IMPORT LIB

In [ ]:
```
# Process Data
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# Detection model
from sklearn.svm import SVC

# Attack model
from art.attacks.evasion import SaliencyMapMethod
from art.estimators.classification import SklearnClassifier

# Metrics and data visualization
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns # for statistical data visualization
```

FUNCTION ZONE

In [ ]:
```
def get_confusion_matrix(y_true, y_pred):
    # Print the Confusion Matrix and slice it into four pieces
    # y_true: is the correct label of instance
    # y_pred: is the predicted label

    cm = confusion_matrix(y_true, y_pred)

    # Binary classification confusion matrix
    print('Confusion matrix\n\n', cm)

    print('\nTrue Negatives(TN) = ', cm[0,0])

    print('\nTrue Positives(TP) = ', cm[1,1])

    print('\nFalse Positives(FP) = ', cm[0,1])

    print('\nFalse Negatives(FN) = ', cm[1,0])
    return cm
```

In [ ]:
```
def draw_binary_confusion_matrix_heatmap(confusion_matrix, title=""):
    # confusion_matrix: from sklearn.metrics import confusion_matrix
    # title: title for your heat map
    # visualize confusion matrix with seaborn heatmap
    cm_matrix = pd.DataFrame(data=confusion_matrix, columns=['Predict Benign: 0', '
                                 index=['Actual Benign: 0', 'Actual Malicious: 1

    sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

```
    # Add title
    plt.title(title)
    # Show the plot
    plt.show()
```

LOAD DATA

In [ ]:
```
# Dataset directory
data_dir = "Dataset\\"
```

In [ ]:
```
# Load Data
data = pd.read_csv(data_dir + "drebin215dataset5560malware9476benign.csv")
print("Total missing values : ",sum(list(data.isna().sum())))
```

```
Total missing values :  0
```

```
C:\Users\hai\AppData\Local\Temp\ipykernel_16624\140858179.py:2: DtypeWarning: Column
s (92) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv(data_dir + "drebin215dataset5560malware9476benign.csv")
```

PREPROCESS DATA

In [ ]:
```
classes,count = np.unique(data['class'],return_counts=True)
#Perform Label Encoding
lbl_enc = LabelEncoder()
print(lbl_enc.fit_transform(classes),classes)
data = data.replace(classes,lbl_enc.fit_transform(classes))

#Dataset contains special characters like ''?' and 'S'. Set them to NaN and use dro
data=data.replace('[?,S]',np.NaN,regex=True)
print("Total missing values : ",sum(list(data.isna().sum())))
data.dropna(inplace=True)
for c in data.columns:
    data[c] = pd.to_numeric(data[c])
data
```

