Mitigating DDoS Attacks Using a Resource Sharing Network

by

Farabi Fardin Khan
17101136
Nafis Mohaimin Hossain
17101133
Toushif Mahmud
17101128
Sad Bin Anwar
21341041
Sirajul Islam
17101467

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Farabi Fardin Khan
17101136

Nafis

Nafis Mohaimin Hossain
17101133

[illegible]

Toushif Mahmud
17101128

Saad

Sad Bin Anwar
21341041

Sirajul

Sirajul Islam
17101467

Approval

The thesis titled "Mitigating DDoS Attacks Using a Resource Sharing Network " submitted by

1. Farabi Fardin Khan(17101136)
2. Nafis Mohaimin Hossain(17101133)
3. Toushif Mahmud(17101128)
4. Sad Bin Anwar (21341041)
5. Sirajul Islam(17101467)

Of Summer,2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October2,2021 .

Examining Committee:

Supervisor:
(Member)

[illegible]

Ms.Jannatun Noor Mukta
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md.Golam Rabiul Alam,PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

Whatever result we have found from this thesis is purely our original work. We have read several papers, journals, and articles to help increase our knowledge on the topic but the findings and analysis are our own work. Materials from other sources have been properly cited in this thesis.

Abstract

Cloud computing is crucial to the internet just as air is crucial to breathing. Most users do not even come to think how the cloud architecture is one of the giant pillars on which the entire internet and all its related services depend. In recent times, cloud computing has gained noticeable popularity due to its ability to radically improve computing power through the application of virtual machines. In this era of the internet, however, security threats are increasing and it costs many businesses. The seemingly legitimate traffic of these application-level attacks renders the previous detection and mitigation methods ineffective. These cyber-attacks have grown ever so sophisticated and the detection and mitigation of these attacks have become one of the major concerns of security researchers and cloud service providers all around the globe. In our paper, we are trying to analyze existing research and implementation of the cloud architecture. This analysis will give us insight into the core mechanisms of the cloud architecture and DDoS attacks. Furthermore, we will apply this understanding of ours to design and build an universal mitigation technique for DDoS attacks. An universal solution that works for all scenarios, may take at least a few dozen developers working years on end to develop. Therefore, the goal of this paper will be to lay the foundation on which one day the universal solution may be created.

Keywords: DDoS; Attack; Detection; Network; Server; Mitigation,System, Requestes, Time.

Dedication

To our parents, we dedicate our thesis. Because of their unending support, love, and encouragement, we are inspired. Above all, we owe our parents' love and support to them. We will continue to draw from both of you for the strength and courage you both gave me as we strive to reach for the heights and chase our ambitions. Thank our brother and sister too.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Ms. Jannatun Noor Mukta madam for her kind support and advice in our work. We really want to thank her for being an incredible listener.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

API	Application Programming Interface
CSP	Cloud Service Provider
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
FS	Feature Selection
HTTP	Hypertext Transfer Protocol
KVM	Kernal-based Virtual Machine
UI	User Interface
VM	Virtual Machine
WOA	Whale Optimization Algorithm

Chapter 1

Introduction

1.1 Introduction

The advent of knowledge and communication technologies and increasing accessibility to the net, the net becomes a target for varied kinds of DDOS attacks. Moreover, DDOS attacks can totally down the servers, and users become unable to use the server. The primary objective of this attack is to totally destroy server resources for example memory, sockets, metrics, etc. As a result, the server becomes inaccessible to the users. Over the years, many major companies has hit by DDoS and infrastructures of internet, feat necessary losses in credits. One in every of the primary major DDoS flooding attacks that created their services offline for regarding two. The reason DDoS attacks keep a significant threat even once such tons of years is as a result of they have matured and progressed over the years. The attacks at first relied on victimization distorted packets or flooding with network layer packets in the device. As a result of the architecture became tons a lot of refined and defenses at the network layer became tons a lot of study against these attacks, attackers enraptured on to the application layer. DDoS attacks at the application layer have been on the increase for some years.

The complexities of DDoS attacks at the application layer are expected to grow over time. Application layer attacks give a lot of refined versions of DDoS attacks within the sense that they're rather more the same as traditional user traffic and therefore cause a significant challenge in however they will be known. The attacks are applied to victimization legitimate user requests, which rule out the chance of inspecting a packet to label it as malicious or not. As a result, each network-layer defenses and a few of the prevailing internet Application Firewalls (WAFs) fail to sight these attacks. The actual fact that these attacks are dead victimization multiple protocols at the applying layer, each association familiarized and connection-less compounds the danger. To detect DDOS attacks there are many ways and most of them are very pricey. In this paper, we propose a way to detect DDoS attacks and in that method, we can handle DDoS attacks. We use some filtering method to track the IP addresses and after that,the unnecessary IP requests send to the proxy server and the unwanted IP packet request will be deleted thereby memory management system. In this way, DDoS can be detected with less cost.This paper is based on the ground values to detect Denial of Service attacks and mitigating it with a minimum cost:

- We tried various storage and hosting services through driver storage of our home devices as well as various premium cloud service providers. By this, we tried to get the idea of storage hosting services.
- We tried analyzing tools to focus on certain attacks and extract valuable data from them. Such analysis helps to move towards our goal, which is maneuvering the number of requests in a server.
- A network of shared computing resources also assisted us to tackle unwanted demands. These processes also have their margins, our tests are not performed in a perspective of a massive amount of traffic a regular host/server is used to.

1.2 Background

In the time period of the web, DDoS attacks failed to get enough attention and it had been a not severe threat. In 1996, the primary DDoS attacks occurred in Panix[29], Panix was knocked by an SYN flood[29]. GitHub, a preferred online code management service employed by several developers was hit with the biggest ever DDoS attack. The platform was accustomed to high levels of traffic however what it had been not ready for was the huge flow of traffic that beat at a record-breaking one.3 terabits per second. The GitHub attack failed to involve botnets however instead used a way referred to as Memcached an info-catching system accustomed to speed up websites and networks.[17]. In 2020 the AWS was hit by a big DDoS attack[29][11]. it had been the foremost extreme DDoS attack ever [29][11]. The spectacular part of the attack was the AWS got to strike a pair of.3 terabytes per second and it lasted for three days. Moreover, Government websites are generally targets of such attacks, and conjointly the inspiration is typically policy disagreements between the govt. and conjointly the attackers. Next, in keeping with Cisco, the entire variety of DDoS attacks have seen around seven.9 billion in 2018 and it'll be doubled one thing over fifteen million at intervals 2023[12] . as a result of around 2023, the number of web users is going to be 5.3 billion [12]. These DDoS attacks bare the inherent at intervals the govt infrastructure and conjointly the shortage of security measures in place. Banking and e-commerce sites are prime targets for DDoS attacks. Moreover, in 2016 the biggest DDoS attack was directed at force unit a serious DNS supplier in Oct 2016 [12]. The attack was massively tumultuous and brought down the websites of over eighty of its customers together with amazon, Netflix, Airbnb, Spotify, Twitter, PayPal and Reddit employing malware known as Mirai hackers created an enormous botnet of a hundred thousand webs of things.

1.3 Problem Statement

DDoS attacks can disrupt the proper functioning of websites, servers, and any other web services. Usually, it temporarily suspends the hosting servers/website but permanent unavailability's are also there. It is hard to detect DDoS attacks when they happen simultaneously. A huge amount of time and money is lost for Denial of Services every year around the globe. The dependency of e-commerce in the world can not be compared, disruption to it brings suffering to both consumer and producer.

The server downtime for just an hour can make a vast impact on the earnings of a company. Legitimate users have been suffering the most from these attacks. As we enter a new era, the implementation of advanced sciences is visual. In order to improve efficiency, it is vital to have a comprehensive understanding of the application of new technology.

1.4 Research Objective

This report is based on the ground values to detect Denial of Service attacks and mitigating it with a minimum cost. The detection of such attacks is difficult to measure let alone diminish. Many renowned Cloud Computing companies have their own secured network designs. We tried various storage and hosting services through driver storage of our home devices as well as various premium cloud service providers. By this, we tried to get the idea of storage hosting services. But, this was the very beginning of our experience. It requires certain tools to load test the number of services a website/server can provide. We tried analyzing tools to focus on certain attacks and extract valuable data from them. Such analysis helps to move towards our goal, which is maneuvering the number of requests in a server. A network of shared computing resources also assisted us to tackle unwanted demands. These processes also have their margins, our tests are not performed in a perspective of a massive amount of traffic a regular host/server is used to.

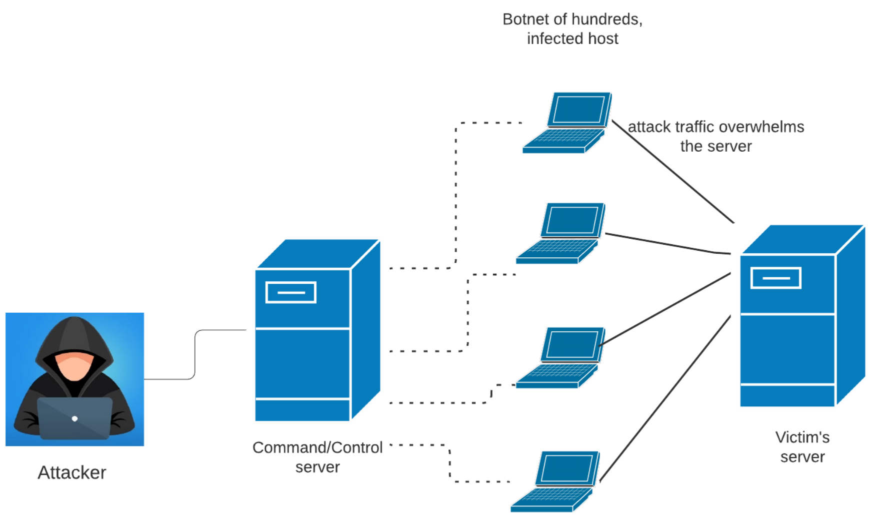

Botnet of hundreds, infected host
attack traffic overwhelms the server
Attacker
Command/Control server
Victim's server

Figure 1.1: DDoS attack

Chapter 2

Related Work

2.1 Literature Review

Hoai Viet Nguyen and Luigi Lo Iacono in their paper [12]. talked about the new system of web store attacks. They proposed a model which inducing an error on the main server that was not recognized by the intermediary caching system, the cache has been tainted by the page created by the server and is programmed to give this meaningless rather than the content one planned. Therefore, rendering the victim service inaccessible. They gathered the default HTTP request header limitations used by various HTTP engines and caching systems in an exploratory investigation. They also experiment with the Feasibility of HMC attacks, HHO attacks. Finally, they described three specific CPDoS attack types driven by inconsistencies in the HTTP method override header, header size constraints, and metacharacter parsing.

Next, In this paper [18], the author presented LUCID which was a viable, lightweight deep learning DDoS detection solution that uses Convolutional Neural Networks (CNNs) to identify as far as traffic flows go, this might be classified as harmful or benign. They suggested a DL-based identification construction that uses CNNs to have a better understanding of with minimal processing overhead and attack detection time, DDoS and benign traffic flows may be detected quickly in online resource-constrained scenarios.

De Assis, M. V. O., Novaes, M. P., Zerbini, C. B., Carvalho in their paper [5] presented an online protection model into the SDN network to reduce DDoS attacks. MLP-DS, PSO-DS, and Wave Detect are three ways of acting on the detecting module that they present. Their first one needed the data from the past and then it is supervised while the other two ways remain unsupervised. They explained in their paper that the MLP-DS can easily catch the anomaly whereas the other two approaches collect the information to mitigate the correct attacks. Finally, they put their method to the test using Data on IP traffic and that was generated. by the Mininet network emulator and a floodlight controller and found that the given mechanism of defense performed well in both detection and mitigation processes.

Additionally, In this paper [22], they Attempt to find new ways to combat DDoS assaults in cloud settings based on containers. They used new methods which are based on a mathematical model. This model mainly surveys the pros and cons of their proposed model and from the feedback, they can develop their model more sharply. dynamic DDoS mitigation solution that may dynamically manage the number of container instances serving different customers while also coordinating to maximize service quality, resources should be allocated to these instances. In the end, through simulations and test bed-based trials, their research confirms the strategy's validity where they find their experiment output is very much positive to detect the low rate attacks by using a small number of resources.

The author of[16], performed a systematic model that is in a sliced directory, invert the directory's Show that the directory may be utilized to bootstrap conflict-based cache attacks on the last-level cache in a non-inclusive cache hierarchy. After that, they showed two attacks on this hierarchy system where one is named prime plus probe attack. It works in state-of-the-art non-inclusive sliced caches like Skylake-X without requiring the victim and attacker to share cores or virtual memory. The second attack is a unique, high-bandwidth Evict plus Reload assault that makes use of a multi-threaded opponent to get around non-inclusive cache replacement restrictions. Finally, a new system was developed which detects the cache lines from the visible slice and the non-inclusive LLC.

