In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import LabelEncoder

# Step 1: Load the dataset
data = pd.read_csv(r"C:\Users\srush\OneDrive\Desktop\IIT Chicago\ACS\Project\combined_data.csv")

# Step 2: Encode categorical labels if 'Label' exists in the data
if 'Label' in data.columns:
    label_encoder = LabelEncoder()
    data['Label'] = label_encoder.fit_transform(data['Label'])

# Step 3: Convert all columns to numeric, handling any non-numeric data
data = data.apply(pd.to_numeric, errors='coerce')
data = data.fillna(0)  # Fill NaNs, which may arise from non-numeric conversion

# Step 4: Replace infinite values (if any) with NaN and then fill with 0
data.replace([float('inf'), float('-inf')], float('nan'), inplace=True)
data.fillna(0, inplace=True)

# Step 5: Save the label column (if it exists) and drop it for anomaly detection
if 'Label' in data.columns:
    labels = data['Label']  # Save the labels separately
    data = data.drop(columns=['Label'])
else:
    labels = None  # If no 'Label' column exists, labels is set to None

# Step 6: Fit Isolation Forest on the data
contamination = 0.05  # Adjust this value based on your dataset
isolation_forest = IsolationForest(contamination=contamination, random_state=42)
anomaly_scores = isolation_forest.fit_predict(data)

# Step 7: Convert anomaly scores to a binary classification
# Instead of directly using the predict results, we'll use decision_function
anomaly_scores = isolation_forest.decision_function(data)
threshold = np.percentile(anomaly_scores, contamination * 100)
data['Anomaly'] = np.where(anomaly_scores <= threshold, 1, 0)

# Optional: Restore the original labels to the dataset (if labels exist)
if labels is not None:
    data['Label'] = labels

# Step 8: Save the processed and labeled data to a new CSV file
data.to_csv('processed_labeled_data.csv', index=False)

print("Anomaly detection complete. Labeled data saved as 'processed_labeled_data.csv'")
print(f"Number of anomalies detected: {data['Anomaly'].sum()}")
print(f"Percentage of anomalies: {data['Anomaly'].mean() * 100:.2f}%")
```

```
Anomaly detection complete. Labeled data saved as 'processed_labeled_data.csv'
Number of anomalies detected: 106
Percentage of anomalies: 5.05%
```

In [3]:
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

# Step 1: Load the dataset
data = pd.read_csv(r"C:\Users\srush\OneDrive\Desktop\IIT Chicago\ACS\Project\combined_data.csv")

# Step 2: Identify the 'Label' column
# Assuming 'Label' is the last column
data.columns = data.columns.str.strip()
label_column = data.columns[-1]

# Step 3: Encode the 'Label' column
label_encoder = LabelEncoder()
data['Label_Encoded'] = label_encoder.fit_transform(data[label_column])

# Step 4: Convert all columns to numeric, handling any non-numeric data
numeric_columns = data.columns.drop([label_column, 'Label_Encoded'])
data[numeric_columns] = data[numeric_columns].apply(pd.to_numeric, errors='coerce')

# Step 5: Replace infinite values with NaN and then fill with 0
data = data.replace([np.inf, -np.inf], np.nan).fillna(0)

# Step 6: Separate features and labels
X = data.drop([label_column, 'Label_Encoded'], axis=1)
y = data['Label_Encoded']

# Step 7: Apply Isolation Forest
```

```
from sklearn.ensemble import IsolationForest

isolation_forest = IsolationForest(contamination=0.1, random_state=42)
anomaly_labels = isolation_forest.fit_predict(X)

# Step 8: Add anomaly detection results to the dataset
data['Anomaly'] = anomaly_labels

# Step 9: Map Isolation Forest results to meaningful labels
# -1 (anomalous) to 1 (Malicious) and 1 (normal) to 0 (Normal)
data['Anomaly'] = data['Anomaly'].map({-1: 1, 1: 0})

# Step 10: Save the processed and labeled data to a new CSV file
data.to_csv('processed_labeled_data.csv', index=False)

print("Anomaly detection complete. Labeled data saved as 'processed_labeled_data.csv'")
print(f"Number of anomalies detected: {data['Anomaly'].sum()}")
print(f"Percentage of anomalies: {data['Anomaly'].mean() * 100:.2f}%")

# Display a sample of the results
print("\nSample of the results:")
print(data[[label_column, 'Label_Encoded', 'Anomaly']].sample(10))
```

```
Anomaly detection complete. Labeled data saved as 'processed_labeled_data.csv'
Number of anomalies detected: 210
Percentage of anomalies: 10.00%

Sample of the results:
       Label  Label_Encoded  Anomaly
675   BENIGN              0        0
1146  BENIGN              0        0
732   BENIGN              0        0
173   BENIGN              0        0
823   BENIGN              0        0
1040  BENIGN              0        1
1199  BENIGN              0        1
908   BENIGN              0        0
1565  BENIGN              0        0
1359  BENIGN              0        0
```