```
C:\Users\hai\AppData\Local\Temp\ipykernel_16624\410007786.py:5: FutureWarning: Downc
asting behavior in `replace` is deprecated and will be removed in a future version.
To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To o
pt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', Tru
e)`
  data = data.replace(classes,lbl_enc.fit_transform(classes))
[0 1] ['B' 'S']
Total missing values :  5
```

Out[ ]:

	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	and
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
...	...	...	...	...	...	
15031	1	1	1	1	1	
15032	0	0	0	0	0	
15033	0	0	0	0	0	
15034	1	1	1	1	1	
15035	1	1	1	1	1	

15031 rows × 216 columns

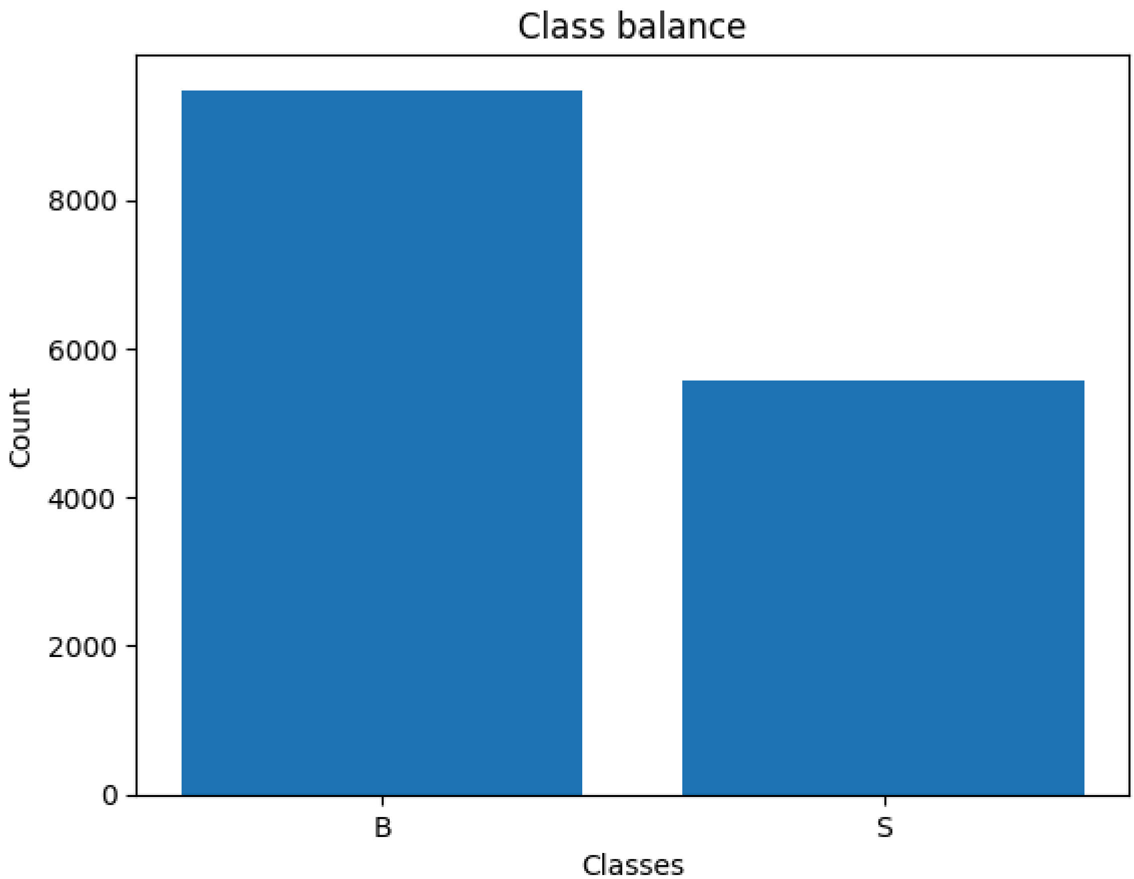
In [ ]:

```
print("Total Features : ",len(data.columns)-1)
```

```
Total Features :  215
```

In [ ]:

```
plt.bar(classes,count)
plt.title("Class balance")
plt.xlabel("Classes")
plt.ylabel("Count")
plt.show()
```



SPLIT DATA

In [ ]:
```
x = data.iloc[:, 1:].values
y = data.iloc[:, 0].values
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
```

In [ ]:
```
print("Train features size : ",len(x_train))
print("Train labels size : ",len(y_train))
print("Test features size : ",len(x_test))
print("Test labels size : ",len(y_test))
```

```
Train features size :  12024
Train labels size :  12024
Test features size :  3007
Test labels size :  3007
```

In [ ]:
```
print("Train features : ",x_train.shape)
print("Train labels : ",y_train.shape)
print("Test Features : ",x_test.shape)
print("Test labels : ",y_test.shape)
```

```
Train features :  (12024, 215)
Train labels :  (12024,)
Test Features :  (3007, 215)
Test labels :  (3007,)
```

TRAIN DETECTION MODEL

In [ ]:
```
svm = SVC(kernel='linear', C=1)
```

In [ ]:
```
svm.fit(x_train,y_train)
```

Out[ ]:

SVC