The author of this study [9] gives an overview of DDoS attacks and REDUCE solutions in the cloud computing domain. They discovered that a DoS attack is the most common type of DDoS assault in the cloud. They proposed three ways to solve this problem. When compared to traditional DDoS solutions, multilevel solutions particularly created for the cloud and their characteristics will undoubtedly perform better than they show in their paper. Finally, they set up a model which is named a multi-layer that can test the attack and provide a successful assessment for this attack. This study was done in a private cloud environment utilizing Tor Hammer as an attacking tool and an Intrusion Detection System to build a fresh dataset. Various methods based on machine learning are employed in this project: like for classification, Support Vector Machine, Naive Bayes, and Random Forest are used and Support Vector Machine, Random Forest, and Naive Bayes, respectively, had an overall accuracy of 99.7.

Wani, A. R., Rana, Q. P., Saxena, U., and Pandey, N. in their paper[15] performed a dataset and a class which is respectively nine and four. Support Vector Machine, Naive Bayes, and Random Forest where the data was analyzed using machine-learning algorithms. They have seen the SVM system provide prominent rightness among the others and Because of its strong performance, the SVM model was utilized for intrusion detection. Finally, they talked about resolving the problem of this attack as the uses of this are growing so fast and this attack is carried out by the transmission of a flood of bogus request packets so it is hard to find out the source of these attacks.

In another paper [21], the author proposed a cloud computing context, which was a machine learning-based solution that has been developed to identify DDoS problems. The system was designed using voting extreme learning machines as a classifier. It detects the attack by using numerous extreme learning machines at the same time. Firstly, they examined the VELM's ELMs that are all trained on the same dataset which computes the result. If the majority of votes are for the attack, the sample is classified as an attack sample; otherwise, it is classified as a normal sample. Finally, they found that their proposed system showed a greater positive result. The Variable system characteristics were used to undertake a complete study of performance.

In a research paper[13], the author deals with potentially harmful distributed denial-of-service attacks and they talked about possibly making the production site useless by flooding the server network with malicious code which might send thousands of malicious requests to the server, or the service might be brought down by exploiting software flaws. Then the researchers evaluated the system to resolve these DDoS attacks. First one names egress and ingress which prevents the attack in the cloud by using the system of NEIF. Whereas the other one is honeypots which can take the attacks and can store the attacks behaviors. Because each field has its own set of dangers, each methodology may be improved further.

Gupta, B.B., Badve, O.P. in their paper[1], gives an overview of DoS attacks, distributed DoS assaults, and various protective techniques, ideas, and devices that may be used in a Cloud setting. In the cloud environment, legitimate packets and illegitimate packets should know very well. They claimed that there are many different types of DoS attacks as a result there is no one-size-fits-all solution to detect this problem. According to them, these defense mechanisms are based on two factors. The first one is the place where this tool has happened and the other is the time of this system. Furthermore, the Author discusses the IP traceback mechanism in this way we can find the true source of the attack. After that, the author discusses the management info base where by analyzing the info it can conclude the destination of attack. After that, the packets are being marked and It features a dynamic technique for filtering the traffic of attackers. This filtering gets the IP from the stored data and it gets through the end station. In their paper, the author talks about the system but When the amount of the attacker's traffic rises, the filtering mechanism becomes ineffective.

In this paper[1], the dynamic detection method is described. This approach is used on edge routers that are close to the victim's location. They also explain how to detect DDOS attacks at several levels. To identify DDOS attacks, neural networks and data mining techniques are described which require shorter memory by getting the speedy detection.

Zerki, Kafahali, Aboutabit, and Saadi in their paper[3], talk about the main methods triggering a DDoS attack, economic/financial gain, cyber warfare, intellectual Challenge, and ideological belief. The authors suggested DDoS detection that is both quicker and more accurate, Using the C4.5 algorithm and a decision tree methodology, which works by eliminating redundant packets in a particular format. In their paper, it proposes a model of Low computing cost, quicker observation rate, network DDoS detection in a Cloud context, scalability, low false positives, and false negatives, and high validity of the data. To provide high detection precision, they coupled their approach with signature detection techniques.

Kriti Bhushan and B. B. Gupta mention in their paper [4], SDN has a number of key characteristics that make it an ideal networking solution for cloud computing. In this SDN architecture, they briefly discuss models of OpenFlow, OpenFlow switch, SDN-based cloud, Security challenges in the system cloud, and attack detection defense in the selected cloud. They also describe the Using of a queuing mathematical model based on theory and then calculate the flow table-space of a switch. This gives them the ideas to solve this selected problem. Moreover, in their paper, they proposed a model named novel flow table which can detect the attack ion the SDN-based cloud. They explain that restricting the dimensions of flow tables is the root cause of flow table overloading DDoS assaults in SDN. They also devised a flow table sharing strategy to solve the problem of insufficient flow table space.

Kesavamoorthy and Soundar in their paper[8], talk about a novel approach It suggests detecting and defending Using autonomous multi-agent systems to defend against DDoS assaults and to communicate effectively, the agents employ particle swarm optimization. They explained a common thought which is to detect the attacks and by cooperating among them it is updated the agent. The coordinator agent examines their situation utilizing the entropy and covariance approaches to look for flaws to detect the problem. The suggested technology in the cloud platform as a consequence may effectively block various types of DDoS attacks.

Other author in their paper [14], talk about the different kinds of DDoS problems and talk about the outcome of these problems. In this paper, they tried to solve issues like Approaches to prevention, detection, and mitigation. They gathered a variety of approaches and It allows a susceptible server to continue serving requests at the same rate. They bring attention to something. Cloud multi-tenancy for efficient hardware Another cloud computing difficulty that requires consideration is the appropriate usage of powerful servers and multi-tenancy. Finally, they point out the software network which helps them to reduce the problem with the help of both cloud and topology. This aids the survey's analysis of the benefits and drawbacks of various assaults and their remedies.

Gaurav, Manoj, Dheeraj, Mauro, and Rajkumar in their paper [2], provide a thorough examination of the characterization, prevention, detection, and mitigation techniques of DDoS attacks. To mitigate these attacks, they describe Some of the important criteria of the required solution are precise auto-scaling choices, multi-layer mitigation, and protection leveraging extensive cloud resources. They also show when compared to traditional DDoS solutions, multilayer solutions are particularly created for the cloud and their characteristics will undoubtedly perform better. The authors finalize stating, guideline-based solutions for multi-layer solutions can be checked for efficient assessment of their cloud infrastructure.

In another paper, the authors talk about the victimization Moving Target Defense [19], the researchers projected a DDoS traffic mitigation framework that is given to support the Cloud-Fog Platform victimization the MTD technique (CFPM). CFPM applies the migration MTD technique at the fog layer to mitigate DDoS attacks within the cloud. It detects attackers among all the legitimate shoppers proactively at the fog layer and isolates it from regular shoppers. Finally, they use CFPM that has good request processing methods for load equalization and assailant segregation procedure that aims to attenuate interruptions to cloud servers also as serving fog servers. What is more, within the paper.

This paper focuses on the mechanism for low-rate [25] DDoS problems and the author then offers a DDoS attack with many features and a Factorization Machine-based detection technique. The information gleaned from the flow rules is utilized so DDoS attacks with a low rate of occurrence can be detected and then the use of FM machine learning techniques to identify low-rate DDoS attacks has been deployed. This paper also got their experimental result which shows that the approach detects low-rate DDoS attacks on the SDN data layer well, and the maximum level of detecting accuracy is achieved. Finally, this research suggests a technique of protection based on the Deletes flow rules dynamically and performs experiments to demonstrate the efficiency of the defensive method, simulation and analysis are used, and the success rate of forwarding normal packets reaches the top.

Shidaganti, Inamdar, Indhuja V. Rai in their paper proposed a system known as SCEF: A model for a bar of DDOS attacks[24] from the cloud, the authors projected a system known as SCEF, wherever the systems check the network is vulnerable or not. The author likes Wireshark and dynamic thresholds for police investigation of the packets. The authors have shown the simulation of protocol SYN flood, DNS reflection, SSH Brute Force attacks, and ICMP flood attacks. The simulation will scale back the victim VM.

In another paper, the researcher observes DDOS attacks, and also the value is low in distinction, within the paper, police investigation uneven application-layer DoS Attacks with the use of FINELAME [10], authors introduced a language-independent framework known as “Finelame” to observe uneven DoS attacks. If a low-level resource is required to cause a considerably higher level of failure of a system that’s known as an associate uneven attack. They collected knowledge to coach a model that utilizes every and each individual request for a period of time. Then resource monitors use the model parameters to catch sinning requests in the time period, keeping with resource allocation.

Some researchers talked about the internet of things matter [6], they claimed the data number is increasing on a daily basis as a result they proposed a new system to detect this problem automated. In their paper, they show that by implementing IoT-specific network characteristics which have a selected number of starting times and the duration time between the passing packets. by using these results the problem of the attack can be measured. These findings suggest that residential gateway routers are useful Other network middle boxes might identify local traffic automatically and flow-based and protocol-independent traffic data.

In another paper[7], the authors propose a detection framework to tackle the efficient analysis of flooding DoS attacks by using tools like MapReduce and HDFS. The author enforced a counter-based DDoS detection rule for major flooding attacks such as, TCP-SYN, HTTP GET, UDP, and ICMP in MapReduce, which is made up of map and scale back functions. The framework is totally automatic, tracing the logs, sending them to the Hadoop detection cluster for starting the detection mechanically. This system consists of major phases and that they square measure Network traffic analysis and log generation, log transfer, DDOS detection, and finally, giving an output of results based on this information. Every part of the above mentioned phases is enforced as different parts that communicate with one another to perform their appointed task. Traffic capturing and log generation square measures are handled at the detection server and DDoS detection and result notification square measures are performed by the detection system. Log transfer is handled through internet services. The author uses the T-shark library to capture live network traffic.

In this paper, the authors[27] performed three methods which are the complication of the neural network. Another one is deep learning which is mainly a dummy and finally, they talk about the short and long-term memories. In their paper, they show the probability and the ability of their methods to reduce the attacks. The act of their proposed system is judged on the basis of the accurate rate, recollection of the data, and the original anti-positive amount. Moreover, this paper focuses on flood assaults against the controller using TCP, UDP, and ICMP and they contrast between machine learning and deep learning. The authors find out the RNN LSTM which is given the best result to detect the attacks among all the detection methods. Finally, they show the split percentages of their models which are the most.

The purpose of this study is to detect attack traffic[23] by focusing on the SDN's centralized control feature as the control layer in unsafe DDoS attacks. The researcher found that for identifying malicious traffic in the SDN area, multiple machine learning approaches are being used for these reasons It's difficult to choose relevant characteristics and accurate classifiers for attack detection. For better detection accuracy, in this work, kernel principal component analysis combined with genetic methods aids the Support Vector Machine. The authors proposed an SVM model and KPCA which is used for reducing the dimension of feature vectors and GA which is used for optimizing different SVM parameters. An improved kernel function is presented to decrease the noise produced by feature difference. The findings of the experiments reveal that, when compared to a single-SVM model, the suggested model exhibits greater generalization and classification accuracy. Furthermore, the suggested model may be implemented inside the controller to build security rules that would restrict attackers from launching attacks.

In this paper, the author[20] suggests that the ISP implement a dynamic learning system. The dynamic learning system is a complete autoencoder-based unsupervised ensemble model to separate the attack. This CA always gets the attack whereas the autoencoder can work only when the CA is active. When the expected quantity of regular IP addresses has been exhausted The CA switches the IP address classes for 50 percent of the total IP addresses.

Chapter 3

Methodology

In this section, we will go through the methods or in other words, the procedural steps which were undertaken by us from the beginning of our research as well as the steps that followed. We will also be analyzing some methodologies of existing DDoS mitigation techniques, which has refined our understanding. Analyzing previous methodologies of DDoS mitigation will unfold our motivations towards the actions that were performed. Also, it will help to understand the paths not taken by us. It is noteworthy that this sort of comparative analysis is what helped us to get to the point we are now, hence its significance.

3.1 Methodology of our research:

The first step is very straightforward and simple. We went through papers of existing research on DDoS mitigation and we looked into the future works put forward by the respective authors. We looked at the cost-benefit ratio of these solutions and selected mitigation to work on which increases the benefits of the cost-benefit ratio. At the same time, it was in the back of our head that the finalized solution has to solve a wide range of problems.

