CHAPTER 4
METHODOLOGY

This chapter details the methodology that was used to design a DDoS Protection System for Cloud Using AWS and Machine Learning. The approach involved the generation of synthetic traffic data, anomaly detection based on the Isolation Forest algorithm, and a robust cloud-based infrastructure that utilized AWS to manage traffic and auto-scale. It also involved stressing the system under simulated DDoS conditions using Apache JMeter and evaluating performance against key metrics such as anomaly detection accuracy, scalability of the system, and availability.

4.1 Addressing Cloud Security Using AI and Cloud-Native Techniques

This factor has made cloud infrastructure critical to businesses, hosting applications, and managing sensitive data. With the upsurge in the usage of cloud infrastructure, attacks by DDoS increase too. DDoS attacks present malicious traffic to overwhelm systems, which in turn disrupts service.

The proposed solution uses ML techniques in conjunction with cloud-native tools such as AWS Auto Scaling and Load Balancers and synthetic traffic simulations for real-time detection and mitigation of DDoS attacks. It focuses on an adaptive architecture that is scalable to achieve availability and performance under heavy traffic conditions.

4.2 Synthetic Traffic Data Generation

A Python script was developed that simulates a realistic web traffic pattern, whether benign or malicious. It includes synthetic data such as:

- Normal traffic which mimics standard HTTP requests with a consistent interval and data sizes.
- Malicious traffic: Simulates DDoS-like behavior, including sudden spikes in requests and high-frequency traffic from multiple sources.

These parameters include request rate, traffic distribution, and payload size to create diverse datasets. These datasets will be used to conduct anomaly detection. Log and save the traffic patterns. It is necessary for later pre-processing and analysis.

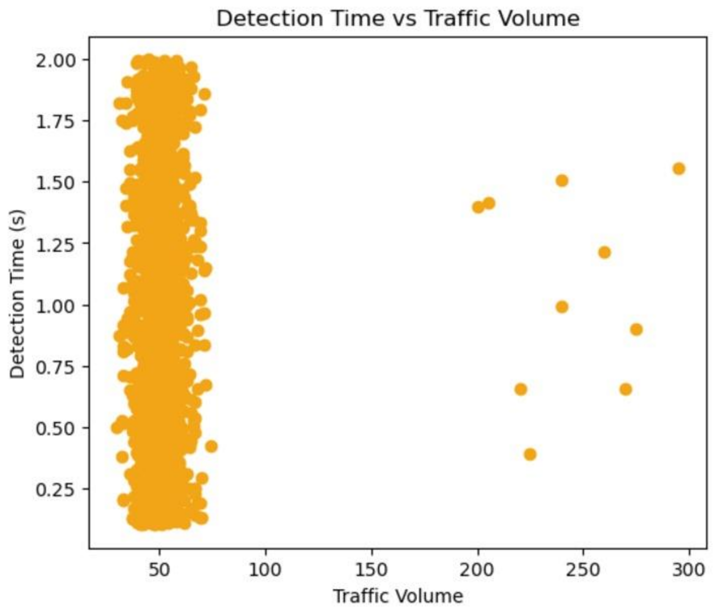


Fig. 4.1 Detection Time vs Traffic Volume

4.3 Anomaly Detection with Machine Learning

The Isolation Forest algorithm was employed with the scikit-learn library to determine anomalies in traffic data. This algorithm isolates anomalies by recursively partitioning the dataset; hence it is very suitable for identifying anomalous traffic patterns. The process encompasses the following steps:

- Preprocessing the data: Standardization of request frequency, payload size, and response times.
- Training the model: Synthetic traffic data was used to train the Isolation Forest to classify traffic as either benign or suspicious.
- Visualization: The model identified outliers were visualized with Matplotlib that gave the suspicious activity insights.

