SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Load Balancing Approach to Mitigating the Impact of DDoS Attack

Fatima Enehezei Usman-Hamza[1], Ghaniyyat B. Balogun[1], Shaibu Muhammed Etudaye[1], Shakirat Aderonke Salihu[1], Peter O. Sadiku[1], Rafiat A. Oyekinle[2], Abdulrauf O. Babatunde[1], and Amos Orenyi Bajeh[1*]

[1]Department of Computer Science, University of Ilorin, P.M.B. 1515, Ilorin, Nigeria
[2]Department of Information Technology, University of Ilorin, P.M.B. 1515, Ilorin, Nigeria
bajehamos@unilorin.edu.ng[*]

[*]Corresponding Author

Abstract

***Background**: Businesses have embraced the burgeoning technology known as cloud computing. This entails the provision of various services over the Internet. Connectivity, availability and security are three major requirements that cloud service providers should always ensure. The cloud computing environment faces several difficult security concerns one of which is the distributed denial of service attacks. Attacks that cause a denial of service puts availability at risk. Conventional methods of mitigating DDoS assaults fall short due to noticeable server downtime and poor quality of service, users still experienced server downtime and performance issues. Also, existing load balancing algorithm used to mitigate DDoS attack all had a common problem which is single point of failure.* ***Method**: This study proposes a load balancing approach to mitigate DDoS attack while eliminating the problem of single point failure associated with the conventional approaches. This is achieved by using cloud automation tools and providing effective configurations for the servers and monitoring tool. The proposed design was implemented, and 20 scenarios of DDoS attacks were used to investigate the effectiveness of the design.* ***Results**: Upon testing, the results show a reduction in the duration of server downtime whenever an attack happens. The measured time is with respect to the server downtime duration. From previous cases of DDoS attack that some organization experienced, the duration from the attack detection to the attack mitigation was about 20 mins. But with the proposed algorithm, the duration from attack detection to mitigation falls within 5 mins.*

***Keywords**: Cloud computing, Cybersecurity, Distributed Denial of Service (DDoS), Load Balancing*

1. Introduction

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell & Grance, 2011). The cloud computing technology has become the de facto form of web service provisioning. According to Potey, Dhote, and Sharma (2013), cloud computing is not a new technology, but it is considered innovative in a delivery paradigm for information and services. By utilizing the Internet infrastructure, cloud computing links the communication between the client and server-side services or applications. Similar to how internet service providers offer clients high-speed broadband to access the internet, cloud service providers (CSPs) provide cloud platforms for their customers to use and construct their web services. Society relies on the internet for everything from work to entertainment, to financial management, personal data storage, and beyond. The market for cloud computing was estimated to be worth USD 371.4 billion in 2020 and grow at a compound annual growth rate of 17.5% (Sumina 2022). As the user base for the internet grows exponentially, security has become more crucial than ever before now. Also, cybersecurity experts are relied upon to protect user data and CSP infrastructure; they have the advanced skills to anticipate security threats and safeguard confidential information from unauthorized users. Cyber threats come in many different forms, which makes them difficult to combat. Many of these threats prey on individuals through digital mining techniques. These attacks rely on various forms of trickery, from phishing emails that are designed to look legitimate, to bogus online alerts that claim an individual's computer is infected with a virus.

SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Distributed Denial of Service (DDoS) attacks are a potential threat to businesses. In DDoS attacks, cybercriminals flood a company's network or servers with an overwhelming amount of internet traffic. This results in system downtime and poor network QoS.

Several studies have proposed system designs and techniques for mitigating DDoS attacks in cloud environments (Belyaev and Gaivoronski, 2014; Zebari; Zeebaree, Sallow, Shukur, Ahmad & Jacksi, 2020; Abdulkareem & Zeebaree, 2022; Somasundarama, Meenakshi, 2021). However, the downtime experienced by these approaches can be improved upon. This will consequently improve the QoS of cloud computing platform. A load balancing approach is proposed that will switch the roles of servers as request traffic grows. A server monitoring configuration is setup to, at interval, check the health of the servers. When the configured threshold of the monitoring tools is reached or surpassed, it triggers the execution of a script that will switch server's role to contain the DDoS attack. The result of the experiments conducted showed that the proposed system is effective in containing DDoS attacks with little or no significant downtime that can affect QoS.

The remaining part of this paper is organized as follows: section 2 discusses the concept of cloud computing, cloud security, DDoS attack, and load balancing. Section 3 presents a review of studies that have proposed solutions to address DDoS attacks. The proposed system and its configurations and implementation are presented in Section 4. Section 5 presents and discuss the results of the experiments conducted to implement and investigate the effectiveness of the proposed approach to DDoS attack.

1.1 Cloud Computing and Security

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell & Grance, 2011). The existing storage and computing facilities are under intense demand because of the Internet's rapid expansion (T. Dillon et al., 2010) The inexpensive commodity PCs become the primary hardware platform used by Internet service providers. One of the main concerns in cloud environments is the provision of security around multi-tenancy and isolation (Bhadauria, R, 2011). Service delivery model is just one of many factors that must be taken into account for a thorough analysis of cloud security. Security at various levels, such as Network level, Host level, and Application level, is necessary to keep the cloud up and running continuously (Bhadauria et al. 2011). In accordance with these different levels, various types of security breaches may occur.

Cyber security is the practice of defending computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It's also known as information technology security or electronic information security. The term applies in a variety of contexts, from business to mobile computing, and can be divided into a few common categories ("What Is Cyber Security?," 2023).

When it comes to blocking crucial server services by flooding the target server with so many packets that it cannot process them, DDoS can be compared to a more sophisticated version of DOS (Bhadauria et al., 2011). A DDoS attack is a malicious attempt to obstruct a server, service, or network's regular traffic by saturating the target or its surrounding infrastructure with an excessive amount of Internet traffic. The most frequent type of DoS attack happens when an attacker floods a network with too many requests to the target server, preventing it from serving regular users (Gruschka & Iacono, 2009). By using numerous compromised computer systems as sources of attack traffic, DDoS attacks are made effective. Computers and other networked resources such as IoT devices, can be exploited machines. The three functional units that make up a DDoS attack are a Master, a Slave, and a Victim. The Master is the attack launcher and is responsible for all attacks that result in DDoS. The Slave is the network that serves as the Master's launch pad and gives the Master the opportunity to attack the Victim. This is why the attack is also referred to as a coordinated attack (Bhadauria et al., 2011). A DoS attack is only considered to occur when access to a computer or network resource is intentionally blocked or degraded because of malicious action taken by another user. These attacks don't necessarily damage data directly or permanently, but they intentionally compromise the availability of the resource (Douligeris & Mitrokotsa, 2004). When viewed from a distance, a

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

DDoS assault resembles an unexpected traffic congestion that blocks the roadway and keeps ordinary traffic from reaching its destination.