3.2 Methodological analysis of previous research:

3.2.1 SCEF

The methods employed by the researchers were to implement a monitor VM in a network of VMs to collect and monitor network traffic in and out of the VMs[24]. Until there was no attack detected, the component that was responsible for the mitigation was not activated and the system is seemingly invisible. When an attack was detected, the mitigation method consisted of taking two parameters as input to detect the source of the attack and the victim VM.

The threat neutralization method employed isolating the victim VMs. Once isolated the victim VM could not be used as a zombie VM in a botnet. Also the mitigation method uses different techniques for mitigating different kinds of attacks. In such an attack named DNS reflection attack, the mitigation method includes taking the source IP, which causes a security issue. Furthermore, this issue has not been solved by the authors.

3.2.2 Hadec

The first step is capturing the live traffic using MapReduce. The traffic log file is then sent to their own HDFS storage where this log file is analyzed[7]. T-shark open source library is used here to analyze the log file to target the foul requests among the log file. After that the Hadoop cluster deletes the malicious requests from the log file. This is how their framework works.

This method of detection does not always work in the case of low-rate DDoS attacks. The MapReduce function takes more time than usual to send notifications to the other two components in case of low-rate DDoS attacks. And as the volume of the traffic climbs higher, the detection rate drops. As well as the time to detect becomes less efficient.

3.2.3 Low Rate Detection

FM algorithm, a defense methodology supported dynamic detection of flow rules. Firstly, in an extreme system, low-rate attacks against the knowledge layer, though DoS attacks are also distributed invisibly victimization low-rate data[25]. It is in charge of constructing packets to forward rules for the entire network. Each animal sends the identical packet to the switch with the real address. The flow rule is the map to the switch once it gets a replacement packet.

Secondly, in feature extension of low-rate attacks in software distributed network, the target of low-rate DDoS attacks on the data layer in SDN or software distributed network is to run through the flow table resources, Over the duration of time, the flow rule refers to the number of packets that are matched to the complete kind of packets. As a result of the attacking flow, the rule ought to be constantly matched in an extremely low-rate DDoS attack against the knowledge layer.

Thirdly, DDoS attacks detection supported FM, here the feature combination technique is developed throughout this study using the FM methodology to work out the correlation between each feature sample, resulting in a great deal of actual featuring samples being used to updating parameters, enhancing the detection rate, fundamental quantity detection of attacking flow rules, and provides dependable criteria for defending against low-rate. Next, a defense methodology supported dynamic detection of flow rules, this analysis provides a defensive strategy for dynamically removing flow rules, that would be a defensive mechanism against DDoS attacks at the knowledge layer, based on this theoretical foundation. This method can't tell the difference between genuine and malicious communications. Moreover, Malicious traffic cannot be eliminated and the address of this attack is hard to find out. As a result, this is not that much effective to detect the mitigation.

3.2.4 Assymetric DDoS detection Using Finelame

If a low-level resource is needed to cause a significantly higher level of failure of a system that is called an asymmetric attack. The authors, introduced a language-independent framework called “Finelame” to detect asymmetric DDoS attacks. Their framework Finelame uses three components to detect DDos attacks. It's made for communicating with modern distributed systems, runs orders of magnitude faster than previous techniques, and can detect attacks on an application that haven't been seen before.They collected knowledge to coach a model that utilizes every and each individual request for a period of time. Then resource monitors use the model parameters to catch sinning requests in the time period, keeping with resource allocation. The framework is less optimized than the two battle-tested Apache and Node.js.

3.3 Implementing previous research and their drawbacks:

e are simulating DDoS attacks, testing the tools we're using to simulate these attacks, testing the servers and various web building technologies to find the best suited from these for our tests as well as finding the standards against which to measure these. These have been explained in detail in the testing and analysis section of our paper. After the analysis of the effects of these test attacks on various types of databases, our focus is narrowed down. Furthermore, the methodological analysis of previous systems helps us to be more confident in our approach.

There are many DDoS detection and defense mechanisms which claim to filter out the malicious requests from the legitimate requests but this mechanism is not full proof. Zhijun, Qing, Jingjie, Meng, Liang in their paper[10] mentions that their mitigation method cannot always distinguish between legitimate and malicious traffic. This can cause a drop in user experience which leaves a huge impact on any company. The system Hadec[7] can not detect slow-rate DDoS attacks in its live detection method. Shidaganti, Inamdar, Rai, Rajeev's paper[24] comes up with a solution that deals with a wide range of application layer DDoS attacks. Yet it raises a security problem of its own while tackling the DNS reflection attack.

3.4 Summary of drawbacks of existing security measures:

As discussed previously, a typical DDoS attack will send multiple simultaneous requests to a server, which generally seems legitimate to the server. The servers are overwhelmed with these sorts of requests when finally, it reaches its peak capacity and finally crashes. Recent history of large-scale DDoS attacks has taught us that no matter the capacity of the server, it can be overwhelmed. In 2020 alone, two of the giants of the cloud computing industry, Google and AWS have been under attack by large scale DDoS attacks[17]This goes to show that DDoS attacks can victimize even those most prepared to neutralize it. Moreover, the most successful DDoS defense services providers use their large capacity of servers to distribute the load of requests to handle malicious requests which can not be filtered but these companies charge a lot of money for their services because creating such a network of servers costs a lot of investment in infrastructure. In addition, these defense mechanisms do not always work as expected. In short, there are many filtering, detection and defense mechanisms which work to some extent but they do not always work.

In 2018, GitHub was under attack by one of the largest DDoS attacks in history [17]. Specialists say that GitHub had already implemented many defense strategies that employed mechanisms which could have prevented an attack even 4 or 5 times larger in volume. The findings methodological analysis is summarized as follows:

- Malicious requests can be filtered to some extent but no filtering method is 100
- The victimization of industry giants prove that a dire need of neutralizing the threat at the very core is desperately needed.
- DDoS attacks only become a considerable threat when volume of requests per second topples the target servers computing capacity but the total number of requests is not always a key threat.
- Finally, an universal solution has to be created with

  -Cost-neutrality in mind

  -It has to be a large enough eco-system of nodes where each node is interlinked

  -All the nodes have to support each other once a threat has been detected.

3.5 Our proposal:

Looking at the facts that DDoS attacks are evolving and adapting, made us realize that we need to combat DDoS attacks at the very core of what makes it dangerous. This realization is what led us to look into a universal solution to DDoS attacks.

A universal solution means creating a network which can share the load of DDoS attacks on any of the nodes of the network, so that any single node can not be overwhelmed with requests. Finally, this network needs to be monitored and maintained by all the nodes of the network which is the only way that such a vast network can exist. This model increases the cost-benefit ratio in favor of benefits. Also it combats a wide range of DDoS attacks. Our inspiration for creating this network came from a project which was used in the research of COVID-19 vaccine. The project boasts a massive 165,000 nodes, 6.8 million cores and 50,000 GPUs on its website[28]. These specifications are more than enough to combat the largest DDoS attacks in history. Our proposal is simply creating a similar network for the purpose of spreading out the number of requests in a large span of time, which may be the only universal solution of DDoS. Meaning we want to create a network which spreads out the number of requests once an attack is detected but also employs the filtering techniques to filter out as many requests as possible.

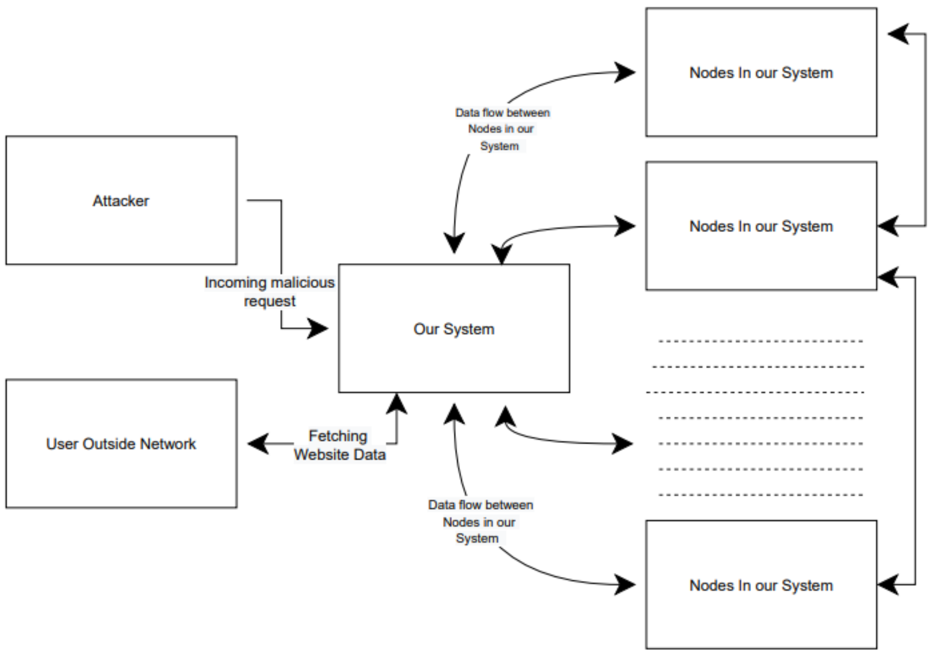


Figure 3.1: Visualization of data Flow in our System

3.6 Methodology of our system:

DDoS attacks are seemingly authentic requests sent by a malicious user for spiteful purposes. So the point is, these are not viruses which spread in seconds within contact with a machine but in most cases of DDoS attacks can not do more damage than taking the server offline for a certain period of time, if not followed up by another type of cyber attack. From our research, we have drawn the conclusion that the only way a DDoS attack can cause damage is by using up the target servers' computing resources. By managing to prevent the target server from receiving all the requests all at once, we can therefore render all sorts of DDoS attacks useless and unable to damage the system in any way. The service may slow down for a little bit but with time, a system can be produced that is developed enough to reduce the overhead.

It is important to note here, the total number of requests is not always enough to take down a server. When the number of requests per second climbs beyond the servers' capabilities, that is when a server fails. We are aiming to devise a strategy that takes the load off the server in a way that the number of requests per second is decreased. In other words, we are not aiming to decrease the overall total number of requests sent by the attacker. Although, we are using an existing DDoS detection and filtering technique in our tests. This filters out more obvious malicious requests. To sum up, our method of combating DDoS is a method that works at the very core of what makes DDoS attacks dangerous, volume of requests per second.

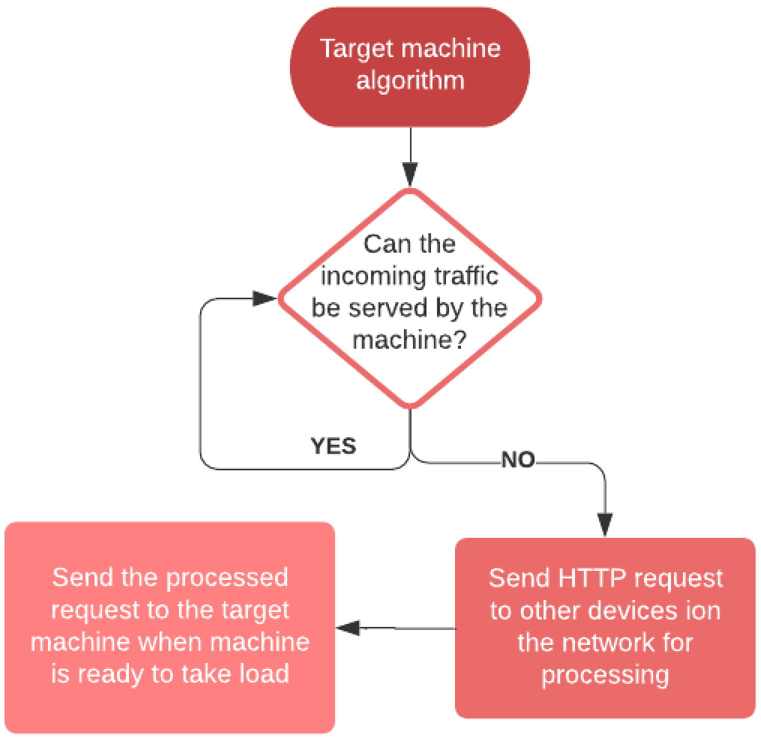

Target machine algorithm
Can the incoming traffic be served by the machine?
YES
NO
Send HTTP request to other devices ion the network for processing
Send the processed request to the target machine when machine is ready to take load

Figure 3.2: Target Machine Algorithm

Chapter 4

System Design

System design is mainly describing a system with the help of graphs, flowchart, model architecture, equation, etc. By system design, we can easily understand how the system is working and its processes. System design helps us to get the core knowledge of the system and its components. It also gives the simulation and data by using those we can assume or predict the result.