In [5]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
import seaborn as sns
import matplotlib.pyplot as plt
```

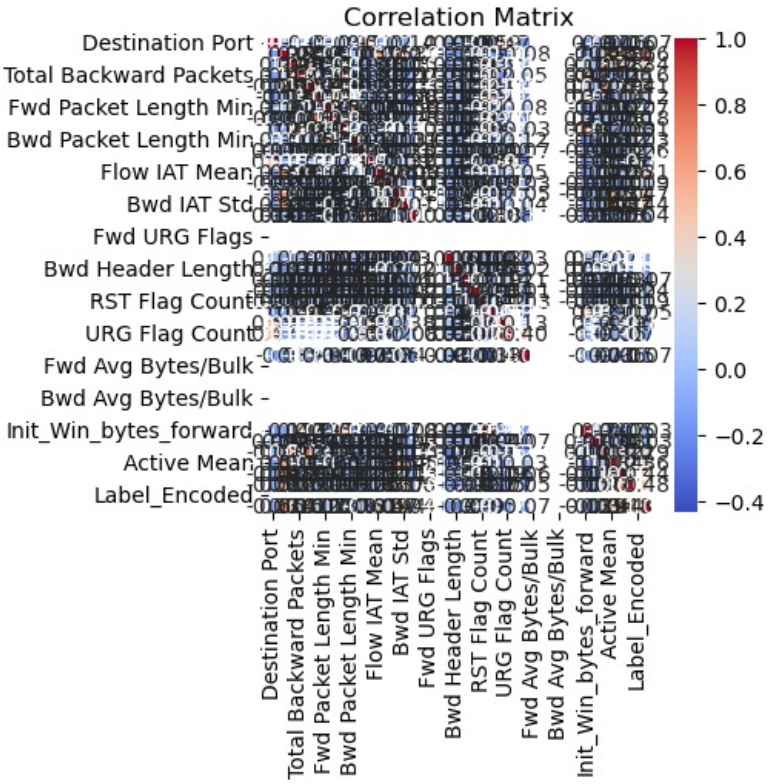
In [8]:
```
# Check for missing values
missing_values = data.isnull().sum()
print("Missing values in each column:\n", missing_values)

data.dropna(axis=0, thresh=int(data.shape[1] * 0.8), inplace=True)
```

```
Missing values in each column:
 Destination Port              0
Flow Duration                  0
Total Fwd Packets              0
Total Backward Packets         0
Total Length of Fwd Packets    0
                              ..
Idle Max                       0
Idle Min                       0
Label                          0
Label_Encoded                  0
Anomaly                        0
Length: 81, dtype: int64
```

In [9]:
```
# Identify features for scaling (e.g., packet lengths and flow durations)
features_to_scale = ['Flow Duration', 'Total Length of Fwd Packets', 'Total Length of Bwd Packets']

# Standardize the features
scaler = StandardScaler()
data[features_to_scale] = scaler.fit_transform(data[features_to_scale])
```

In [15]:
```
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Plotting the correlation matrix for visualization
plt.figure(figsize=(4, 4))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
# Removing highly correlated features (threshold can be adjusted)
threshold = 0.8
to_drop = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > threshold:
            colname = correlation_matrix.columns[i]
            to_drop.add(colname)

data.drop(columns=to_drop, inplace=True)
```



In [18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Split data into features and labels
X = data.drop(columns=['Anomaly'])  # Assuming 'label' column indicates normal or malicious traffic
y = data['Anomaly']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Random Forest Classifier

In [19]:
```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
rf_model.fit(X_train, y_train)
```

Out[19]:
```
RandomForestClassifier(max_depth=10, random_state=42)
```

Neural Network Model

In [21]:
```
pip install tensorflow
```

```
Collecting tensorflow
  Using cached tensorflow-2.18.0-cp312-cp312-win_amd64.whl.metadata (3.3 kB)
Collecting tensorflow-intel==2.18.0 (from tensorflow)
  Using cached tensorflow_intel-2.18.0-cp312-cp312-win_amd64.whl.metadata (4.9 kB)
Collecting absl-py>=1.0.0 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow-intel==2.18.0->tensorflow)
```

