Stream Analytics
Assignment – 1

Network Traffic Analysis Using Silk Suite

Overview

This project involves the analysis and classification of network traffic using the SiLK Suite. The objective is to:

1. Retrieve TCP traffic from a dataset.
2. Classify nodes based on TCP traffic volume.
3. Detect anomalies in the traffic.
4. Generate visual and textual outputs for the analysis.

The dataset used for this project is the `FCCX-silk.tar.gz` file, downloaded from the CERT NetSA Security Suite reference data page.

Step 1: Prerequisites

1. Operating System: Ubuntu 18.04 or higher.
2. Tools:
   - SiLK Suite (installed and configured).
   - Python 3.x with the following libraries:
     - `pandas`
     - `matplotlib`
   - Basic UNIX command-line tools.

Step 2: Steps to be performed

1. Data Setup

❖ Download Data

Download the `FCCX-silk.tar.gz` file from [SiLK Reference Data]
(https://tools.netsa.cert.org/silk/referencedata.html#).

❖ Extract Data

```
cd ~/Downloads
gzip -d -c FCCX-silk.tar.gz | tar xf -
cd FCCX-silk
export SILK_DATA_ROOTDIR=$(pwd)
```

2. TCP Traffic Retrieval

❖ Filter TCP Traffic

```
rwfilter --proto=6 --type=in,inweb,out,outweb \
    --start-date=2015/06/02T13 --end-date=2015/06/18T18 \
    --pass-destination=tcp_traffic.rw
```

❖ Summarize TCP Traffic

```
rwstats tcp_traffic.rw --fields=sip,dip --count=10
```

❖ Convert Flow Data to CSV

```
rwcut --fields=sip,dip,packets,stime --no-titles --output-path=tcp_data.csv
tcp_traffic.rw
```

3. Node Classification

Using Python, classify nodes based on their TCP traffic volume.

```
import pandas as pd
import matplotlib.pyplot as plt

# Load data
data = pd.read_csv("tcp_data.csv", names=["SourceIP", "DestinationIP", "Packets",
"StartTime"])

# Define classification thresholds
def classify_traffic(packets):
    if packets <= 100:
        return "Low"
    elif 101 <= packets <= 1000:
        return "Medium"
    else:
        return "High"

# Apply classification
data["TrafficClass"] = data["Packets"].apply(classify_traffic)

# Save classified data
data.to_csv("classified_data.csv", index=False)

# Plot classification distribution
traffic_counts = data["TrafficClass"].value_counts()
traffic_counts.plot(kind="bar")
plt.title("Node Traffic Classification")
plt.xlabel("Traffic Class")
plt.ylabel("Number of Nodes")
plt.savefig("classification_plot.png")
plt.show()
```

4. Anomaly Detection

Detect nodes with anomalously high traffic (e.g., >1000 packets/sec).

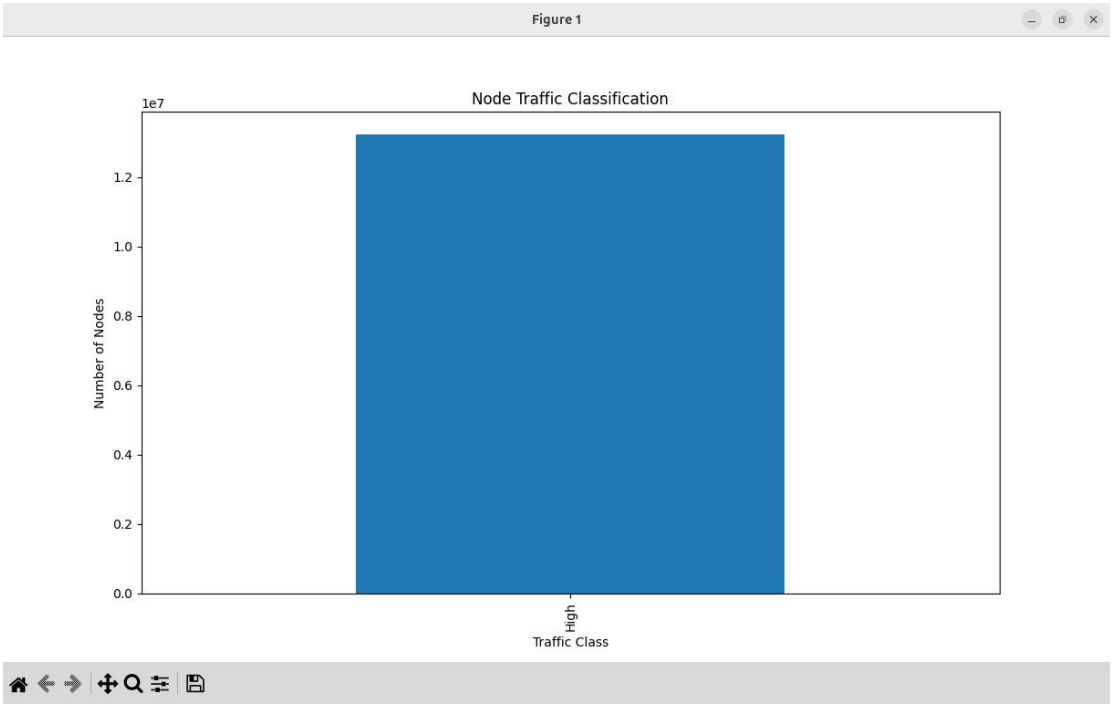
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data
data = pd.read_csv("tcp_data.csv", names=["SourceIP", "DestinationIP", "Packets",
"StartTime"])