```
SVC(C=1, kernel='linear')
```

In [ ]:
```
y_pred = svm.predict(x_test)
```

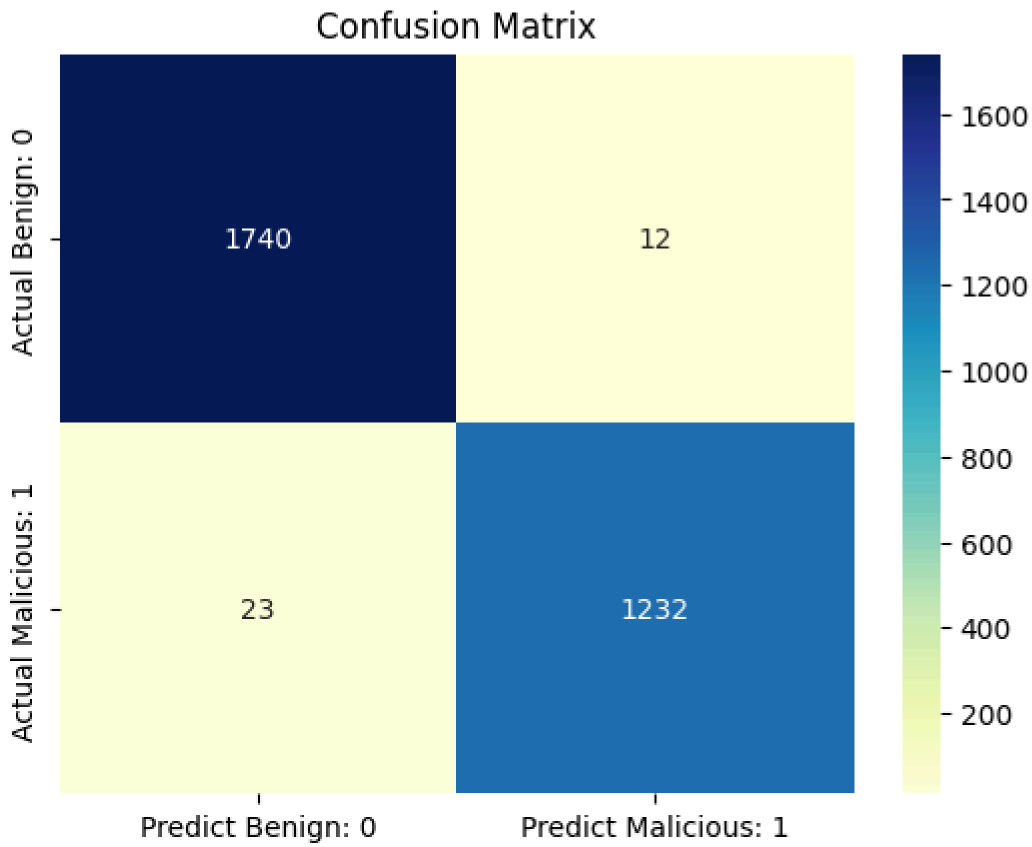
In [ ]:
```
print("Accuracy: ", accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy:  0.9883604921849019
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1752
           1       0.99      0.98      0.99      1255

    accuracy                           0.99      3007
   macro avg       0.99      0.99      0.99      3007
weighted avg       0.99      0.99      0.99      3007
```

In [ ]:
```
cm = get_confusion_matrix(y_test, y_pred)
```

```
Confusion matrix

 [[1740   12]
 [  23 1232]]

True Negatives(TN) =  1740

True Positives(TP) =  1232

False Positives(FP) =  12

False Negatives(FN) =  23
```

In [ ]:
```
draw_binary_confusion_matrix_heatmap(cm, "Confusion Matrix")
```



SELECT ATTACK INSTANCES

In [ ]:
```
# Select an instance to attack (assuming we want to flip a '1' to '0')
positive_indices = np.where(y_test == 1)
x_to_attack = x_test[positive_indices]
y_to_attack = y_test[positive_indices]
```

CREATE ATTACK MODEL

In [ ]:
```
# Create an ART classifier wrapper for the SVM
classifier = SklearnClassifier(model=svm, clip_values=(0, 1))
# Create attack model
attack = SaliencyMapMethod(classifier)
```

CREATE ADVERSARIAL EXAMPLES

In [ ]:
```
# Generate perturbed instance
perturbed_instance = attack.generate(x_to_attack,y=np.zeros_like(y_to_attack))
```

