National College of Ireland

Configuration Manual

MSc Research Project
Research in Computing CA2

Mayuri Umrikar
Student ID: 22151630

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mayuri Umrikar
Student ID:	22151630
Programme:	Research in Computing CA2
Year:	2024
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	1506
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mayuri Umrikar
Date:	24th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mayuri Umrikar
22151630

1 Introduction

The detailed installation instructions of this setup guide rule-based Intrusion Detection System, specifically to be set up in the Fog-to-Cloud computing environment, which applies AWS services that include, namely AWS Lambda, and API Gateway, while processing data received from network traffic to track security threats. The pre-configured UNSW-NB15 data set from which the data is captured through analysis by pre-defined IDS rules for diverse malicious attacks. This guide outlines every step of configuration process to ensure smooth deployment that avoids any code specifics.

2 Project Overview

The project titled **Leveraging Intrusion Detection System: Based on Fog-to-Cloud Computing** focuses on implementation within Fog-to-Cloud computing frameworks. As it executes intrusion detection logic using AWS Lambda and facilitates interactions with external entities via AWS API Gateway, the system is capable of exhibiting scalability, flexibility, and robust security monitoring. IDS processes network traffic data against rule-based heuristics to flag potential threats that make the overall security posture of the cloud environment better.

3 Steps Followed

The configuration procedure is split into 4 essential sections:

1. Dataset Preparation
2. Intrusion Detection Rules
3. AWS Lambda Function Setup

4. API Gateway Configuration

Each segment is distinct underneath, observed by relevant screenshots to aid in the configuration technique.

4 Dataset Preparation

4.1 Dataset Selection

The IDS utilizes the **UNSW-NB15** dataset, renowned for its comprehensive series of community visitors facts encompassing diverse benign and malicious sports. This dataset presents a stable foundation for education and checking out the intrusion detection good judgment.

4.2 Downloading the Dataset

Download the subsequent documents from the reliable UNSW-NB15 repository:

- `UNSW_NB15_training-set.parquet`
- `UNSW_NB15_testing-set.parquet`

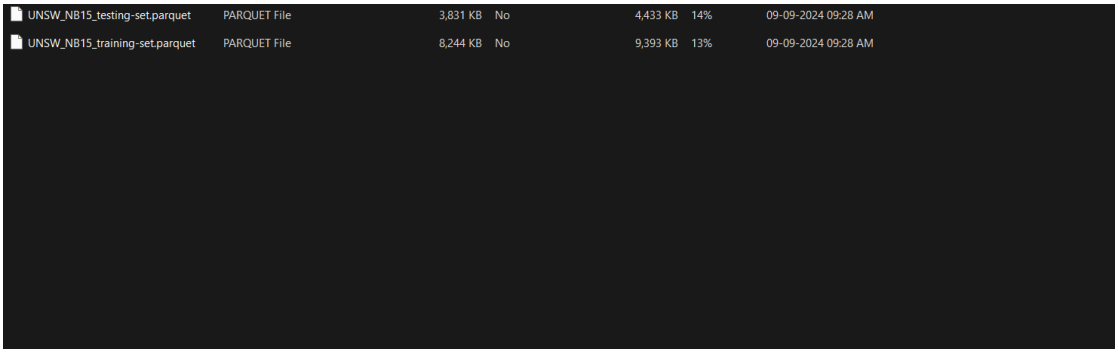

Figure 1: Dataset Download Screen

4.3 Data Analysis

4.3.1 Inspecting Dataset Properties

Begin by inspecting the dataset to understand its structure and key features. Important features include:

- `sbytes`: Source bytes

- `dbytes`: Destination bytes
- `dur`: Duration of the connection
- `proto`: Protocol used (e.g., TCP, UDP, ICMP)

4.3.2 Label Distribution Examination

Analyze the distribution of labels to differentiate between benign and malicious traffic. Understanding the balance between these categories is crucial for effective rule formulation.