In this section firstly we described some of the systems of other authors how they design their system and also the work method and lastly, we talked about the system design of our proposed system

4.1 System design of previous systems

We started by analysing the system, individual components of those system and how the components works for DDoS detection and mitigation thus we came through some exciting new methods and techniques for tackling DDoS.So we chose some of the existing papers which are closely related to our proposed topic and started to analyze their papers and understand what they tried to implement in their work.
After analyzing their system and related components we find some drawbacks in their system design which can be fatal for security concerns. To handle the drawbacks and problems of the previous system we required to list the drawback of the system and thus we thought of implementing a new system of our own that can be a much more efficient method to handle the drawbacks and DDoS attacks.

4.1.1 DDoS Detection by deep learning

In this paper, the authors tried to implement an IDS-based security mechanism to automatically notify the administrator about any harmful activity[26]. For that purpose, a lightweight attack recognition system with a deep learning method is suggest to arrange simple and attack details. The mechanism consists of a data-set and algorithm. In the data preparation and prepossessing module min-max formalization is executed which falls under the concept, transformations that are linear by portraying the allocated value.
Analyzing Based Whale Optimization Algorithm (shortly FS - WOA) is an algorithm that can be compared to a whale's hunting behavior because it selects the best

feature sets. Then those feature sets reclaimed from FS-WOA are added as inputs of the DNN(Deep Neural Network). The output function acts as the output of the entire system. The final result is set on by reducing the error function getting the network model. when error reaches its lowest point the network automatically closes the iteration and starts to protect the sensitive data in the cloud database.
Then that sensitive details acquired from the classified algorithm is saved in the cloud storage after securing by doing encryption and further prepossessed.The safety process is done by using homomorphic encryption algorithm. For preventing further attacks the cloud service provider(CSP) checks all access requests. Upon getting a request, the CSP asks for the verifying process. If the user gives the correct information which is required for verifying then the CSP gives access to the iser; otherwise, it will simply reject the whole process.

4.1.2 Hadec: Hadoop based DDoS Detection

In this paper, the authors work on four major phases for higher results and conjointly the square of the system measure network traffic capture and generations of log, transfer of log, half detection of DDOS, and notification of result[7]. Every of the phases is enforced as separate components that communicate with one another to perform their own assigned task. Among the initial half, an online interface is provided by HADEC where the capturing server can be tuned by the admin with wanted parameters.
Throughout this half, the live detection process starts in operation to capture traffic networks. The echo class gets the property file from the traffic handler and starts capturing the live network whenever the admin is completed with configuration.
During the second half (log transfer), once the file of the log is generated, the handler of the traffic will notify the server for detection and share the information with it via an online service. There are two interfaces this half capturing server has, one is for incoming traffic and another one is for human activity to the server of detection. The detection server in the main works to transfer log data files from native storage to HDFS.
Moreover, to require care of a healthy storage capability both the servers delete the precise file from their native storage. Thirdly, their square measures two core components which consist of an Apache server. Throughout this technique, Name Node disjoints the knowledge into identical size big blocks and allocates them among nodes of the cluster. The method transfers pre-packed codes for nodes to the technique in parallel.
Finally (in the result notifications), when the method task is finished the result will save in HDFS, and once the result is notified every input and output folder square measure progressing to be deleted for higher memory management. r square measure progressing to be deleted for higher memory management.

4.1.3 Low-Rate DDoS Detection Based on Factorization Machine

The authors worked supported four system designs and conjointly the systems square measure low rate DDOS attack, feature extension of attacks in SDN and detection

algorithm supported FM, a defense methodology supported dynamic detection of flow rules[25].
Firstly, in an extreme system, low-rate attacks against the knowledge layer, though DoS attacks are also distributed invisibly victimization low-rate data. The controller may well be a robust half in SDN. It is in charge of constructing packets to forward rules for the entire network. Each animal sends the identical packet to the switch with the real address. The flow rule is the map to the switch once it gets a replacement packet. Per second, the attackers send out a few dozen packets, but the attack on the button sends out over thousand packets per second, all with wrong addresses.

Secondly, in feature extension of attack the distributed network, targeted on the data layer in SDN software distributed network is to run through the flow table resources, hence this DDOS attack will endlessly transmit packets to occupy the flow table for a lengthy time. Over the duration of time, the flow rule refers to the number of packets that are matched to the complete kind of packets. As a result of the attacking flow, the rule ought to be constantly matched in AN extremely low-rate DDoS attack against the knowledge layer; the attacking flow rule's field value is greater than the legal flow rule's field value. The DoS attack flow is regular of the packet delivery interval and somewhat smaller than the timeout, according to the DDoS attack model provided in the study. The everyday user, on the other hand, will send several data packets per unit time attributable to their objective needs, and conjointly the interval between these data packets square measure progressing to be unpredictable and variable. For this reason, they accept the relative dispersion for his or her detection feature.

Thirdly, DDOS attacks detection supported FM, here the feature combination technique is developed throughout this study using the FM methodology to work out the correlation between each feature sample, resulting in a great deal of actual featuring samples being used to updating parameters, enhancing the detection rate, fundamental quantity detection of attacking flow rules, and provides dependable criteria for defending against low- rate DDoS attack in software distributed network data layer. Because each flow entry samples the input sample, this algorithm can also observe that the attack flow rules are followed precisely, therefore A more fine-grained DoS attack detection can be carried out. Next, the function has been hashed to the desired space of order of magnitude and conjointly the work parameters, the rate of learning. Next, a defense methodology supported dynamic detection of flow rules, this analysis provides a defensive strategy for dynamically removing flow rules, that would be a defensive mechanism against DDoS attacks at the knowledge layer, based on this theoretical foundation.

The strategy relies on police investigation low-rate attacking against the knowledge layer successfully. The flow rule grouping will continue if it's reached seconds. If the timer reaches t seconds, the switch will receive the message of FLOW MOD with the command to delete all flow rules in SDN, reducing the number of flow rules among the switch, then clearing the chip and information SDN.

4.1.4 Detecting Asymmetric Application-layer DDoS Attacks In-Flight with FineLame

In this paper, annotations are superimposed by programmers to purpose once a request is being processed[10]. Even for advanced applications, FineLame entirely needs a few annotations to properly assign resource usage to requests. A multi-layer anomaly detection model that learns ancient behavior and detects attacks as shortly as they depart from it. Request mapping: FINELAME permits programmers to annotate their applications victimization the three request mapping operations therefore as for resource monitors to properly assign usage to the textual matter of invite (start method, attribute request, and processing). Ideally, the annotations have to be compelled to cowl as much of the code base as possible; however, not all resource utilization is attributed to 1 request. In such instances, programmers have a great deal of state in but they map: for true application overhead—rather than request method overhead—utilization is left unattributed, and for shared overhead, utilization is divided or delegated stochastically.

4.1.5 Machine Learning DDoS Detection for Consumer Internet of Things Devices

In another paper on Machine Learning DDoS Detection, the author worked on anomaly detection pipeline, feature engineering, and stateful features. Moreover, the anomaly detection pipeline has four parts which square measure trafficking capture, the groups of packets by time and device, extraction of feature, and classification of binary. In traffic capture, The process of traffic recording is documented at the availability science address[3].All packets have a timestamp, a port, a destination's own IP address, a packet area, and a port. science packets sent from smart home devices. Likewise, in the second half, the IoT device packets square measure separated by offer science address. By timestamps, each device's packets are recorded at the middlebox and separated into non-overlapping time intervals. In the extraction of features, Stateful and stateless choices square measure individual packets which are dependent on the domain knowledge behavior. The stateless choices square measure principally packet header fields, on the hand mixture flow information measures by the stateful choices over within very little time, requiring restricted memory to support on-router activity[29]. Moreover, classify- cation of binary, neighbors of K-nearest, random forests, call trees, vector machines support, and deep neural networks can differentiate normal traffic from DoS attack trafficking with high precision. On the alternative hand, IoT devices square measure distinguished by the limited kind of goals thereupon they convey.

4.1.6 SCEF: A Model for Prevention of DDoS Attacks From the Cloud

The system model suggested throughout this paper detects associated actions once an attack is detected. The SCEF system is effectively clear beneath usual conditions until an associated attack is detected from one or a great deal of VMs[24]. The system is deployed once the collected knowledge from the detector VM triggers an alarm that an associate attack is ongoing.

Once an associate attack is not detected, network traffic flows swiftly as a result of it might within the different network. Once an associated attack is detected, the system filters out the malicious traffic from entering the network, thus protecting the alternative VMs among the network. And simply just in case, any VM among the network is already affected by the attack, the system isolates the affected VM to protect it from being utilized in AN extremely botnet.
The SCEF detects two parameters as input from the VMM for each VM beneath its control-whether the associate attack is current and conjointly the sort of attack being performed. The algorithm that drives the SCEF system can be drawn as for each VM, it receives parameters, whether the associate attack is current and conjointly the sort of attack. If there is no attack, the packet square measure is allowed to own. When the system detects the associate attack and conjointly the attack kind is believed, then the component that works as a result of the mitigation of a section of that attack is activated. For associate unknown attack kind, the default mitigation techniques square measure used.
Finally, The SCEF then returns to being transparent. All of these three systems square measure customary and kind at the very core of the system.

4.1.7 Detection and Mitigation of DDOS Attack on SDN Controllers using Deep Learning

The author experimented with some parameters for this experience. At each switch, the number of packets received is counted, Each flow packet is counted, the IP address of the Source and Destination and the number of packets passed on at each switch is counted[4].
According to the Author, the network's regular functioning is constant, which is the foundation of their anomaly detection and protection mechanism. Moreover, the detection engine is trained off-device, with the model only being exported and utilized on the controller. In their research, a Mininet is implemented which consists of seven switches and eight hosts. Then Flood attacks on TCP, UDP, and ICMP were simulated. UDP, TCP, and ICMP were the three types of traffic created by the hping3 program. The first batch of data was categorized as "usual traffic." Then Hping3 was used to produce malicious traffic for different UDP, TCP, and ICMP floods, which were classified as malicious traffic.
The information obtained was used to create binary classification models using machine learning models. The following key performance metrics were used to evaluate each model's performance: recall, accuracy, true negative rate, and the time spent detecting and mitigating DDoS attacks. In the study, three possibilities were examined. In the first case, 80percent of the data was utilized for training purposes, and the 20percent was used for testing. In the second case, 70percent of the data was used for training and 30percent for testing. Sixty percent of the data was utilized for training in the third scenario, while forty percent was used for testing. Finally, they implemented automatically matching packets based on the detection algorithm's output that was used to construct the defensive mechanism.

To summarize, the key drawbacks of system design of the papers discussed above are:

- Security issues during DNS reflection mitigation due to IP capture. [SCEF]

- Can not always accurately differentiate between legitimate and malicious traffic.[Low Rate]
- Not suitable for both volumetric and low-rate DDoS attacks. [Hadoop]

This tells us that these systems are not always successful in detecting and mitigating DDoS attacks of all sorts. So, it is not that safe to rely on those systems fully because in real-life scenarios, DDoS attacks may consist of both huge amounts of packets or a low-rate attack.
With that in mind, we are designing a system which counters all the aforementioned drawbacks as well as can serve as an universal system for combating DDoS. For this reason, we are setting up a list of minimum requirements for our system, which includes:

- Using a hash function to store or capture IPs, if necessary.
- Works for both slow-rate DDoS and volumetric attacks. In other words, it can combat all forms of DDoS attacks.
- Does not disrupt regular traffic for authentic requests in case the detection technique does not work.
- Reduces data-redundancy of load balancing.
- No central authority. Network is maintained by all the nodes of the system. This removes any influence that any single organization may have on the network and its workings.

4.2 Introduction of our System Design

The system we have designed is based on HTTP request re-routing (Nginx reverse proxy/Nginx load balancing). In previous research, we have seen that one of the many traditional DDoS mitigation techniques is using multiple cache servers to balance the load of the incoming traffic. This mitigation technique causes companies to invest highly in infrastructure, which would be better spent elsewhere.

Our system is based on the same design with one key difference. The difference is that, unlike conventional load balancing methods, the HTTP requests will not be redirected to other servers bought or paid for by the company. This network will be open to use for everyone starting from an individual to a technological giant. All the nodes of the network will have to allocate some computing resources which can be considered an ‘entry fee’ to enter the network. After joining the network, all the computing resources of all the nodes in the network will be used as a giant distributed computer.

Whenever any node of the network detects a DDoS attack, all the malicious requests that are sent to that node will be redirected to other nodes of the network. But no node will be overwhelmed with more requests than it can handle. In other words, only the amount of computing resources that any node has decided to share with the network will be used for this purpose.