A DDoS attack basically consists of two stages: the first is the intrusion phase, where the Master attempts to compromise less critical machines to assist in flooding the more critical one, and the second is the installation of DDoS tools and attacking the victim server or machine. Thus, a DDoS attack causes the service to be unavailable to the authorized user similarly to how it is done in a DoS attack but different in the way it is launched (Bhadauria et al., 2011). DDoS assaults are conducted using networks of computers linked to the Internet. These networks are made up of computers and other devices, such as Internet of Things (IoT) devices, that have been infected with malware, enabling an attacker to remotely manage them. These gadgets are known as bots (or zombies), and a botnet is a collection of bots. Once a botnet has been created, the attacker can control an attack by giving each bot remote commands. When the botnet targets a victim's server or network, each bot sends requests to the target's IP address, potentially causing the system to be overwhelmed with unexpected requests which not only prevent legitimate users from accessing the server legitimately but also has the potential of eventually halting the operation of the server bringing it to its knees. According to the level of the attacked protocol, DoS attacks can be categorized into five. Fig 1 presents the categories of DDoS (Douligeris & Mitrokotsa, 2004).

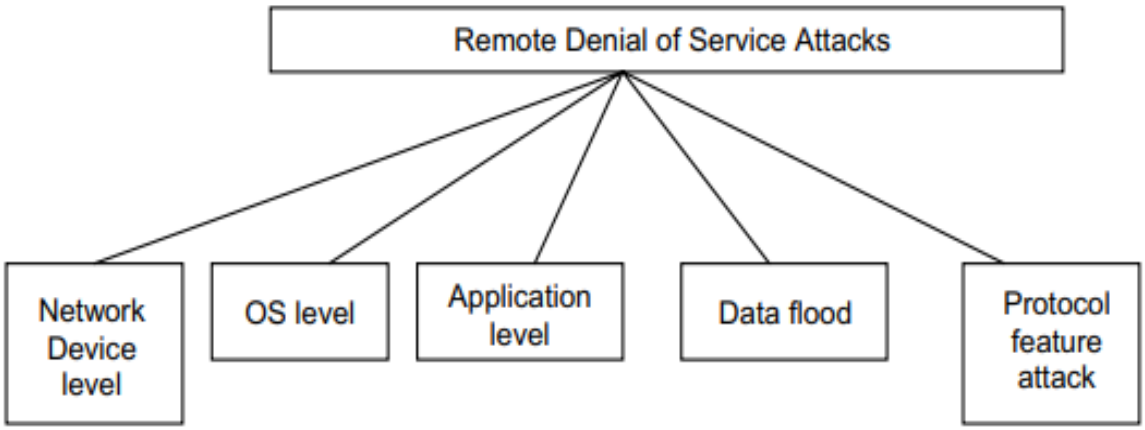

Figure 1: Categories of DDOS Attack (Douligeris & Mitrokotsa, 2004)

The Network Device Level of DoS attacks includes attacks that may be caused by exploiting bugs or weaknesses in software, or by attempting to exhaust the hardware resources of network devices. One example of a network device exploit is the one that is caused by a buffer overrun error in the password checking routine. Using these exploits, certain Cisco 7xx routers could be crashed by connecting to the routers via telnet and entering extremely long passwords (Douligeris & Mitrokotsa, 2004).

The Ping of Death attack, a type of DoS attack where ICMP echo requests with total data sizes larger than the maximum IP standard size are sent to the targeted victim, frequently results in the victim's computer crashing, is an example of this category of DoS attacks that take advantage of the ways operating systems implement protocols (Douligeris & Mitrokotsa, 2004). Application-based attacks aim to put a machine or a service out of commission by either using network applications that are running on the target host to exploit specific bugs or by using those applications to drain the victim's resources. It's also possible that the attacker may have discovered points of high algorithmic complexity and exploited them in order to consume all resources on a remote host (Douligeris & Mitrokotsa, 2004).

DoS attacks based on protocol features take advantage of certain standard protocol features. For example, several attacks exploit the fact that IP source addresses can be spoofed (Douligeris & Mitrokotsa, 2004). Several types of DoS attacks have focused on DNS, and many of these involve attacking DNS cache on name servers. An

SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

attacker who owns a name server may persuade a victim name server into caching false records by inquiring the victim about the attacker's own site (Abhishek, 2017).

1.2 Load Balancing and Cloud Monitoring

Load balancing is a key aspect of cloud computing and avoids the situation in which some nodes become overloaded while the others are idle or have little work to do. Load balancing improves Quality of Service (QoS) which includes response time, cost, throughput, performance and resource utilization.

The load balancing in the cloud, also known as load balancing as a service (LBaaS), may be among physical hosts or VMs; this balancing mechanism distributes the dynamic workload equally among all the nodes (hosts or VMs). There are two versions of load balancing algorithms: static and dynamic. The static-based balancing algorithms are mostly suitable for stable environments with homogeneous systems. Algorithms for dynamically based balancing are more flexible and efficient in both homogeneous and heterogeneous environments (Mishra, Sahoo, & Parida, 2018). For this research, the load balancing tool that would be used is called Nginx. Nginx is a web server that can also be used as a reverse proxy, mail proxy and HTTP cache.

Monitoring of Cloud is a task of paramount importance for both Providers and Consumers. On the one side, it is a key tool for controlling and managing hardware and software infrastructures; on the other side, it provides information and Key Performance Indicators (KPIs) for both platforms and applications. The continuous monitoring of the Cloud and of its SLAs (for example, in terms of availability, delay, etc.) supplies both the Providers and the Consumers with information such as the workload generated by the latter and the performance and QoS offered through the Cloud, also allowing to implement mechanisms to prevent or recover violations (for both the Provider and Consumers. In cloud computing, monitoring can be of two types: high-level and low-level. High-level monitoring is related to the virtual platform status. The low-level monitoring is related to information collected for the status of the physical infrastructure (Caron et al., 2012; Spring, 2011). Cloud monitoring system is a self-adjusting and typically multi-threaded system that is able to support monitoring functionalities (Anand, 2012). It comprehensively monitors pre-identified instances/resources on the cloud for abnormalities. On detecting an abnormal behavior, the monitoring system attempts to auto-repair this instance/resource if the corresponding monitor has a tagged auto-heal action (Anand, 2012).

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes. *Infrastructure as code (IaC)* uses DevOps methodology and versioning with a descriptive model to define and deploy infrastructure, such as networks, virtual machines, load balancers, and connection topologies. Just as the same source code always generates the same binary, an IaC model generates the same environment every time it deploys (Mijacobs, 2022). IaC avoids manual configuration and enforces consistency by representing desired environment states via well-documented code in formats such as JSON. Infrastructure deployments with IaC are repeatable and prevent runtime issues caused by configuration drift or missing dependencies. Release pipelines execute the environment descriptions and version configuration models to configure target environments. To make changes, the team edits the source, not the target (Juliakm, 2022).

2. Related Works

Belyaev and Gaivoronski (2014), proposed a system of mitigating and detecting attacks. The system involves two-level balancing solution in SDN networks, which includes traditional balancing between servers and load balancing between network devices as well. Zebari et al. (2020) proposed a system that uses network load balancing (NLB) and high availability proxy (HAProxy) as DDoS mitigation techniques. The NLB is used in the Windows platform and HAProxy in the Linux platform. Moreover, Internet Information Service (IIS) 10.0 is implemented on Windows server 2016 and Apache 2 on Linux Ubuntu 16.04 as web servers. They evaluated each load balancer efficiency in mitigating synchronized DDoS attack on each platform separately. The evaluation

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

process is accomplished in a real network and average response time and average CPU were measured as performance metrics.