5 Intrusion Detection Rules

Developing effective intrusion detection rules is pivotal to the system's success. The following rule-based heuristics are designed to identify potential security threats based on the selected dataset features.

5.1 Rule 1: Potential Data Exfiltration

Condition:

- `sbytes` $>$ 25,000
- `dbytes` $<$ 3,000
- `dur` < 5

Detection: Potential Data Exfiltration
Explanation: High source byte count coupled with low destination byte count and short duration may indicate large volumes of data being exfiltrated from the source.

5.2 Rule 2: Potential Ping Flood

Condition:

- `proto` = "ICMP"
- `dbytes` $<$ 3,000

Detection: Potential Ping Flood
Explanation: Low destination byte count in ICMP traffic can be indicative of a ping flood attack, where numerous ICMP packets are sent to overwhelm the target.

5.3 Rule 3: Potential Long-Duration Attack

Condition:

- `dur` > 20
- `sbytes` and `dbytes` are low

Detection: Potential Long-Duration Attack
Explanation: Extended session durations with minimal data transfer may suggest attempts to maintain persistent access or reconnaissance activities.

5.4 Rule 4: Potential Worm Attack

Condition:

- Small `sbytes` and `dbytes`
- Short `dur`

Detection: Potential Worm Attack
Explanation: Worms often operate by sending small packets rapidly, which can be characterized by low byte counts and short connection durations.

5.5 Rule 5: Potential Exploits in OSPF and SCTP Traffic

Condition:

- Protocol-specific anomalies in `OSPF` and `SCTP` traffic

Detection: Potential Exploits
Explanation: Unusual patterns in protocols like OSPF and SCTP may indicate exploitation attempts targeting specific vulnerabilities within these protocols.

```python
# Updated rule function

def detect_intrusion(row):
    # Rule 1: High Source Byte Count with refined conditions
    if row['sbytes'] > 10000 and row['dbytes'] < 5000 and row['dur'] < 7:
        return "Potential Data Exfiltration"

    # Rule 2: Low Destination Byte Count for ICMP traffic
    if row['dbytes'] < 8000 and row['sbytes'] < 2000 and row['proto'] == "ICMP":
        return "Potential Ping Flood"

    # Rule 3: Abnormal Long Duration
    if row['dur'] > 15 and row['sbytes'] < 5000:
        return "Potential Long-Duration Attack"

    # Rule 4: Low Bytes and Duration for Worms
    if row['sbytes'] < 2000 and row['dbytes'] < 500 and row['dur'] < 1.5:
        return "Potential Worm Attack"

    # Rule 5: Protocol-Specific ICMP Worms
    if row['proto'] == "ICMP" and row['sbytes'] < 1500 and row['dbytes'] < 500:
        return "Potential ICMP Worm Attack"

    return "Benign"
```

Figure 2: Intrusion Detection Rules

6 AWS Lambda Function Setup

AWS Lambda serves as the backbone for executing the intrusion detection logic. This section outlines the configuration steps to set up the Lambda function effectively.

6.1 Creating the Lambda Function

1. Log in to the AWS Management Console.
2. Navigate to the **AWS Lambda** service.
3. Click on **Create function**.
4. Select **Author from scratch**.
5. Configure the following settings:
    - **Function name**: `IntrusionDetectionFunction`
    - **Runtime**: Python 3.x
    - **Permissions**: Choose or create an appropriate execution role with necessary permissions.
6. Click **Create function**.

