Project Report

Network Traffic Analyzer

By: Harisankar R Nair

1. Introduction

In today's digital age, understanding and analyzing network traffic is crucial for maintaining the performance, security, and reliability of network systems. The **Network Traffic Analyzer** project aims to provide a tool that helps visualize and analyze network traffic patterns using Python and Flask. This tool leverages synthetic data to demonstrate basic network traffic analysis and can be adapted to analyze real network data.

2. Objective

The primary objectives of the Network Traffic Analyzer project are:

- To develop a web-based application that visualizes network traffic data.
- To create a user-friendly interface that allows users to initiate traffic analysis and view results.
- To provide a framework that can be extended to analyze real network traffic data.

3. Problem Statement

With the increasing complexity of network infrastructures, there is a growing need for tools that can help in understanding and troubleshooting network traffic. Existing solutions often require specialized knowledge and are not always user-friendly. The Network Traffic Analyzer aims to address these challenges by offering an accessible and interactive tool for network traffic analysis, using both synthetic and real data.

4. Working Procedure

4.1 System Architecture

The system comprises a Flask web application that interacts with Python scripts to perform network traffic analysis. The key components are:

- **Flask Web Application**: Handles user requests and serves web pages.
- **Network Traffic Analyzer Module**: Contains the logic for analyzing network traffic and generating visualizations.
- **Templates and Static Files**: Provides the user interface and displays results.

4.2 Data Flow

1. **User Interaction**: The user accesses the home page of the Flask application.

2. **Data Analysis**: Upon clicking the "Run Analysis" button, the application triggers the analysis function.
3. **Visualization**: The analysis results are visualized in a scatter plot and saved as an image file.
4. **Result Display**: The generated image is displayed on the results page.

4.3 Detailed Working

4.3.1 Data Generation

The network_traffic_analyzer.py script contains a function to generate synthetic network traffic data. This includes fields such as source IP network, destination ASN, and flow bytes. The data is used to simulate network traffic for analysis.

4.3.2 Analysis and Visualization

The analyze_traffic() function performs a basic analysis by converting categorical variables and plotting a scatter plot using Matplotlib and Seaborn. The generated plot is saved in the static folder and is displayed to the user.

4.3.3 Web Interface

The Flask application serves two main pages:

- **Home Page (index.html)**: Provides a button to initiate the analysis.
- **Result Page (result.html)**: Displays the analysis result as an image and provides a link to return to the home page.

5. Implementation Details

5.1 Code Overview

- **app.spyder**: The main Flask application script. It handles routing, triggers the analysis function, and serves web pages.
- **network_traffic_analyzer.py**: Contains functions to generate synthetic data and perform traffic analysis.
- **Templates**: HTML files for user interface, including index.html for initiating analysis and result.html for displaying results.
- **Static Files**: Folder to store generated images, such as the analysis plot.

5.2 Example Code

app.spyder

python

Copy code

```
from flask import Flask, render_template, request

import pandas as pd

import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from network_traffic_analyzer import analyze_traffic

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/analyze', methods=['POST'])
def analyze():
    fig = analyze_traffic()
    fig.savefig('static/analysis.png')
    plt.close()
    return render_template('result.html')

if __name__ == '__main__':
    app.run(debug=True)
```

network_traffic_analyzer.py

python

Copy code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

def load_data():
    data = {
        'date': pd.date_range(start='1/1/2023', periods=100),
```

```
        'l_ipn': np.random.randint(0, 255, 100),
        'r_asn': np.random.randint(0, 1000, 100),
        'f': np.random.rand(100) * 1000
    }
    df = pd.DataFrame(data)
    return df


