



UNIFIED MENTOR
YOUR SKILL. SUCCESS & JOURNEY

Cybersecurity: Suspicious Web Threat Interactions

Reported By: Adhwai Sherina
Rajeev

Internship ID: UMIP276956

Contents

Introduction.....	3
Research Questions:.....	3
Data Preparation.....	4
Feature Engineering:.....	6
Data Cleaning	7
Exploratory Data Analysis.....	9
Predictive Analysis	14
Anomaly Detection using Isolation Forest	14
Results:.....	16
Conclusion.....	20

Introduction

In the age of digital communication, monitoring network traffic is critical for identifying potential threats and ensuring efficient operations.

This project focuses on the analysis of a cloud-based network traffic dataset, aiming to extract meaningful patterns through Exploratory Data Analysis (EDA) and detect anomalies indicative of suspicious activities.

The key objectives are:

- Understand the underlying structure and characteristics of the network traffic.
- Perform feature engineering to enrich the dataset.
- Visualize important relationships and patterns.
- Detect anomalous network sessions that may represent cyber threats.

Research Questions:

- Which source IP addresses and countries contribute most to the network traffic?
- At what time of day does the network experience the highest volume of traffic?
- Are there any unusual patterns or anomalies in the network traffic?
- Which source IP addresses and countries are most associated with suspicious activity?
- How can anomaly detection techniques help identify potential security threats in network traffic?

Data Preparation

Taking a look at the dataset:



```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

[2] df=pd.read_csv('/content/drive/MyDrive/Unified Projects/CloudWatch_Traffic_Web_Attack.csv')

df.head()
```

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_code	protocol	response.code	dst_port	dst_ip	rule_names	observation_name	source.meta	source.name
0	5602	12990	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	147.161.161.82	AE	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversary Infrastructure Interaction	AWS_VPC_Flow	prod_webserve
1	30912	18186	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.33.6	US	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversary Infrastructure Interaction	AWS_VPC_Flow	prod_webserve
2	28506	13468	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.212.255	CA	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversary Infrastructure Interaction	AWS_VPC_Flow	prod_webserve
3	30546	14278	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	136.226.64.114	US	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversary Infrastructure Interaction	AWS_VPC_Flow	prod_webserve
4	6526	13892	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.240.79	NL	HTTPS	200	443	10.138.69.97	Suspicious Web Traffic	Adversary Infrastructure Interaction	AWS_VPC_Flow	prod_webserve

Shape of the dataset:

	df.shape
	(282, 20)

Rows: 282

Columns: 20

Dataset info:

```

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 282 entries, 0 to 281
Data columns (total 16 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   bytes_in                             282 non-null    int64
1   bytes_out                             282 non-null    int64
2   creation_time                         282 non-null    object
3   end_time                             282 non-null    object
4   src_ip                               282 non-null    object
5   src_ip_country_code                  282 non-null    object
6   protocol                             282 non-null    object
7   response.code                        282 non-null    int64
8   dst_port                             282 non-null    int64
9   dst_ip                               282 non-null    object
10  rule_names                           282 non-null    object
11  observation_name                      282 non-null    object
12  source.meta                           282 non-null    object
13  source.name                           282 non-null    object
14  time                                  282 non-null    object
15  detection_types                       282 non-null    object
dtypes: int64(4), object(12)
memory usage: 35.4+ KB

```

The dataset contains network traffic logs with the following key columns:

- creation_time, end_time, time — timestamps associated with session initiation, termination, and event recording.
- bytes_in, bytes_out — amount of data transferred inbound and outbound.
- src_ip_country_code, src_ip, dst_ip — geographical and IP information.

Conversion of time fields into datetime format for accurate time-based operations.

```

df['creation_time'] = pd.to_datetime(df['creation_time'])
df['end_time'] = pd.to_datetime(df['end_time'])
df['time'] = pd.to_datetime(df['time'])

```

Feature Engineering:

- **Session Duration:** Calculated as the difference between end_time and creation_time in seconds.
- **Average Packet Size:** Estimated by dividing the total bytes transferred by session duration.

```
df['session_duration'] = (df['end_time'] - df['creation_time']).dt.total_seconds()  
df['avg_packet_size'] = (df['bytes_in'] + df['bytes_out']) / df['session_duration']
```

- **Hour of Day:** Extracted from the time field to capture temporal patterns.

```
df['hour_of_day'] = df['time'].dt.hour
```

Data Cleaning

Handling Duplicates and Missing Values:

```
df.isnull().sum()
```

	0
bytes_in	0
bytes_out	0
creation_time	0
end_time	0
src_ip	0
src_ip_country_code	0
protocol	0
response.code	0
dst_port	0
dst_ip	0
rule_names	0
observation_name	0
source.meta	0
source.name	0
time	0
detection_types	0

dtype: int64

```
[5] df.duplicated().sum()
```

```
np.int64(0)
```

There is no null values or duplicate rows in the data frame.