```
C:\Users\hai\AppData\Local\Temp\ipykernel_16624\1723329802.py:2: DeprecationWarning:
`product` is deprecated as of NumPy 1.25.0, and will be removed in NumPy 2.0. Please
use `prod` instead.
  perturbed_instance = attack.generate(x_to_attack,y=np.zeros_like(y_to_attack))
JSMA: 100%|██████████| 1255/1255 [00:35<00:00, 35.37it/s]
```

In [ ]:

```python
original_prediction = svm.predict(x_to_attack)
perturbed_prediction = svm.predict(perturbed_instance)
print("Accuracy before attack: ", accuracy_score(y_to_attack, original_prediction))
print(classification_report(y_to_attack, original_prediction))
print("Accuracy after attack: ", accuracy_score(y_to_attack, perturbed_prediction))
print(classification_report(y_to_attack, perturbed_prediction))
```

```
Accuracy before attack:  0.9816733067729083
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.98      0.99      1255

    accuracy                           0.98      1255
   macro avg       0.50      0.49      0.50      1255
weighted avg       1.00      0.98      0.99      1255

Accuracy after attack:  0.0
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       0.0
           1       0.00      0.00      0.00    1255.0

    accuracy                           0.00    1255.0
   macro avg       0.00      0.00      0.00    1255.0
weighted avg       0.00      0.00      0.00    1255.0
```

```
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\hai\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metri
cs\_classification.py:1517: UndefinedMetricWarning: Recall is ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

VISUALIZE RESULT

AVERAGE DISTORTION

In [ ]:

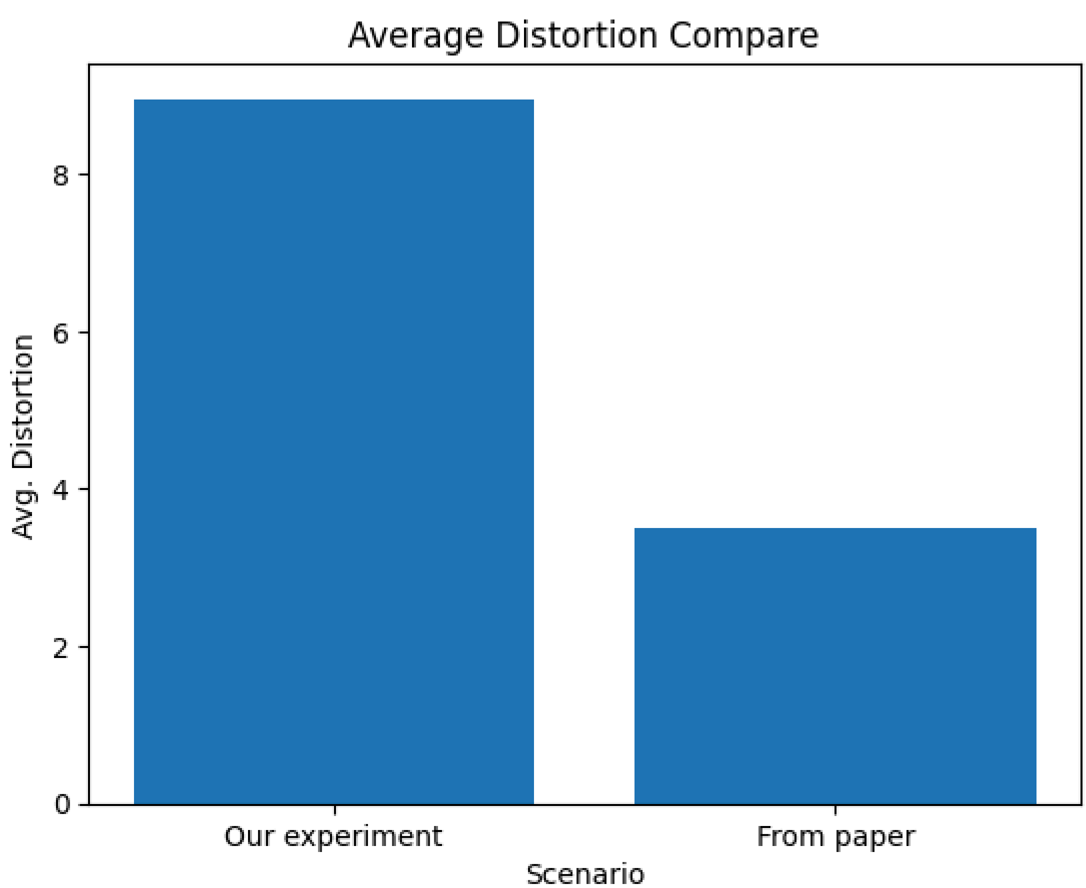
```
# Distortion is the number of value that has been changed to evade detection
temp = 0
```

```
count = 0
while temp < len(x_to_attack):
    array1 = x_to_attack[temp]
    array2 = perturbed_instance[temp]

    # Calculate the differences between consecutive elements
    result = np.diff(array1 - array2)
    for i in result:
        if i != 0:
            count+=1
    temp+=1
avg_distortion = count/len(x_to_attack)
print("Average Distortion: ", avg_distortion)
```

```
Average Distortion:  8.952988047808764
```