# Detect anomalies
anomalies = data[data["Packets"] > 1000]

# Save anomalies
anomalies.to_csv("anomalies.csv", index=False)

# Plot anomalies
plt.scatter(anomalies["SourceIP"], anomalies["Packets"], color="red")
plt.title("Anomalous Nodes")
plt.xlabel("Source IP")
plt.ylabel("Packets")
plt.savefig("anomaly_plot.png")
plt.show()
```

5. Outputs

- ❖ Textual Outputs:
  - **tcp_data.csv:** Raw TCP traffic data
  - **classified_data.csv:** Classified TCP traffic
  - **anomalies.csv:** Detected anomalies

- ❖ Graphical Outputs:

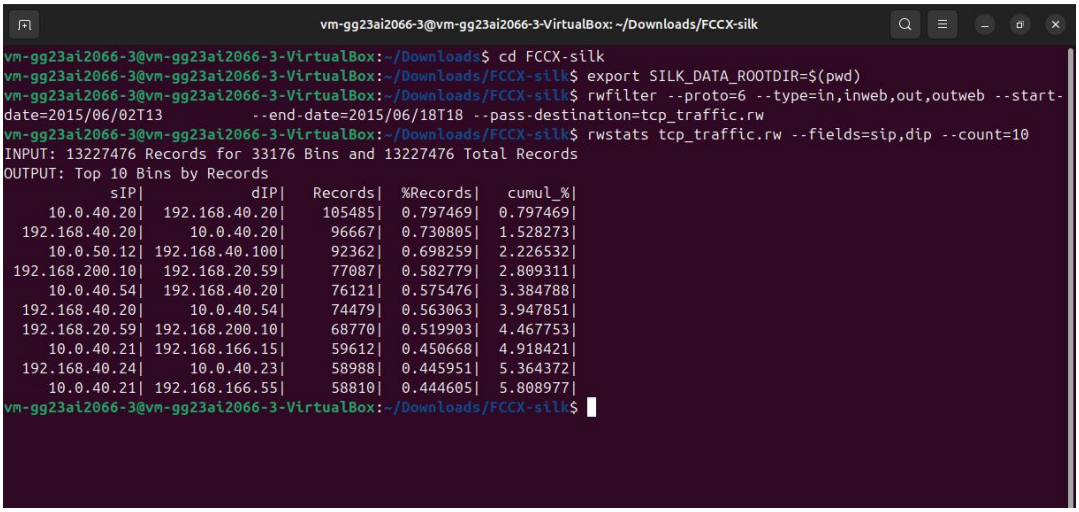
- **classification_plot.png:** Bar chart of traffic classification.



- **anomaly_plot.png:** No anomalies found (>1K).

❖ Terminal Screenshot:

Show summary output of the `rwstats` command.



Step 3: How to Run the Code

1. Ensure all required tools and libraries are installed.
2. Place the `tcp_data.csv` file in the same directory as the Python scripts.
3. Run the Python scripts for classification and anomaly detection:

```
python classify_nodes.py
python detect_anomalies.py
```

4. View the generated CSV files and plots in the current directory.