Outlier Analysis:

- Significant outliers were found in bytes_in, bytes_out, and session_duration.

```
# Selecting numerical columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
# Detecting outliers using IQR
outlier_summary = {}
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    outlier_summary[col] = {
        'lower_bound': lower_bound,
        'upper_bound': upper_bound,
        'outlier_count': len(outliers),
        'percentage': len(outliers) / len(df) * 100
    }

# Print the summary
for col, stats in outlier_summary.items():
    print(f"\n📊 Column: {col}")
    print(f"  Outliers: {stats['outlier_count']} ({stats['percentage']:.2f}%)")
    print(f"  Lower Bound: {stats['lower_bound']}")
    print(f"  Upper Bound: {stats['upper_bound']}")
```

📊 Column: bytes_in
Outliers: 40 (14.18%)
Lower Bound: -32795.75
Upper Bound: 69010.25

📊 Column: bytes_out
Outliers: 37 (13.12%)
Lower Bound: -11564.25
Upper Bound: 48985.75

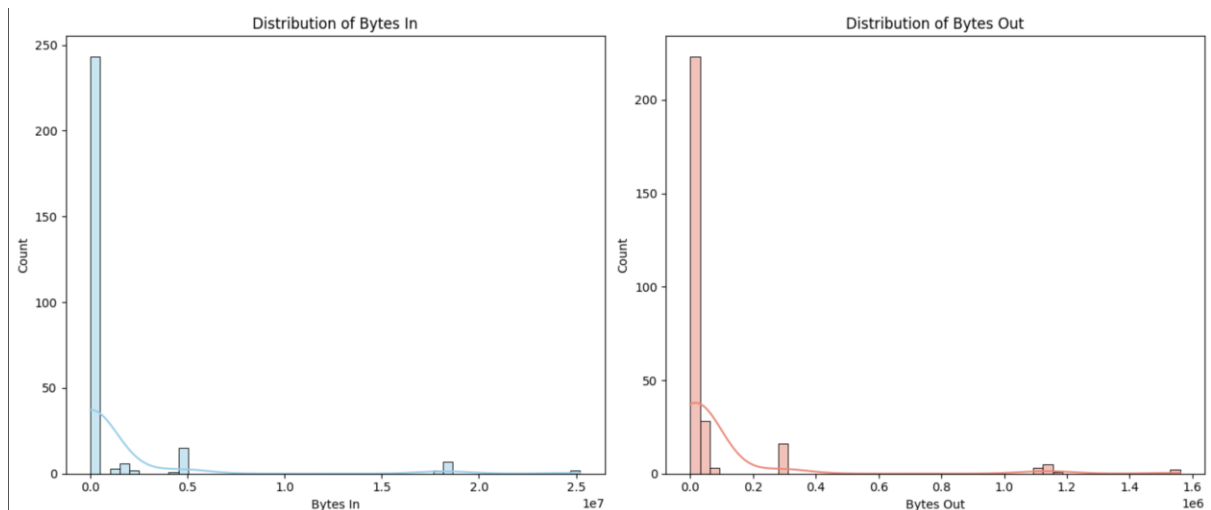
- For the EDA, outliers were retained to preserve data authenticity, especially important in security-related datasets where extreme values might indicate attacks.

Exploratory Data Analysis

A detailed EDA was conducted to gain insights into the behaviour of network traffic.