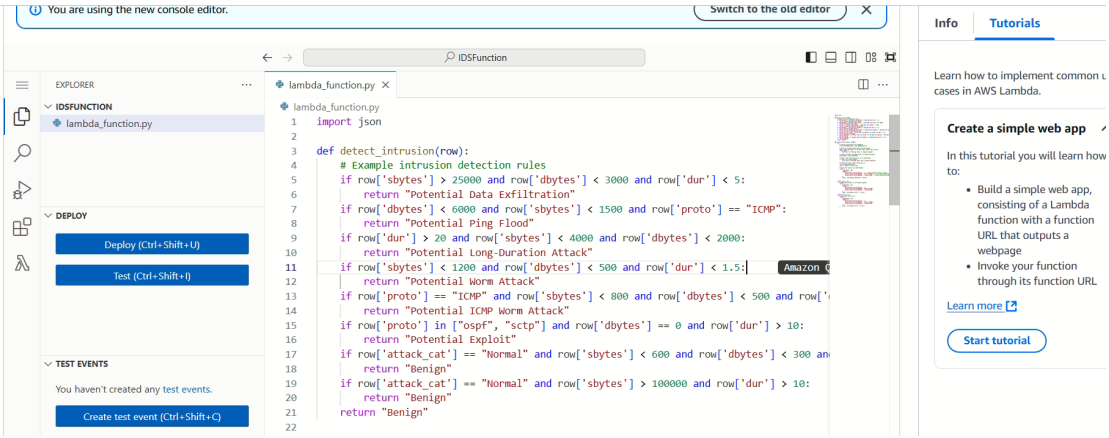

Figure 3: Creating the Lambda Function

6.2 Configuring the Lambda Function

6.2.1 Function Code

While the configuration manual does not include code, ensure that the Lambda function is set up to handle JSON payloads containing network traffic features and return appropriate detection results based on the predefined rules.

6.2.2 Function Structure

The Lambda function should be structured to accept inputs and produce outputs as follows:

Input JSON Structure:

```
{
    "data": {
        "sbytes": 30000,
        "dbytes": 1000,
        "dur": 4.5,
        "proto": "TCP",
        "attack_cat": "Normal"
    }
}
```

Output JSON Structure:

```
{
    "detection": "Potential Data Exfiltration"
}
```

6.2.3 Error Handling

Implement robust error handling to manage missing keys or malformed JSON inputs. Additionally, integrate logging mechanisms to facilitate debugging through AWS CloudWatch.

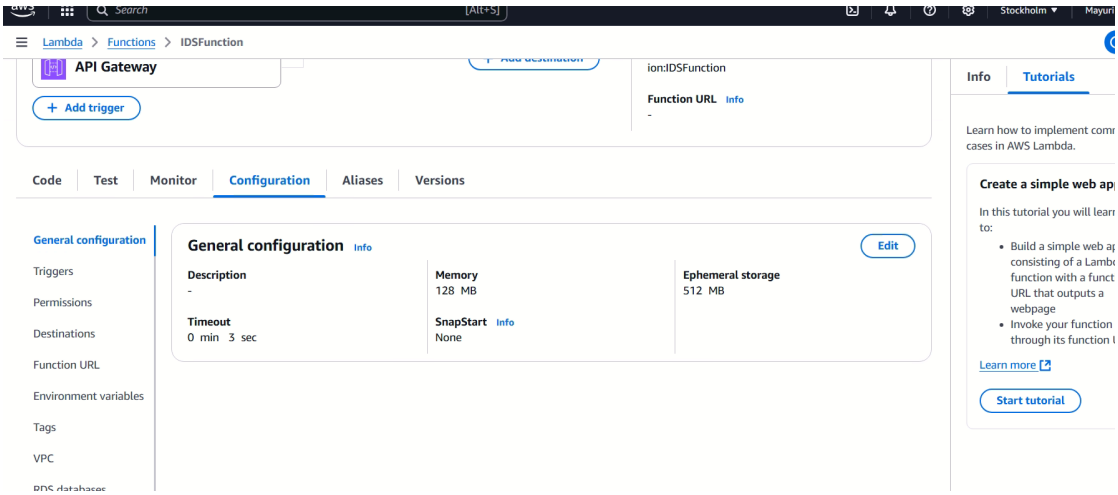

Figure 4: Lambda Function Configuration

6.2.4 Setting Up CORS Headers

To ensure compatibility with external applications, configure the Lambda function to include the following CORS headers in responses:

