Malware Detection from IoT Traffic

Konnor Mascara
Syracuse University, kmascara@syr.edu
CIS662: Machine Learning, Professor Mohan

INTRODUCTION

IoT devices have grown exponentially and have improved the way we have lived and developed. But these devices can be malicious without the user knowing any difference. Wireshark is a common tool used for getting any traffic over the connected network. Multiple different captured various types of attacks and devices were taken by Stratosphere Lab located in Czech Republic. The data was put into a conn.log file and can easily be put into a CSV file to be used for training and testing a model.

FILES

To cut back on the size of files I have only included the necessary files while omitting ones that were either used but not needed or not used. Some not included was most of the big conn.log files and the proposal as that was already handed in. The items handed in includes:

- Four csv files used in the main python code.
- One conn.log file that was used to convert to csv in logTOcsv.py.
- This report is in pdf form.
- Mlprj.py that shows only one model running.
- Mlprj-mod.py that includes all classification models used.

FORMATTING DATA

```
import csv

#change these when needed. a-c is non-malware, 1-20 is malicious
input_file_path = './/log_files//2conn.log.labeled'
output_file_path = './/csv_files//2.csv'

with open(input_file_path, 'r') as input_file, open(output_file_path, 'w', newline='') as output_file:
    csv_writer = csv.writer(output_file, delimiter=',')

    #Get rid of header, and the last row. Then change value 20 to do T or F based on Beign or Malicous
    for _ in range(8):
        next(input_file)
    lines = list(input_file)[:-1]
    for line in lines:
        values = line.strip().split('\t')
        if values[20] not in ['-   Benign   -', '-   benign   -','(empty)   Benign   -']:
            values[20] = True
        else:
            values[20] = False

        csv_writer.writerow(values)

print('Conversion complete')
```

I. The first step is to import csv and get the path for input and output files. Creating a writer that uses a comma for separating each column.
II. The first eight rows is information about the conn.log file and is cut off then the last line is cut off.
III. Each line of the file gets split up every time a tab is found. The result being the 20th value is changed from a string to Boolean where benign rows are False and malicious is True.

MAIN MODEL

```
#Konnor Mascara CIS662 Project
#Most code modified from given guide on sclearn website (scikit-learn.org)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

#Step 1: Load file (Dataset 1)
#Step 2: X is data, Y is answer
#Step 3: Get encoder to change all strings to int
df1 = pd.read_csv(".//csv_files//1.csv", header=None)
X1 = df1.iloc[:, :-1]
y1 = df1.iloc[:, -1]
label_encoders1 = {}
for col in X1.select_dtypes(include=['object']).columns:
    label_encoders1[col] = LabelEncoder()
    X1[col] = label_encoders1[col].fit_transform(X1[col])
```

P1

```
#Split data train and test (Some are higher test due to size of file, takes a minute)
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.99, random_state=42)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2, random_state=42)
X_train10, X_test10, y_train10, y_test10 = train_test_split(X10, y10, test_size=0.2, random_state=42)
X_trainA, X_testA, y_trainA, y_testA = train_test_split(XA, yA, test_size=0.99, random_state=42)

#Put data together for training (Maybe testing later)
X_trainD = pd.concat([X_train1, X_train2, X_train10], axis=0, ignore_index=True)
y_trainD = pd.concat([y_train1, y_train2, y_train10], axis=0, ignore_index=True)

#Get model and train
model = RandomForestClassifier()
model.fit(X_trainD, y_trainD)

#Test with 1st data (Lower due to train set being small)
y1_pred1 = model.predict(X_test1)
accuracy1 = accuracy_score(y_test1, y1_pred1)
print(f"Accuracy: {accuracy1}")
```

P2

The images are labeled to explain what is happening.
P1: Imports and reading the data.
I. import pandas is being used for the CSV file. Train_test_split makes splitting the data easier. RandomForestClassifier is the main model that was used for testing. Accuracy_score is used to give the percentage the model goes correct. Lastly, label encoder was used to change any other data formats into int.

II. Reading the csv file and splitting the first 19 columns (input) with the last (result). Then with each column in the input that is not int is changed. Fit_transform makes sure the change goes into the correct location again. This is repeated for all other files.

P2: Split, combine training data, training the model, and testing with accuracy.

I. For each data set needs to be split into train and test with X being the input and Y being the results. The test size changes based on the size of the file. Training can take a while, and some files have millions of rows, so the test size is big to save training. Random state is to help reproduce the same results and is set to 42.

II. Combining the train all the train data from multiple files is required since each file used had a different type of malicious attack. Without this testing most of the files will have low accuracy.

III. Simple step of training the data (This takes around 1 minute of running on my computer)

IV. Use the test data to get the predictions. Then get the accuracy of the results and print. This is done with all files the same way.

RESULTS OF MAIN MODEL

```
Accuracy: 0.9935523666189027
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 0.8973214285714286
```

Top to bottom of dataset: 1, 2, 10, A. Reminder this is how all results will be displayed in the next section.

Dataset 2 & 10:

Datasets 2 & 10 had a 0.2 test size meaning most of it was used for training the data and getting a perfect probability of success.