```
  Using cached astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached gast-0.6.0-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached libclang-18.1.1-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached opt_einsum-3.4.0-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: packaging in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-intel==2
.18.0->tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3
in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\srush\anaconda3\lib\site-packages (from tensorflo
w-intel==2.18.0->tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-intel==
2.18.0->tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-intel=
=2.18.0->tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached termcolor-2.5.0-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\srush\anaconda3\lib\site-packages (from tens
orflow-intel==2.18.0->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-inte
l==2.18.0->tensorflow) (1.14.1)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached grpcio-1.67.1-cp312-cp312-win_amd64.whl.metadata (4.0 kB)
Collecting tensorboard<2.19,>=2.18 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached tensorboard-2.18.0-py3-none-any.whl.metadata (1.6 kB)
Collecting keras>=3.5.0 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached keras-3.6.0-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in c:\users\srush\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.18.0->tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\users\srush\anaconda3\lib\site-packages (from tensorflow-intel
==2.18.0->tensorflow) (3.11.0)
Collecting ml-dtypes<0.5.0,>=0.4.0 (from tensorflow-intel==2.18.0->tensorflow)
  Using cached ml_dtypes-0.4.1-cp312-cp312-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\srush\anaconda3\lib\site-packages (from astunparse
>=1.6.0->tensorflow-intel==2.18.0->tensorflow) (0.44.0)
Requirement already satisfied: rich in c:\users\srush\anaconda3\lib\site-packages (from keras>=3.5.0->tensorflow
-intel==2.18.0->tensorflow) (13.7.1)
Collecting namex (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow)
  Using cached namex-0.0.8-py3-none-any.whl.metadata (246 bytes)
Collecting optree (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow)
  Using cached optree-0.13.1-cp312-cp312-win_amd64.whl.metadata (48 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\srush\anaconda3\lib\site-packages (from requ
ests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\srush\anaconda3\lib\site-packages (from requests<3,>=2.2
1.0->tensorflow-intel==2.18.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\srush\anaconda3\lib\site-packages (from requests<3
,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\srush\anaconda3\lib\site-packages (from requests<3
,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in c:\users\srush\anaconda3\lib\site-packages (from tensorboard<2
.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (3.4.1)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensor
flow)
  Using cached tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\srush\anaconda3\lib\site-packages (from tensorboard<2
.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\srush\anaconda3\lib\site-packages (from werkzeug>=1
.0.1->tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\srush\anaconda3\lib\site-packages (from rich->k
eras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\srush\anaconda3\lib\site-packages (from rich-
>keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\srush\anaconda3\lib\site-packages (from markdown-it-py>=2.
2.0->rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.1.0)
Using cached tensorflow-2.18.0-cp312-cp312-win_amd64.whl (7.5 kB)
Using cached tensorflow_intel-2.18.0-cp312-cp312-win_amd64.whl (390.3 MB)
Using cached absl_py-2.1.0-py3-none-any.whl (133 kB)
Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Using cached flatbuffers-24.3.25-py2.py3-none-any.whl (26 kB)
Using cached gast-0.6.0-py3-none-any.whl (21 kB)
Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Using cached grpcio-1.67.1-cp312-cp312-win_amd64.whl (4.3 MB)
Using cached keras-3.6.0-py3-none-any.whl (1.2 MB)
Using cached libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
Using cached ml_dtypes-0.4.1-cp312-cp312-win_amd64.whl (127 kB)
Using cached opt_einsum-3.4.0-py3-none-any.whl (71 kB)
Using cached tensorboard-2.18.0-py3-none-any.whl (5.5 MB)
```