Abdulkareem & Zeebaree (2022) addresses the optimization of load balancing algorithms to deal with DDoS attacks using Whale Optimization Algorithm (WOA) with other algorithms: Round-Robin (RR), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA). The results obtained from implementing these algorithms showed that the WOA performed better than other algorithms in terms of speed of the response time to client requests. The whale optimization algorithm can prevent unexpected traffic and block the regular operation of Internet websites by providing a proper plan for distributing requests between servers and reducing the average response speed. Somasundarama & Meenakshi (2021) proposed a system which uses two components, a verifier module and an elastic loan balancer module, for mitigating DDoS attack in the cloud environment.

3. The Proposed System Architecture

Figure 2 depicts the architecture of the proposed load balancing approach to mitigating DDoS attack in Cloud environment.

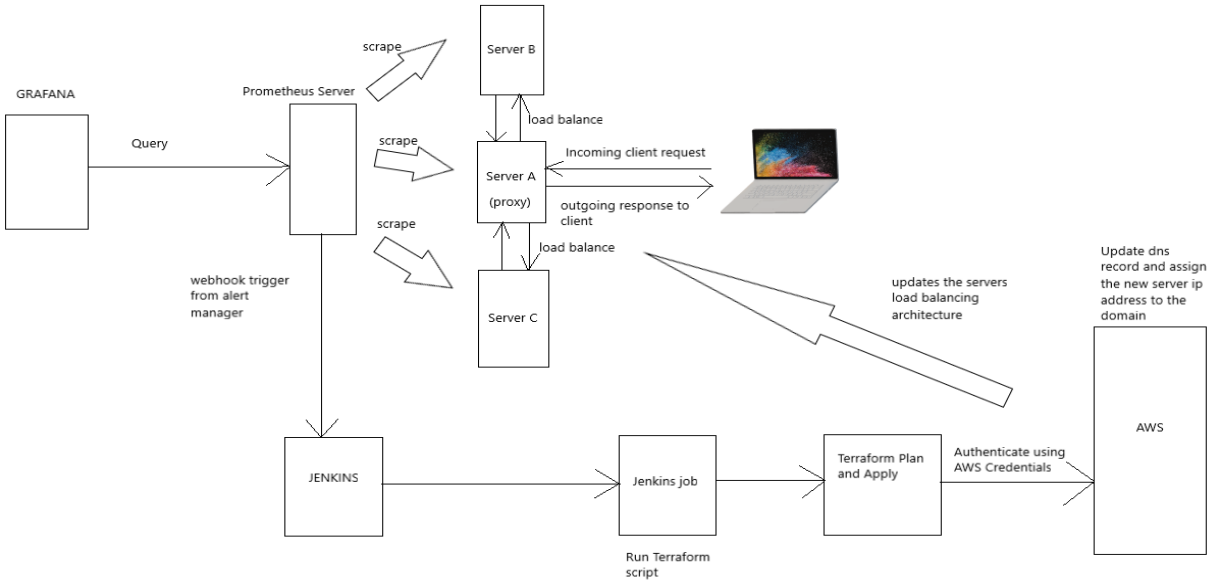

Figure 2: Proposed system architecture

Figure 2 depicts a typical scenario of the proposed system design. It consists of three servers with node exporter installed on them. Depending on the prevailing traffic, one of the servers is selected to play the role of a proxy server; this role is switched between the servers as the traffic size grows. A monitoring tool, Prometheus, is setup to fetch the prevailing state of the server by communicating with the node exporter installed on the servers. The node exporters send metric data about the state of the server such as the servers' health, CPU usages, and network traffics. Prometheus is configured with monitoring rules that specify the attributes of the server that should be monitored and their thresholds. If a threshold is exceeded, an alert would be triggered which will invoke a Jenkins job through an API endpoint. The job runs a script that update DNS record, reassigning the next server IP address to the domain.

3.1 Load Balancing in the Proposed System

The first server, server A, acts as a proxy server and a load balancer between the client and the servers. It accepts all incoming traffic and distribute the traffic among the other servers based on the load balancing algorithm specified in the system. Depending on the traffic level, these other remaining servers can take the role of serving as proxy server and load balancer.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

When there is an attack, server A receives the attack which increases the traffic level on it. This situation will lead to the triggering of the monitoring alerts as configured in the Prometheus monitoring tool. Consequently, the alert initiates a Jenkin job that will run the script that will switch the role of Server A to a mere server while any of the other servers, say Server B, takes the role of a proxy server and load balancer. Listing 1 presents the pseudocode of the proposed system showing how requests are handled by the servers through load balancing.

```
Initialize an empty list of servers
Initialize a counter to 0

function addServer(server):
    Add the server to the list of servers
Pick a server from the list to act as the proxy server

function getNextServer():
    if the list of servers is not empty:
        Get the server at the index of the counter in the list of servers
        Increment the counter
        if counter >= length of the list of servers:
            Reset the counter to 0
        return the server
   else:
        return an error message indicating no servers are available

function handleRequest(request):
    Send request to proxy server
    Proxy server uses getNextServer() to distribute request among servers

function checkProxyServerLoad():
    if proxy server CPU load > 80%:
        Old proxy server joins the rest of the servers
        Pick another server from the list to act as the new proxy server
```

Listing 1: Proposed system pseudocode

In this pseudocode, a server from the list is chosen to act as a proxy server. The proxy server accepts all incoming requests and uses the round-robin algorithm implemented in the getNextServer() function to distribute these requests among all other servers. If the CPU load of the proxy server exceeds 80%, it joins the rest of the servers in receiving requests and another server from the list is chosen to act as a new proxy server.

3.2 Prometheus Configuration

Prometheus is a versatile monitoring tool, used to monitor server health metrics. Figure 3 presents the configuration and rules specified in the proposed design. As shown in Figure 3(a), the Prometheus configuration file configures Prometheus to scrape metrics every 10s. The ***global*** section configures global settings and default values, and the ***scrape_config*** section tells Prometheus which targets to scrape. A ***scrape_config*** entry allows specifying a list of targets and a common label set for them. It contains the list of servers from which metrics should be drawn from. Figure 3 (b) shows the alert rules that have been configured in Prometheus. The rules monitor the operating system of the server to identify if the CPU usage exceeds a certain threshold. The operating system state data is collected through a node exporter which is then published to Prometheus via an HTTP endpoint. The rules specify the metrics that should be monitored and collected from the node exporter.

(a) Prometheus configuration

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

ubuntu@ip-172-31-57-218: /etc/prometheus

```
  GNU nano 4.8
# my global config
global:
  scrape_interval: 10s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).


# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - alert.rules.yml
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - 'localhost:9093'
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: []
  - job_name: 'node_exporter_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['54.159.1.91:9100']
```

(b) Configuration alert rules

ubuntu@ip-172-31-57-218: /etc/prometheus

```
  GNU nano 4.8                                                    alert.rules
groups:
- name: alert.rules
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: "critical"
    annotations:
      summary: "Endpoint {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1 minutes."
```