- `Access-Control-Allow-Origin: *`
- `Access-Control-Allow-Methods: POST, OPTIONS`
- `Access-Control-Allow-Headers: Content-Type`

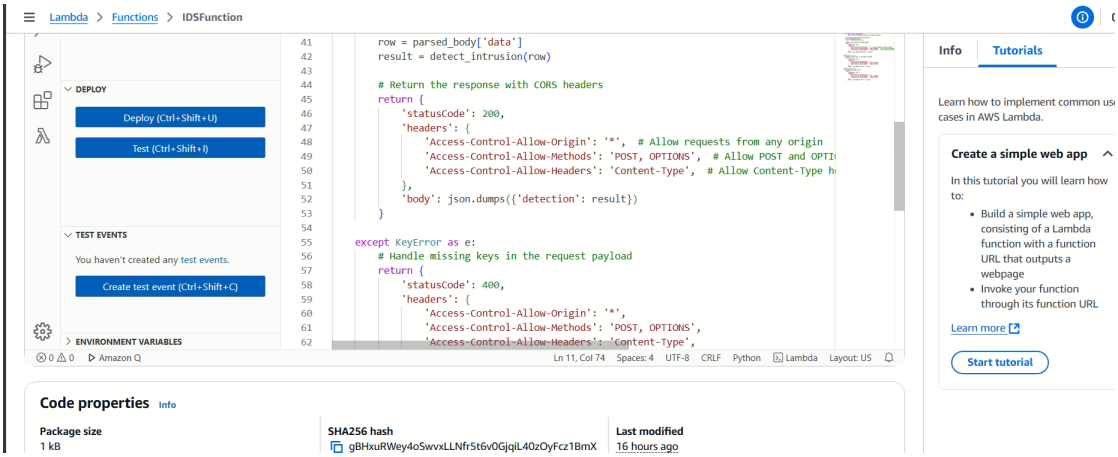

Figure 5: Setting Up CORS Headers in Lambda

7 API Gateway Configuration

AWS API Gateway facilitates interactions between the Lambda function and external applications or users. This section details the steps to set up and configure the API Gateway.

7.1 Creating the REST API

1. Navigate to the **API Gateway** service in the AWS Management Console.
2. Click on **Create API**.
3. Select **REST API** and choose **Build**.
4. Configure the API settings:
   - **API name**: `IntrusionDetectionAPI`
   - **Endpoint Type**: Regional
5. Click **Create API**.

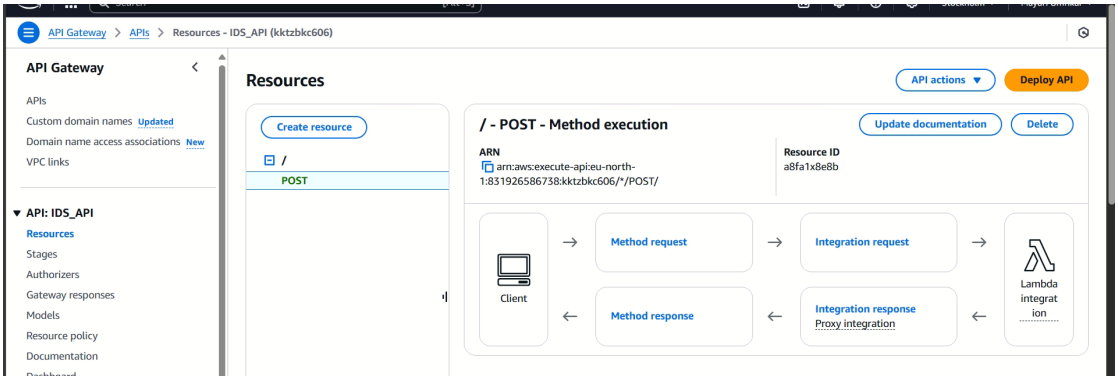

Figure 6: Creating the REST API

7.2 Configuring the Root Resource

7.2.1 Setting Up the / Resource

1. In the API Gateway console, select the created API.
2. Under **Resources**, select the root (/) resource.
3. Click on **Actions** and choose **Create Method**.
4. Select **POST** from the dropdown and click the checkmark.