def analyze_traffic():
    df = load_data()
    df['l_ipn'] = df['l_ipn'].astype('category').cat.codes
    df['r_asn'] = df['r_asn'].astype('category').cat.codes
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='l_ipn', y='f', data=df)
    plt.title('Network Traffic Analysis')
    plt.xlabel('Source IP Network')
    plt.ylabel('Flow Bytes (f)')
    return plt
```

6. Libraries, Languages, and Tools Used

6.1 Libraries

- **Flask**: A micro web framework used to create the web application.
- **Pandas**: A library for data manipulation and analysis, used to handle and process data.
- **NumPy**: A library for numerical computing, used for generating random data.
- **Matplotlib**: A plotting library used to create visualizations of the data.
- **Seaborn**: A statistical data visualization library built on top of Matplotlib, used for creating attractive plots.
- **Scikit-Learn**: A machine learning library that provides preprocessing utilities like StandardScaler.

6.2 Languages

- **Python**: The primary programming language used for developing the Flask application and performing data analysis.

6.3 Operating Systems

- **Windows**: The project was developed and tested on a Windows operating system. However, the application is cross-platform and should work on other operating systems such as macOS and Linux as long as the required libraries are installed.

6.4 Tools

- **Spyder or Other IDE/Text Editor**: Used for writing and editing the Python code.
- **Web Browser**: Used for accessing the Flask application and viewing results.
- **Command Line Interface**: Used for running the Flask application and managing dependencies.

7. Conclusion

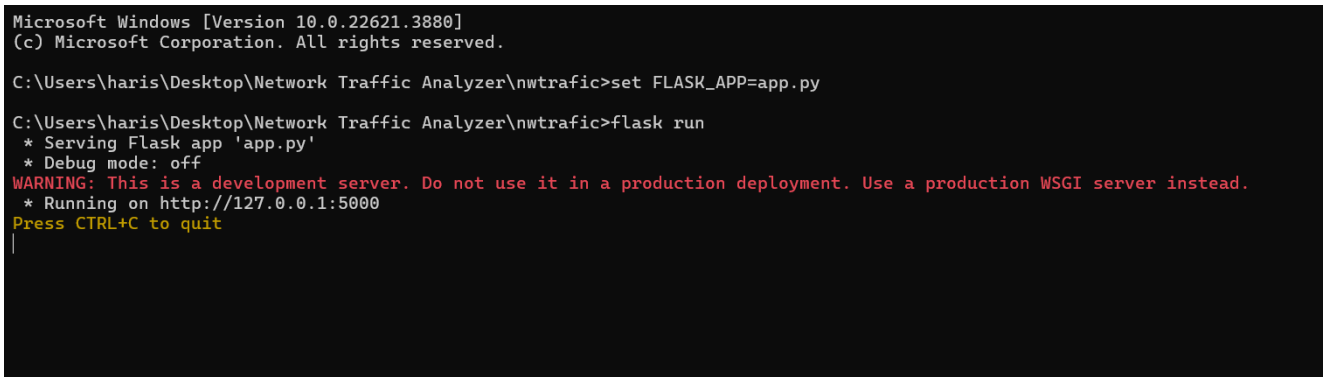
The Network Traffic Analyzer project provides a foundational tool for visualizing and analyzing network traffic data. While the current implementation uses synthetic data, the framework can be adapted for real-world applications by integrating with network traffic data sources. This project demonstrates the use of Python and Flask for web-based data analysis and visualization, offering a basis for further development and enhancement.

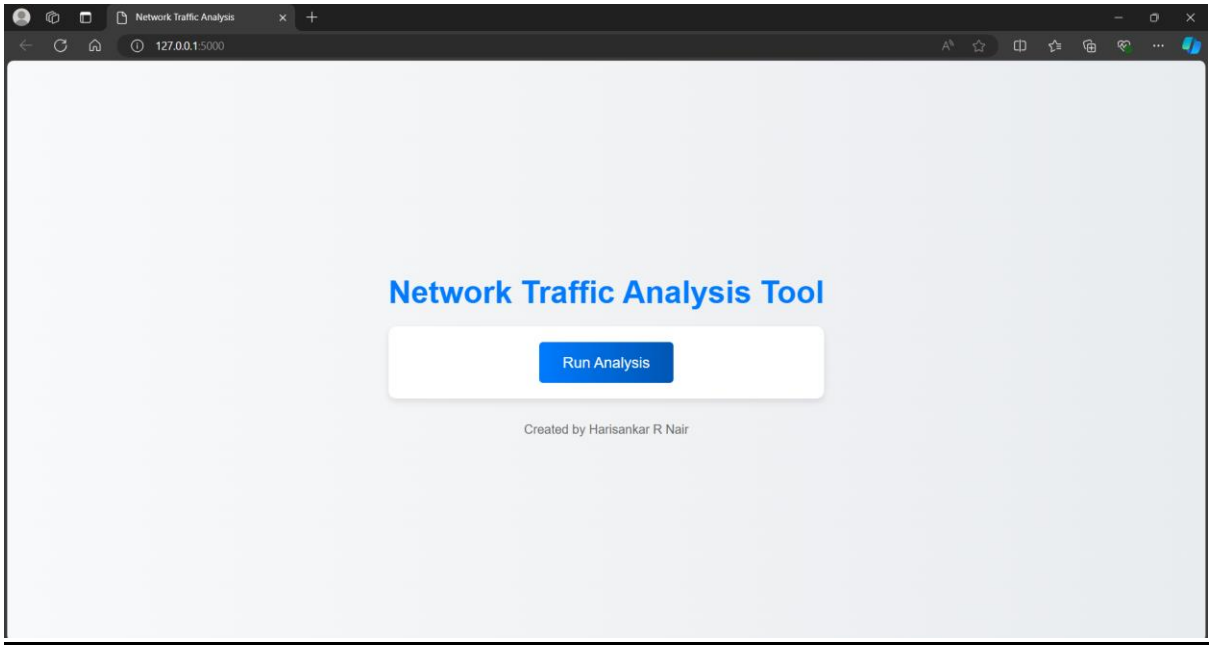
8. Future Work

Future enhancements could include:

- Integrating with real-time network monitoring tools.
- Implementing advanced data analysis techniques, such as anomaly detection.
- Improving the user interface and adding more interactive features.

Output:




Network Traffic Analysis
127.0.0.1:5000
Network Traffic Analysis Tool
Run Analysis
Created by Harisankar R Nair


Analysis Result
127.0.0.1:5000/analyze
Analysis Result
Network Traffic Analysis
Flow Bytes (f)
Source IP Network
Back to Home
Created by Harisankar R Nair