an appropriate way.

Whereas, Chonka and Abawajy [2012] use supervised machine learning in ENDER. Based on predefined rules, machine learning algorithms implemented in the system decide whether the packet is genuine or malicious. The malicious packets are discarded and the legitimate packets are sent forward for processing.

Casas, Mazel, and Owezarski [2012] and Song et al. [2013] establish their detection systems based on unsupervised machine learning. Casas et al. [2012] develop an Unsupervised Network Intrusion Detection System (UNIDS) which is adept in detecting unknown attacks. Without the help of any pre-defined rules or labelled data, it is capable of detecting attacks (such as DDoS, probing attacks etc.) using sub-space clustering and evidence accumulation techniques.

However, Song et al. [2013] propose an unsupervised anomaly detection system for DDoS attacks which does not use labelled data. If the incoming data is normal, it will have more occurrences and the others will be very few. Unsupervised machine learning is used by grouping the data in two sections: sparse (normal data) and dense (others). The elements belonging to the dense group act as the training data for the system. The drawback of this approach is that parameters must be provided before building an intrusion detection system.

It is interesting to note that, Jin and Sendhoff [2008] discuss a new type of machine learning known as Pareto based multiobjective machine learning in which the objective is a vector instead of a scalar where there are a number of optimal solutions. The solutions here refer to the optimal number of ways a target can be achieved. This approach is suitable for DDoS attacks but it is fragile against any other attack. Although being an exceptional approach it is suitable for DDoS attacks but is unresponsive against any other attack.

In a nutshell, it is evident that either unsupervised or supervised machine learning is essential as it strengthens the decision making process and makes the system more intelligent.

2.8.2 Fuzzy Logic

Fuzzy Logic is a machine learning algorithm which deals with approximation reasoning of values rather than fixed reasoning. According to Novak et al. [1999] it introduces partial truth whose values can vary between completely true (binary 1) or completely false (binary 0).

> 'Fuzzy Logic is a precise logic of imprecision. More concretely, fuzzy logic is a system of reasoning and computation in which the objects of reasoning and computation are classes with unsharp boundaries.'Zadeh [2010]

In other words, Fuzzy Logic is a processing algorithm that is able to deal with varied and subjective data. It uses precise algorithms to process imprecise data, wherein linguistic notations are used primarily which makes it much easy to enforce and understand. Simplicity, better performance, easier and faster development are some of the benefits mentioned by Anderson [1994].

Some researchers have suggested that probability is enough to represent linguistic information as fuzzy logic introduces unwanted redundancy. According to Novak et al. [1999], Fuzzy Logic possess some advantages, however, it also exhibits considerable limitations. For instance, Fuzzy Logic can address the complex problems but surprisingly, this research area has been neglected until recently. Woolf and Wang [2000] reinforce the argument by expressing that if fuzzy sets are used, the analysis is made more complex rather than keeping it in general terms.

Nevertheless, Fuzzy Logic has many advantages over other methods for analyzing data and has been applied in various architectures for different purposes. In the context of DDoS detection, there have been many applications of fuzzy logic and one of them is in intrusion detection systems IDS. Goss et al. [2007] implement fuzzy logic for intrusion detection of attacks originating in wired and wireless networks. Here, the fuzzy engine is responsible for detecting the probability of attack from a particular user.On the other hand, Chapke and Deshmukh [2015] propose a signature based intrusion detection system that uses fuzzy rules. Here, Fuzzy Logic helps in analyzing the signatures that are added beforehand.

Hence, the stellar advantage of Fuzzy Logic is rapid prototyping. The fact that fuzzy logic can be used for systems that are in need of a precise result or a phenomenon that is imprecise predominates over other principles.Also, fuzzy logic enables a user to effortlessly interface with an automated system compared to other conventional methods that have a natural tendency towards uncertainty. Consequently, the research intends to design an algorithm based on Fuzzy Logic for DDoS mitigation.

2.9 Conclusions

Cloud-related technologies and the architecture of cloud was explained in the initial sections of this chapter. This chapter also provided an outline of Distributed Denial of Service attacks (DDoS), the classification, the severity and the various surviving mitigations designed to tackle such attacks. Section 2.4 provides an overview about DDoS attacks and their consequences in cloud.

A taxonomy of the surviving Application and Transport Layer DDoS defense mechanisms were presented. Prior modeling the proposed solution, the research compares and contrasts other surviving strategies (see Table 2.2) for mitigating DDoS attacks.

Moreover, mitigation of DDoS attacks in cloud is a domain that is ripe with uncertainty. Hence, Fuzzy Logic is suggested as the preferred system over existing ways of mitigating such attacks. The next chapter 3 specifies the features of the proposed framework for mitigating DDoS attacks.

Chapter 3

Design

3.1 Preface

This chapter discusses the design features of the proposed mitigation system and provides the necessary specification . The proposed mitigation framework is based on fuzzy logic and application layer mitigation algorithms. The workflow and the architecture of the system will be discussed in the subsequent sections.

The primary focus of this research is on the Application Layer based DDoS attacks. As discussed in Chapter 2, network-level protection is useless against such attacks. They are getting very prominent in cloud and mitigating them is a serious concern for cloud-based enterprises.

The main body of this chapter consists of three sections. Section 3.2 gives an outline of the specification of the proposed method and how it is going to be achieved. Application Layer attack generating tools have been discussed in Section 3.2.1 .VMware workstation and its purpose has been outlined in Section 3.2.2. Fuzzy Logic based Mitigation approach has been explained in detail in the Section 3.3. Finally, the summary and conclusions have been outlined in Section 3.4.

3.2 Specification

The proposed system is to apply Fuzzy Logic algorithm in a proactive mitigation system for defending against Application-Layer DDoS attacks in cloud. This section outlines the system requirements which underlies the architecture of the mitigation framework.

A high level overview and The attack generating tools have been discussed in this section.

As previously discussed in Section 2.4.1, such attacks are targeted after an anonymous user has gained a genuine connection with the victim server, hence the server is not able to determine if the requests coming from the anonymous user are legitimate or not. That particular user keeps sending HTTP GET request, and the web server is busy responding to those requests and ignores the legitimate requests sent by genuine users. Hence denial of service attained easily and detecting such threats becomes very difficult. Figure 3.1 shows a high level overview of the proposed system hosted on an application in cloud. The mitigation system blocks the non-legitimate HTTP requests and only allows the genuine HTTP requests to be processed by the application.

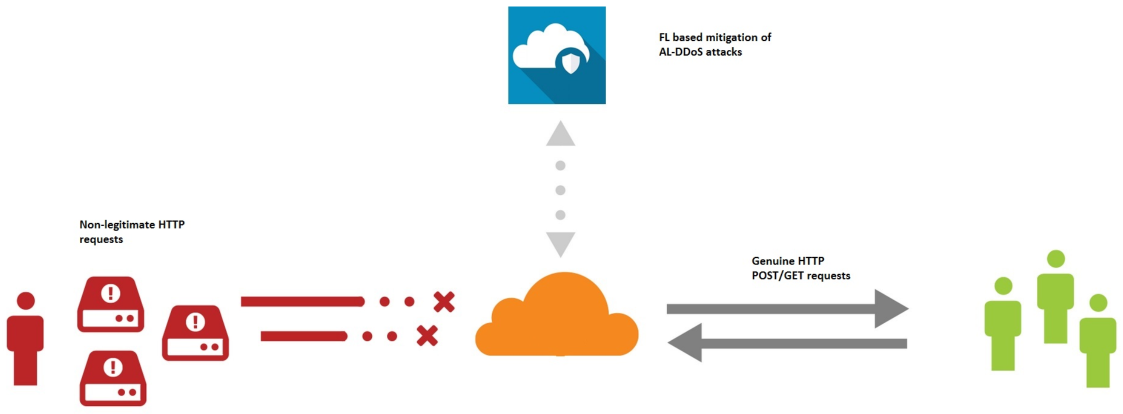


Figure 3.1: High Level Overview

DDoS attacks can be tested by the tools described in section 3.3. The tools will help in testing the proposed method in a testing environment. The non-legitimate HTTP POST/GET requests are targetted onto the system, where the response time of the system is analyzed. R U Dead Yet(RUDY) ,LOIC(Low Orbit Ion Cannon) HULK (HTTP unbearable Load King) and Slowloris are some of the application layer DDoS attack generating tools.

3.2.1 Application Layer DDoS attack generating tools

There are several DDoS attack generating tools out there, but the primary focus of this thesis is on the tools that generate Application-Layer DDoS attacks against a victim web server. These tools have unique features and provide a varied play-style for new hackers.

Tracking the attacks generated from these tools is not easy compared to network layer attack generating tools.

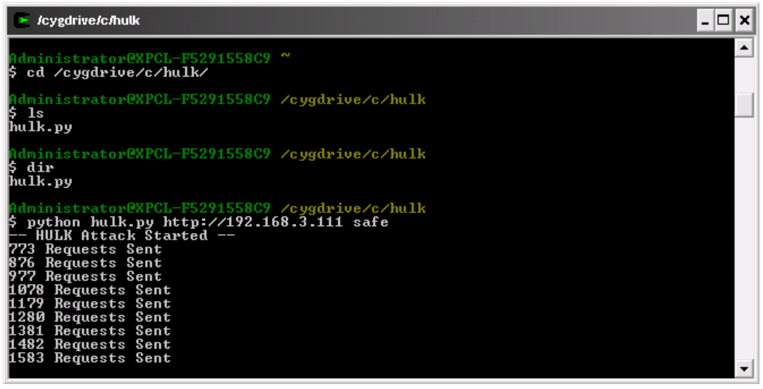
3.2.1.1 RUDY (R-U-Dead-Yet)

RUDY is one of the most popular open source Layer-7 DDoS attack generation tools developed by Raz [2011] in python. RUDY is categorized as a low and slow attack tool as the requests sent by RUDY are slow and are sent in minimal volume. When a genuine user fills a web form, an HTTP POST request is forwarded to the web server. The web server closes the connection after receiving that specific request.

Whereas when an attacker using the RUDY tool sends a HTTP POST request to the server , it goes into denial as the RUDY sends thousands of requests of one custom byte in HTTP web form fields at random time intervals. The application threads start acting as zombies awaiting end of requests. The important feature of RUDY is that it is able to impersonate users having poor Internet connection.