What we are proposing is creating an open-source network that is similar to the blockchain network but due to time constraints, we are not able to implement it in an actual blockchain network. So, what we are doing with this system is emulating the blockchain network using Nginx to show the potential of our proposal.

4.2.1 Description of the system

For this system, we're using a demo website that will be hosted in localhost using Nginx. We will be creating multiple VMs (25-500) which will emulate the nodes in the network. These VMs will be used to redirect traffic during our tests. These VMs will not hold any data of the actual website but will work as a buffer so that the actual server from which the website is hosted, does not get more requests than it can handle.

A DDoS detection method has been installed on this server to detect or initially filter out the malicious requests. This detection method only filters out requests which are obvious. In this method, after filtering HTTP requests we use a proxy server to pass the HTTP request. Note that, only those requests that are very much similar to authentic requests will be able to pass through this filtering method. After that, we have used JMeter, Wireshark for detecting HTTP requests. For data collection, we have used the JMeter to gather data and represent it in an MS Excel file and sort them according to packet count, request count, average response time, and median response time. By Request tracking, we monitor how many requests have been received and possibly identify correlations between them. Recently, hackers are creating HTTP requests that most filtering methods can not filter out. For our design, we are assuming that is the majority case.

After the initial filtering, if the server is still about to be overwhelmed with more requests, we are using the 'least connected' method which is built-in Nginx, to redirect the requests from the server. To be specific, within the network the nodes that have served the least of the requests will be forwarded the next incoming request. The IPs of these buffer VMs are stored in the Nginx config files.

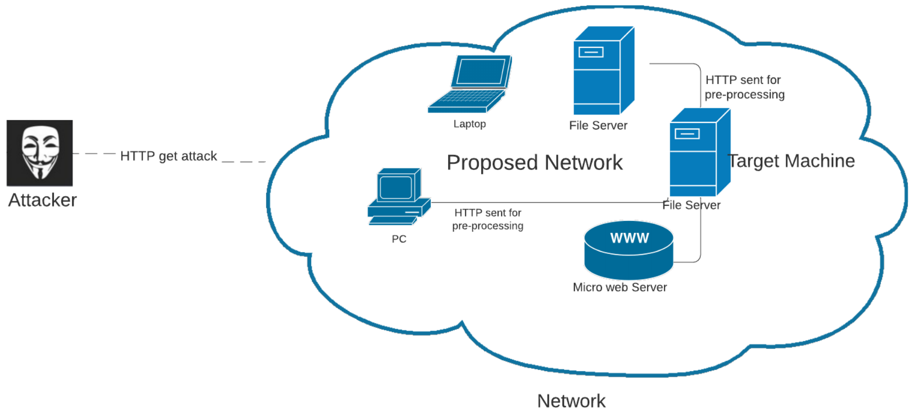


Figure 4.1: Http get attack

4.2.2 Detection used by us

We have started our detection technique by capturing live network from free hosting sites and by Wireshark we captured IP address, packets request, source and destination. Wireshark permits us to filter the log either before the capture starts or throughout analysis. Wireshark is an open-source library that is suitable for detecting DDOS attacks.

After detecting the live network, the information's about packet requests, source and destination send to the detection server. We have used two types of interfaces; among them one is for capturing live network and another one is for notifying the details about the attacks to the detection server. Next, the detection server is started working after being notified and in the server, there are two major roles, one is notifying our proxy server and transferring the detected data's information including IP address, source, destination and the size of the packets to the memory management system. While send IP address's information we use hashing so that we can filter the authentic IP address for example: if a user is trying to login to the server at the same time of attacking and there is no filtering process, the users request will consider as a part of DDOS attack and the server will deny the request of the users. That is why we used hashing to prevent this problem, otherwise there is no outcome of detecting DDOS attack.

Furthermore, the memory management system is started working to make deleting unnecessary files from the storage. After that the detection phase is began to split the attacker request and by using JMeter the detection phase was completed. Whenever we have finished detecting the DDOS attacks, the detection phase was notified to the proxy servers and the memory management system. Next, the disassociate packet is sent to the proxy servers. We have used few proxy servers for handling DDOS attacks, these servers will receive the detected http request, IP addresses etc. as block size and by this time the notification will be sent to every step of throughout the detection.

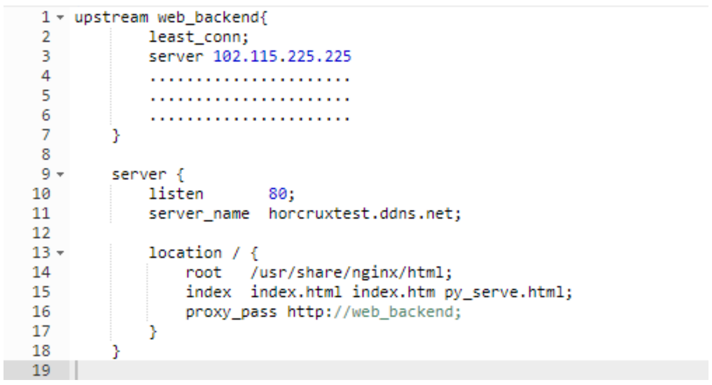
The block size packets are handled by the proxy servers and because of that DDOS attacks are able to detect and handle. There is an infinite loop in the proxy servers and these block size requests is going to fell in the loop and continuously rotating to the servers and the memory management is deleting the attack request by this time. And that is how storage can stay healthy and at the same time DDOS can be detected.

4.2.3 Configuration of VM

Each VM is using Ubuntu 18.04 desktop image. We are using 2 cores and 2048 MB of RAM for each VM.

4.2.4 Configuration of Nginx

We are configuring the Nginx servers to listen to port 80 in each case. The server name after the listen port value is the IP of the demo website. The upstream contains the IPs of the alternative servers meaning the buffer VMs. More upstream IPs are being added as the nodes or the number of buffer VMs is increasing. In the nginx configuration of the VM, the upstream value will be the IP of the website. This configuration is what is making the traffic flow to be redirected back to the original server.



```
upstream web_backend{
        least_conn;
        server 102.115.225.225
        ......................
        ......................
        ......................
    }

    server {
        listen       80;
        server_name  horcruxtest.ddns.net;

        location / {
            root   /usr/share/nginx/html;
            index  index.html index.htm py_serve.html;
            proxy_pass http://web_backend;
        }
    }
```

Figure 4.2: Nginx Code

4.2.5 Topology of our network

In our system, we used mesh topology. Mesh topology is a setup where each computer and network system is interconnected with each other so that if one system faces a problem then another system can recover it easily so it does not hamper or waste any work and time. As we are working with a heavy load of data and if a huge amount of DDoS attack happens we will not lose any data and time for recovery as other systems connected to the network will come to aid with the problem

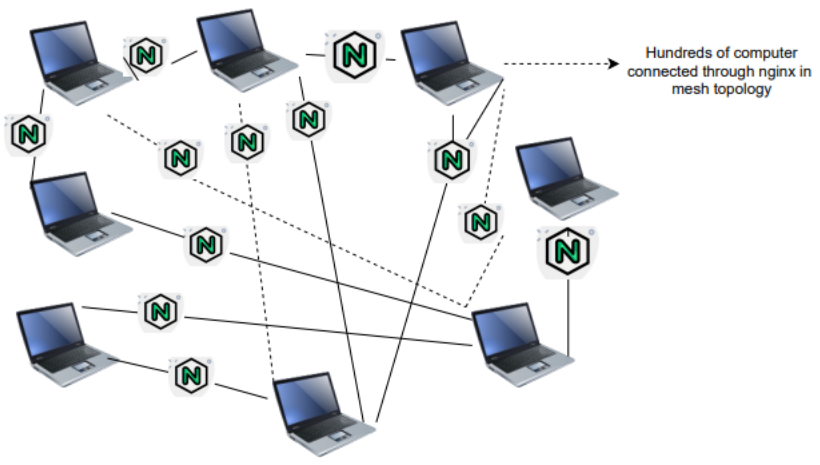

Hundreds of computer connected through nginx in mesh topology

Figure 4.3: nginx in mesh topology

4.2.6 Details of the demo site

We created a demo website to test the performance of the buffer VM network. It is a very basic single-page login page which has not been made too complicated because our focus is on testing the performance of the network, not the website itself. We are serving the website using nginx as mentioned before form our localhost but using a public domain. The issue of not having a real ip at home has been solved by using the 'No ip' website.The features and configurations of the new network:

- Consists of multiple VMs. We start from only 10 VMs to initially create the network. We are increasing the number to a maximum of 500, if the tests require.
- Each VM is configured to detect malicious traffic. The filtering method filters out as many malicious requests as possible.
- Once an attack is detected, the extra load on the particular VM is re-directed via nignx to other VMs in the network.
- The requests that are redirected to the VMs are redirected back to the target server once the load of the DDoS wears off.

Chapter 5

Tools and Components

This part of the report is the description of all the tools that we're using for the tests, deriving the results, and finally analyzing the results. We are describing these tools to the extent of our concern. In other words, we are not describing these tools in full length. Just the parts and significance that concern our work. Some necessary and noteworthy components are also being described in brief including the discussion of its significance. Without understanding the tools and these components that have been discussed here, we would never have been able to follow through with our plan. Let alone, derive the necessary results and the analysis for this report. We are listing these tools firstly, in two groups. 1. Tools used in creating the network. 2. Tools used in testing. The tools that were used for both purposes are mentioned in both places. The VM is used to create the network and the JMeter, Docker is using for the Testing.

5.1 Virtual Machines(VM)

VMs are an emulation of an operating system inside another. The system that contains VM is known as the host operating system and another VM is known as the visitor operating system. At the time VM is put in a host, it simulates the behavior of an actual operating system. A virtual machine is a software-based machine that simulates the functionality of a physical machine. The CPU, RAM, hard drive, and network are the major components of a physical machine, and in a virtual machine, the software makes the operations of these components act as a genuine machine. Virtual machines are usually hosted on or run on physical hardware.

A hypervisor is provided by the hosted physical machine in order for a virtual machine to function. By separating a portion of the CPU, memory, hard disk, and network, assigning these to the virtual machine, the hypervisor virtualizes a physical machine. The virtual machine's OS identifies these components as those used to build a computer and consequently uses them to run the virtual machine's operating system and apps. Next, we will discuss the need for VMs in our tests.

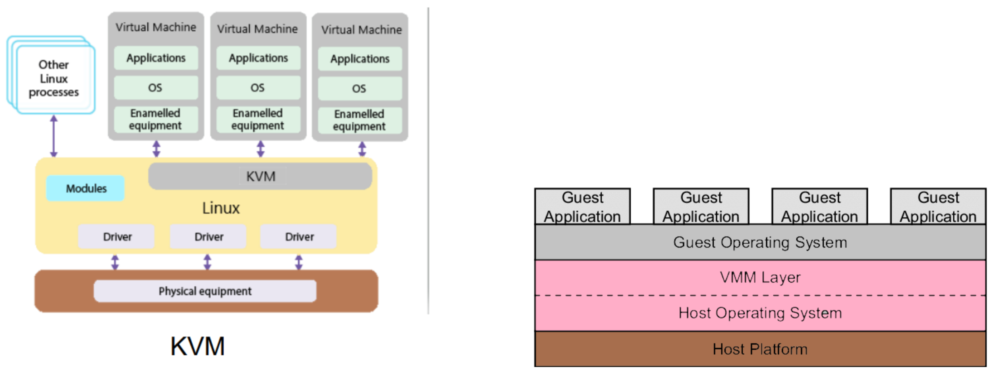


Figure 5.1: KVM vs VM

5.1.1 Virtual Machines simulation

VMs are ideal for creating test environments. It is even better when Linux machines are used for testing purposes. They are usually very lightweight meaning they require very little computational resources to operate efficiently. This feature makes them ideal for test environments as they can be deployed with ease and very quickly. Moreover, multiple VMs can be used in a single machine which allows for more resources to be put to work simultaneously. This multiplies the testing capabilities of our machines.

These VMs are limited to the use case of these tests only and we will not be using this to create our proposed network. Virtual machines can be saved as appliances once the operating system has been deployed and fully configured in them. This appliance can be used by another person and easily copied and pasted onto a different computer.

5.1.2 Description of Virtual Machines used in our tests

We are using Virt-manager to create these VMs which is known as KVM. Unlike other virtual machines which emulate an operating system inside another operating system, kernel-based virtual machines emulate virtual machines based on the kernel rather than on top of another operating system. In comparison to real computers, these KVMs are easier to maintain. On these types of Virtual Machines, the host process is simple to carry out. It saves time and value because numerous devices are created to run on the same machine's hardware. On the same machine, different OS environments can exist and run. The KVM can make several copies of itself on the same machine, reducing the cost of purchasing multiple machines for different activities. In other words, kernel-based VMs give us more control over the physical machine even as we are simulating an OS while testing. This is why KVMs(Kernel-based Virtual Machine) is more useful during testing.