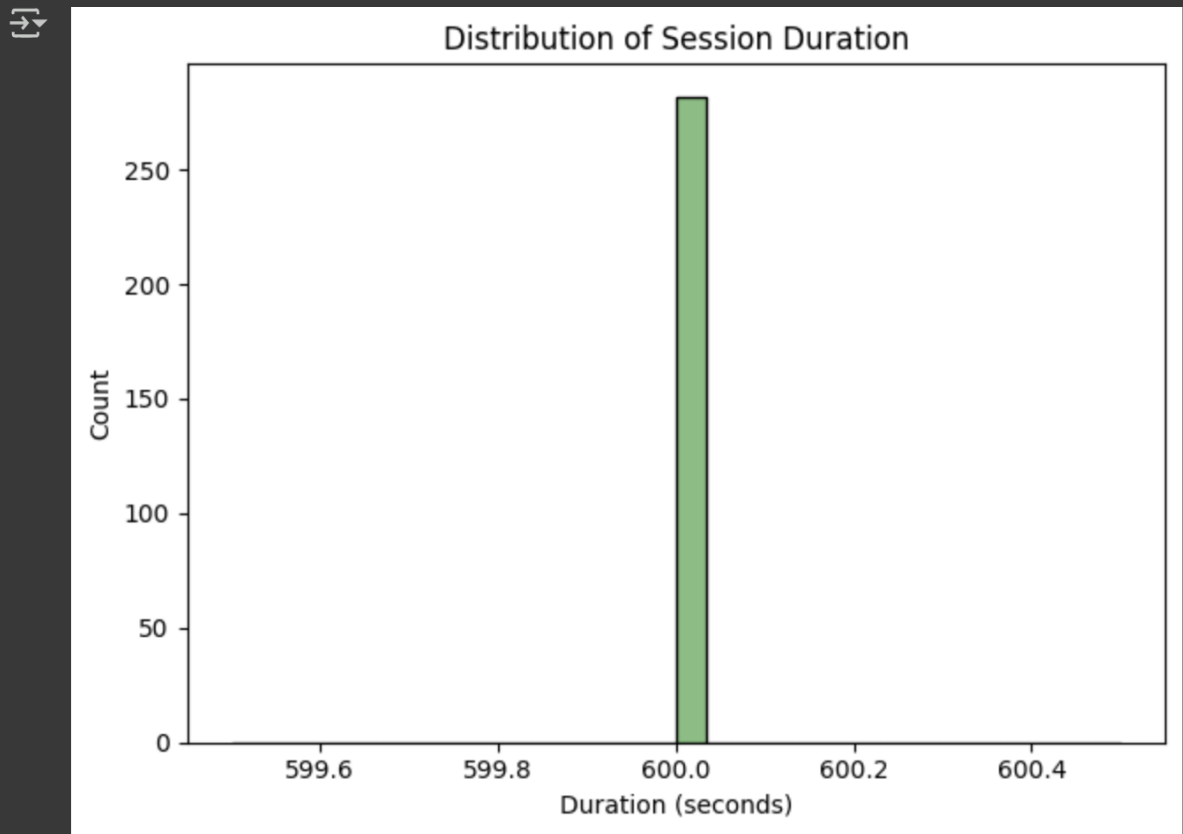
- Distribution of Traffic Volumes

```
plt.figure(figsize=(14, 6))
# bytes_in distribution
plt.subplot(1, 2, 1)
sns.histplot(df['bytes_in'], bins=50, kde=True, color='skyblue')
plt.title('Distribution of Bytes In')
plt.xlabel('Bytes In')
# bytes_out distribution
plt.subplot(1, 2, 2)
sns.histplot(df['bytes_out'], bins=50, kde=True, color='salmon')
plt.title('Distribution of Bytes Out')
plt.xlabel('Bytes Out')
plt.tight_layout()
plt.show()
```



- Session Durations

```
sns.histplot(df['session_duration'], kde=True, bins=30, color='green')  
plt.title('Distribution of Session Duration')  
plt.xlabel('Duration (seconds)')  
plt.tight_layout()  
plt.show()
```