3.2.1.2 HULK (HTTP unbearable Load King)

HULK developed by Shteiman [2012] stands apart from the traditional DDoS attack generation tools. What makes it special is that, each and every time an attack is generated from HULK tool, the attack has a unique pattern every-time it hits the victim server. Whereas, unlike other tools the pattern of attacks generated by them is repetitive and predictable.



```
Administrator@XPCL-F5291558C9 ~
$ cd /cygdrive/c/hulk/

Administrator@XPCL-F5291558C9 /cygdrive/c/hulk
$ ls
hulk.py

Administrator@XPCL-F5291558C9 /cygdrive/c/hulk
$ dir
hulk.py

Administrator@XPCL-F5291558C9 /cygdrive/c/hulk
$ python hulk.py http://192.168.3.111 safe
-- HULK Attack Started --
773 Requests Sent
876 Requests Sent
977 Requests Sent
1078 Requests Sent
1179 Requests Sent
1280 Requests Sent
1381 Requests Sent
1482 Requests Sent
1583 Requests Sent
```

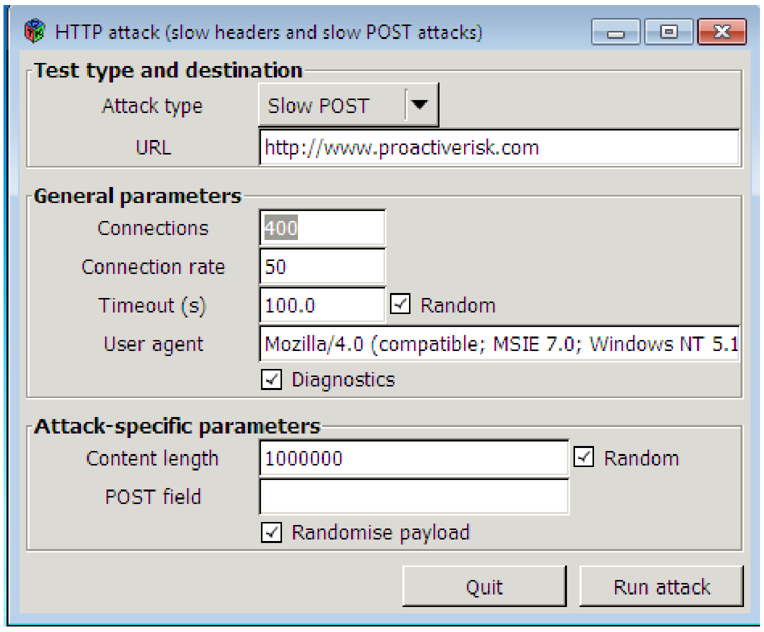
Figure 3.2: HULK - HTTP Unbearable Load King

HULK has distinct user agents which are used when an attack is generated which results in unique pattern whenever a request is created. HULK is also written in native python script.

The following sub-section will talk about other tools for Layer-7 DDoS attack generation. Since they can be countered effectively, in-depth analysis will not be carried out.

3.2.1.3 Other attack generating tools

A Layer-7 HTTP POST/GET attack tool developed by OWASP (Open Web Application Security Project) which is used for loading a target server with HTTP requests on a web application running on Apache or Microsoft based IIS (Internet Information Services) server. It has a user-friendly GUI for sending out all types of slow HTTP GET/POST attacks in a couple of clicks.


HTTP attack (slow headers and slow POST attacks)

Test type and destination

Attack type	Slow POST
URL	http://www.proactiverisk.com

General parameters

Connections	400
Connection rate	50
Timeout (s)	100.0 ☑ Random
User agent	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1
	☑ Diagnostics

Attack-specific parameters

Content length	1000000 ☑ Random
POST field	
	☑ Randomise payload

Figure 3.3: OWASP HTTP POST Tool

LOIC (Low-orbit Ion cannon) and SlowLoris are equally good tools for DDoS attack generation but can be easily countered by following some prerequisites. When an attacker uses LOIC for sending Layer-7 HTTP POST requests to a victim server in cloud, the original IP is revealed, hence the attackers are forced to anonymize the IP address before targetting a server. Whereas SlowLoris can be countered by using load balancers and switching to a Microsoft based IIS server as they are not prone to attacks rendered by

SlowLoris.

3.2.2 VMware Workstation

VMware Workstation 11.0 is used as a hypervisor (VMM) for using Ubuntu 14.0.4 LTS as a guest OS on Windows 10 Pro. Sahoo et al. [2010] states that it acts as an abstraction layer for the operating system, hardware and applications. Also, the physical resources are hidden from the guest OS.

VMware network adapters VMnet1 and VMnet8 are used for establishing bridge connection between the Host OS and the Guest OS. They are helpful in maintaining a common NAT connection between the operating systems. These adapters are scanned for the analysis of network packets that are traveling between the two operating systems.

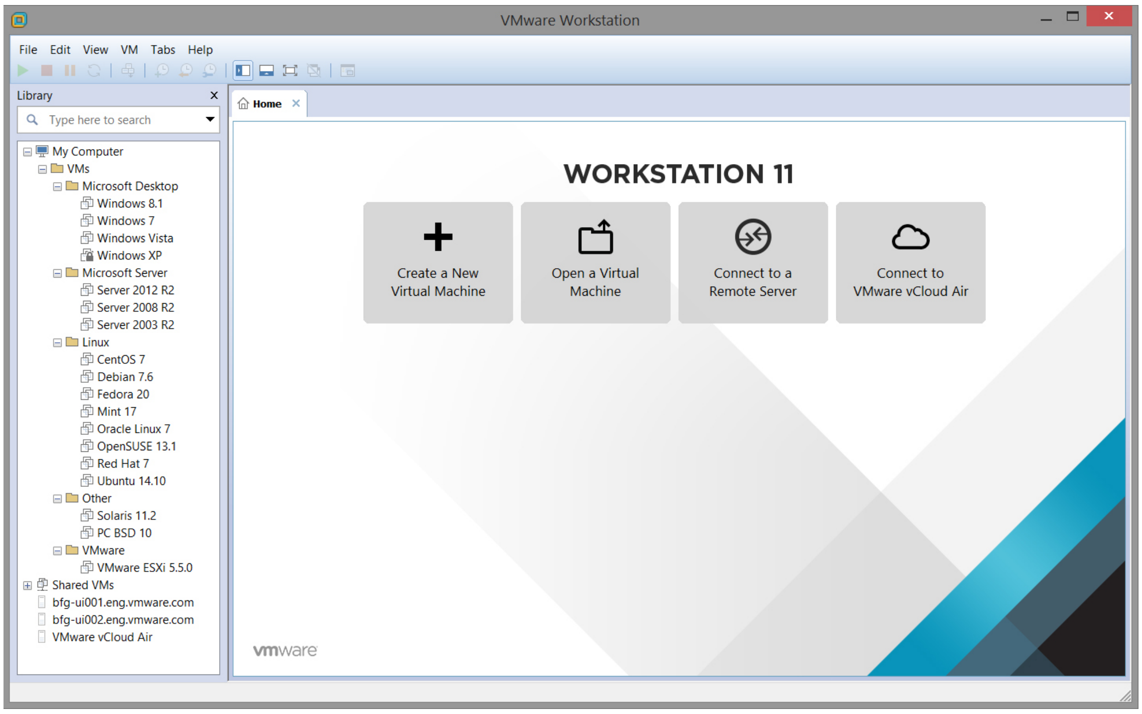


Figure 3.4: VMware Workstation 11

3.3 Fuzzy Logic based Proactive Mitigation

The proposed system as depicted in Figure 3.1 involves fuzzification of the incoming HTTP GET/POST requests. Fuzzification is the process of converting the crisp values into approximated scales or fuzzy sets. The proposed framework is shown for a legitimate user and a non-legitimate user where the difference in results is highlighted. A high level overview of the mitigation system is outlined below followed by a complete description which will be explained in the Section 3.3.2 and Section 3.3.3.

Any IP that attempts to access the cloud application, is assigned a buffer and a separate mitigation process so that the availability of service for other users is not affected. As a result, the non-legitimate users are blocked from aceesing the application.

There are two tiers of defense in the whole mitigation process. The first tier is called Fuzzy Elimination and the second tier is Application Layer DDoS mitigation. Fuzzy Elimination uses fuzzy logic and authorization techniques to filter the incoming requests. Whereas, the second tier is responsible for mitigating the remaining rrequests forwarded by a flagged IP.

The two layer model is preferred over a singular interface as it ensures the reduction of anonymous invalid HTTP requests at the application layer in cloud. Even if some of the invalid requests pass throught the first tier, the second tier will make sure the requests are out of bounds for the hosted application.

3.3.1 Workflow of the mitigation process

Consider a legitimate and a non-legitimate user that are trying to connect to a cloud application that has the proposed mitigation installed. For better understanding of the workflow refer to Figure 3.1 and follow the operation of Application Layer DDoS Mitigation described below :

(i) initially, the attacker sends a flood of HTTP requests against the server. As the attacker is sending the requests, Fuzzy Elimination becomes active and starts to filter the requests based on the request patterns from pre-defined rules and training data.

(ii) If it was a legitimate user sending HTTP requests, the user is allowed by an authorization process. Otherwise, most of the requests are not allowed.

(iii) After the elimination process is complete, the requests are passed through a mitigation process for application layer.

(iv) If the buffer exceeds the `max requests` then the IP that generated the HTTP flood is saved in temporary cache.

(v) The temporary cache is then used to block that specific IP for an assigned `block duration`. All the IP addresses saved in cache cannot access the application unless they re cleared in back-end.

(vi) On the other hand, if the `max requests` do not exceed buffer, the server receives a steady flow of HTTP requests.

3.3.2 Fuzzy Elimination

Fuzzy Elimination tier is responsible for filtering to detect application layer DDoS attacks by validating the requests and authenticating an anonymous user trying to access the application hosted on the cloud. The algorithm compares the incoming requests with the profiled data as well as the rules set in the mitigation system. The Figure 3.1 shows the components of fuzzy elimination.

Algorithm 1 Fuzzy Elimination

```
Require: HTTP requests forwarded to the application
1: if request.len >rl && request.freq >rf then
2:     if IsAuthorized(User) then
3:         call(application)
4:     else
5:         call(fuzzye)
       end
6: end
```

Algorithm 1 denotes the algorithm for fuzzy elimination. Initially, the incoming requests from an IP are sent to this tier. Based on some pre-defined rules (`request.len and request.freq`) and a collection of HTTP requests data, the elimination of the requests are carried out using Fuzzy Logic and authentication. The rules as well as the training data of normal and attack HTTP requests are retrieved beforehand.

`request.len and request.freq` validate the incoming HTTP requests. If the HTTP requests do not match the rules, then most of the requests are discarded. But if it does match the criteria, then all the requests are sent for authentication.

Almost all of the invalid and malicious are filtered out by authentication. A legitimate user will know the username and password for the cloud application. Once the user is validated all the HTTP requests from that IP are forwarded to the next tier.