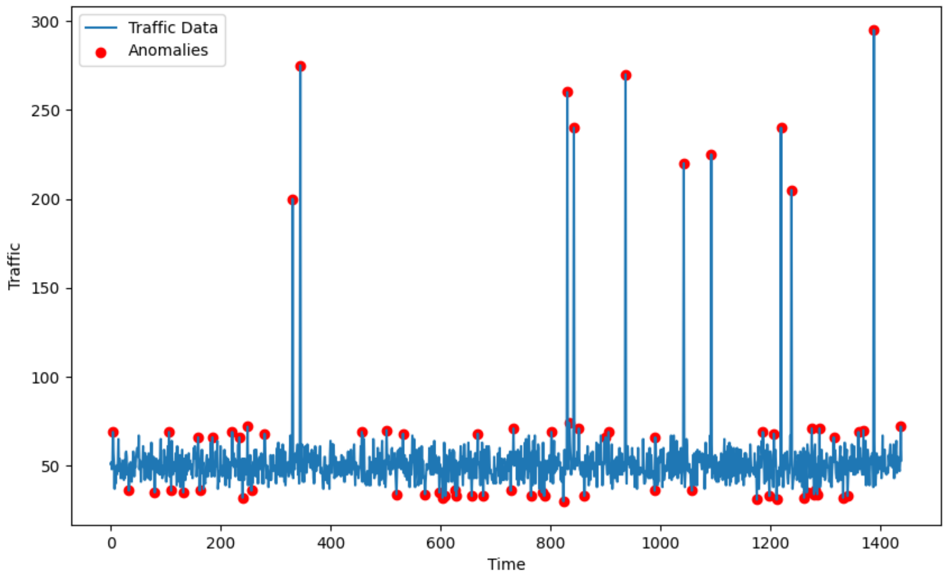


Fig. 4.2 Anomalies Detection Plot

4.4 Load Testing with Apache JMeter

4.4.1 Simulation of realistic traffic

We used the Apache JMeter tool for stress testing our system under the following traffic scenarios, in order to emulate real attacks:

- Number of users: Simulated concurrent requests by multiple sources
- Ramp-up period: Gradually increasing traffic with respect to time.
- Loop count: Continuous sending of requests simulating persistence DDoS attack behavior.

These parameters enabled all possible tests that could prove the system's response for normal and malicious loads.

4.4.2 DDoS attack resilience tests

Tests created in JMeter aimed at testing the system's resiliency to sudden increases in traffic. Response time, throughput, and error rate metrics were used to establish how effective the system was in withstanding stress conditions.

4.5 Cloud Integration with AWS

4.5.1 Deploying on AWS

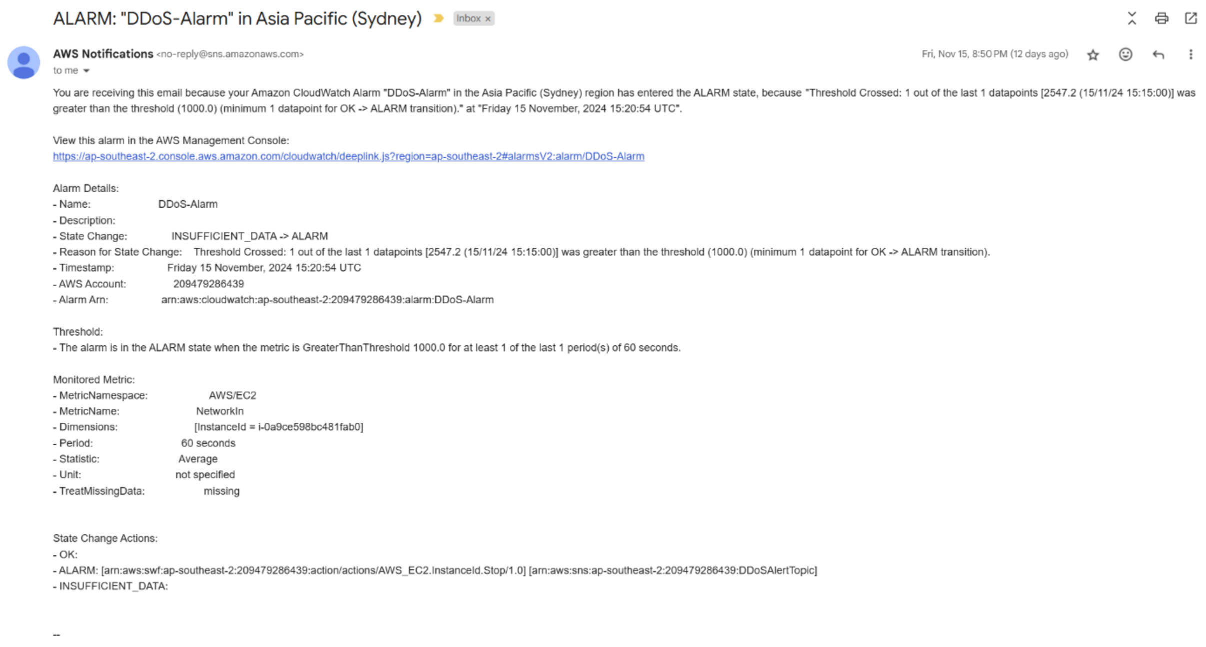
The anomaly detection script and traffic monitoring components were deployed on an AWS EC2 instance. This cloud-based setup enabled real-time detection and monitoring of incoming traffic.