In [ ]:

```
x_bar = ["Our experiment","From paper"]
y_bar = [avg_distortion, 3.5]

plt.bar(x_bar,y_bar)
plt.title("Average Distortion Compare")
plt.xlabel("Scenario")
plt.ylabel("Avg. Distortion")
plt.show()
```



CONFUSION MATRIX

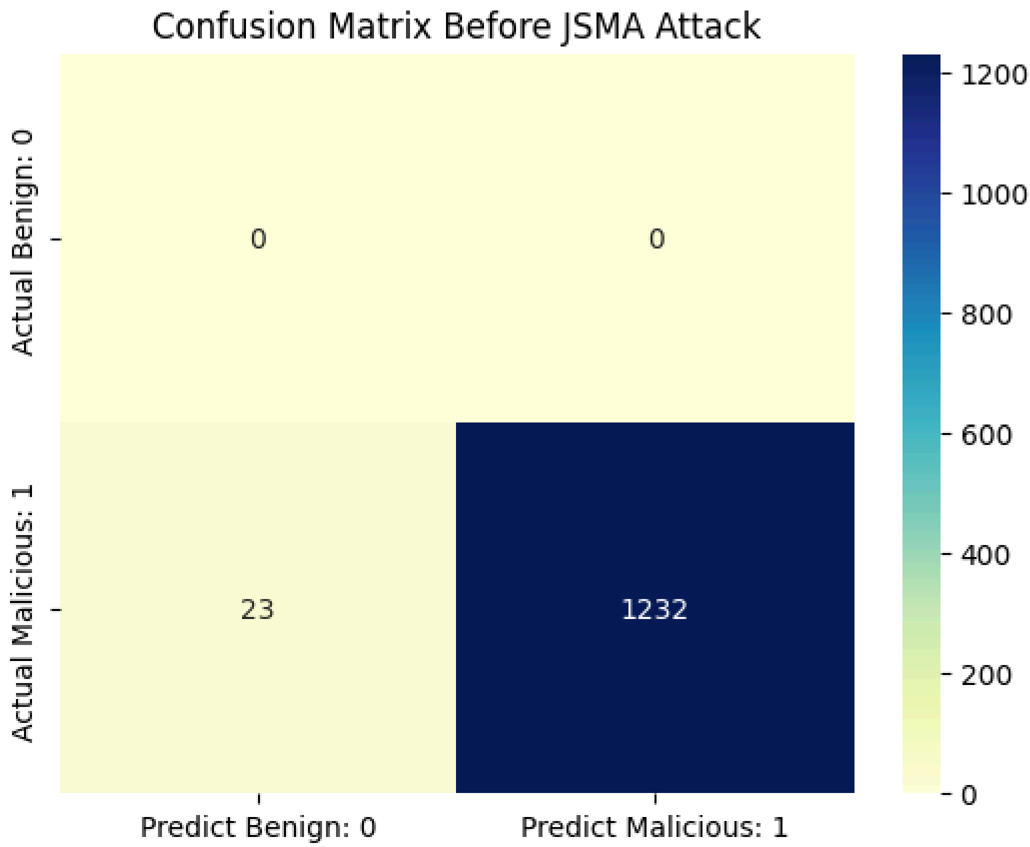
In [ ]:
```
cm_before_attack = get_confusion_matrix(y_to_attack, original_prediction)
```

```
Confusion matrix

 [[   0    0]
 [  23 1232]]

True Negatives(TN) =  0

True Positives(TP) =  1232

False Positives(FP) =  0

False Negatives(FN) =  23
```