After this tier, all the IP addresses are assigned an individual buffer (acts as a maximum threshold for incoming HTTP requests) and are then forwarded to the core interface Application Layer DDoS Mitigation.

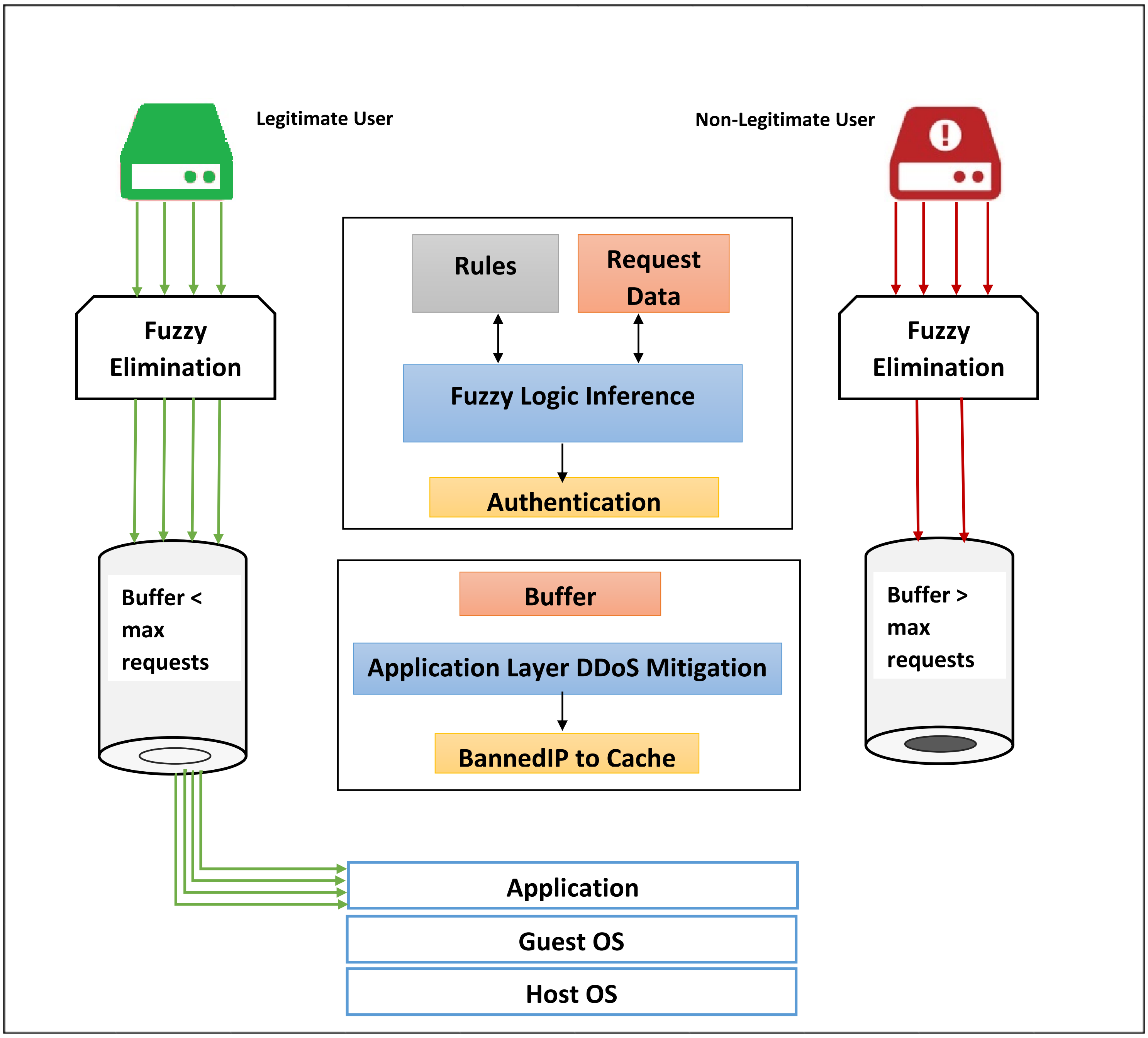


Table 3.1: Proposed Application Layer DDoS Mitigation System in Cloud

3.3.3 Application Layer DDoS Mitigation

This is the tier of the whole mitigation process. It is responsible to check if a user exceeds the assigned amount of HTTP requests generated in a period of time and blocks the user for a duration if the former fails. This process has been described in detail below.

Initially, as the HTTP requests are sent from fuzzy elimination, they are enclosed in buffer which has a threshold limit that is assigned beforehand. Until the incoming requests do not exceed the `max requests`, the user in the safe zone and can aceess the application without any interruption.

Whereas, when the overflow of HTTP requests is detected, the algorithm writes a temporary file in the cache which includes the banned IP address and the block duration. The attacker connected with this IP cannot access the cloud application unless the file is removed from cache or the block duration expires. Finally, the HTTP requests from a legitimate user are then forwarded to the Application for further processing.

Algorithm 2 Mitigation algorithm

```
 1: if cache.read ≠ nil then
 2:     baz ← cache.read.to integer
 3:     cache.write(baz+1).to string
 4: else
 5:     cache.write(block duration)
 6: if cache.read.to integer > max requests then
 7:     banned IP ← cache.read
 8:     banned IP = banned IP + Remote Address
 9:     cache.write(banned IP)
10: else
11:     cache.write(Remote Address)
12: return Response
```

Algorithm 2 denotes the flow of read/write operation of the temporary cache created within the application's domain. In this context, when a cache file is absent, it creates a new file including the block duration. Now, if the IP under inspection, exceeds the assigned `max requests`, then the IP is flagged as a `banned IP`. Otherwise, the IP is not flagged and is allowed to access the application until it exceeds `max requests`. Finally, a *Response* is displayed if the IP is blocked. Here, *Response* is a custom page rendered in the attacker's window which is displayed if the IP is blocked
Algorithm 3 illustrates the purge operation. At line 1, a statement is used which describes a loop. The `custom_IP` is removed as long as the condition remains true. Immediately

Algorithm 3 Purge Algorithm

```
1: while custom IP $\neq$ nil do
2:     cache.delete(custom IP)
   end
3: cache.read.each do
4: cache.delete(banned IP)
5: end
```

after the condition is satisfied, the control from the loop is transfered to the remaining instructions where all the banned ips are removed if they are present in the cache.

3.4 Conclusions

This chapter has provided the detailed design and specification of the mitigation framework designed for application layer DDoS attacks. A high level overview was outlined and a proposed mitigation framework (see figure 3.1) was described using algorithms.

The proposed framework is able to avoid bogus and anomalous requests by using fuzzy logic to differentiate between legitimate and non-legitimate requests. Moreover, the anonymous requests are once again mitigated in the second tier and the IP under inspection is banned until further examination

The attack tools mentioned in this chapter have been used for the implementation and testing of the network under inspection. To sum up, RUDY and HULK prove to be better tools for attack testing rather than Low Orbit Ion Cannon (LOIC), SlowLoris or any other traditional Layer-3 DDoS tools as they have practical defenses and can be countered effectively.

The design that was described within this chapter is utilized to construct a reference implementation of the framework in Chapter 4. The corresponding results and testing procedures have been described in Chapter 5.

Chapter 4

Implementation

4.1 Preface

Chapter 3 explained about the algorithms and methods involved in creating the logic behind the application layer DDoS mitigation system. The findings of the previous chapter will be used to generate a framework for defending against such attacks.

This chapter provides the working of the whole mitigation process of application-layer DDoS attacks. This section also characterizes implementation of the tools and resources that have been used for achieving the research goal. It is also necessary to know about the configurations and dependencies that have to be installed or modified. The structure of this chapter is divided into sections that will cover the development of the proposed DDoS Mitigation System against attack tools RUDY and HULK.

The primary programming language used for the implementation of the framework is Ruby on Rails. Minor python is also used for generating HTTP requests for testing purposes. Only the utility of attack tools described in Section 3.2.1 is discussed here. The configuration of the attack tools and the related dependencies have been specified in Appendix 6.1.

4.2 Fuzzy Elimination

Fuzzy Elimination is the first tier of the mitigation process. This tier specializes in blocking malicious HTTP requests based on some pre-defined parameters and rules.Based on the nature of the incoming HTTP requests, the algorithm will decide either to filter or to authorize them to the next tier of the mitigation process.

The `request.length` and `request.frequency` is checked before authorizing an IP for using the cloud application. An anonymous IP is blocked when the length and frequency of the requests do not match the pre-defined parameter range. On the other hand, if the HTTP request parameters are validated, the legitimate user is forwarded to the authentication screen, where the user has to enter credentials to access the application.

The incoming HTTP requests are significantly reduced with the help of rules and data collected prematurely, if they are found to be invalid. In case, there are requests that penetrate the first tier, they will be caught by the buffer. When the buffer overflows (exceeds maximum requests) , then the second tier will mitigate the penetrated HTTP DDoS requests successfully. Listing 4.1 represents the Fuzzy Elimination and Authorization Middleware that uses the fuzzy logic libraries[1]

Listing 4.1: Fuzzy Elimination

```
require 'net/http'
require 'fuzzy-logic'
class FuzzyAuthorizeEnvironment

    def initialize(app)
    @app = app
      @size = 0
      @frequency = 0
  end

  def fuzzye
  incoming_request = FuzzyLogic::Set.new(1) { |request|
    # default output is zero
    o = 0.0
    if request.frequency == 12 and request.size <= 24 then
      # set is completly true
```

[1]https://fuzzy.io/docs

```
        o = 1.0 if request.size >= 20
        # set is fuzzy when its between
        o = 1.0 - (24.0 - request.size)/(24.0 - 6.0) if request.size >= 6
            and request.size < 20.0
      end
      # just the correct return
      fuzzye.o          }
end

def results
incoming_request.get( request.now )
incoming_request.support( request.new(12, 8) ) # => true
incoming_request.support( request.new(12, 6) ) # => false
incoming_request.core( request.new(12, 23) )   # => true
incoming_request.core( request.new(12, 8) )    # => false
end
```

4.3 Application Layer DDoS Mitigation

The application layer DDoS mitigation is implemented in rails framework as a middleware (depicted in Figure 4.1) which acts as a standardized intermediary layer between requests coming from client-side and the back-end of the application on which it is protecting. It is the second tier of the mitigation process wherein the request flow is blocked when the buffer is exceeded.