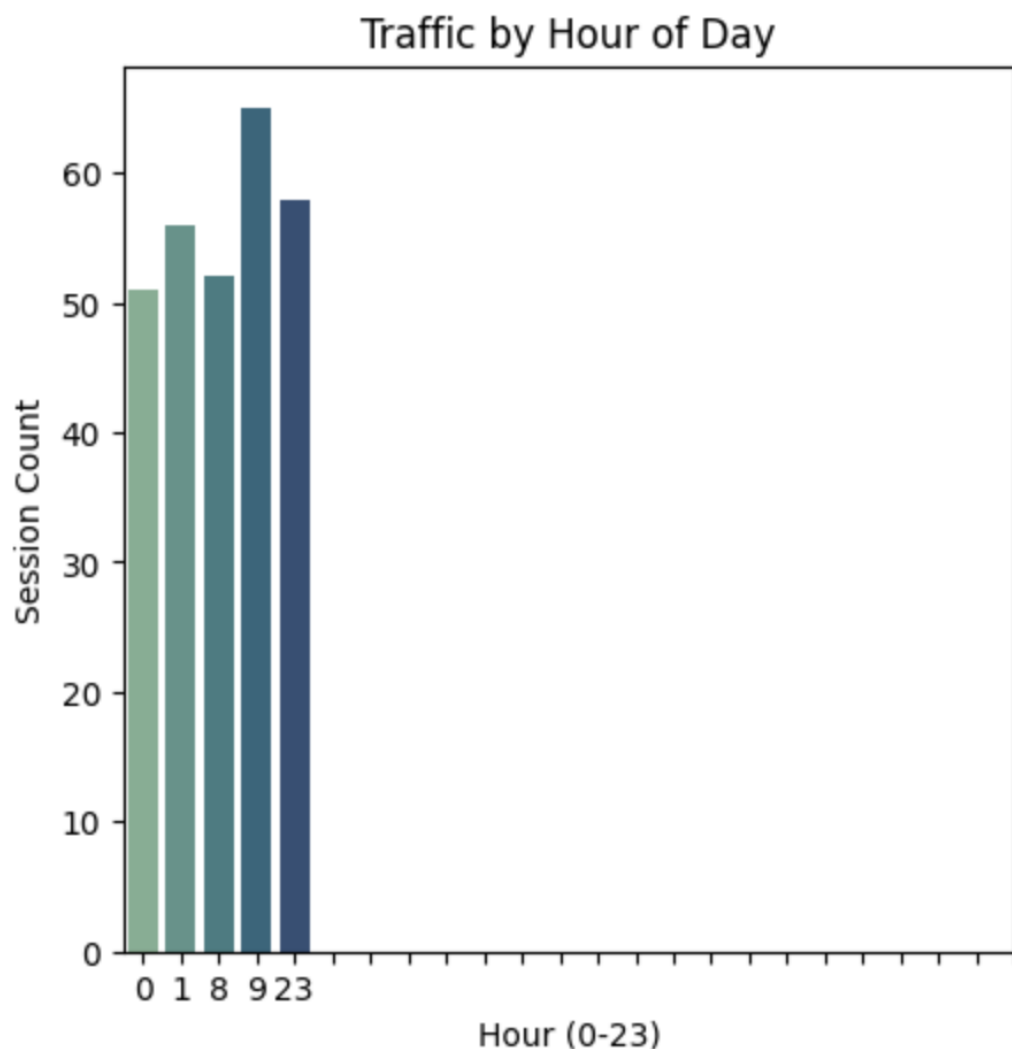
- Top Source Countries



- Traffic Over Each Hour

```
plt.figure(figsize=(5,5))
sns.countplot(x='hour_of_day', data=df, palette="crest")
plt.title('Traffic by Hour of Day')
plt.xlabel('Hour (0-23)')
plt.ylabel('Session Count')
plt.xticks(range(24))
plt.show()
```

 <ipython-input-34-c1b652961d3f>:2: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will
sns.countplot(x='hour_of_day', data=df, palette="crest")



Top 10 Source IP Frequencies

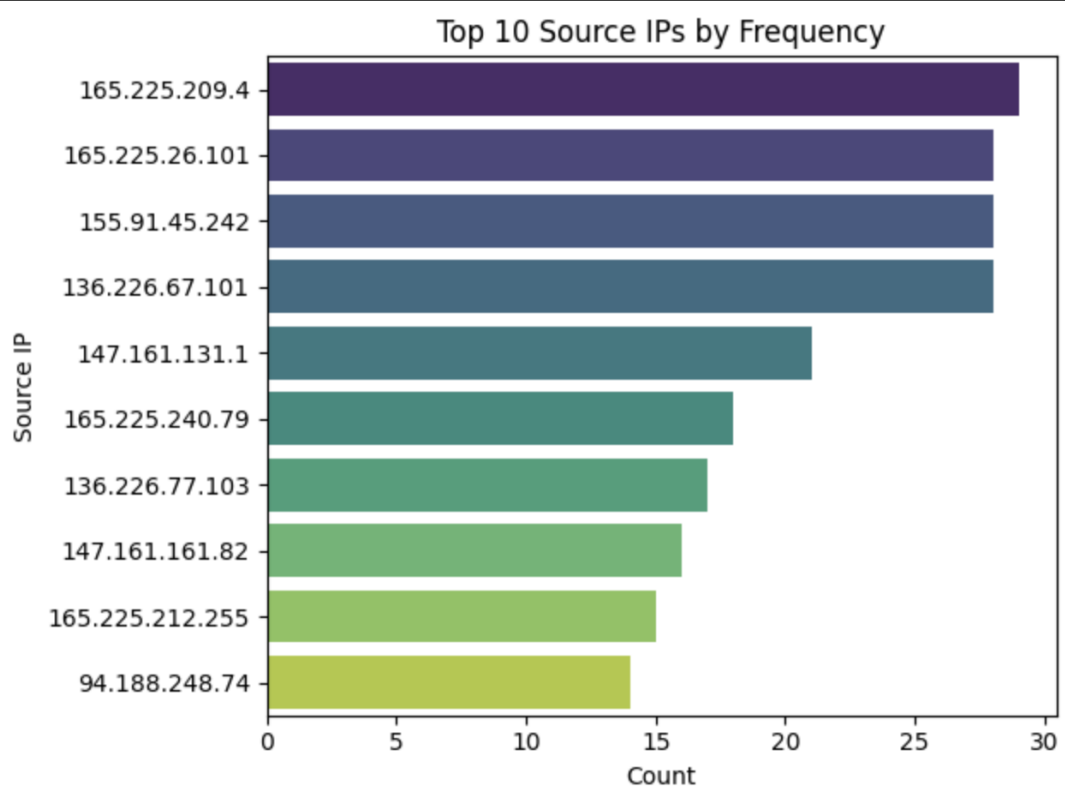
```

top_ips = df['src_ip'].value_counts().head(10)
sns.barplot(x=top_ips.values, y=top_ips.index, palette='viridis')
plt.title('Top 10 Source IPs by Frequency')
plt.xlabel('Count')
plt.ylabel('Source IP')
plt.tight_layout()
plt.show()