5.2 JMeter

An open-source performance load testing tool, is one of the most widely used. The Apache JMeter program is free and open-source software that is entirely written in Java. This tool is developed by the Apache Software Foundation using the java framework. It was created to test Web applications, but it has now been expanded to include other test functions like Stress testing, Load testing. With an importance on online applications, this tool is mostly used to analyze and measure the performance of a range of services. This tool uses the HTTP and FTP servers for testing. In a normal JMeter test, a loop and a thread group are created. With a predetermined delay, the loop replicates successive requests to the server. This thread group side by side simulates the whole testing. A loop and a thread group are generated in a typical JMeter test. The loop sends consecutive requests to the server with a preset delay. It also offers an API that allows Java applications to conduct JMeter-based tests.

5.2.1 Jmeter in our testing

Our initial test consists of two significant testing phases. One of which is testing the existing server's load capacity. We are testing to find out the capacity which breaks these servers. Jmeter helps to create multiple thousands of HTTP requests within minutes and can create necessary graphs, charts, and tables for direct comparison of these existing servers. We are also testing out some of the existing DDoS mitigation techniques which require a lot of loads to be put under stress. This is where Jmeter shines. Jmeter's simple UI also helps to complete these tests within a short time and the learning curve of Jmeter is not so steep. This ease of learning helped us to focus more time on researching a mitigation technique and figuring out how to simulate it.

5.3 Operating Systems used for testings

Two operating systems were used for our testing, the main server which is an Arch-based Linux distribution named Manjaro another is the popular Ubuntu 18.04-desktop which is a popular and common Linux-based distro. Hence, its use in our testing. We wanted to create test environments that are very similar to real-world scenarios.

5.3.1 Significance of Linux distributions in tests

Linux-based distributions allow access to the hardware more easily which means these OSs can be used to gain more control and access to the actual CPU, RAM, and memory management. This feature gives it a significant advantage to be used for research and development purposes. Linux distributions are operating systems based on the Linux kernel. It's benefitted users by giving them variety. The user has the option of selecting a distribution as their needs. Rolling release distros like Arch Linux can be utilized when a user needs a distro that always has the newest software. Its main function is to make use of time-consuming articles. The number of Linux distros available reduces the amount of time required to learn non-essential information and allows users to focus on the distro's intended function. The finest feature about Linux, in our opinion, is that it allows you to gain a deeper understanding of a system by allowing you to directly watch the machine's statistics and instructions flow. Windows, on the other hand, is more like a BlackBox, blocking out a great deal of information about the computer. Other dominant operating systems such as Microsoft make it very difficult to do the same thing because they have multi-layer protection in place which creates a gap between the hardware and the actual user.

5.4 NGINX

Nginx is a web-server technology that is widely used for load balancing and as a reverse proxy server. It is easy to deploy. Nginx is an open-source free HTTP server that can be utilized as a mail proxy or a reverse proxy server. It can efficiently handle a large number of concurrent users while using minimum resources. When it comes to dealing with large amounts of online traffic, it comes in very helpful. This is likely the only web server capable of handling large amounts of traffic with little hardware resources.
Nginx protects Apache servers by acting as a sort of shock absorber when challenged with security flaws and unexpected traffic spikes. It operates on the Linux operating system and can therefore be called a stand-alone server. It doesn't require a second server, but it works best in situations where there are multiple servers and With a dedicated pair of Load Balancers in front of it, it is highly recommended. Nginx is a lightweight and fast web server that is well optimized and can provide an advantage to some of its applications over hosting with Apache. IIS is capable of hosting more complex websites, such as.net and SharePoint sites. It has a lot of functions, including basic layer-4 load balancing, but it can be improved much more with a dedicated load balancer.

5.4.1 Uses of NGINX

It can handle hundreds of concurrent requests without the strange process and child setup that Apache requires. Some of the uses that we used in our testing by using NGINX are:

- Web Server: It can serve static websites and contains modules that allow it to run PHP, Python, and other programming languages.
- Load Balancer: It can distribute the load among numerous servers running the same application.
- Reverse Proxy: Depending on the desired domain, it can send requests to several servers. For example, Having several domains but only one server.
- API Gateway: Based on the request URI and other parameters, it can route requests to several servers.

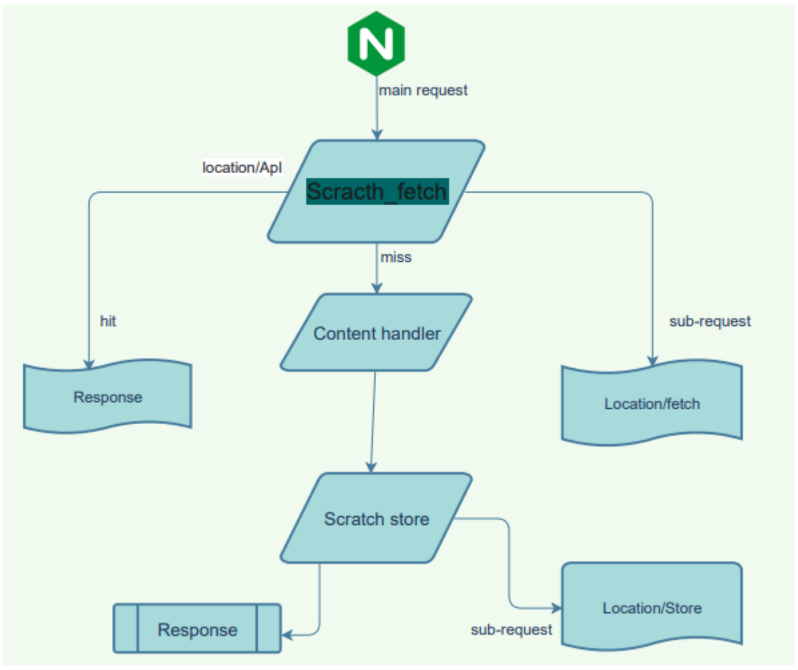


Figure 5.2: Mesh Topology with Nginx

5.4.2 Utilization of NGINX in our tests

We have used Nginx to deploy our demo website from the Manjaro desktop server. We are configuring it to use the default server in case no DDoS attack is detected. But in the case of detection, the configuration file includes an 'upstream' configuration which is basically a list of alternative servers. We are configuring this upstream to take the IPs of the VMs that are designated to be used as buffers. NGINX increases the speed of our website by caching both static and dynamic content replies. When a visitor comes to our website, they are sending a request to the webserver on our hosting account. The web server processes the request, and the desired material is delivered to the visitor.

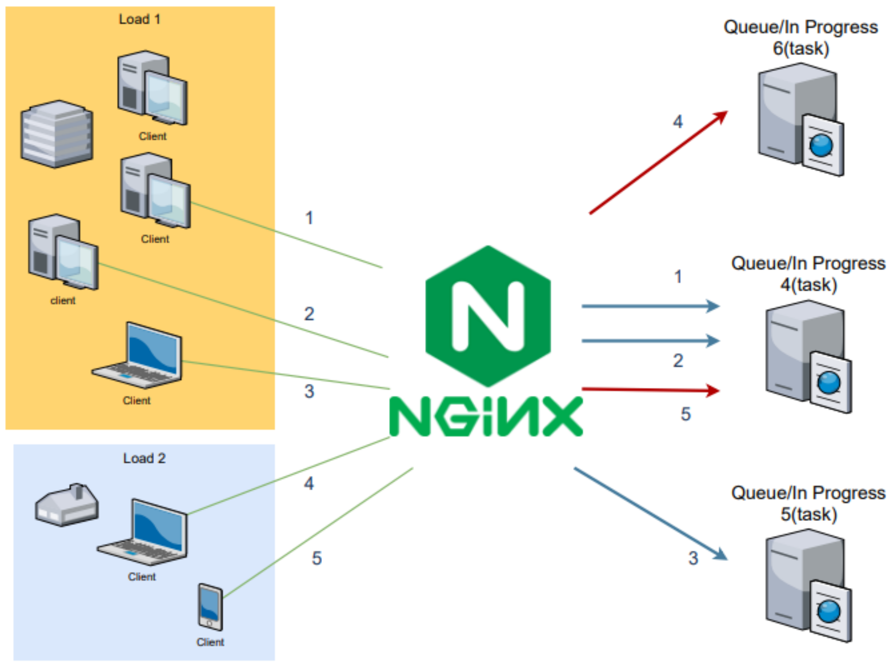


Figure 5.3: nginx least connected method

5.5 Docker

Docker is a service for managing containers. The main point of Docker is to create apps that can be deployed from any location as containers. The Docker platform was created to make developing distributed applications easier as In the cloud, it could be more productive. It is essentially a technology that supports the creation of isolated environments. It is comparable to virtual machines in that it can help us launch operating systems and run programs on them, but it is much lighter. It's the only container-based system available that In a hybrid environment provides an address for all apps. More than a technology, a container platform that provides businesses with solutions to handle mixed problems arising from a variety of resource and information requirements. It removes the software's connection from its background. For example, Isolation between deployments and staging environments. It aids in the resolution of disagreements across teams utilizing various applications on the same infrastructure. Docker now allows infrastructure, apps, and developers to be truly independent.

5.5.1 Uses of docker

Docker is an open-source project that makes developing, deploying, and running applications easier. It allows apps to execute in a lightweight environment. The applications are bundled in a Docker container that includes all of the requirements required to deploy the app. An application can be simply transported from the developer's laptop to the testing environment, and finally to production, using Docker. Since all of the application dependencies are contained within the container, we don't have to worry about whether the application will operate on the production server. As it is lightweight, this consumes only a small portion of the resources that are allocated to them. It is free and open-source software. It offers perfect separation from other programs. Each application is carried out, complete with its dependencies and resources, and therefore each container is isolated from the others.

Tool name	Creating network	Used in testing	Usage	Max output capacity
VM	No	Yes	1. Creating test environment. 2. As a node in the network.	N/A
Jmeter	No	Yes	1. DDoS simulation. 2. Response time, deviation time. 3. Graph generation.	10000 requests/ second
Nginx	Yes	No	Used as load balancer.	N/A
Docker	Yes	No	1. Duplicating test environment. 2. Duplicating VM, nginx configuration.	8 containers per host.

Table 5.1: Summary of most used tools in the tests and network creation.

Chapter 6

Testing and Result

In our final analysis, there are three phases of testing. This section discusses the tests starting from testing out the tools, servers, existing mitigation techniques to testing our mitigation techniques. We are going in-depth and analyzing each test to draw the conclusion of each test in its own perspective.

6.1 Phase 1 (Testing the tools)

We are testing all the tools that we have selected for our testing and development purposes first. We have initially set some metrics for these tools. These metrics are set by us based on some prior assumptions. These assumptions are discussed at the beginning of each test.

- 1 **Jmeter:** We are using Jmeter for the purpose of simulating application layer DDoS. DDoS attacks can range from 1000 users per second to millions of users per second. We are assuming that an average DDoS attack will send at least 2000-5000 users per second. Based on this assumption, we start our tests on the Jmeter itself. The purpose of this test is to find out how many concurrent requests can be sent by Jmeter. We start the tests slowly with only 100 users per second and will be doubled with each test. If we find that for a number of requests, Jmeter is failing, we will go back to the previous number and this time we will not double the number but increase ever so slightly. After repeating this process over and over again multiple times we find that Jmeter can send at most 5000 users per second without crashing. Although there is no limit set by Jmeter, after increasing the number from 5000 to 6000, Jmeter fails to respond. We have taken this as a clear indication that Jmeter can not send more than 5000 users per second. Even if it does send more than that, it will not be an efficient testing environment.

- 2 **Free hosting sites:** The free hosting sites that we've created use the following hosting. One is '000webhost' and the other is named 'zyrosite' and finally 'firebase' from google. We have used basic wordpress templates to create websites to be hosted on the first two of these servers. For 'firebase' we have created an online chatroom application. The purpose for doing this is to test the response time of all these servers under different sorts of DDoS attacks. These attacks will start from only 100 users per second to 5000 users per second. We're using servers from three different categories for the purpose of comparing servers of the lowest capacity side by side with the servers of the highest capacity.

- 3 **Wireshark:** We have used Wireshark to capture network traffic on the native network and Wireshark will capture network traffic from local area networks, Wi-Fi etc. Wireshark permits either before or after the capture begins, we can filter the log and throughout analysis, thus we have a tendency to slender down and that we may simply trace around for what we want within the network trace. for instance, if we have a tendency to set a filter to check transmission control protocol traffic between two IP addresses. We are able to set it solely to indicate to us the packets sent from the PC. What is more, one amongst the most important reasons for exploitation Wireshark is it shows details regarding the chosen packets. It shows the amount order of the packet that got captured, the capturing time is additionally shown by Wireshark. After that, the supply of the packets, the destination of the packets, the protocol of the packets, the length of the packets in bytes and therefore the info of the packets and therefore the actual quantity of packet size in hex value.