Figure 3. Prometheus Configuration and Alert Rules

3.3 Alert Manager Configuration

Figure 4 shows the configuration of the alert manager for the proposed system. The alert manager handles alerts sent by Prometheus monitoring tool and forwards the alert to appropriate receiver integration. In the configuration, the receiver type used is a webhook receiver. The webhook receiver forwards alerts to the webhook URL that has been configured in the ***receivers*** block.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

```
global:
  resolve_timeout: 5m
  http_config: {}
  smtp_hello: localhost
  smtp_require_tls: true
  pagerduty_url: https://events.pagerduty.com/v2/enqueue
  opsgenie_api_url: https://api.opsgenie.com/
  wechat_api_url: https://qyapi.weixin.qq.com/cgi-bin/
  victorops_api_url: https://alert.victorops.com/integrations/generic/20131114/alert/
route:
  receiver: webhook_receiver
receivers:
- name: webhook_receiver
  webhook_configs:
  - send_resolved: false
    http_config: {}
    url: http://admin:e3fedfc4779c46e399e59230517f8e7b@44.202.4.63:8080/generic-webhook-trigger/invoke
    max_alerts: 0
templates: []
```

Figure 4. Alert Manager Configuration

3.4 Terraform

Figure 5 shows the code that switches server when a DDoS attack occurs. It switches the server by updating the DNS records. The ***provider*** block specifies the cloud service provider used; in this case the Amazon Web Services(AWS) is used. The AWS Provider is used to interact with the many resources supported by AWS. In order to use the provider, the provider was configured with the appropriate credentials of the AWS account that is being used. The ***resource*** block indicates the resource being used from AWS. AWS Route 53 resource was used to manage DNS records and for easy and automatic updating and configuring.

```
variable AWS_ACCESS_KEY_ID {}
variable AWS_SECRET_ACCESS_KEY {}
provider "aws" {
  region     = "us-east-1"
  access_key = var.AWS_ACCESS_KEY_ID
  secret_key = var.AWS_SECRET_ACCESS_KEY
}

resource "aws_route53_zone" "finalyearproject" {
    name = "finalyearproject.click"

}

resource "aws_route53_record" "www" {
  zone_id = Z03488433S662LBOSFSJ2
  name    = "www.finalyearproject.click"
  type    = "A"
  ttl     = 300
  records = ["10.0.0.1"]
}
```

Figure 5. Terraform Script

3.5 Jenkins

In order to automatically run the terraform script created, Jenkins, an open-source automation tool is used. To accomplish the task, a job is created in Jenkins. The job created is named terraform as shown in figure 6. The job contains the configuration for automating specific tasks.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

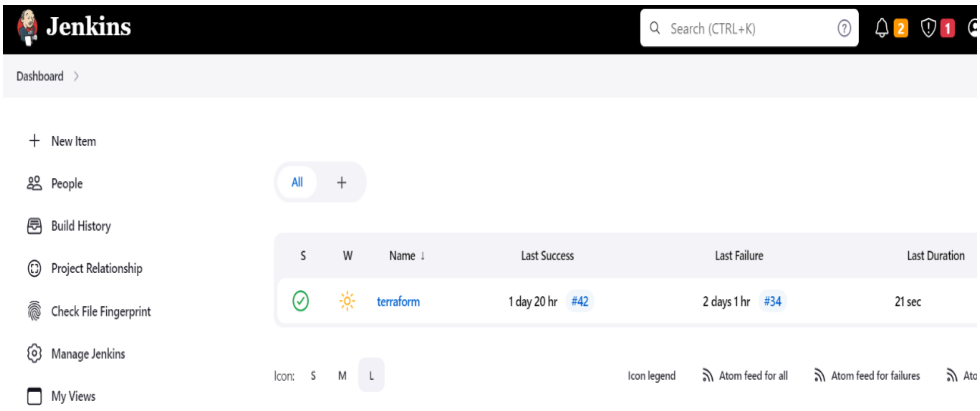

Figure 6. Jenkins Job

A build trigger configuration specifies how the job defined in Jenkin is triggered. Webhook build trigger is selected as shown in figure 7. The webhook build trigger makes an API endpoint that receives HTTP requests from alert manager.

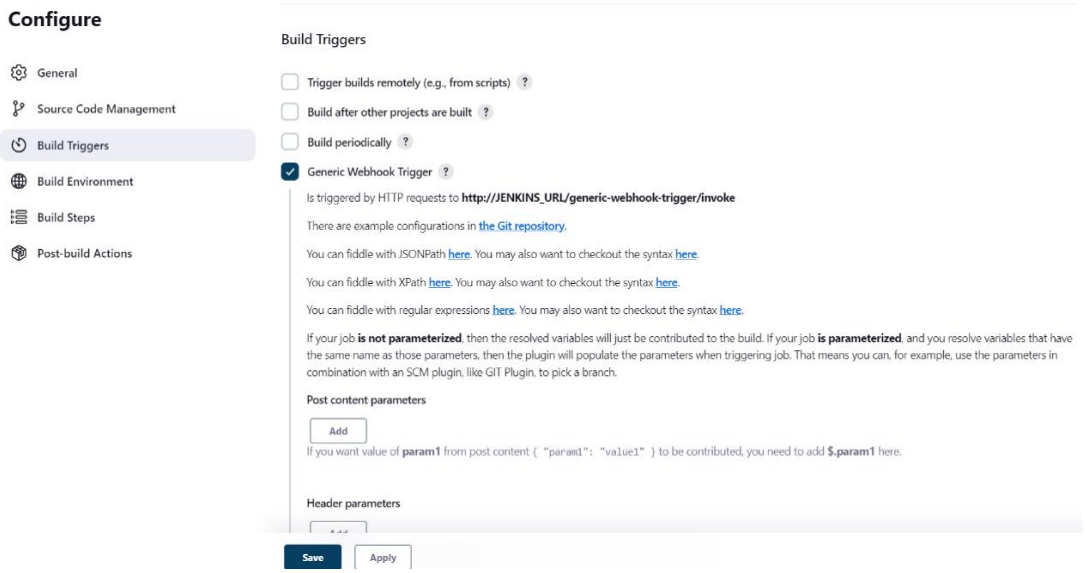

Figure 7. Jenkins Build Trigger Configuration

Then finally, the build step configuration is added. This step runs a couple of shell commands as shown in the figure above. The ***cd*** command navigates to the directory of the terraform script. The ***terraform init*** command initializes the working directory of the terraform script. The 'terraform plan' command allows previewing of

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

changes that are going to be made to the infrastructure. The ***terraform apply*** command then finally executes the actions i.e. updating the DNS records and assigning the new IP address.