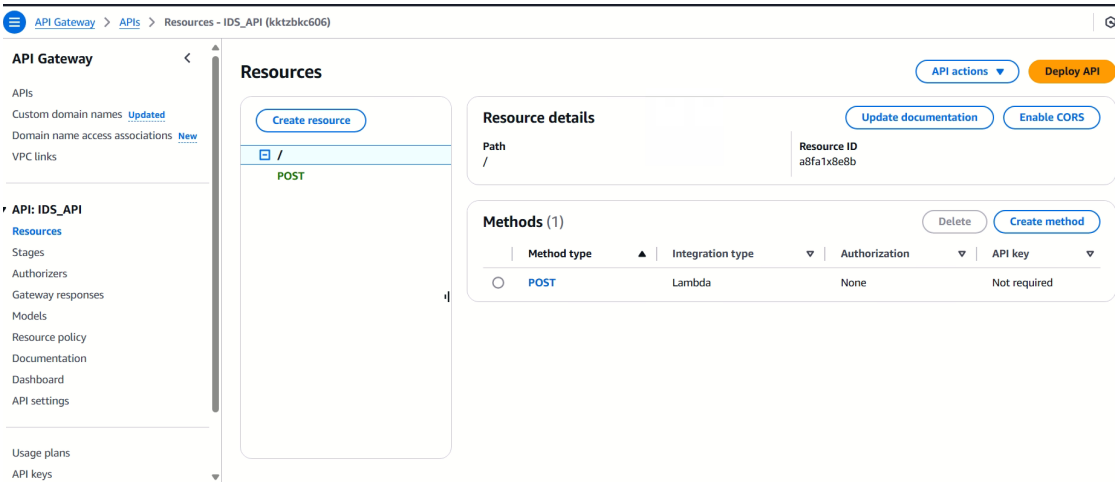

Figure 7: Configuring the Root POST Method

7.2.2 Integrating the POST Method with Lambda

1. In the **POST** method setup, choose **Lambda Function** as the integration type.
2. Select the appropriate region.
3. Enter the name of the Lambda function created earlier (`IntrusionDetectionFunction`).
4. Click **Save** and grant API Gateway permission to invoke the Lambda function.

7.2.3 Setting Response Headers for CORS Compliance

Ensure the following headers are included in the method response:

- `Access-Control-Allow-Origin`: `*`
- `Access-Control-Allow-Methods`: `POST`
- `Access-Control-Allow-Headers`: `Content-Type`

7.3 Deploying the API

7.3.1 Creating a Deployment Stage

1. In the API Gateway console, select **Actions** and choose **Deploy API**.
2. Create a new stage:
    - **Stage name**: `production`
3. Click **Deploy**.

7.3.2 Obtaining the Public Endpoint URL

After deployment, the API Gateway provides a public endpoint URL. This URL is used by external applications or users to interact with the IDS.

Example Endpoint URL:

`https://kktzbkc606.execute-api.eu-north-1.amazonaws.com/production`

8 Conclusion

This paper details the detailed configuration procedure of setting up an Intrusion Detection System on a Fog-to-Cloud computing framework using AWS services, through a rule-based deployment. In this light, preparation of the UNSW-NB15 dataset, definition of intrusion detection rules, setup of AWS Lambda, and configuration of AWS API Gateway can thus enable a reliable system that accurately identifies potential security threats and offers effective responses to such threats. This will ensure that the users will see each step of the configuration process, which will make the deployment smoother.

Further possible improvements could include the addition of machine learning-based models for more dynamic threat detection, additional expansion of the rule set with more attack vectors, and comprehensive monitoring and alerting mechanisms to ensure an optimal security posture.