UI-load testing: In this phase, To determine the standard behavior, reliability, and peak load of the system we tested two free web-hosted websites which is called UI load testing. We created these two websites using the free hosting service '000webhost'. The first website was hosted in the service's default hosting which is a Word Press based website. The second website was hosted using a different hosting named 'zyrosite'.

- (1st website is https://springtest2021.000webhostapp.com
- And the 2nd website is https://spring2021.zyrosite.com)

Our reason for using these two hosting services are diffrent, electing 000webhost because, it is one of the most common free hosting services and as we explained previously, we want to test all categories of hosting services from the lowest grade to the top tier hosting services and find out the gap between them. That's why our tests began with the most commonly available free hosting service. Conversely, zyrosite is unique in its interface and have the most powerful tools.

The summary of tests (1st website): At first, we tested the 1st website with 100 users or Threads within 1 second. After successfully finishing this we upgraded our threads to 500 users within 1 second but unfortunately our website could not take that load and crashed

Analyzing the summary of the data from the tests on the first website, we can see that the throughput is 8.5/sec for each of the websites. The average response time of the two tests is sequential, 2355ms and 3605ms.

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
000webhost	300	3315	404	21004	5134.62	40.67	4.3
000webhost	100	3419	292	21003	6115.33	100.00	2.5
000webhost	100	457	290	5199	684.93	100.00	1.6
000webhost	100	308	288	600	41.08	100.00	1.6
Total	600	2355	288	21004	4630.53	70.33	8.5

Table 6.1: Summary of 100 users per sec(1st website)

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
000webhost	1780	4731	377	22194	5785.49	67.47	4.5
000webhost	596	5577	292	21259	7741.31	100.00	1.6
000webhost	591	850	290	6033	77.50	100.00	1.6
000webhost	589	963	288	21002	1711.17	100.00	1.4
Total	3556	3605	288	22194	5577.71	83.72	8.5

Table 6.2: Summary of 500 users per sec(1st website)

This means that each of the users was getting a response around 2 seconds later and it increased to around 3.5 seconds as the users count became 500. The deviation is 4630.53 for the first test and 5577.51 for the second test. The difference in the average response time was only around 1.5 seconds. The difference in the deviation of the two tests is 946.98. Analyzing the differences in the average response time and deviation tells us that this website cannot handle traffic of more than 500 users but the user experience does not change that much no matter how many users there are in that range. So, we can conclude that to DDoS this website we'll need to throw over 500 requests in 0.5 seconds which will crash the server immediately and we'll keep the background traffic to 250 users while simulating an attack on this site.

The summary of tests (2nd website): As the 1st website crashed so we proceeded to our 2nd website. Similar to our first step for the 1st website we recorded it in Badboy Software then exported the JMX file to JMeter and started our UI load test again.

At first, we gave the website 100 users within 1 second and it ran successfully. Then we updated the User count to 500 and it also worked fine. So, we upgraded it to 1000 users this time, and the load-testing went smoothly as well. As the load testing is running successfully for our 2nd website, we now upgrade the user count to a massive amount of 5000 users in 1 second. The website was also able to take that kind of load. At last, the website ran down when we tried to upgrade our user count to more than 5000 users. JMeter is stopping at a certain period. The Apache JMeter was crashing so we assumed that our 2nd website cannot take a load of more than 5000 users within one second. Now, we are graphically comparing the tests on all three websites.

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
zyrosite	100	3781	2940	5150	468.28	0.00	18.1
zyrosite	200	0	0	0	0	100.0	0.0
zyrosite	100	18836	10298	25856	3356.05	0.00	3.8
Total	400	5654	0	25856	7948.30	50.0	15.3

Table 6.3: Summary of 100 users per sec(2nd website)

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
zyrosite	600	14229	2323	1071642	61352.29	1.50	25.2
zyrosite	1196	0	0	0	0	100.0	0.0
zyrosite	598	62116	3704	106269	30478.56	0.33	1.2
Total	2394	19082	0	1071642	42729.94	50.42	2.2

Table 6.4: Summary of 500 users per sec(2nd website)

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
zyrosite	1599	271817	1180	1881282	644880.38	19.57	28.1
zyrosite	2753	0	0	0	0	100.0	0.0
zyrosite	1376	100261	3704	1879771	196449.14	27.40	24.2
Total	5728	99964	0	1881282	372035.50	60.11	3.0

Table 6.5: Summary of 1000 users per sec(2nd website)

Label	#Samples	Average (ms)	Min (ms)	Max (ms)	St. Deviation (ms)	Error (%)	Throughput (per second)
zyrosite	1065	62338	12871	163650	61447	66.20	6.5
zyrosite	1558	0	0	0	0	100.0	0.0
zyrosite	1544	9791	286	35129	6882.18	93.59	10.8
Total	4167	19560	0	163650	40356.75	88.98	25.5

Table 6.6: Summary of 5000 users per sec(2nd website)

Analyzing the summary of the data from the tests on the second website, we can see that the throughput is 15.3/sec, 2.2/sec, 3.0/sec, and 25.5/sec consecutively for each of the tests. The average response time of the four tests are, sequentially, 5654ms, 19082ms, 99964 ms, and 19560 ms. This means that each of the users was getting a response around 5.6 seconds later and it gradually changed to 19.08 seconds, 99.96 seconds, and 19.56 seconds as the later tests added more users. The deviation was 7948.30, 42729.94, 372035.50, and 40365.75. The difference between the highest and lowest average response time was only around 94 seconds. The difference between the highest and lowest deviation of the tests was 364087.2.
Analyzing the differences of the average response time and deviation tells us that this website can handle traffic of around 5000 users but the user experience takes a major downgrade as soon as we hit 500 users. So, we can conclude that to DDoS

this website, we'll need to throw over 1000 requests in 0.5 seconds which will downgrade the service but the server will keep running. If we want to completely ruin the server, we'll test it with 5000 requests in 0.5 seconds which may cause the server to be permanently down. We'll keep the background traffic to 250 users as well while simulating an attack on this site, as a site like this in real life would never target to host more than that number. Because if it does, the downgrade in user experience will mean that the site will continuously lose customers due to low user experience.

	Test bed details				Attacker details		
Serial no.	Domain	Average RT(ms)	Standard deviation (ms)	User count	Attack Type	No. of requests	Attack duraiton (minutes)
1	Zyrosite	471	600	100	N/A	N/A	1
2	Zyrosite	5990	27544.95	100	HTTP GET	5604	1
3	Zyrosite	1083	16764	100	HTTP POST	9989	1
4	Zyrosite	1037	10303	100	HTTP PUT	9956	5
5	000webhost	1316	1558.74	100	N/A	156731	5
6	000webhost	19128	26520.71	100	HTTP GET	171331	5
7	000webhost	15949	23571.67	100	HTTP POST	175050	5
8	000webhost	22460	2499.59	100	HTTP PUT	160982	5
9	Firebase	271	344.43	500	HTTP GET	254092	5

Table 6.7: Summary of all UI load tests

6.2 Phase 2 (Live detecting, Traffic handler testing)

In this second phase, we have tested many existing mitigation systems, among them first we have tested in such a way where we captured live detection networks by using Jmeter and Wireshark library. Both are open source libraries and by using them we were able to capture HTTP requests. We also have tuned Jemeter and during the detection phase, Wireshark should only output the information that is requested.
After detecting http requests and IP addresses the admin can forward the information to the next step and that is log transfer. In log transfer, the admin will notify the detection server with the file's IP address, file name, file data path, etc. The traffic handler sends the property file to the echo class and starts capturing the live network whenever the admin is completed with configuration. During the second half (log transfer), The traffic handler will alert the detection server and share file information after the log file has been generated with it via an online service.

Throughout this half capturing server has two interfaces, one is for incoming traffic and another one is for human activity to the detection server. Next, these files will go to the proxy memory server, the proxy server is used to pass the HTTP request by the admin and these requests are going to be deleted by the system. Once the

HTTP request is deleted, the result will be notified to every step because the memory management system can delete the attack request, and also for healthy memory management it will delete the spam requests. Moreover, The SCEF system is effectively clear beneath usual conditions until an associated attack is detected from one or a great deal of VMs.

The system is deployed once the collected knowledge from the detector VM triggers an associate alarm that an associate attack is ongoing. Once an associate attack is not detected, network traffic flows swiftly as a result of it might within the different network. Once an associated attack is detected, the system filters out the malicious traffic from entering the network, thus protecting the alternative VMs among the network. And simply just in case, any VM among the network is already affected by the attack, the system isolates the affected VM to protect it from being utilized in AN extremely botnet.

Furthermore, in this method the method can detect two types of parameters as input from the VMM for each VM beneath its control-whether the associate attack is current and conjointly the sort of attack being performed. The algorithm that drives the SCEF system can be drawn as for each VM, it receives parameters, whether the associate attack is current and conjointly the sort of attack. If there is no attack, the packet square measure is allowed to own. When the system detects the associate attack and conjointly the attack kind is believed, then the component that works as a result of the mitigation of a section of that attack is activated. For associate unknown attack kind, the default mitigation techniques square measure used.

Applying k-nearest neighbors, random forest, binary classification will differentiate normal traffic from Dos attack traffic with high accuracy. On the other hand, we try to achieve in a simple way detecting DDoS attacks through request tracking, analyzing the request stream. In this method, we tried to observe users' HTTPS requests and analyze the request.

By Request tracking, we Keep track of the number of inquiries gotten. and possibly identify correlations between them. Moreover, we have used the metrics for analyzing the results are throughput, average response time, and deviation of response time. Throughput indicates how many requests were successful. Average response time takes a few response times and gives an average. But the key metric will be the deviation of response time because it indicates the difference of response time of each of the responses. A higher deviation or a higher number indicates that the difference of the response that the users are getting is very high meaning, the user experience is very poor. A lower deviation number, on the contrary, indicates a better user experience.

6.3 Phase 3 (Testing response time of websites using our network)

After completing the initial setup of the network we are replicating the VMs using Docker. We then use JMeter to attack the website using the public IP address of the website. We complete the test using 10 VMs and users in JMeter are 5000 with other configurations and settings set to previous values. We keep JMeter configurations the same throughout the tests and increase the number of VMs gradually by 10. The second test is conducted using 20 VMs, the third test with 30 VMs, the fourth test with 40 VMs and lastly the final test is for 50 VMs. This process is repeated for two other websites we tested previously. Now we are comparing the final tests using on our network using all three of the websites. The metrics that are being used is response time over time.

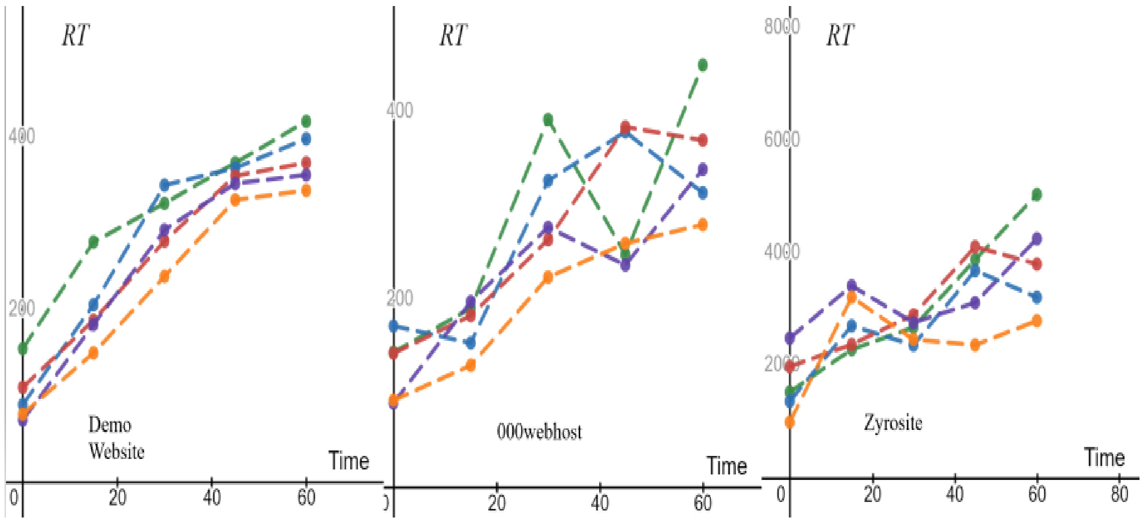


Figure 6.1: Graphical comparison of final tests on all three websites

Finally, we are comparing the final test results side by side with the test results on the websites derived without using the network. The first bar is the response time of the respective website with DDoS simulation without using our network. The second bar is the response time of the same website with DDoS simulation using our network. We can see a significant improvement of response time for each website using our network.