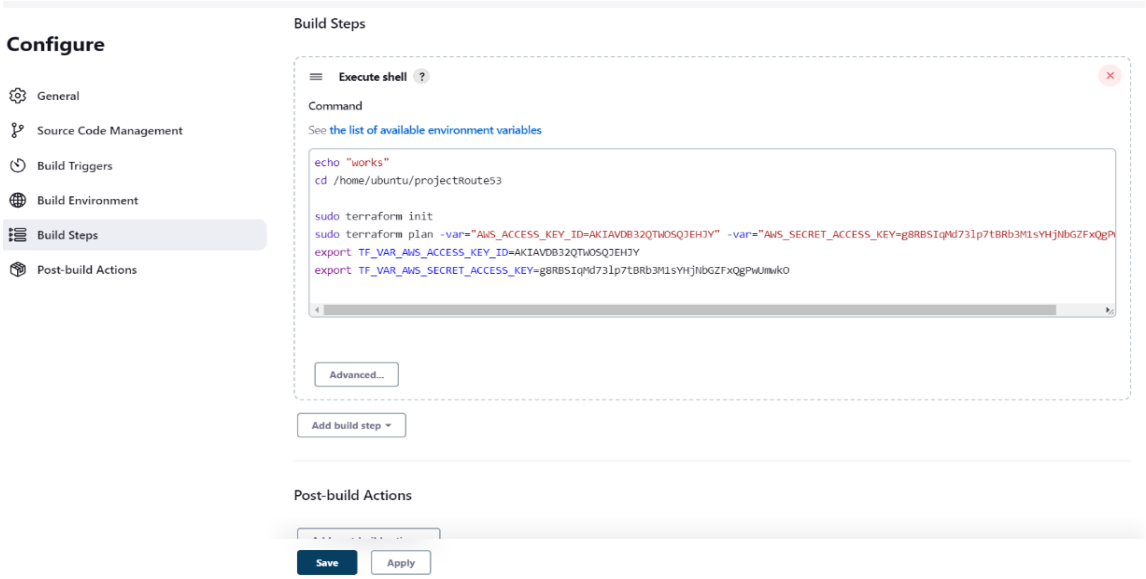

Figure 8. Build Steps/Tasks Configuration

4. Experimental Result

Having implemented the system by setting up all the tools and their configurations as presented in the previous section, the system was tested and evaluated. A DDoS attack was simulated on the servers used. The results of the implementation of the proposed system are presented using the UIs of the following tools used in the study: Google Chrome, Alertmanager, Grafana, Command line interface, and AWS Route53 console. The states of the setup before, during and after arresting the simulated attack are presented.

4.1 Before the Attack

Figures 9, 10 and 11 present a status of the network and server CPU before a DDoS attack. Figure 9 shows a website running on a server running normally as expected, with its IP address.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

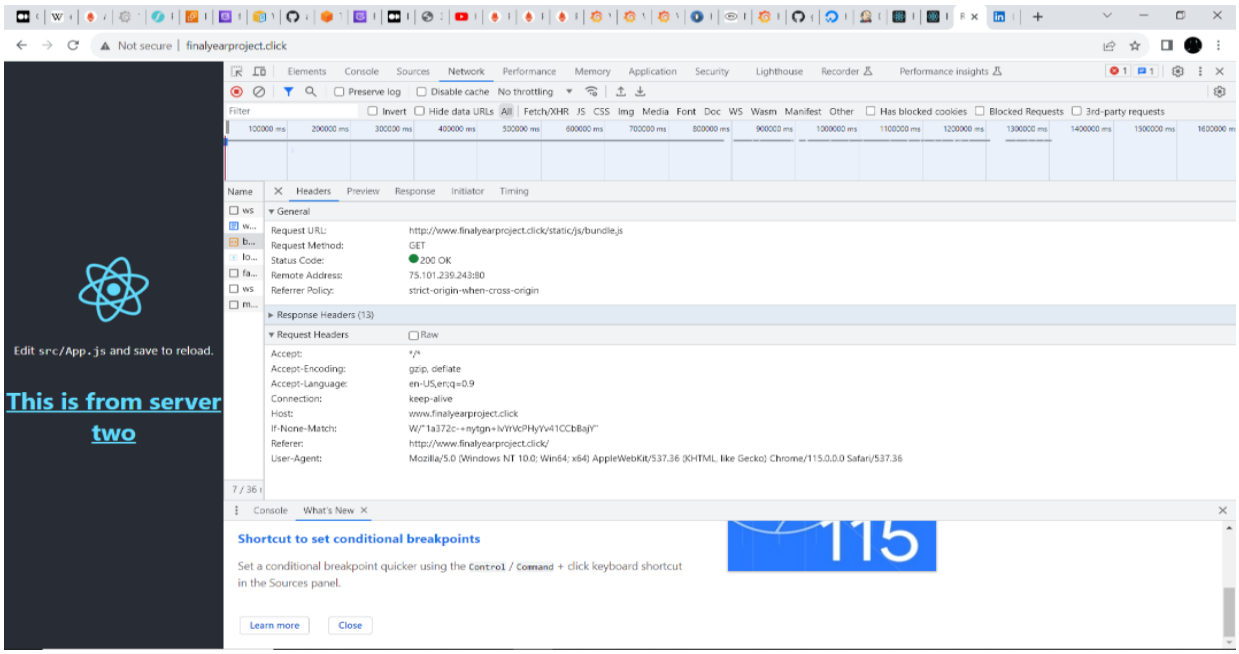

Figure 9. Chrome Browser Network Status of a Website before Attack

Figure 10 presents a graphcal representation of the CPU usage before an attack. The time series graph shows the CPU running normally before an attack.

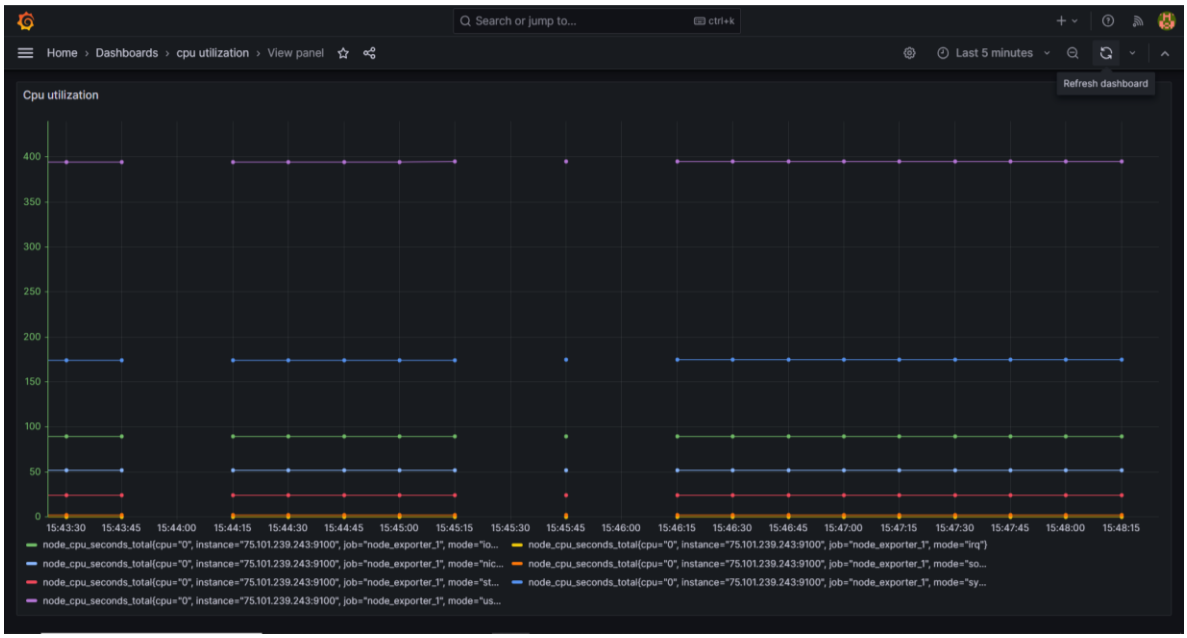

Figure 10. Grafana Dashboard showing Server CPU Usage before an Attack.