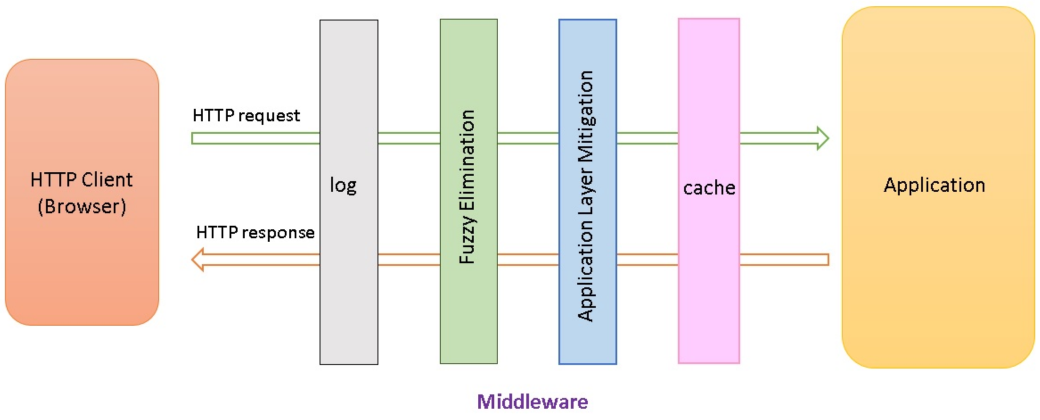


Figure 4.1: Overview of Rack Middleware

The remaining HTTP requests after fuzzy elimination are passed on to this tier for further analysis of DDoS attacks. (Refer to Section 3.3.1). The configuration and the development involved in proper functioning of this tier will be explained in the sections below.

4.3.1 Configuration

For using the mitigation Framework on an application hosted in the cloud, configurations have to be made in the application before launching. Minor changes to the application middleware and updating options within the framework are made.

Listing 4.2: Configuration of mitigation middleware

```ruby
module Muse
  class Application < Rails::Application

    # The required middleware is added below. Make sure you add the
        middleware in doublequotes

    config.middleware.use "ApplicationLayer"
    config.middleware.use "FuzzyAuthorizeEnvironment"
  end
end
```

This will enable Application Layer Mitigation framework to work on all interfaces of the application when the server is initialized. Based on a few hard-coded parameters, the algorithm will work as intended.

4.3.2 Development

When, the fuzzy elimination ends, the requests are then limited by a threshold which is defined by this tier. This tier specializes in blocking any remote adress that voids the defined rules. The algorithm helps in scanning for non-legitimate users flooding the server with HTTP requests. The IP that floods the server more than the pre-defined `max requests`, gets blocked temporarily for the defined `block duration`. This section explains how the mitigation has been carried out.

Figure 4.2 represents the UML class diagram of the mitigation framework and the others from which it is composed. The Application layer involves class variables (discussed

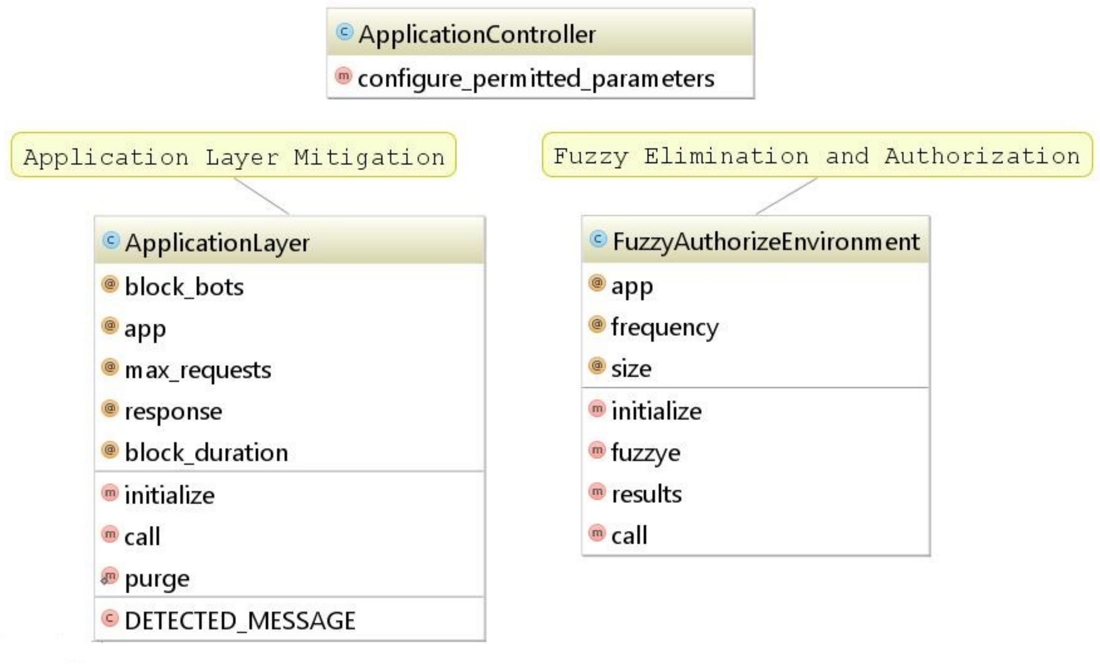


Figure 4.2: UML Class Diagram of the framework

in Section 4.3.2.1) and methods initialize, call and purge have been discussed in the subsequent sections. The class variables ans methods of fuzzy elimination have been discussed in Section 4.2.

Section 3.3 described the respective algorithms (Application Layer Mitigation and Fuzzy Elimination) in this framework and Fuzzy Elimination has been explained in Section 4.2. The code development has been classified into 3 sub sections : Invoking the Environment, Read/Write Cache and Purge.

4.3.2.1 Initialize

The initialize method is used for adding default values to the instance variables and objects associated to that particular class. The parameters that can be modified are shown in the listing 4.3 below

Options such as: `max requests` and `block duration` can be changed. As previously mentioned in Section 3.3, `max requests` is used to vary the number of HTTP GET/POST requests before the IP is blocked. Whereas, the `block duration` can be tweaked to set the block duration when an IP is blocked from accessing the web application. The configurations can also be tweaked as per the requirement.

Listing 4.3: Invoking environment variables

```
def initialize(app, options = {})

   @app = app
   # sets the duration to deny the remote address for a specific
      interval
   @block_duration = 1.hour

   # sets the threshold for maximum requests by the remote adress
   @max_requests = 70

   @block_bots = options[:block_bots] || false
   @response = [403,{'Content-Type' => (options[:content_type] || 'text/
      html')},[options[:message] || DETECTED_MESSAGE]]

end
```

4.3.2.2 Fetch Operation

Initially, `call(env)` is used to invoke the hash accessor for environment variables. Here, IP address, remote address, port can be denoted as the environment variables of the cloud application.

Listing 4.4: Invoking environment variables

```
# remote adress of the source under inspection
remote_addr = env['REMOTE_ADDR']

# cache key assigned w.r.t remote address
cache_key = "ip_#{remote_addr}"
```

Subsequently, when the HTTP requests start to arrive from an anonymous source, the remote address of that particular source is fetched from the environment and stored in `remote_addr`.

Additionally, the cache key is stored based on the remote address of the source under inspection. For example, if the remote address is 194.96.xxx.xxx then the cache key will be assigned as `ip_194.96.xxx.xxx`

4.3.2.3 Cache Operation

After the environment has been invoked, a temporary cache file is generated based on the remote address and the corresponding cache key under inspection. The file in cache contains the remote address, block expiration and the time when the cache was created.

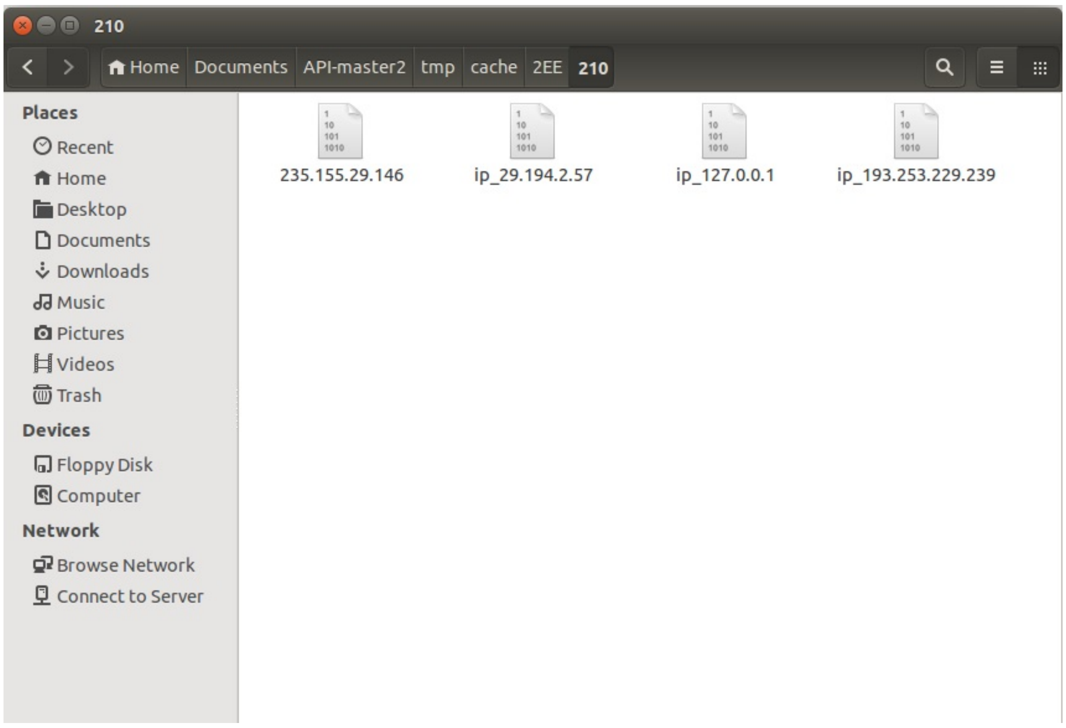


Figure 4.3: Temporary Cache of banned users

Listing 4.5: Read/Write Cache

```
if !Rails.cache.read(cache_key, :raw => true).blank?  # => false if
    cache is nil
      baz= Rails.cache.read(cache_key, :raw => true).to_i
     Rails.cache.write(cache_key, (baz+1).to_s, :raw => true)
   else
       Rails.cache.write(cache_key, "1", :expires_in => @block_duration)
   end
```

Listing 4.5 represents the read and write operation (based on algorithm 2) that is carried out when the IP under inspection is checked if it is already present in the cache. Otherwise a new file is generated in cache with the `block duration`

4.3.2.4 Purge operation

Based on the algorithm 3 mentioned in Chapter 3, a method is designed for excluding any IP address that needs access to the cloud application permanently. The `custom_IP` can be 127.0.0.1 or any other IP that is being used for development purposes.According to the algorithm 3 presented in Chapter 3, the reference coding is shown in listing 4.6

Listing 4.6: Purge operation

```
   def self.purge(custom_ip = nil)
     return Rails.cache.delete("ip_#{custom_ip}") if !custom_ip.blank?
     Rails.cache.read("application_layer").each do |banned_ip|
     Rails.cache.delete("ip_#{banned_ip}")
     end
   end
end
```

The proposed solution is efficient in mitigating Application Layer DDoS attacks. Other methods and procedures were considered which were found to be obsolete relatively. As the proposed framework has a multi-tier protection, it is able to reduce the incoming requests and block the requests that surpass the first tier.