```

<ipython-input-16-61d2b5305534>:2: FutureWarning:
 Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0

```
sns.barplot(x=top_ips.values, y=top_ips.index, palette='viridis')
```



Predictive Analysis

Anomaly Detection using Isolation Forest

Given the critical nature of identifying suspicious activities, **anomaly detection** was chosen as the predictive modelling approach.

Methodology:

- Selected key features: bytes_in, bytes_out, session_duration, avg_packet_size.
- Applied the **Isolation Forest** algorithm:
 - contamination=0.1 parameter assumed roughly 10% of sessions might be anomalous.
 - Each session was labelled either "**Highly Suspicious**" or "**Normal**".

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer

# Time of Day
df['hour_of_day'] = df['time'].dt.hour

# Interaction count per IP
ip_interaction_counts = df['src_ip'].value_counts()
df['interaction_count'] = df['src_ip'].map(ip_interaction_counts)

# Selecting features for modeling
features = df[['bytes_in', 'bytes_out', 'session_duration', 'avg_packet_size',
               'hour_of_day', 'interaction_count', 'src_ip_country_code']]

# Preprocessing: Scaling and One-Hot Encoding
# One-hot encode country codes, scaling the rest
numeric_features = ['bytes_in', 'bytes_out', 'session_duration', 'avg_packet_size',
                    'hour_of_day', 'interaction_count']
categorical_features = ['src_ip_country_code']
# Creating the transformer pipeline
preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), numeric_features),
    ('cat', OneHotEncoder(sparse_output=False), categorical_features)
])
# Fit and transform the features
X_scaled = preprocessor.fit_transform(features)

# Get transformed column names
encoded_cat_names = preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features)
all_feature_names = numeric_features + list(encoded_cat_names)
# Creating the final DataFrame
df_features = pd.DataFrame(X_scaled, columns=all_feature_names, index=df.index)
# Show the first few rows
print(df_features.head())
```

	bytes_in	bytes_out	session_duration	avg_packet_size	hour_of_day	\
0	-0.288219	-0.281223	0.0	-0.287850	1.766389	
1	-0.282108	-0.260804	0.0	-0.280910	1.766389	
2	-0.282689	-0.279344	0.0	-0.282531	1.766389	
3	-0.282197	-0.276161	0.0	-0.281883	1.766389	
4	-0.287996	-0.277678	0.0	-0.287435	1.766389	

	interaction_count	src_ip_country_code_AE	src_ip_country_code_AT	\
0	-0.445092	1.0	0.0	
1	-0.936349	0.0	0.0	
2	-0.567907	0.0	0.0	
3	-0.813535	0.0	0.0	
4	-0.199464	0.0	0.0	

	src_ip_country_code_CA	src_ip_country_code_DE	src_ip_country_code_IL	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	1.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	src_ip_country_code_NL	src_ip_country_code_US
0	0.0	0.0
1	0.0	1.0
2	0.0	0.0
3	0.0	1.0
4	1.0	0.0

In this section, additional feature engineering and pre-processing steps were performed to prepare the data for machine learning models:

- **New Features Created:**
 - hour_of_day: Extracted from the time column to capture patterns based on the time of session initiation.
 - interaction_count: Number of interactions initiated by each source IP address (src_ip), representing activity levels.
- **Feature Selection:**
 - Key features were selected, including bytes_in, bytes_out, session_duration, avg_packet_size, hour_of_day, interaction_count, and src_ip_country_code.
- **Pre-processing Pipeline:**
 - **Numerical Features** (bytes_in, bytes_out, etc.) were **standardized** using StandardScaler to ensure they have a mean of 0 and a standard deviation of 1, improving model performance.
 - **Categorical Feature** (src_ip_country_code) was **one-hot encoded** using OneHotEncoder to handle country code categories numerically.
- **Transformation:**
 - The numerical and categorical transformations were combined into a ColumnTransformer.

- After applying transformations, a new feature matrix `df_features` were created, ready for model training.

This structured pre-processing ensures that the machine learning models can efficiently learn from the data without being biased by scale differences or categorical encodings.