Figure 11 shows the CPU usage on the console. Comparing it to the time series graph , it can be seen that all is running as would be expected of a normal network traffic.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

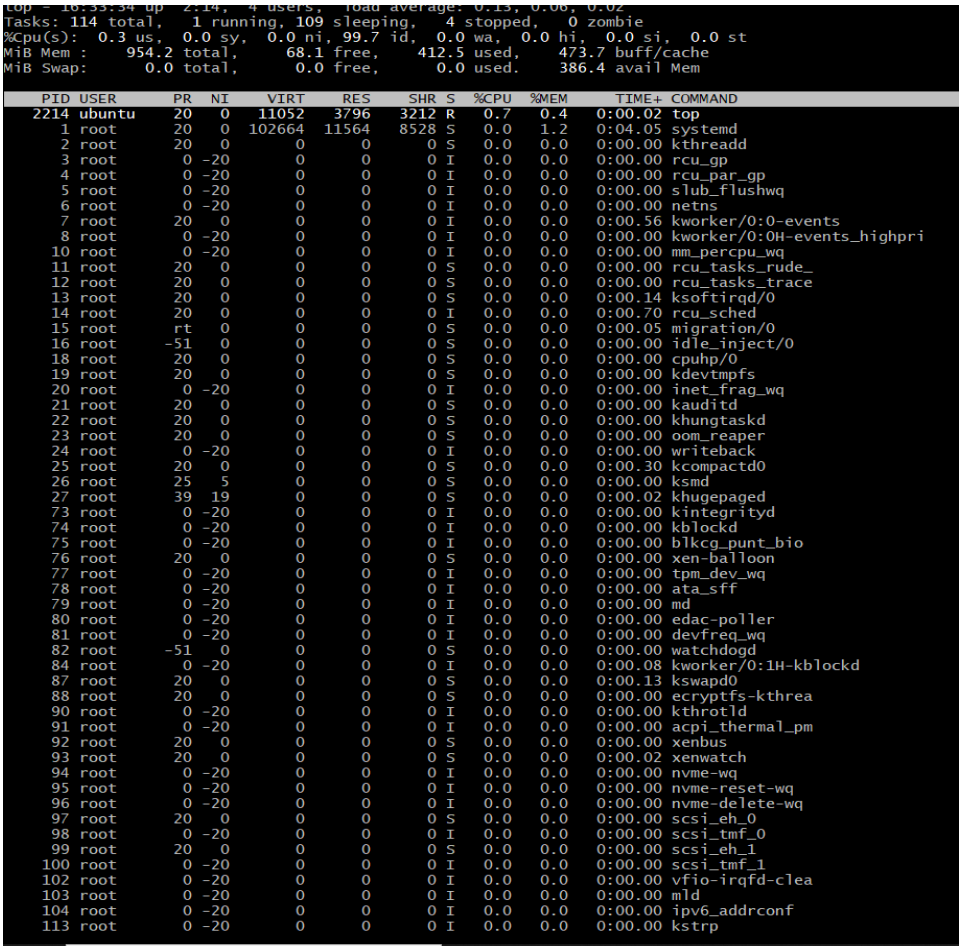

Figure 11. Console showing CPU Usage

4.2 During the Attack

Figure 12 shows a CPU usage of almost 100% during the simulated DDoS attack. Figure 13 gives a graphical representation of the CPU showing a sudden rise in the usage of the server processor.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