Chapter 5

Evaluation

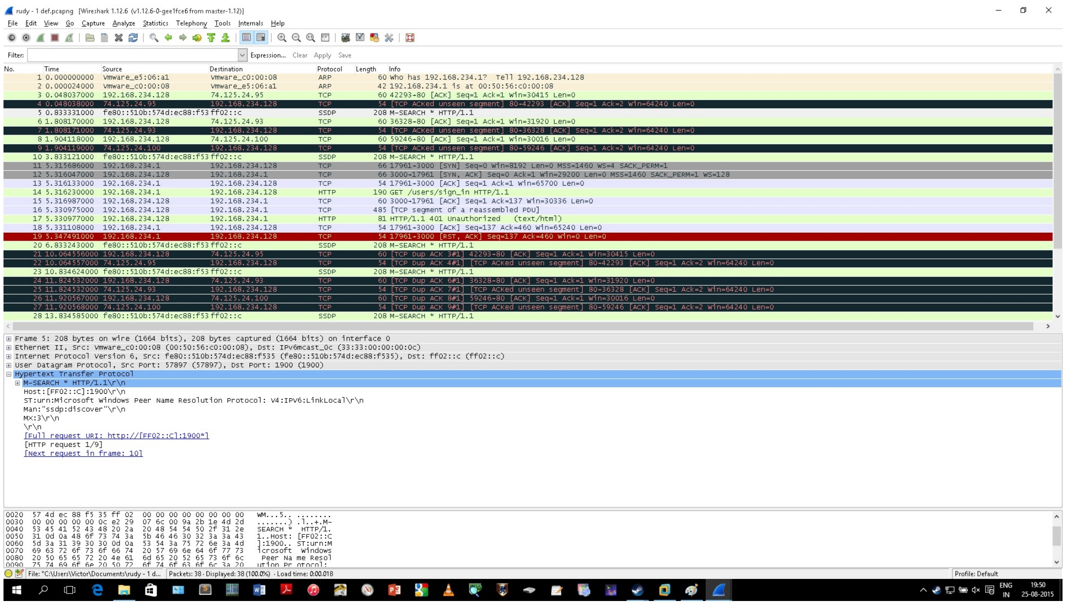
5.1 Preface

This chapter depicts the process of evaluation regarding the detection rate, response time and the efficiency of the the proposed fuzzy logic based mitigation system. There are various measurements when mitigation of application layer DDoS attacks are considered. Section 4.1 discusses about the datasets collected based on user behavior, demographic profiling, nature of requests and user session intervals. Section 4.2 presents the metrics involved in evaluating the proposed system.

The primary aim of the proposed solution is to minimize the impact caused by Layer 7 DDoS requests. Mitigating such threats is a tedious job, as it involves a lot pattern matching and determining the nature of the incoming requests. Hence a two-tier protection against HTTP DDoS attacks has been proposed.

Moreover, RUDY and HULK are the tools that are very powerful in generating HTTP GET/POST attacks that can bring down a cloud infrastructure. Both the tools have unique features that aid in denying the services of a server. Both the tools are relatively easy to setup which opens up a lot of threat to the existing cloud systems. Hence, the proposed solution is tested against the mentioned tools and are compared when the proposed solution absent in the application on cloud.

For the evaluation of the proposed algorithm, Wireshark Network protocol analyzer has been used as benchmarking tool to determine the HTTP DDoS attack trends and the response time. In the context of Layer-7 DDoS detection, response time corresponds to the time taken by the server for processing the HTTP POST/GET requests that are forwarded.


rudy - 1 def.pcapng [Wireshark 1.12.6 (v1.12.6-0-gee1fce6 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Profile: Default

```
No.  Time          Source                     Destination      Protocol Length Info
   1 0.000000000   Vmware_e5:06:a1            Vmware_c0:00:08  ARP       60 Who has 192.168.234.1?  Tell 192.168.234.128
   2 0.000024000   Vmware_c0:00:08            Vmware_e5:06:a1  ARP       42 192.168.234.1 is at 00:50:56:c0:00:08
   3 0.048037000   192.168.234.128            74.125.24.95     TCP       60 42293→80 [ACK] Seq=1 Ack=1 Win=30415 Len=0
   4 0.048038000   74.125.24.95               192.168.234.128  TCP       54 [TCP ACKed unseen segment] 80→42293 [ACK] Seq=1 Ack=2 Win=64240 Len=0
   5 0.833331000   fe80::510b:574d:ec88:f53   ff02::c          SSDP     208 M-SEARCH * HTTP/1.1
   6 1.808170000   192.168.234.128            74.125.24.93     TCP       60 36328→80 [ACK] Seq=1 Ack=1 Win=31920 Len=0
   7 1.808171000   74.125.24.93               192.168.234.128  TCP       54 [TCP ACKed unseen segment] 80→36328 [ACK] Seq=1 Ack=2 Win=64240 Len=0
   8 1.904118000   192.168.234.128            74.125.24.100    TCP       60 59246→80 [ACK] Seq=1 Ack=1 Win=30016 Len=0
   9 1.904119000   74.125.24.100              192.168.234.128  TCP       54 [TCP ACKed unseen segment] 80→59246 [ACK] Seq=1 Ack=2 Win=64240 Len=0
  10 3.833121000   fe80::510b:574d:ec88:f53   ff02::c          SSDP     208 M-SEARCH * HTTP/1.1
  11 5.315686000   192.168.234.1              192.168.234.128  TCP       66 17961→3000 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  12 5.316047000   192.168.234.128            192.168.234.1    TCP       66 3000→17961 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
  13 5.316133000   192.168.234.1              192.168.234.128  TCP       54 17961→3000 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  14 5.316230000   192.168.234.1              192.168.234.128  HTTP     190 GET /users/sign_in HTTP/1.1
  15 5.316987000   192.168.234.128            192.168.234.1    TCP       60 3000→17961 [ACK] Seq=1 Ack=137 Win=30336 Len=0
  16 5.330975000   192.168.234.128            192.168.234.1    TCP      485 [TCP segment of a reassembled PDU]
  17 5.330977000   192.168.234.128            192.168.234.1    HTTP      81 HTTP/1.1 401 Unauthorized  (text/html)
  18 5.331108000   192.168.234.1              192.168.234.128  TCP       54 17961→3000 [ACK] Seq=137 Ack=460 Win=65240 Len=0
  19 5.347491000   192.168.234.1              192.168.234.128  TCP       54 17961→3000 [RST, ACK] Seq=137 Ack=460 Win=0 Len=0
  20 6.833243000   fe80::510b:574d:ec88:f53   ff02::c          SSDP     208 M-SEARCH * HTTP/1.1
  21 10.064556000  192.168.234.128            74.125.24.95     TCP       60 [TCP Dup ACK 3#1] 42293→80 [ACK] Seq=1 Ack=1 Win=30415 Len=0
  22 10.064557000  74.125.24.95               192.168.234.128  TCP       54 [TCP Dup ACK 4#1] [TCP ACKed unseen segment] 80→42293 [ACK] Seq=1 Ack=2 Win=64240 Len=0
  23 10.834624000  fe80::510b:574d:ec88:f53   ff02::c          SSDP     208 M-SEARCH * HTTP/1.1
  24 11.824532000  192.168.234.128            74.125.24.93     TCP       60 [TCP Dup ACK 6#1] 36328→80 [ACK] Seq=1 Ack=1 Win=31920 Len=0
  25 11.824532000  74.125.24.93               192.168.234.128  TCP       54 [TCP Dup ACK 7#1] [TCP ACKed unseen segment] 80→36328 [ACK] Seq=1 Ack=2 Win=64240 Len=0
  26 11.920567000  192.168.234.128            74.125.24.100    TCP       60 [TCP Dup ACK 8#1] 59246→80 [ACK] Seq=1 Ack=1 Win=30016 Len=0
  27 11.920568000  74.125.24.100              192.168.234.128  TCP       54 [TCP Dup ACK 9#1] [TCP ACKed unseen segment] 80→59246 [ACK] Seq=1 Ack=2 Win=64240 Len=0
  28 13.834585000  fe80::510b:574d:ec88:f53   ff02::c          SSDP     208 M-SEARCH * HTTP/1.1

⊞ Frame 5: 208 bytes on wire (1664 bits), 208 bytes captured (1664 bits) on interface 0
⊞ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: IPv6mcast_0c (33:33:00:00:00:0c)
⊞ Internet Protocol Version 6, Src: fe80::510b:574d:ec88:f535 (fe80::510b:574d:ec88:f535), Dst: ff02::c (ff02::c)
⊞ User Datagram Protocol, Src Port: 57897 (57897), Dst Port: 1900 (1900)
⊟ Hypertext Transfer Protocol
  ⊞ M-SEARCH * HTTP/1.1\r\n
    Host:[FF02::C]:1900\r\n
    ST:urn:Microsoft Windows Peer Name Resolution Protocol: V4:IPV6:LinkLocal\r\n
    Man:"ssdp:discover"\r\n
    MX:3\r\n
    \r\n
    [Full request URI: http://[FF02::C]:1900*]
    [HTTP request 1/9]
    [Next request in frame: 10]

0020  57 4d ec 88 f5 35 ff 02  00 00 00 00 00 00 00 00   WM...5.. ........
0030  00 00 00 00 00 0c e2 29  07 6c 00 9a 2b 1e 4d 2d   .......) .l..+.M-
0040  53 45 41 52 43 48 20 2a  20 48 54 54 50 2f 31 2e   SEARCH *  HTTP/1.
0050  31 0d 0a 48 6f 73 74 3a  5b 46 46 30 32 3a 3a 43   1..Host: [FF02::C
0060  5d 3a 31 39 30 30 0d 0a  53 54 3a 75 72 6e 3a 4d   ]:1900.. ST:urn:M
0070  69 63 72 6f 73 6f 66 74  20 57 69 6e 64 6f 77 73   icrosoft  Windows
0080  20 50 65 65 72 20 4e 61  6d 65 20 52 65 73 6f 6c    Peer Na me Resol
0090  75 74 69 6f 6e 20 50 72  6f 74 6f 63 6f 6c 3a 20   ution Pr otocol:

File: "C:\Users\Victor\Documents\rudy - 1 d...   Packets: 38 · Displayed: 38 (100.0%) · Load time: 0:00.018
```

Figure 5.1: Wireshark network protocol analyzer

In this context, Wireshark is used to scan the incoming and outgoing HTTP requests in the application. The tool has helped in generating I/O graphs for determining attack trends when DDoS attack tools are used. There are various filters available in Wireshark that can be configured according to preference. The benchmarking tool also provides real-time I/O HTTP request flow for an IP or Virtual Machine in cloud.