```
from sklearn.ensemble import IsolationForest

features = df[['bytes_in', 'bytes_out', 'session_duration', 'avg_packet_size']]
model = IsolationForest(contamination=0.1, random_state=42)
df['anomaly'] = model.fit_predict(features)
df['anomaly'] = df['anomaly'].map({1: 'Normal', -1: 'Highly Suspicious'})
```

In this section, an **anomaly detection model** was implemented to identify unusual network sessions that may indicate suspicious activity:

- **Feature Selection:**
 - Key traffic-related features (`bytes_in`, `bytes_out`, `session_duration`, and `avg_packet_size`) were selected as inputs for the model.
- **Modelling:**
 - An **Isolation Forest** model was used, which is well-suited for unsupervised anomaly detection in high-dimensional datasets.
 - The contamination parameter was set to 0.1, assuming approximately 10% of the data may be anomalous.
 - The model was fitted on the selected features to learn normal behaviour patterns and isolate outliers.
- **Labelling Results:**
 - After prediction, the sessions were labelled as either:
 - 'Normal' (for regular behaviour)
 - 'Highly Suspicious' (for detected anomalies)

This approach helps flag potential cybersecurity threats or unusual traffic behaviour without requiring labelled attack data.

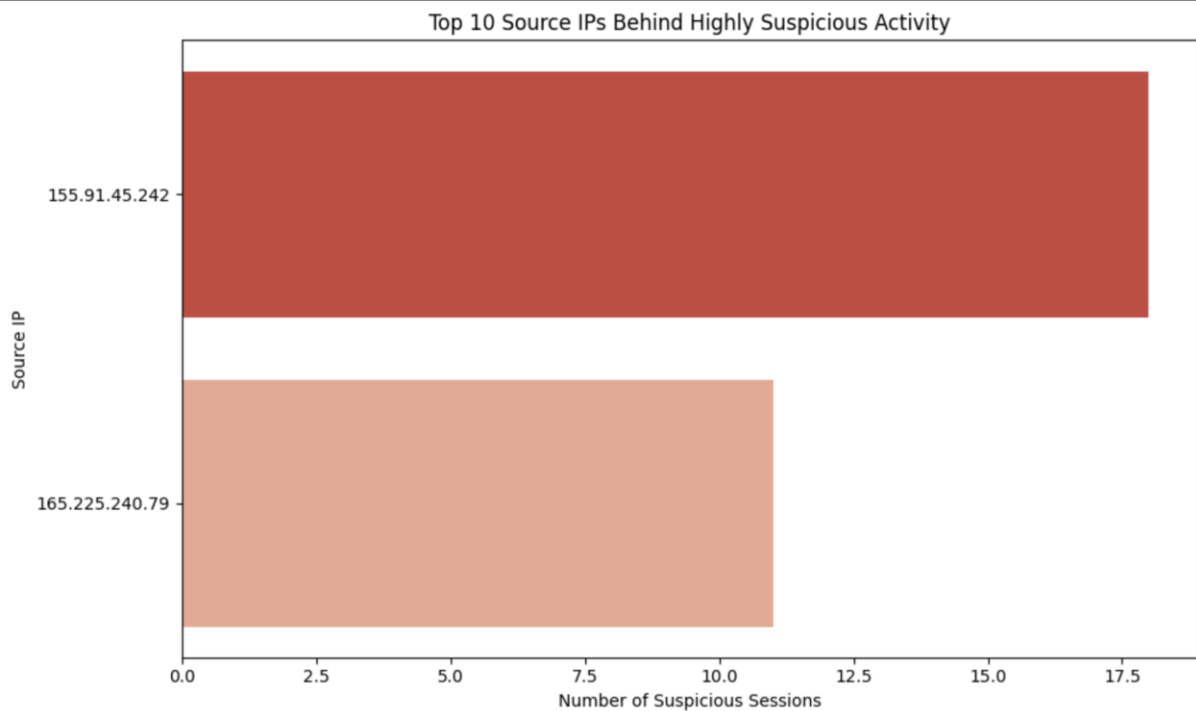
Results:

- Highly Suspicious Source IP's