```
Using cached termcolor-2.5.0-py3-none-any.whl (7.8 kB)
Using cached tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Using cached namex-0.0.8-py3-none-any.whl (5.8 kB)
Using cached optree-0.13.1-cp312-cp312-win_amd64.whl (292 kB)
Installing collected packages: namex, libclang, flatbuffers, termcolor, tensorboard-data-server, optree, opt-ein
sum, ml-dtypes, grpcio, google-pasta, gast, astunparse, absl-py, tensorboard, keras, tensorflow-intel, tensorflo
w
Successfully installed absl-py-2.1.0 astunparse-1.6.3 flatbuffers-24.3.25 gast-0.6.0 google-pasta-0.2.0 grpcio-1
.67.1 keras-3.6.0 libclang-18.1.1 ml-dtypes-0.4.1 namex-0.0.8 opt-einsum-3.4.0 optree-0.13.1 tensorboard-2.18.0
tensorboard-data-server-0.7.2 tensorflow-2.18.0 tensorflow-intel-2.18.0 termcolor-2.5.0
Note: you may need to restart the kernel to use updated packages.
```

In [22]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Neural Network model
nn_model = Sequential([
    Dense(64, input_shape=(X_train.shape[1],), activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification
])

# Compile and train the model
nn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
nn_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
C:\Users\srush\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input
_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as
the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 0.5352 - loss: 0.6828 - val_accuracy: 0.9256 - val_loss: 0.2
928
Epoch 2/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9402 - loss: 0.2614 - val_accuracy: 0.9673 - val_loss: 0.13
79
Epoch 3/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9410 - loss: 0.1603 - val_accuracy: 0.9762 - val_loss: 0.09
89
Epoch 4/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9620 - loss: 0.1085 - val_accuracy: 0.9881 - val_loss: 0.08
14
Epoch 5/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9561 - loss: 0.1047 - val_accuracy: 0.9881 - val_loss: 0.07
26
Epoch 6/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9673 - loss: 0.0945 - val_accuracy: 0.9881 - val_loss: 0.06
46
Epoch 7/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9748 - loss: 0.0765 - val_accuracy: 0.9881 - val_loss: 0.06
17
Epoch 8/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9709 - loss: 0.0795 - val_accuracy: 0.9732 - val_loss: 0.05
93
Epoch 9/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9684 - loss: 0.0716 - val_accuracy: 0.9881 - val_loss: 0.05
40
Epoch 10/10
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9713 - loss: 0.0741 - val_accuracy: 0.9851 - val_loss: 0.05
27
```

Out[22]:
```
<keras.src.callbacks.history.History at 0x2c429ba5fd0>
```

Model Evaluation

In [23]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Evaluate Random Forest model
rf_preds = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
print("Random Forest Precision:", precision_score(y_test, rf_preds))
print("Random Forest Recall:", recall_score(y_test, rf_preds))
print("Random Forest F1 Score:", f1_score(y_test, rf_preds))

# Evaluate Neural Network model
nn_preds = (nn_model.predict(X_test) > 0.5).astype("int32")
print("Neural Network Accuracy:", accuracy_score(y_test, nn_preds))
print("Neural Network Precision:", precision_score(y_test, nn_preds))
print("Neural Network Recall:", recall_score(y_test, nn_preds))
print("Neural Network F1 Score:", f1_score(y_test, nn_preds))
```

```
Random Forest Accuracy: 0.9857142857142858
Random Forest Precision: 0.94
Random Forest Recall: 0.94
Random Forest F1 Score: 0.94
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Neural Network Accuracy: 0.9714285714285714
Neural Network Precision: 0.9318181818181818
Neural Network Recall: 0.82
Neural Network F1 Score: 0.8723404255319149
```

ZTA Framework

In [24]:
```python
class ZTAFramework:
    """Basic Zero Trust Framework Simulation"""
    def __init__(self, model, scaler):
        self.model = model
        self.scaler = scaler

    def calculate_threat_score(self, input_data):
        """Calculate threat score based on model prediction probability."""
        input_scaled = self.scaler.transform([input_data])
        threat_score = self.model.predict_proba(input_scaled)[0][1]  # Assuming model outputs probability
        return threat_score

    def verify_and_authorize(self, username, password, user_role, threat_score):
        """Authorization based on ZTA principles."""
        if username == "valid_user" and password == "secure_password" and threat_score < 0.5:
            return f"Access Granted to {user_role}"
        else:
            return "Access Denied: High Risk Detected or Unauthorized"
```

ZTA Integration with AI

In [27]:
```python
# Define ZTA process with threat detection
def zta_process(input_data):
    zta = ZTAFramework(model=rf_model, scaler=scaler)
    threat_score = zta.calculate_threat_score(input_data)
    decision = zta.verify_and_authorize("valid_user", "secure_password", "employee", threat_score)
    print(f"Threat Score: {threat_score}, Decision: {decision}")

# Example usage
input_data_example = X_test[27]  # Can Replace with a specific row of test data
zta_process(input_data_example)
```

```
Threat Score: 0.04465922451978741, Decision: Access Granted to employee
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
```

Real-Time Data Simulation and Alerting

In [28]:
```python
import time
import numpy as np