There are two cases where the algorithm is tested for the mitigation of HTTP DDoS attacks. The first case involves the evaluation of algorithm against RUDY tool. The second case involves testing against HULK. Thalmeier [2015] specifies some of the metrics involved in testing a mitigation framework. Among them, Response time and the number of packets are the metrics that are analyzed in the attack trends. The analysis has been performed for a specific number of HTTP attack requests and has been recorded in real-time. Both the cases have been explained below.

5.2 Case 1: RUDY attack analysis

As previously mentioned in Section 3.2.1, RUDY follows a Slow HTTP GET/POST attack pattern and tries to impersonate the traffic trend of a legitimate user, the number of requests are low and the rate of attack is also very low. The proposed algorithm is meant to tackle slow HTTP DDoS attacks and has been successful in mitigating attacks from the tool.

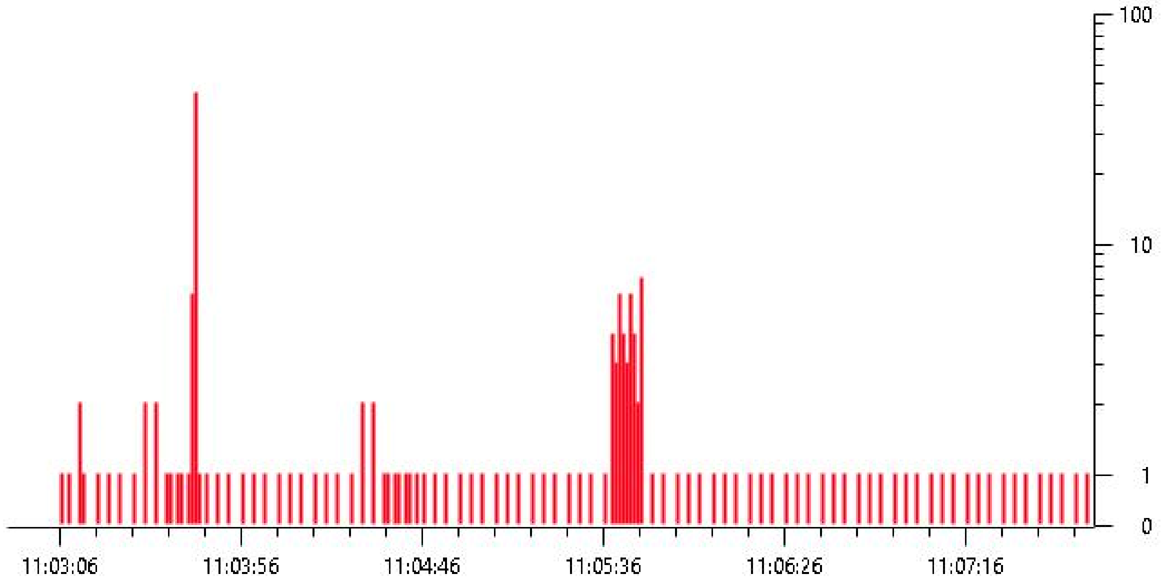


Figure 5.2: RUDY attack trend without proposed defense

The application is first analyzed without the proposed algorithm against RUDY using Wireshark. The attacks from RUDY are able to penetrate the web application easily. The Figure 5.2 represents the attack traffic without the proposed defense system.

The application is now tested with the fuzzy elimination and application layer DDoS mitigation system. A significant reduction in number of packets and response time is noted. The graph 5.3 represents the HTTP traffic trend with the proposed algorithm. The attacks last only for a response time of 20 seconds and it gets terminated.

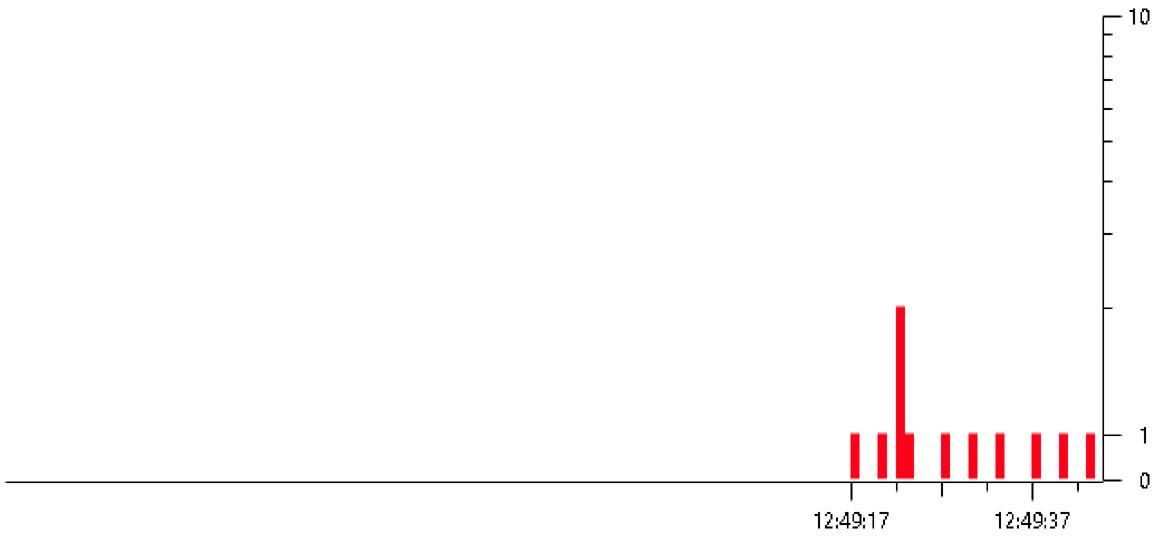


Figure 5.3: RUDY attack trend with proposed defense

5.3 Case 2 : HULK attack analysis

Unlike RUDY, HULK attacks the target with different patterns continuously. The attack rate of HULK tool is very high compared to other Layer 7 DDoS attack tools. It generates requests of high volume in a short period of time which makes the application unavailable for other users.

Similar to the previous case, the hosted application is tested against HULK. It is observed that the malicious HTTP requests are able to completely deny the services. The application was not accessible due to the high volume of requests generated per second. Around 3000 non-legitimate requests were targeted in a span of 25 seconds and the attack was followed by minor HTTP requests at the end. The graph 5.4 represents the attack trend of HULK without the proposed defense.

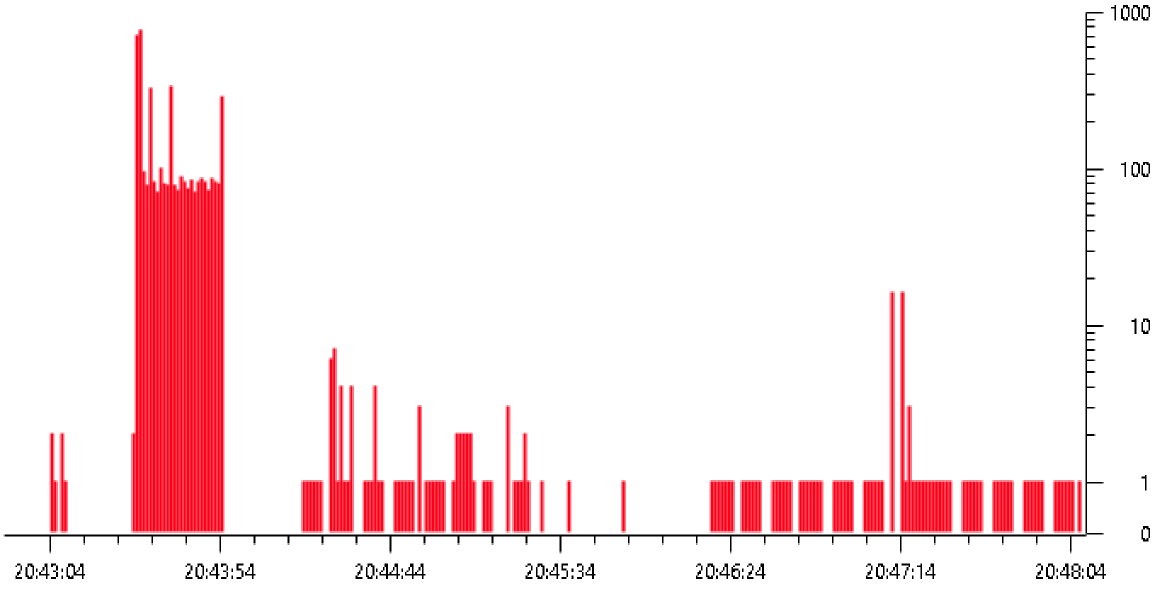


Figure 5.4: HULK attack trend without proposed defense

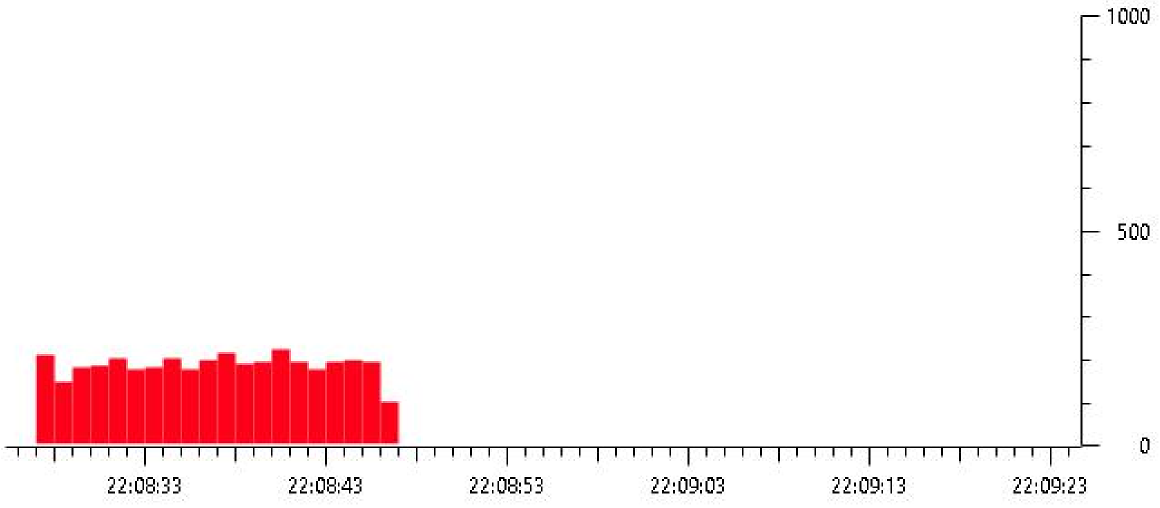


Figure 5.5: HULK attack trend with proposed defense

Now, the application is tested with the proposed solution. The graph 5.5 represents the attack trend of HULK with the proposed solution. Significant reduction in the number of request is observed. Fuzzy Elimination is able to eliminate most of the requests in the first phase(1st minute) of the attack from HULK. After the overflow of the buffer assigned to the IP and successful confirmation of the nature of the HTTP requests, all the incoming requests are completely blocked as the IP is blacklisted. The response time is also minimal in the initial phase of attack trend.