4.5.2 Traffic Monitoring with CloudWatch

Amazon CloudWatch was configured to track key performance metrics, including:

- NetworkIn: Monitors incoming traffic volume.
- CPU utilization: Tracks server load under varying traffic conditions.

CloudWatch alerts were set to notify administrators whenever thresholds were being breached, so responses in case of potential DDoS attacks could be expedited.



ALARM: "DDoS-Alarm" in Asia Pacific (Sydney)

AWS Notifications <no-reply@sns.amazonaws.com>
to me
Fri, Nov 15, 8:50 PM (12 days ago)

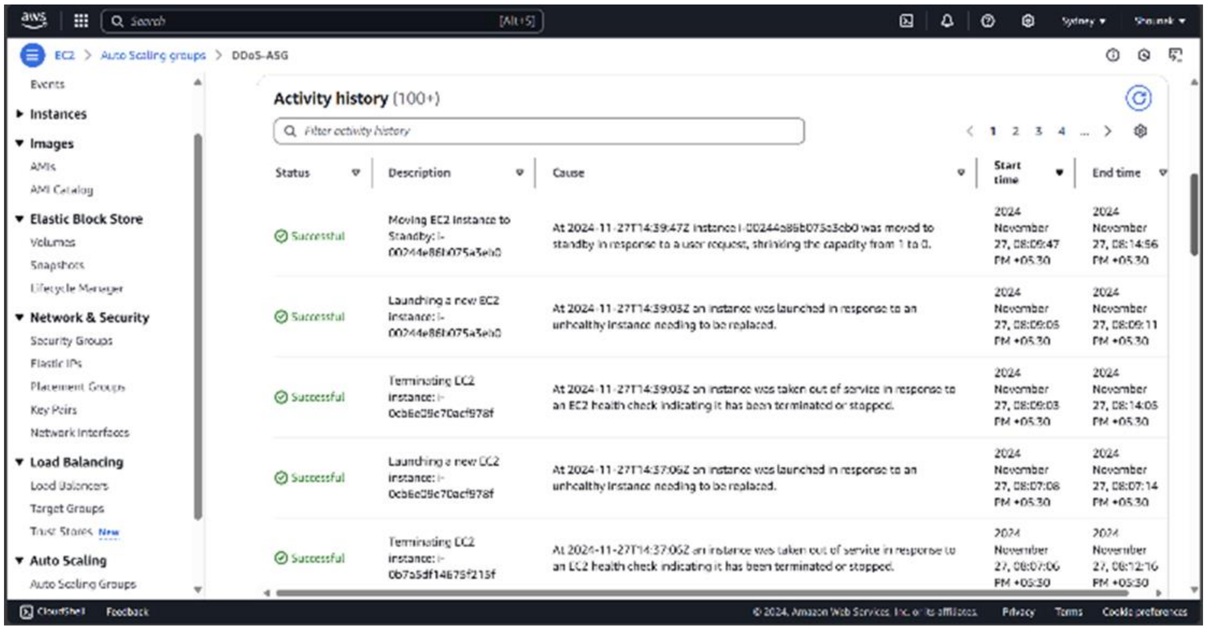
```
You are receiving this email because your Amazon CloudWatch Alarm "DDoS-Alarm" in the Asia Pacific (Sydney) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [2547.2 (15/11/24 15:15:00)] was greater than the threshold (1000.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Friday 15 November, 2024 15:20:54 UTC".

View this alarm in the AWS Management Console:
https://ap-southeast-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=ap-southeast-2#alarmsV2:alarm/DDoS-Alarm

Alarm Details:
- Name:                 DDoS-Alarm
- Description:
- State Change:           INSUFFICIENT_DATA -> ALARM
- Reason for State Change:    Threshold Crossed: 1 out of the last 1 datapoints [2547.2 (15/11/24 15:15:00)] was greater than the threshold (1000.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp:              Friday 15 November, 2024 15:20:54 UTC