```

top_suspicious_ips = df[df['anomaly'] == 'Highly Suspicious']['src_ip'].value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_suspicious_ips.values, y=top_suspicious_ips.index, palette='Reds_r')
plt.title("Top 10 Source IPs Behind Highly Suspicious Activity")
plt.xlabel("Number of Suspicious Sessions")
plt.ylabel("Source IP")
plt.tight_layout()
plt.show()

```



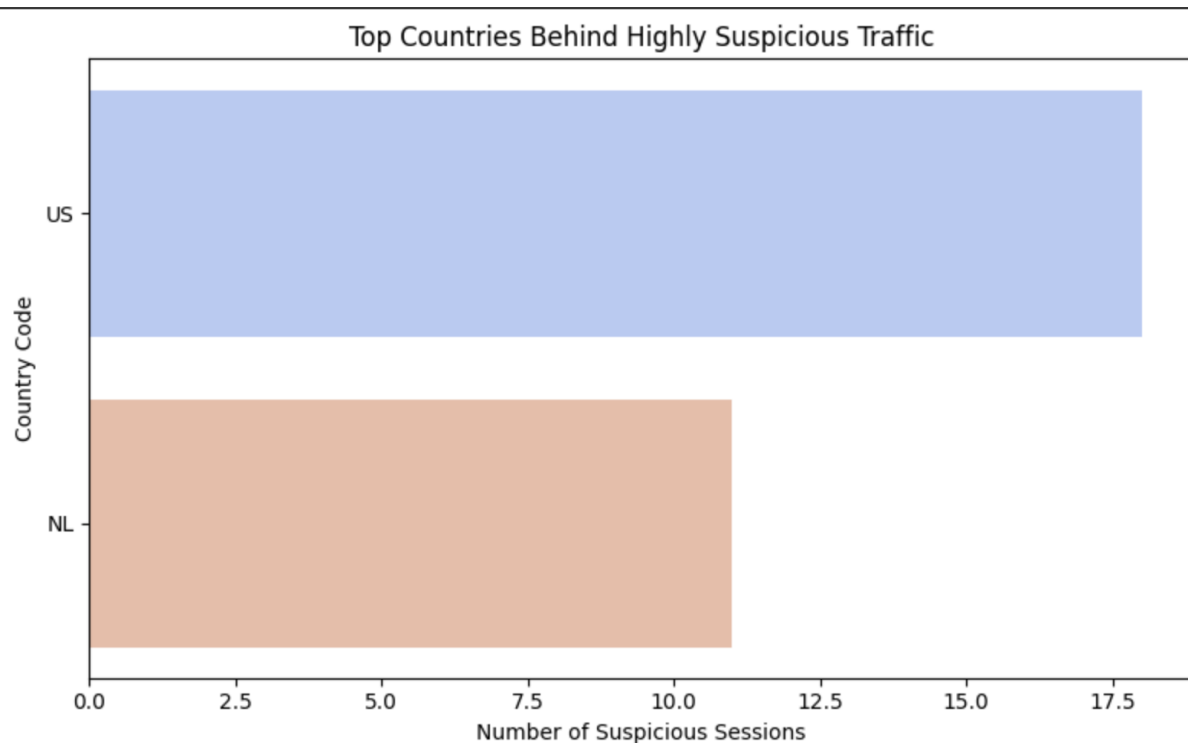
This graph identifies the **top 10 source IP addresses** responsible for the highest number of sessions labelled as **"Highly Suspicious"** by the anomaly detection model.

- Top Suspicious Countries

```

[40] top_suspicious_countries = df[df['anomaly'] == 'Highly Suspicious']['src_ip_country_code'].value_counts().head(10)
plt.figure(figsize=(8, 5))
sns.barplot(x=top_suspicious_countries.values, y=top_suspicious_countries.index, palette='coolwarm')
plt.title("Top Countries Behind Highly Suspicious Traffic")
plt.xlabel("Number of Suspicious Sessions")
plt.ylabel("Country Code")
plt.tight_layout()
plt.show()

```



This graph highlights the **top 10 countries** from which the highest number of **"Highly Suspicious"** network sessions originated, based on the anomaly detection results.

- Percentage distribution of highly suspicious network traffic based on the source IP country code.