# Define a function to simulate real-time data generation
def simulate_real_time_data():
    """Simulate incoming network traffic data with features like packet size, flow duration, and protocol."""
    while True:
        # Simulate a single network packet's features (replace with real data in production)
        packet_data = np.random.rand(1, X_train.shape[1])  # Random data for example; replace with actual traff
        yield packet_data
        time.sleep(1)  # Simulate data stream with a 1-second interval

# Define a function to generate alerts based on threat detection
def detect_and_alert(model, scaler):
    """Use the model to predict and trigger alerts for anomalies."""
    for packet_data in simulate_real_time_data():
        packet_data_scaled = scaler.transform(packet_data)
        threat_score = model.predict_proba(packet_data_scaled)[0][1]  # Probability of malicious activity
        if threat_score > 0.5:  # Threshold for threat alert
            print(f"Alert! Potential Threat Detected with Threat Score: {threat_score}")
        else:
            print(f"Normal Traffic Detected with Threat Score: {threat_score}")

# Example usage
detect_and_alert(rf_model, scaler)
```

```
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.06653196494695192
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.0758470334401026
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.0777294191006203
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.12181910982852763
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.14233766595726116
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.109463843749986
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.06561447530056772
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.14475415468101638
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.14968151043028993
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.1594885366549312
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
Normal Traffic Detected with Threat Score: 0.1699140685698248
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[28], line 25
     22             print(f"Normal Traffic Detected with Threat Score: {threat_score}")
     24 # Example usage
---> 25 detect_and_alert(rf_model, scaler)

Cell In[28], line 16, in detect_and_alert(model, scaler)
     14 def detect_and_alert(model, scaler):
     15     """Use the model to predict and trigger alerts for anomalies."""
---> 16     for packet_data in simulate_real_time_data():
     17         packet_data_scaled = scaler.transform(packet_data)
     18         threat_score = model.predict_proba(packet_data_scaled)[0][1]  # Probability of malicious activit
y

Cell In[28], line 11, in simulate_real_time_data()
      9 packet_data = np.random.rand(1, X_train.shape[1])  # Random data for example; replace with actual traffi
c features
     10 yield packet_data
---> 11 time.sleep(1)

KeyboardInterrupt:
```

Deploy and Monitor Model with Feedback Loop

In [29]:

```python
import logging

# Set up logging for feedback and monitoring
logging.basicConfig(filename="model_feedback.log", level=logging.INFO)

def deploy_model_for_monitoring(input_data, model, scaler):
    """Deploy model and monitor, logging feedback for model improvement."""
    input_data_scaled = scaler.transform([input_data])
    threat_score = model.predict_proba(input_data_scaled)[0][1]
```

```
    # Define thresholds and provide feedback
    if threat_score > 0.5:
        result = "Threat Detected"
        logging.info(f"Alert generated at {time.ctime()}: Threat Score {threat_score}")
    else:
        result = "Normal Traffic"

    # Log result for feedback loop
    logging.info(f"{time.ctime()}: {result} with Threat Score {threat_score}")
    return result

# Simulate monitoring loop
for input_data in X_test[:10]:  # Replace with real-time or batch data in production
    result = deploy_model_for_monitoring(input_data, rf_model, scaler)
    print(result)
```

```
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
Normal Traffic
```

```
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\srush\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names
, but StandardScaler was fitted with feature names
  warnings.warn(
```

Security Auditing and Continuous Monitoring

In [30]:
```
from sklearn.metrics import confusion_matrix

# Function to periodically audit model performance
def audit_model_performance(true_labels, predictions):
    """Audit predictions to check for false positives/negatives and overall accuracy."""
    cm = confusion_matrix(true_labels, predictions)
    tn, fp, fn, tp = cm.ravel()
    print("Confusion Matrix:")
    print(cm)
    print(f"False Positives: {fp}, False Negatives: {fn}")

# Function to update model with new data if needed
def retrain_model(X_new, y_new, model):
    """Retrain the model with new data, adding it to the existing training set."""
    model.fit(X_new, y_new)
    print("Model updated with new training data.")

# Example usage of auditing and updating process
preds = rf_model.predict(X_test)  # Get predictions
audit_model_performance(y_test, preds)  # Audit performance
```

```
Confusion Matrix:
[[367   3]
 [  3  47]]
False Positives: 3, False Negatives: 3
```

For new data (X_new, y_new) for updating the model periodically X_new, y_new = load_new_data() retrain_model(X_new, y_new, rf_model)

Retrain model with new data