- AWS Account:            209479286439
- Alarm Arn:              arn:aws:cloudwatch:ap-southeast-2:209479286439:alarm:DDoS-Alarm

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 1000.0 for at least 1 of the last 1 period(s) of 60 seconds.

Monitored Metric:
- MetricNamespace:              AWS/EC2
- MetricName:                 NetworkIn
- Dimensions:                 [InstanceId = i-0a9ce598bc481fab0]
- Period:                   60 seconds
- Statistic:                 Average
- Unit:                    not specified
- TreatMissingData:             missing


State Change Actions:
- OK:
- ALARM: [arn:aws:swf:ap-southeast-2:209479286439:action/actions/AWS_EC2.InstanceId.Stop/1.0] [arn:aws:sns:ap-southeast-2:209479286439:DDoSAlertTopic]
- INSUFFICIENT_DATA:


--
```

Fig. 4.3 Alert Email Screenshot

4.5.3 Auto Scaling and Load Balancing

To ensure scalability and availability of the system:

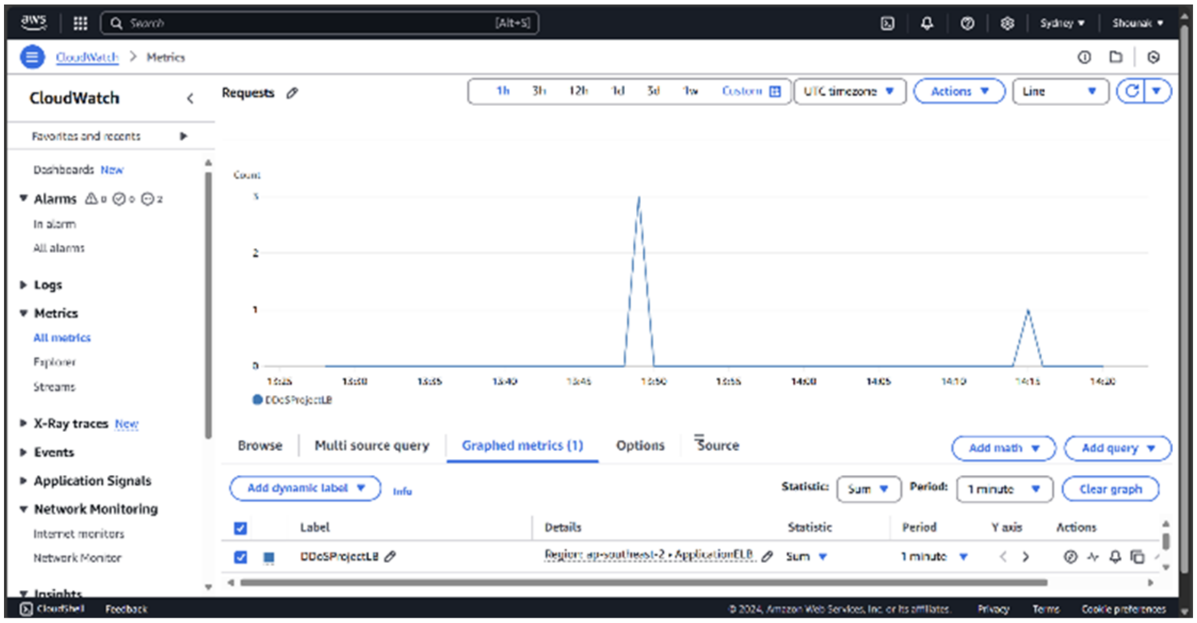
- AWS Auto Scaling will dynamically adjust the number of EC2 instances depending upon the incoming traffic load; these instances scale up during peak usage and scale down during normal usage.
- An AWS Elastic Load Balancer (ELB) distributed incoming traffic evenly across instances, preventing overload on any single server and maintaining service continuity.