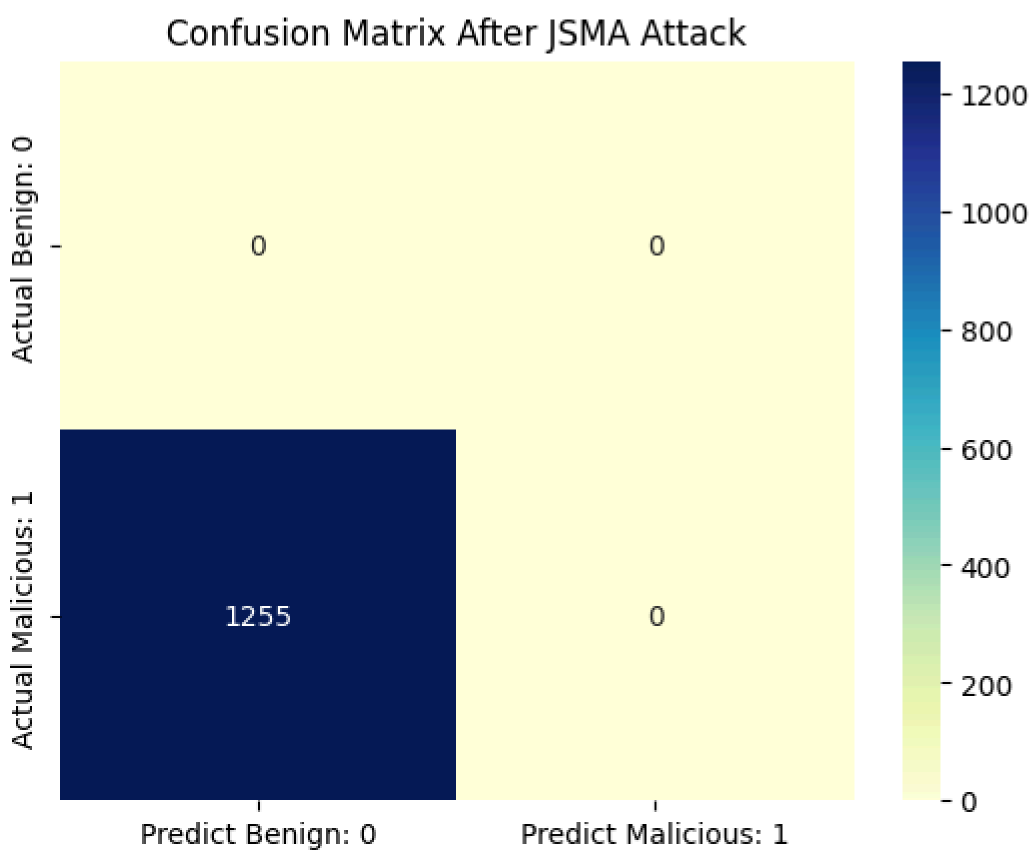
In [ ]:
```
cm_after_attack = get_confusion_matrix(y_to_attack, perturbed_prediction)
```

```
Confusion matrix

 [[   0    0]
 [1255    0]]

True Negatives(TN) =  0

True Positives(TP) =  0

False Positives(FP) =  0

False Negatives(FN) =  1255
```

In [ ]:
```
draw_binary_confusion_matrix_heatmap(cm_before_attack, "Confusion Matrix Before JSM
```



In [ ]:
```
draw_binary_confusion_matrix_heatmap(cm_after_attack, "Confusion Matrix After JSMA
```



EVASION RATE

In [ ]:
```
# evasion rate = (FN)/len(attack_instances)
evasion_rate = cm_after_attack[1,0] / len(x_to_attack)

x_bar = ["Our experiment","From paper"]
y_bar = [evasion_rate, 1]

plt.bar(x_bar,y_bar)
plt.title("Evasion Rate Compare")
plt.xlabel("Scenario")
plt.ylabel("Evasion Rate")
plt.show()
```