When comparing the graphs, it is evident that the impact caused by first phase of the attack has been minimized by fuzzy elimination and the application layer mitigation is able to discard all the requests after the initial phase.

5.4 Detection Rate

Detection Rate represents the proposed system's extent of detecting AL-Layer DDoS attacks. A simple evaluation criteria has been shared by Chonka and Abawajy [2012] which is shown below :

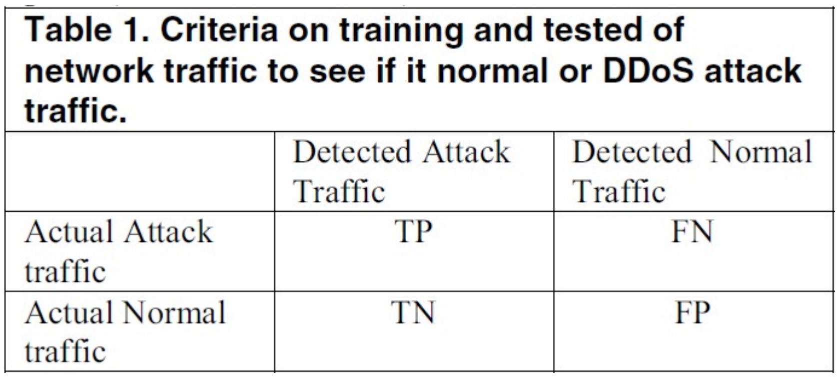


Table 1. Criteria on training and tested of network traffic to see if it normal or DDoS attack traffic.

	Detected Attack Traffic	Detected Normal Traffic
Actual Attack traffic	TP	FN
Actual Normal traffic	TN	FP

Figure 5.6: Criteria for incoming traffic

The table 5.6 denotes the True positive values when the detected requests are the actual attack traffic, whereas when normal HTTP requests are considered as attack traffic, then it is a false negative. True negative cases are those HTTP requests which are genuine but are detected as attack traffic.The false alarm ratio corresponds to how many times the system detects attacks falsely when the incoming requests are normal HTTP requests. Based on the criteria the detection rate can be formulated as follows:

$$\text{Detection Rate} = \frac{TP}{(TP + FN)} * 100$$

The graph 5.7 represents the comparison of detection rates of the proposed framework against RUDY and HULK. The detection rate has been calculated using the formula and has been represented for each case. As RUDY is a SLOW HTTP DDoS attack tool, the requests do not have a unique pattern and are easy to detect as the trend is repetitive. Whereas HULK is a brute force HTTP DDoS tool that overloads the buffer and makes it difficult to mitigate using the proposed framework. Hence, the detection rate for RUDY is better than that for HULK.

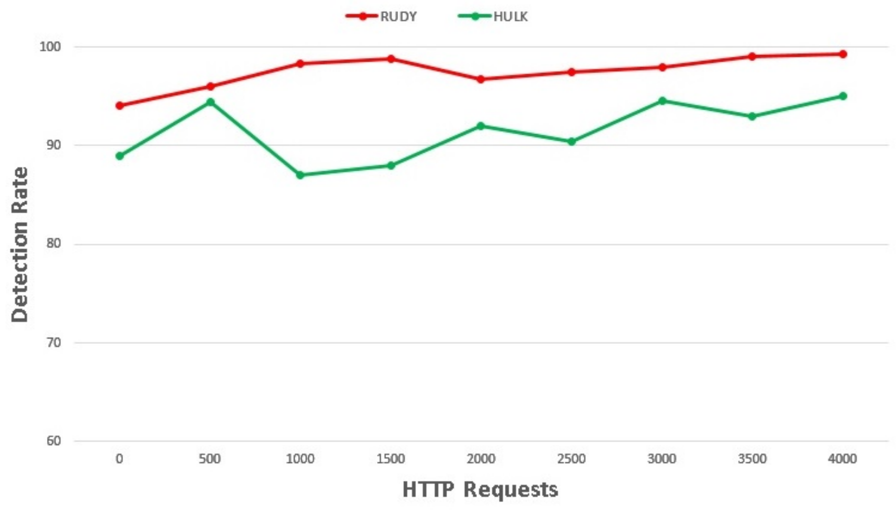


Figure 5.7: Detection Rate of proposed system against RUDY and HULK

5.5 Data collection

This section talks about the collection of pre-processed data regarding user behaviour, geographic/demographic based profiling of user models. Data regarding the session times of the user is also noted. Most importantly ,for categorizing the HTTP requests, two seperate datasets are collected -attack requests data and normal requests data

5.5.1 User behavior and Demographic Profiling of user models

Users from across the world will be accessing a cloud-based website. Based on the structure of the website and interaction of users, a user model can be created. Such a user model will portray the frequency/popularity of a web page, number of clicks and HTTP access.

On the other hand it is evident that users with heterogeneous behavior would be accessing that particular server. Hence a demographic user model can be created. Therefore, for instance, a sudden access from japan will flag up and requests from such sources can be avoided.

Hence, before access is given , such profiles and datasets can be denied access for the server. This in turn will help in minimizing the load on the server.

5.5.2 Normal and Attack requests data

Before the testing of the proposed system , data pre-collection of normal HTTP requests and non-legitimate HTTP requests is profiled. For categorizing the HTTP requests, two seperate datasets are collected -attack requests data and normal requests data
This helps in filtering the incoming HTTP requests based on the profiled data which has been collected beforehand. Wherein, the fuzzy logic interface helps in judging the nature of the attack.

5.5.3 User session

The uptime of the user i.e the duration of user session is recorded until the connection is terminated. Similarly, the downtime is also recorded.
This helps in analyzing the average operational time of the user. A sudden spike in a lot of user session time can aid in blacklisting that particular user.

Chapter 6

Conclusions

The thesis proposed a Fuzzy Logic based Application Layer DDoS Mitigation Framework that is able to tackle Layer 7 based attacks successfully. This paper has put forward the surviving DDoS Mitigation Techniques and has suggested how to extend the core capabilities of the former for an efficient defense. The multi-tier approach for protecting DDoS attacks in a cloud application makes it more efficient than the existing mitigation mechanisms.

Various concepts and their challenges are critically analyzed and compared in this paper. The concepts and methods that have been used in the implementation of the proposed framework have also been discussed. Layer 3/4 and Layer 7 based mitigation techniques have also been discussed. In addition, proactive systems are preferred over reactive measures as the enterprises are keen on protecting their data before it is compromised.

Fuzzy Logic can act as a backbone to analyze datasets, organize pre-defined rules and also aid in the decision making process of HTTP requests. Fuzzy Logic is helpful in identifying the nature of the incoming requests and can avoid any brute force block of an IP by the mitigation system.

This paper is focused on Layer 7 (Application layer) based DDoS attacks as they are the most prominent threats currently in cloud. Layer 7 attack generating tools such as RUDY and HULK were used against the proposed system. The regarding performance statistics of the solution have been analyzed in the evaluation section. Minimal response time and reduced number of attack requests were observed against HTTP DDoS attack tools. The solution has aided in improving the performance and page load of the cloud application.

The development of the proposed framework is focused on using fuzzy logic libraries to detect the incoming HTTP requests and mitigate using the second tier of the framework. Based on the request length and frequency, the HTTP requests are categorized and are forwarded to the application layer mitigation which then protects the cloud application by banning the IPs that target bogus HTTP requests.

The proposed Mitigation Framework has the following advantages: it can be configured easily and there are no dependencies. It is efficient in mitigating all Application Layer attacks with minimal overhead. The response time and the number of attack requests are also low.

6.1 Future Work

Features like blacklisting and whitelisting can be added to the framework as a part of future work to enhance the recognition and user experience. Pre-defined parameters have been assigned for the experimental setup. However, the parameters can be tweaked and rules can be assigned for any specific data center. The future work of the proposed system is to make the framework more intelligent and well-trained which will enable fuzzy logic to detect non-legitimate requests from the HTTP traffic. Also developing a knowledge base for normal and attack HTTP requests can help in training the fuzzy logic to eliminate anonymous threats.

With constant advancements in technology, attackers will have sophisticated DDoS generating tools. Hence, all types of DDoS attacks in cloud cannot be mitigated with a single tool. DDoS defense systems should be modified and updated frequently to protect an application or a server from such threats. Fuzzy Logic based Application Layer DDoS Mitigation is apt for future research.However, a DDoS attack would have multiple sources and tracing the primary source of the attack poses a great challenge for future research.

Bibliography

Glenn Anderson. Fuzzy logic: What it is; what it does; what it can do. *Production*, 106(10): 38, 10 1994. URL https://ezproxy.ncirl.ie/login?url=http://search.proquest.com/docview/217432479?accountid=103381. Copyright - Copyright Gardner Publications, Incorporated Oct 1994; Last updated - 2014-05-21; CODEN - PDTNAG.

E. Anitha and S. Malliga. A packet marking approach to protect cloud environment against ddos attacks. In *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, pages 367–370, Feb 2013. doi: 10.1109/ICICES.2013.6508330.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL http://doi.acm.org/10.1145/1721654.1721672.

A. Belenky and N. Ansari. Ip traceback with deterministic packet marking. *Communications Letters, IEEE*, 7(4):162–164, April 2003. ISSN 1089-7798. doi: 10.1109/LCOMM.2003.811200.