```
# Country-based anomaly percentage
anomaly_country_pct = (
    df[df['anomaly'] == 'Highly Suspicious']['src_ip_country_code']
    .value_counts(normalize=True) * 100
).round(2)

print("\n Suspicious Traffic by Country (%):\n")
print(anomaly_country_pct)
```

```
\n Suspicious Traffic by Country (%):

src_ip_country_code
US      62.07
NL      37.93
Name: proportion, dtype: float64
```

```

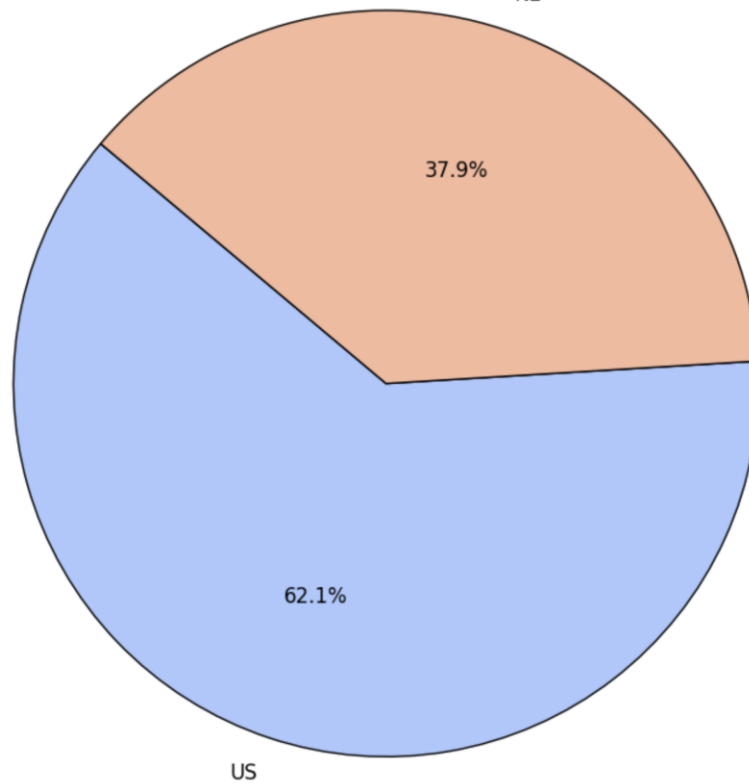
# Create a Pie Chart for Suspicious Traffic by Country
plt.figure(figsize=(10, 7))
colors = sns.color_palette('coolwarm', len(anomaly_country_pct))

plt.pie(anomaly_country_pct,
        labels=anomaly_country_pct.index,
        autopct='%1.1f%%',
        startangle=140,
        colors=colors,
        textprops={'fontsize': 12},
        wedgeprops={'edgecolor': 'black', 'linewidth': 1})

plt.title("Distribution of Highly Suspicious Traffic by Country", fontsize=16, fontweight='bold')
plt.axis('equal') |
plt.tight_layout()
plt.show()

```

Distribution of Highly Suspicious Traffic by Country



Conclusion

The exploratory data analysis and anomaly detection provided several key insights into the network traffic dataset:

- **Session Duration and Destination Port:**
Every network session lasted exactly **600 seconds**, and all connections were directed to **destination port 443**, indicating that the traffic likely involved encrypted HTTPS communication.
- **Source Country Analysis:**
The **United States (US)** emerged as the dominant source of network traffic, with the highest volume of sessions originating from US-based IP addresses.
- **Traffic Patterns Over Time:**
Analysis showed that the **9th hour of the day** (9 AM) experienced the **highest amount of traffic**, suggesting peak network activity during that time window.
- **Frequent and Suspicious IP Addresses:**
 - The **most frequent source IP address** across all traffic was **165.225.209.4**, indicating heavy activity from this IP.
 - Anomaly detection revealed **two highly suspicious source IP addresses**:
 - **155.91.45.242** (most suspicious)
 - **165.225.240.79** (the second highly suspicious IP)
- **Country-Level Suspicious Activity:**
 - Only **two countries** were found to be sources of highly suspicious traffic: **United States** and **Netherlands**.
 - Of the suspicious sessions, **62.1%** originated from the United States, while **37.9%** originated from the Netherlands.
- **Anomaly Detection and Security Insight:**
 - The Isolation Forest model was effective in detecting a small subset of network sessions that behaved abnormally.
 - These findings highlight specific IP addresses and countries that warrant closer monitoring and possibly stricter security controls.

In summary, the analysis successfully identified the main sources of normal and suspicious traffic, peak activity times, and potential security threats. The results provide actionable insights that can guide future security measures, traffic monitoring strategies, and anomaly detection improvements.