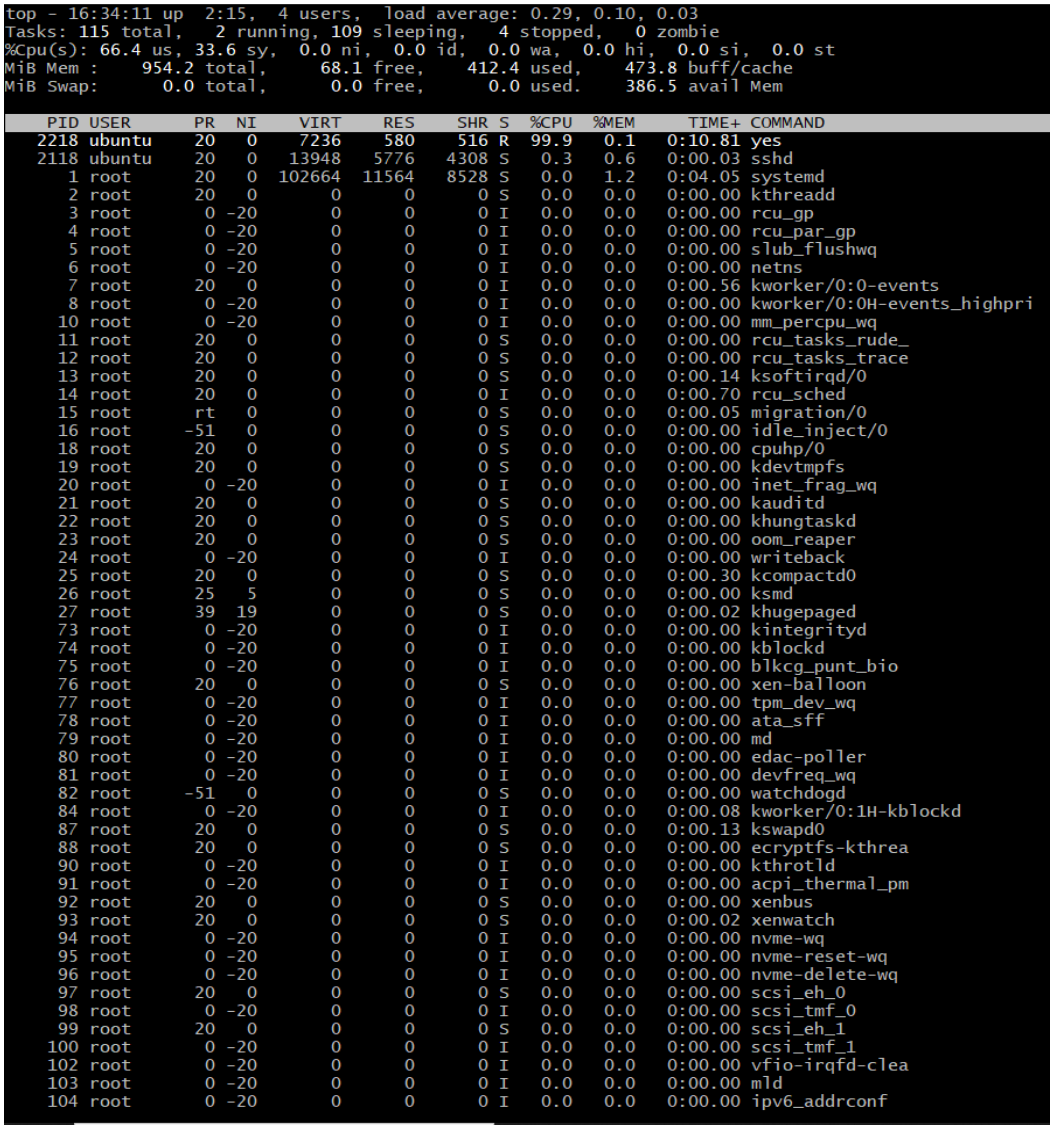

Figure 12. Console showing CPU Usage during an Attack

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

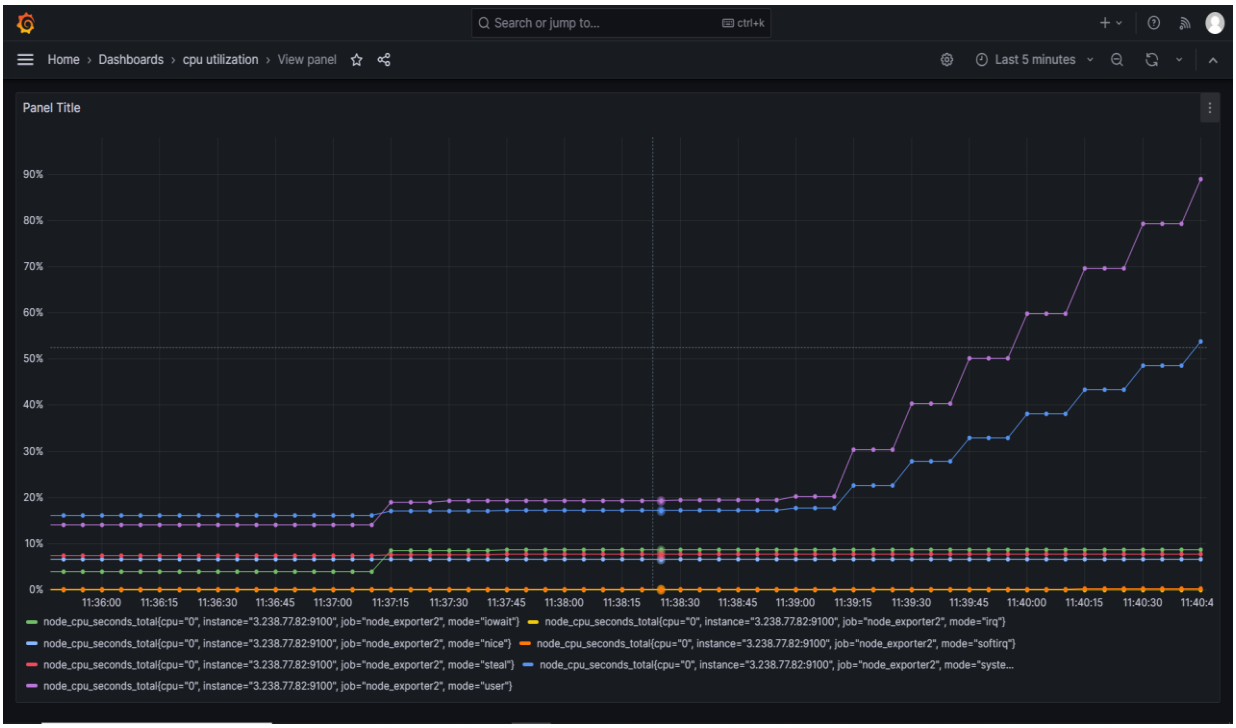

Figure 13. Grafana showing the Rise in CPU Usage.

4.3 After Arresting the Attack

Figures 14 and 15 presents the state of the system immediately after the proposed system has handled the flood of attacks by switching the role of the servers as described in section 4. Figure 14 shows that the website in Figure 11 now has a different server IP address it communicates indicating that the proposed system has switched to another server serving to serve the same website.

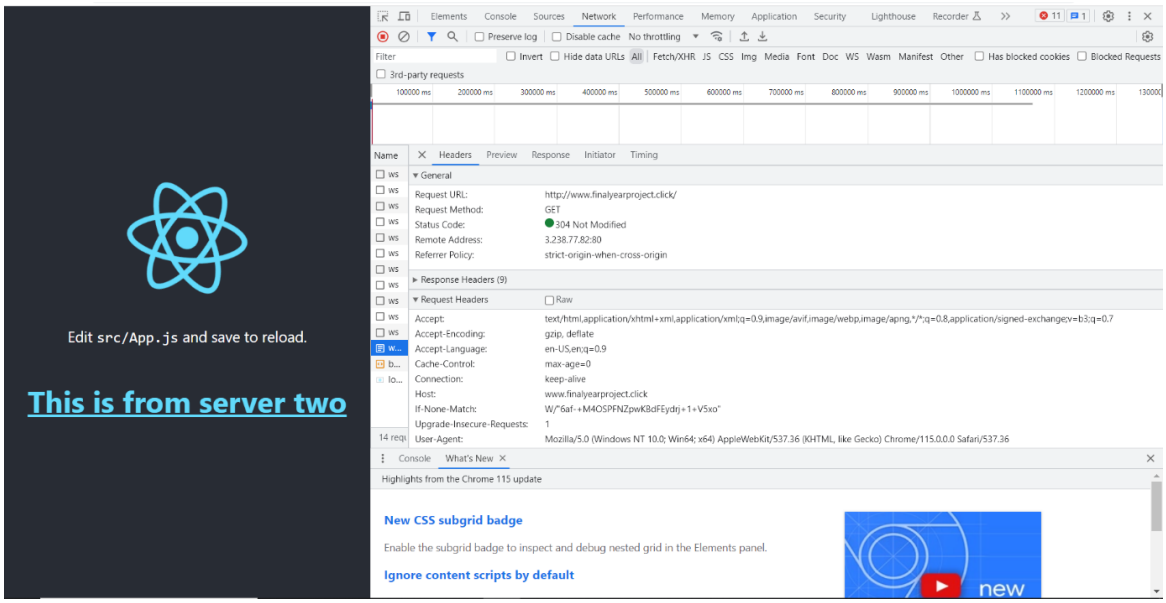

Figure 14. Chrome showing the New IP Address after the Attack has been Arrested.

Figure 15 shows the output of the terraform script that updates the DNS record to switch the server from the one under attack to another.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 1 [June, 2024], pp. 1-17
https://doi.org/10.56471/slujst.v8i.483
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

```
# aws_route53_zone.finalyearproject will be created
  + resource "aws_route53_zone" "finalyearproject" {
      + arn                  = (known after apply)
      + comment              = "Managed by Terraform"
      + force_destroy        = false
      + id                   = (known after apply)
      + name                 = "finalyearproject.click"
      + name_servers         = (known after apply)
      + primary_name_server  = (known after apply)
      + tags_all             = (known after apply)
      + zone_id              = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
aws_route53_zone.finalyearproject: Creating...
aws_route53_record.www: Creating...
aws_route53_zone.finalyearproject: Still creating... [10s elapsed]
aws_route53_record.www: Still creating... [10s elapsed]
aws_route53_record.www: Still creating... [20s elapsed]
aws_route53_zone.finalyearproject: Still creating... [20s elapsed]
aws_route53_record.www: Creation complete after 25s [id=Z03488433S662LBOSFSJ2_www.finalyearproject.click_A]
aws_route53_zone.finalyearproject: Still creating... [30s elapsed]
aws_route53_zone.finalyearproject: Creation complete after 33s [id=Z0696168UMAFVYAQW6S5]
[1m[32m
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
Finished: SUCCESS
```

Figure 15. Output of the terraform script.

5. Discussion

The above experiment was repeated 20 times with different attack scenarios to ascertain the effectiveness of the proposed system. The results obtained showed that the proposed system was able to arrest the DDoS attacks by switching the roles of the servers configured for the system. Figure 13 showed that the DDoS attacks suddenly spike up the CPU usage of the server under attack. This sudden increase in requests over a very short period triggered the execution of the terraform script by the Prometheus monitoring rules configured for the system. The script switches the proxy and load balancing server from the server under attack to the one of the servers servicing requests. As long as the request traffic hits the thresholds configured in the monitoring server, the script keeps switching the proxy and load balancing server until the attack is contained and QoS is restored. Figure 14 shows that the attack arrest was successful with the site still up and running from a server with a different IP address indicating that the server was switched from the previous one to this new one. The output of the script indicates the time taken to handle the attack; it shows a time of 55 seconds for the scenario presented. On average over the scenarios tested out in the experiment, the time taken is approximately 63 seconds. Thus, the downtime or low QoS experienced by the system due to the DDoS attack in the experiments falls within the neighborhood of 1 minute period. This is negligible and users may not notice the effect of the attack.