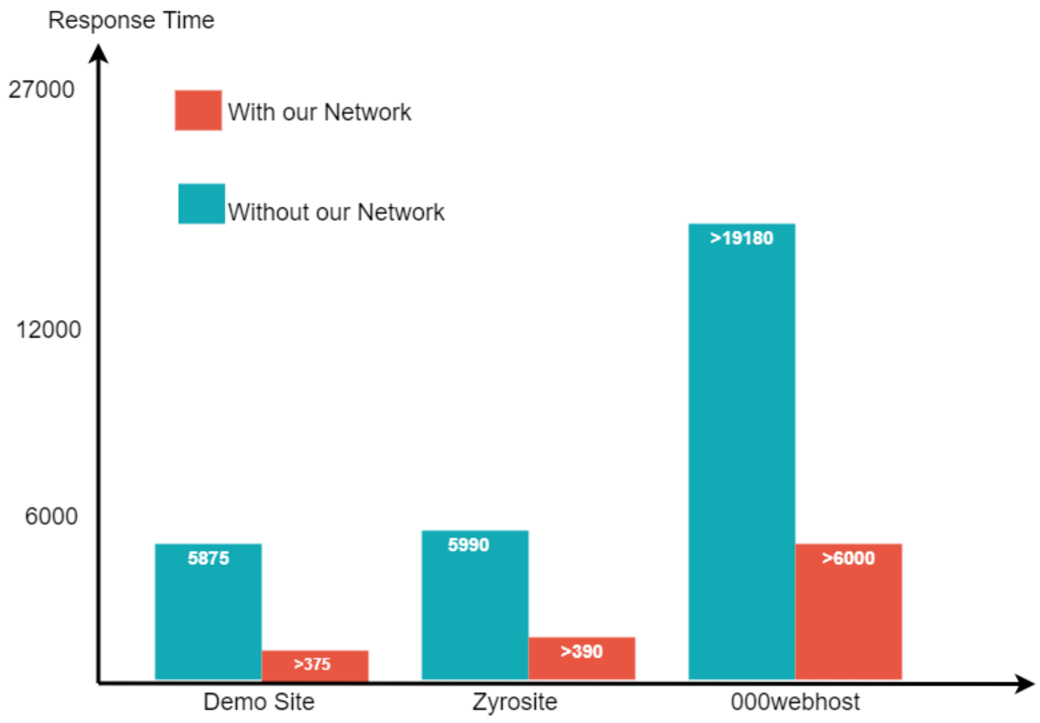


Figure 6.2: Testing Result comparison

6.3.1 Analysis of findings

For the tests on the first two sites, the average response time is less than 400 ms. For the tests on 'zyrosite', the response time is greater than 6000 ms. Meaning, only for zyrosite, the DDoS attack was successful. For the rest of the tests, the proposed method was successful. We conducted a total of 15 tests on all three websites using our proposed solution. A total of 5 tests failed to mitigate DDoS in our case. In other words, the success rate of DDoS mitigation in our tests is 66.67%. We need to keep in mind that, we used nginx to complete the tests because the technology for creating our proposed network does not yet exist. Moreover, the configuration of nginx is not optimized for our proposed network. So, we can hope for greater results in future if we can create an optimized solution for this network.

Chapter 7

Conclusion

Cloud computing is becoming more popular, but with increased cloud usage comes to the challenge of cloud security. The Distributed Denial of Service Attack (DDoS) or simply Denial of Service Attack is one of the most serious risks to Cloud security (DDoS) and a threat to thousands of business. Money, time, clients, and even credibility may be lost in the case of DDoS attacks. It is critical to provide a strategy to avoid DDoS attacks in order to increase resource availability. The detection trigger technique may efficiently identify abnormal flow occurrences and conserve controller resources. DDoS attacks may also be efficiently mitigated with the defense plan in place. It is high time that the DDoS, a name of terror for many businesses and services, be nipped in the bud. We need to focus our attention, our resources and efforts towards this goal. We hope our mitigation technique can be the guideline for future works.

7.1 Future work

In this paper we initially planned to test our system using 500 VM but we could only test the network using 50 VMs. In the future we are going to increase the number of VMs for our testing purpose and we believe that the results that we get will be enough for the network to be deployed in a LAN at least. Moreover, we have planned for using 500 VM but for real life scenarios, meaning for networks larger than a LAN, we need to test the network using at least few thousands VMs and analyze the results but again due to time constraints we could not make it. So, in near future we will further test our system using the specifications mentioned above.

In addition, we have planned to build a mitigating system that is based on a ledger system which can be something like a block chain network. The resources that any single node shares with the whole network can be an entry in the ledger system. We predict that our system combined with the block chain technology can become the ultimate-universal solution for DDoS mitigation, although the network will have to be working faster than block chain networks work today.

Bibliography

[1] B. B. Gupta and O. P. Badve, "Taxonomy of dos and ddos attacks and desirable defense mechanism in a cloud computing environment," *Neural Computing and Applications*, vol. 28, pp. 3655–3682, Apr. 2016. DOI: 10.1007/s00521-016-2317-5.

[2] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "Ddos attacks in cloud computing: Issues, taxonomy, and future directions," *Computer Communications*, vol. 107, pp. 30–48, Jul. 2017. DOI: 10.1016/j.comcom.2017.03.010.

[3] M. Zekri, S. E. Kafhali, N. Aboutabit, and Y. Saadi, *Ddos attack detection using machine learning techniques in cloud computing environments*, IEEE Xplore, Oct. 2017. DOI: 10.1109/CloudTech.2017.8284731. [Online]. Available: https://ieeexplore.ieee.org/document/8284731 (visited on 06/12/2021).

[4] K. Bhushan and B. B. Gupta, "Distributed denial of service (ddos) attack mitigation in software defined network (sdn)-based cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 1985–1997, Apr. 2018. DOI: 10.1007/s12652-018-0800-9. [Online]. Available: https://link.springer.com/article/10.1007/s12652-018-0800-9.

[5] M. V. De Assis, M. P. Novaes, C. B. Zerbini, L. F. Carvalho, T. Abrãao, and M. L. Proença, "Fast defense system against attacks in software defined networks," *IEEE Access*, vol. 6, pp. 69 620–69 639, 2018. DOI: 10.1109/ACCESS.2018.2878576. [Online]. Available: https://ieeexplore.ieee.org/document/8514012?fbclid=IwAR0BwrGkD_HVCQsEylYMB_cFzb5NQrOx6mzz4Dum9BdEL9F1qjWtY
BA (visited on 10/03/2021).

[6] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35. DOI: 10.1109/SPW.2018.00013.

[7] S. Hameed and U. Ali, "Hadec: Hadoop-based live ddos detection framework," *EURASIP Journal on Information Security*, vol. 2018, Jul. 2018. DOI: 10.1186/s13635-018-0081-z.

[8] R. Kesavamoorthy and K. Ruba Soundar, "Swarm intelligence based autonomous ddos attack detection and defense using multi agent system," *Cluster Computing*, Mar. 2018. DOI: 10.1007/s10586-018-2365-y. (visited on 10/25/2019).

[9] A. Bakr, A. A. A. El-Aziz, and H. A. Hefny, "A survey on mitigation techniques against ddos attacks on cloud computing architecture," *International Journal of Advanced Science and Technology*, vol. 28, pp. 187–200, Oct. 2019. [Online]. Available: http://sersc.org/journals/index.php/IJAST/article/view/1211 (visited on 10/03/2021).

[10] H. M. Demoulin, I. Pedisich, N. Vasilakis, V. Liu, B. T. Loo, and L. T. X. Phan, *Detecting asymmetric application-layer denial-of-service attacks in-flight with finelame*, www.usenix.org, 2019. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/demoulin (visited on 10/03/2021).

[11] Z. Li, L. Wei, W. Li, L. Wei, M. Chen, M. Lv, X. Zhi, C. Wang, and N. Gao, "Research on ddos attack detection based on elm in iot environment," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 2019, pp. 144–148. DOI: 10.1109/ICSESS47205.2019.9040855.

[12] H. V. Nguyen, L. L. Iacono, and H. Federrath, "Your cache has fallen," *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2019. DOI: 10.1145/3319535.3354215. [Online]. Available: https://cpdos.org/paper/Your_Cache_Has_Fallen___Cache_Poisoned_Denial_of_Service_Attack___Preprint_.pdf.

[13] U. M. Shahil, M. Deekshitha, N. Anam M, and M. Basthikodi, *Ddos attacks in cloud computing and its preventions*, papers.ssrn.com, May 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3708159.

[14] K. Srinivasan, A. Mubarakali, A. Saad Alqahtani, and A. Dinesh Kumar, *A survey on the impact of ddos attacks in cloud computing: Prevention, detection and mitigation techniques*, springerprofessional.de, 2019. [Online]. Available: https://www.springerprofessional.de/en/a-survey-on-the-impact-of-ddos-attacks-in-cloud-computing-preven/17060314 (visited on 10/03/2021).

[15] A. R. Wani, Q. P. Rana, U. Saxena, and N. Pandey, "Analysis and detection of ddos attacks on cloud computing environment using machine learning techniques," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 870–875. DOI: 10.1109/AICAI.2019.8701238.

[16] M. Yan, R. Sprabery, B. Gopireddy, C. Fletcher, R. Campbell, and J. Torrellas, "Attack directories, not caches: Side channel attacks in a non-inclusive world," *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019. DOI: 10.1109/sp.2019.00004. (visited on 10/03/2021).

[17] Cisco, *Cisco annual internet report - cisco annual internet report (2018–2023) white paper*, Cisco, Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[18] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, pp. 876–889, Jun. 2020. DOI: 10.1109/tnsm.2020.2971776.

[19] V. Kansal and M. Dave, "Proactive ddos attack mitigation in cloud-fog environment using moving target defense," *arXiv:2012.01964 [cs]*, vol. V1, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2012.01964 (visited on 10/04/2021).

[20] I. Ko, "Adaptable feature-selecting and threshold-moving complete autoencoder for ddos flood attack mitigation," *Journal of Information Security and Applications*, vol. 55, Oct. 2020. DOI: 10.1016/j.jisa.2020.102647.

[21] G. S. Kushwah and V. Ranga, "Voting extreme learning machine based distributed denial of service attack detection in cloud computing," *Journal of Information Security and Applications*, vol. 53, p. 102 532, Aug. 2020. DOI: 10.1016/j.jisa.2020.102532. (visited on 06/04/2020).

[22] Z. Li, H. Jin, D. Zou, and B. Yuan, "Exploring new opportunities to defeat low-rate ddos attack in container-based cloud environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, pp. 695–706, Mar. 2020. DOI: 10.1109/tpds.2019.2942591. (visited on 03/13/2020).

[23] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, "An evolutionary svm model for ddos attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132 502–132 513, 2020. DOI: 10.1109/ACCESS.2020.3009733.

[24] G. I. Shidaganti, A. S. Inamdar, S. V. Rai, and A. M. Rajeev, "Scef: A model for prevention of ddos attacks from the cloud," *International Journal of Cloud Applications and Computing*, vol. 10, pp. 67–80, Jul. 2020. DOI: 10.4018/ijcac.2020070104. (visited on 10/14/2020).

[25] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate ddos attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17 404–17 418, 2020. DOI: 10.1109/access.2020.2967478. (visited on 02/02/2021).

[26] A. Agarwal, M. Khari, and R. Singh, "Detection of ddos attack using deep learning model in cloud storage application," *Wireless Personal Communications*, Mar. 2021. DOI: 10.1007/s11277-021-08271-z. (visited on 09/19/2021).

[27] D. Gadze, A. Bamfo-Asante, J. Owusu Agyemang, H. Nunoo-Mensah, and K. Opare, "An investigation into the application of deep learning in the detection and mitigation of ddos attack on sdn controllers," Feb. 2021. DOI: 10.3390/technologies9010014.

[28] K. Moskvitch, *Covid-19 hpc consortium*, localhost, Apr. 2021. [Online]. Available: https://covid19-hpc-consortium.org/?fbclid=IwAR0fus4vQxBfs8rufv30AxIf96hwcA3ro7I (visited on 10/04/2021).

[29] M. Pinho, *Aws shield threat landscape review: 2020 year-in-review*, Amazon Web Services, May 2021. [Online]. Available: https://aws.amazon.com/blogs/security/aws-shield-threat-landscape-review-2020-year-in-review/ (visited on 10/04/2021).