Dataset 1:

The size of the dataset is massive and only 0.01 was used for training and that is what caused it to go from perfect accuracy to only 99.35%.

Dataset A:

This was a non-malicious file and was split for training but was not put into training. Meaning that the accuracy of this file was 89.73% while no training from this file was used.

CHANGES TO MODEL

```
Accuracy: 0.9936284685193474
Accuracy: 0.9997757901395706
Accuracy: 0.9988658457550227
Accuracy: 0.796875
```

Same model but Dataset 2 & 10 test case changed from 0.2 to 0.8. This is a good example that less training data caused worse results for dataset 2, 10, and A.

```
Accuracy: 0.7912715125553116
Accuracy: 0.9833043728930279
Accuracy: 0.9954633830200907
Accuracy: 0.9977678571428571
```

With the modifications above but the model changed to KNeighborsClassifier. Showing better results for dataset A but worse on dataset 1.

```
Accuracy: 0.7284223575367417
Accuracy: 0.9709327930943363
Accuracy: 0.9154785050766905
Accuracy: 0.0
```

LogisticRegression results. The only major change was the last with 0 correct. Even when adding dataset, A into the train data did not change this.

```
Accuracy: 0.9934642486289141
Accuracy: 0.9993033479336659
Accuracy: 0.9956794124000864
Accuracy: 0.47098214285714285
```

GradientBoostingClassifier does good in the first three but again does poorly in the last dataset.

```
Accuracy: 0.9936284685193474
Accuracy: 0.9997677826445553
Accuracy: 0.992546986390149
Accuracy: 0.9955357142857143
```

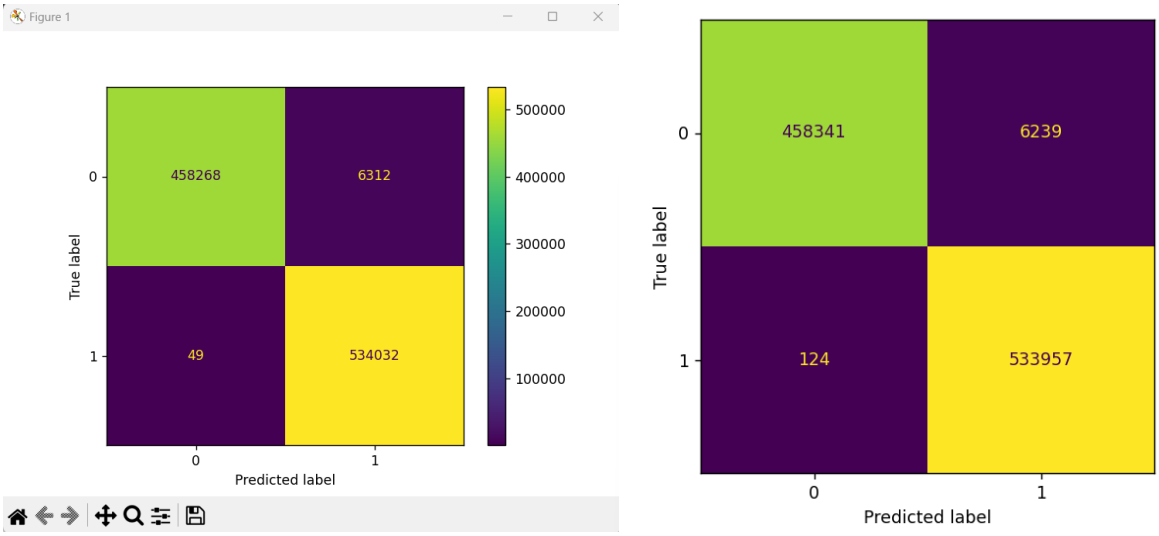
AdaboostClassifier is the best model so far with the smaller data used for testing.

```
Accuracy: 0.5879953257411674
Accuracy: 0.9759534924689509
Accuracy: 0.9837977965003241
Accuracy: 0.020089285714285716
```

GaussianNB doesn't seem to be as good in the data I gave it for training either.

GRAPHS

The left is using the best model (Adaboost) and the right is the original model (RandomForestClassifier). Then displaying it in a Confusion Matrix. Top left: true negative. Top right: false positive. Bottom left: false negative. Bottom right: True negative. Note that other graphs can be used for displaying accuracy either dot plots, line graphs, and colored graphs.



DISCUSSION

The above models covered were under classification models. Some other models include regression, clustering, and dimensionality reduction. The Stratosphere Lab also included 12 conn.log files and the four picked where the four smallest total size. This is done to save on time as running the file does take a little while on a mid-level laptop. But the datasets used have over a few million packets total. Changing the percentage of training/test as well as using more or different datasets can change the accuracy that was given above. Using python instead of Julia was my preference as having more experience with python. GradientBoositingClassifier was the only regression model that got 47% accuracy for dataset A and 99% for the rest.

CONCLUSION

Overall, the main model used had high accuracy but took a lot of data for testing while most of the other models did worse except for AdaBoostClassifier taking fewer testing data getting over 99% accuracy for every test set. Using a benign dataset for only testing and 3 dataset that use Mirai, Muhstik, and Hide and Seek malware techniques. Machine Learning has helped tremendously with finding malicious traffic in IoT devices.