The highest duration recorded during the experiments was 4 minutes 23 seconds with the servers switched 15 times due to the increase in volume of the attack request in the scenario. This implies that the downtime that will be experienced by the proposed system is directly proportional to the volume of attack requests.

6. Conclusion

The internet is used by society for a wide range of purposes, including employment, entertainment, managing finances, storing personal data, and more. According to estimates, the cloud computing industry would be valued USD 371.4 billion in 2020 and expand at a pace of 17.5% CAGR. Security is now more important than ever in a setting where the number of internet users is increasing constantly. We rely on cybersecurity professionals as a result to safeguard our privacy when sending and receiving data. They possess the sophisticated abilities to foresee security issues and protect sensitive data from unauthorized users. Cyber dangers come in a variety of shapes and sizes, making it challenging to stop them. This project work has been able to address the issue of ddos attack which is a threat to availability in cloud computing environment, by developing a load balancing algorithm. After integrating and testing the system, there were few to zero downtime encountered, and even when there was a downtime, the duration didn't exceed 5 minutes. Thus, the adoption of the proposed system with appropriate tools and system configurations, it can contain DDoS attack with little or no service downtime and poor QoS.

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Further studies could be conducted to demonstrate the effectiveness of the proposed system in containing DDoS attack by increasing the volume of attack request and the number of servers configured for the system. Also, the number of proxy and load balancing servers will be increased to investigate the effect on the performance of the proposed system. It was observed in the study that the number of switching of servers done is directly proportional to the volume of attack request sent to the servers. This observation will be further investigated by hypothesis testing using different setups of the system with different volumes of attack request and the same server configurations and tool settings.

References

Abdulkareem, N. M., & Zeebaree, S. R. M. (2022). Optimization of Load Balancing Algorithms to Deal with Ddos Attacks Using Whale optimization Algorithm. Optimization of Load Balancing Algorithms to Deal with Ddos Attacks Using Whale Optimization Algorithm, 25(2), 65–85. https://doi.org/10.26682/sjuod.2022.25.2.7

Ahsan, M., Nygard, K. E., Gomes, R., Chowdhury, M. M., Rifat, N., & Connolly, J. F. (2022). Cybersecurity threats and their mitigation approaches using Machine Learning—A Review. Journal of Cybersecurity and Privacy, 2(3), 527-555.

Anand, M. (2012, October). Cloud monitor: Monitoring applications in cloud. In 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 1-4). IEEE.

Bhadauria, R., Chaki, R., Chaki, N., & Sanyal, S. (2011). A survey on security issues in cloud computing. arXiv preprint arXiv:1109.5388, 1-15.

Bhadauria, R., & Sanyal, S. (2012b). Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques. International Journal of Computer Applications, 47(18), 47–66. https://doi.org/10.5120/7292-0578.

Caron, E., Rodero-Merino, L., Desprez, F., & Muresan, A. (2012). Auto-scaling, load balancing and monitoring in commercial and open-source clouds (Doctoral dissertation, INRIA).

Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer Networks, 44(5), 643–666. https://doi.org/10.1016/j.comnet.2003.10.003

Frankenfield, J. (2023). What is Cloud Computing? Pros and Cons of Different Types of Services. Investopedia. Retrieved from https://www.investopedia.com/terms/c/cloud-computing.asp

H. Tianfield, "Security issues in cloud computing," 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Korea (South), 2012, pp. 1082-1089, doi: 10.1109/ICSMC.2012.6377874.

M. Belyaev and S. Gaivoronski, "Towards load balancing in SDN-networks during DDoS- attacks," 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), Moscow, Russia, 2014, pp. 1-6, doi: 10.1109/MoNeTeC.2014.6995578.

Mell, P., & Grance, T. (2011). SP 800-145. The NIST Definition of Cloud Computing. The NIST Definition of Cloud Computing. Retrieved from https://dl.acm.org/citation.cfm?id=2206223

Mijacobs. (2022, November 28). What is infrastructure as code (IaC)? - Azure DevOps. Retrieved from https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

Mishra, S. K., Sahoo, B., & Parida, P. P. (2018). Load balancing in cloud computing: A big picture. Journal of King Saud University - Computer and Information Sciences, 32(2), 149–158. https://doi.org/10.1016/j.jksuci.2018.01.003

N. Gruschka and L. L. Iacono. “Vulnerable Cloud: SOAP Meaage Security Validation Revisited” IEEE International Conference on Web Service, pp.635-631, Jul 2009.

Nuaimi, K. A., Mohamed, N., Nuaimi, M. A., & Al-Jaroodi, J. (2012). A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms. https://doi.org/10.1109/ncca.2012.29

P. Mell and T. Grance, "Draft nist working definition of cloud computing - v15," 21. Aug 2009, 2009.

Perry, G., Minimizing public cloud disruptions, TechTarget, [online]. Available at: http://searchdatacenter.techtarget.com/tip/Minimizing-public-cloud-disruptions, 2011.

Potey, M. M., Dhote, C. A., & Sharma, D. H. (2013). Cloud computing-understanding risk, threats, vulnerability and controls: a survey. International Journal of Computer Applications, 67(3).

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Sahu, S. K., & Khare, R. K. (2020). DDOS Attacks & Mitigation Techniques in Cloud Computing Environments. Gedrag Organ. Rev, 33(2), 2426-2435.

Somasundaram, A. (2021). DDOS Mitigation In Cloud Computing Environment By Dynamic Resource Scaling With Elastic Load Balancing. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(11), 3346-3362.

Sumina, V. (2022, March 18). 26 cloud computing statistics, facts & trends for 2022. Cloudwards. https://www.cloudwards.net/cloud-computing-statistics/

T. Dillon, C. Wu and E. Chang, "Cloud Computing: Issues and Challenges," 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, Australia, 2010, pp. 27-33, doi: 10.1109/AINA.2010.187.

Trend Micro. (2021, October 25). The most common cloud misconfigurations that could lead to security breaches. https://www.trendmicro.com/vinfo/us/security/news/virtualization-and-cloud/the-most-common-cloud-misconfigurations-that-could-lead-to-security-breaches

Tissir, N., Kafhali, S. E., & Aboutabit, N. (2021). Cybersecurity management in cloud computing: semantic literature review and conceptual framework proposal. Journal of Reliable Intelligent Environments, 7(2), 69–84. https://doi.org/10.1007/s40860-020-00115-0

What is Cyber Security? (2023, May 18). Retrieved from https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security

Yan, Q., Yu, F. R., Gong, Q., & Li, J. (2016). Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges. Ddos Attack, 18(1), 602–622. https://doi.org/10.1109/comst.2015.2487361

Zebari, Rizgar & Zeebaree, Subhi & Sallow, Amira & Shukur, Hanan & Ahmed, Omar & Jacksi, Karwan. (2020). Distributed Denial of Service Attack Mitigation using High Availability Proxy and Network Load Balancing. 174-179. 10.1109/ICOASE51841.2020.9436545.