Activity history (100+)

Status	Description	Cause	Start time	End time
Successful	Moving EC2 instance to Standby: i-00244e86b075a3eb0	At 2024-11-27T14:39:47Z instance i-00244a86b075a3eb0 was moved to standby in response to a user request, shrinking the capacity from 1 to 0.	2024 November 27, 08:09:47 PM +05:30	2024 November 27, 08:14:56 PM +05:30
Successful	Launching a new EC2 instance: i-00244e86b075a3eb0	At 2024-11-27T14:39:03Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 November 27, 08:09:05 PM +05:30	2024 November 27, 08:09:11 PM +05:30
Successful	Terminating EC2 instance: i-0cb6e09c70acf978f	At 2024-11-27T14:39:03Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2024 November 27, 08:09:03 PM +05:30	2024 November 27, 08:14:05 PM +05:30
Successful	Launching a new EC2 instance: i-0cb6e09c70acf978f	At 2024-11-27T14:37:06Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 November 27, 08:07:08 PM +05:30	2024 November 27, 08:07:14 PM +05:30
Successful	Terminating EC2 instance: i-0b7a5df14675f215f	At 2024-11-27T14:37:06Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2024 November 27, 08:07:06 PM +05:30	2024 November 27, 08:12:16 PM +05:30

Fig. 4.4 Auto Scaling Behavior



Statistic: Sum Period: 1 minute

Label	Details	Statistic	Period
DDoSProjectLB	Region: ap-southeast-2 • ApplicationELB	Sum	1 minute

Fig. 4.5 Load Balancer Behavior

4.6 Validation and Mitigation Strategies

4.6.1 Simulation of Attack Scenarios

Several attack scenarios were simulated using JMeter to validate the system's ability to:

- Detect anomalies in real-time using the Isolation Forest algorithm.
- Scale resources dynamically to handle traffic surges.
- Maintain availability during DDoS attacks.

4.6.2 Mitigation with Blocking Mechanisms

Automated blocking mechanisms were put in place based on the detection of suspicious activity to:

- Restrict traffic from identified malicious IP addresses.
- Maintain normal traffic flow and minimize disruption.

4.6.3 Performance Evaluation

The performance of the system was measured using metrics like:

- Detection accuracy: Capability to correctly identify malicious traffic.
- Scalability: How well Auto Scaling keeps up with performance under loads.
- Availability: Percentage uptime during stress conditions.

Improvements in detection algorithms and architectures were made based on the feedback from the evaluations.

4.7 Summary

This methodology outlines a comprehensive approach to DDoS protection using synthetic traffic generation, machine learning for anomaly detection, and AWS for scalability and resilience. By integrating these components, the system effectively addresses the challenges posed by DDoS attacks, ensuring continuous availability and performance in cloud environments.