Pedro Casas, Johan Mazel, and Philippe Owezarski. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Comput. Commun.*, 35(7):772–783, April 2012. ISSN 0140-3664. doi: 10.1016/j.comcom.2012.01.016. URL http://dx.doi.org/10.1016/j.comcom.2012.01.016.

Prajkta P. Chapke and Rupali R. Deshmukh. Intrusion detection system using fuzzy logic and data mining technique. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering &#38; Technology (ICARCSET 2015)*, ICARCSET '15, pages 63:1–63:5, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3441-9. doi: 10.1145/2743065.2743128. URL http://doi.acm.org/10.1145/2743065.2743128.

A. Chonka and J. Abawajy. Detecting and mitigating hx-dos attacks against cloud web services. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, pages 429–434, Sept 2012. doi: 10.1109/NBiS.2012.146.

A. Chonka, Wanlei Zhou, J. Singh, and Yang Xiang. Detecting and tracing ddos attacks by intelligent decision prototype. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 578–583, March 2008a. doi: 10.1109/PERCOM.2008.76.

A. Chonka, Wanlei Zhou, and Yang Xiang. Protecting web services with service oriented traceback architecture. In *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, pages 706–711, July 2008b. doi: 10.1109/CIT.2008.4594761.

Ashley Chonka, Yang Xiang, Wanlei Zhou, and Alessio Bonti. Cloud security defence to protect cloud computing against http-dos and xml-dos attacks. *J. Netw. Comput. Appl.*, 34(4):1097–1107, July 2011. ISSN 1084-8045. doi: 10.1016/j.jnca.2010.06.004. URL http://dx.doi.org/10.1016/j.jnca.2010.06.004.

Dipen Contractor and DhirenR. Patel. Trust management framework for attenuation of application layer ddos attack in cloud computing. In Theo Dimitrakos, Rajat Moona, Dhiren Patel, and D.Harrison McKnight, editors, *Trust Management VI*, volume 374 of *IFIP Advances in Information and Communication Technology*, pages 201–208. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29851-6. doi: 10.1007/978-3-642-29852-3_14. URL http://dx.doi.org/10.1007/978-3-642-29852-3_14.

M. Darwish, A. Ouda, and L.F. Capretz. Cloud-based ddos attacks and defenses. In *Information Society (i-Society), 2013 International Conference on*, pages 67–71, June 2013.

T. Dillon, Chen Wu, and E. Chang. Cloud computing: Issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33, April 2010. doi: 10.1109/AINA.2010.187.

Yinghong Fan, H. Hassanein, and P. Martin. Proactively defeating distributed denial of service attacks. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 2, pages 1047–1050 vol.2, May 2003. doi: 10.1109/CCECE.2003.1226075.

Robert Goss, Martin Botha, and Rossouw von Solms. Utilizing fuzzy logic and neural networks for effective, preventative intrusion detection in a wireless environment. In *Proceedings of the 2007 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, SAICSIT '07, pages 29–35, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-775-9. doi: 10.1145/1292491.1292495. URL http://doi.acm.org/10.1145/1292491.1292495.

Imperva. *Denial of Service Attacks: A Comprehensive Guide to Trends, Techniques, and Technologies*. ADC Monthly Web Attacks Analysis, September 2012.

ISO. *Information processing systems – Open Systems Interconnection – Basic Reference Model*. November 1989.

Jos De Jess. Navigating the ibm cloud, part 1: A primer on cloud technologies. *IBM WebSphere Developer Technical Journal.*, 2012. ISSN 1867-4828. URL http://www.ibm.com/developerworks/websphere/techjournal/1206_dejesus/1206_dejesus.html.

Yaochu Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, May 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.919172.

Dongwoo Kwon, J.W.-K. Hong, and Hongtaek Ju. Ddos attack forecasting system architecture using honeynet. In *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, pages 1–4, Sept 2012. doi: 10.1109/APNOMS.2012.6356055.

Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 23–31, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-3713-9. doi: 10.1109/CLOUD.2009.5071529. URL http://dx.doi.org/10.1109/CLOUD.2009.5071529.

Yadong Li, Wenqiang Cui, Danlan Li, and Rui Zhang. Research based on osi model. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 554–557, May 2011. doi: 10.1109/ICCSN.2011.6014631.

Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

Cade Metz. Bitbucket's amazon ddos-what went wrong. http://www.theregister.co.uk/2009/10/09/amazon_cloud_bitbucket_ddos_aftermath/, 2009. Accessed: 2014-12-05.

Vilem Novak, Irina Perfilieva, and Jiri Mockor. *Mathematical Principles of Fuzzy Logic*. Springer US, 1999. ISBN 978-1-4615-5217-8. doi: 10.1007/978-1-4615-5217-8.

Gunter Ollman. Web application dos/ddos testing methodology. Technical report, IOActive, Inc., 01 2014.

R. Porter, J. Theiler, and D. Hush. Interactive machine learning in data exploitation. *Computing in Science Engineering*, 15(5):12–20, Sept 2013. ISSN 1521-9615. doi: 10.1109/MCSE.2013.74.

Raviv Raz. R-u-dead-yet or rudy http denial of service tool. https://code.google.com/p/r-u-dead-yet/, 2011. Accessed: 2015-01-05.

Farzad Sabahi. Cloud computing security threats and responses. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 245–249. IEEE, 2011.

J. Sahoo, S. Mohapatra, and R. Lath. Virtualization: A survey on concepts, taxonomy and associated security issues. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 222–226, April 2010. doi: 10.1109/ICCNT.2010.49.

Barry Shteiman. Hulk, web server dos tool. http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/, 2012. Accessed: 2015-01-05.

Jungsuk Song, Hiroki Takakura, Yasuo Okabe, and Koji Nakao. Toward a more practical unsupervised anomaly detection system. *Information Sciences*, 231(0):4 – 14, 2013. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2011.08.011. URL http://www.sciencedirect.com/science/article/pii/S0020025511004245. Data Mining for Information Security.

P. Srivastava, S. Singh, A.A. Pinto, S. Verma, V.K. Chaurasiya, and R. Gupta. An architecture based on proactive model for security in cloud computing. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 661–666, June 2011. doi: 10.1109/ICRTIT.2011.5972392.

Kevin Surace. The new security threat we all face. http://www.infotech.com/research/special-letter-all-your-data-is-being-stolen-as-you-read-this/, 2013. Accessed: 2015-12-05.

Hassan Takabi, James B.D. Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security and Privacy*, 8(6):24–31, 2010. ISSN 1540-7993. doi: http://doi.ieeecomputersociety.org/10.1109/MSP.2010.186.

Werner Thalmeier. How to evaluate a vendor for ddos and cyber attack protection. https://blog.radware.com/security/2015/11/evaluate-vendor-ddos-protection/, 2015. Accessed: 2015-01-12.

Thomas Vissers, Thamarai Selvi Somasundaram, Luc Pieters, Kannan Govindarajan, and Peter Hellinckx. {DDoS} defense system for web services in a cloud environment. *Future Generation Computer Systems*, 37(0):37 – 45, 2014. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2014.03.003. URL http://www.sciencedirect.com/science/article/pii/S0167739X1400048X. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.

Huangxin Wang, Quan Jia, Dan Fleck, Walter Powell, Fei Li, and Angelos Stavrou. A moving target {DDoS} defense mechanism. *Computer Communications*, 46(0):10 – 21, 2014. ISSN 0140-3664. doi: http://dx.doi.org/10.1016/j.comcom.2014.03.009. URL http://www.sciencedirect.com/science/article/pii/S0140366414000954.

Peter J. Woolf and YIXIN Wang. A fuzzy logic approach to analyzing gene expression data. *Physiological Genomics*, 3(1):9–15, 2000. ISSN 1094-8341.

Yang Xiang, Wanlei Zhou, and Minyi Guo. Flexible deterministic packet marking: An ip traceback system to find the real source of attacks. *Parallel and Distributed Systems, IEEE Transactions on*, 20 (4):567–580, April 2009. ISSN 1045-9219. doi: 10.1109/TPDS.2008.132.

Lanjuan Yang, Tao Zhang, Jinyu Song, JinShuang Wang, and Ping Chen. Defense of ddos attack for cloud computing. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 2, pages 626–629, May 2012. doi: 10.1109/CSAE.2012.6272848.

Lotfi A. Zadeh. A summary and update of fuzzy logic. In *Granular Computing (GrC), 2010 IEEE International Conference on*, pages 42–44, Aug 2010. doi: 10.1109/GrC.2010.144.

Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010. ISSN 1867-4828. doi: 10.1007/s13174-010-0007-6. URL http://dx.doi.org/10.1007/s13174-010-0007-6.

Appendix A

Configuration of attack tools

Before using any of the following tools, Python binaries and libraries have to be installed. Windows users, download the latest version of Python[1] and run the following command

```
msiexec /i python<version>.msi
```

Linux users can install Python by executing the following commands in the terminal.

```
wget http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
tar -xzf Python-2.7.6.tgz
cd Python-2.7.6 make   sudo make install
```

HULK and RUDY both require python. After installing Python, extract the tools in a directory and follow the instructions written in the subsequent sections.

Generally, RUDY has two modes of operation: Configuration-based execution and interactive mode. Configuration-based execution requires a config file `rudeadyet.conf` placed within the tool directory. Listing A.1 indicates the `rudeadyet.conf` which contains parmeters that can be changed before executing the tool.

Interactive mode involves calling the tool from command line with this command : `r-u-dead-yet.py <URL>`. Wherein, URL is a FQDN link leading to a web form to attack. Additionally, several options are displayed when the given command is executed. Options such as :form to attack, form field to use, number of conccurent connectons of attack and whether a SOCKS proxy should be used or not.

[1] https://www.python.org/downloads/windows/

HULK can be executed only from the command line where the URL to be targeted is added after `hulk.py`. Alternatively, the Target IP, Host IP, headers and other options can be hard-coded in the python file. Listing A.1 shows all the global parameters that can be tweaked within the python script as per requirement.

Listing A.1: Configuration of attack tools

```
#RUDY
[parameters]
URL: "http://192.xxx.xxx.128:3000/users/sign_in"
number_of_connections: 50
attack_parameter: Password
proxy_port: 0
proxy_addr: ""

# URL = POST URL
# number_of_connections = concurrent processes to execute
# attack_parameter = POST parameter to fuzz
# proxy_addr = IP of the SOCKS4 proxy to use (empty if not required)
# proxy_port = TCP port on which the SOCKS4 proxy is listening

#HULK
#global params
url='http://192.xxx.xxx.128:3000/users/sign_in'
host=192.168.xxx.xxx
headers_useragents=['Mozilla']
headers_referers=[]
request_counter=0
flag=0
safe=1
```