Bioinformatics Summer School Report

Name: Manar Abdullah Zain Aldeen
Internship Organization: Duke University (Data +) Program
Supervisor's Name: Bob Zhang
Project title: MISTRAL:Dynamic Data Analysis of Security Threats within Biomedical Research environment

1. Introduction

MISTRAL (Massive Internal System Traffic Research Analysis and Logging) is a comprehensive project designed to enhance Duke University's internal network monitoring and data collection capabilities, with a critical focus on protecting sensitive patient data, such as genome and exome sequences, from potential cyber threats. Funded by NSF and Cisco, this initiative involves close collaboration with the Duke University IT Security Office (ITSO) and additional expert teams. The project aims to establish a baseline of 'normal' network traffic, enrich datasets with synthetic malicious data, and develop robust machine learning models. These models are essential for automatically detecting security breaches and ensuring the integrity and security of our research and operational environments, ultimately safeguarding vital patient information from unauthorized access or tampering.

2. Materials and Methodology

We followed a systematic, step-by-step approach to achieve our objectives. Initially, we used the Splunk Cloud Platform to analyze our lab's network traffic. We then performed Exploratory Data Analysis (EDA) on static network logs using Python libraries like NumPy and Pandas, focusing on identifying and removing anomalous data sets, such as truncated data or misidentified ports, to establish consistent trends. After enriching our dataset with synthetic attack data, we conducted further EDA to better understand the characteristics of normal and malicious traffic. With this foundation, we created a Machine Learning model to classify network traffic, utilizing both standard and attack data. Finally, we compared various machine learning techniques and evaluated the performance of our models across different scenarios to determine the most effective approach for detecting and mitigating potential cyber threats.

3. Results and Discussion

Our investigation employed two primary models: Logistic Regression and Neural Network, each tailored to detect port scanning activity within the network. The Logistic Regression model focused on analyzing Packet-Octet Ratio, Packet-Reverse Packet Ratio, number of connections, and unique destination ports for each IP over 24-hour periods. In contrast, the Neural Network model served as a broader monitoring tool, analyzing data flow, including flow counts, flow count ratios, flow duration, and destination ports, to predict port scanning activity.

In addition to these, we explored other machine learning models such as Random Forest, Support Vector Machine, and Isolation Forest. However, given our limited synthetic attack data and specific objectives, our primary focus remained on the Logistic Regression and Neural Network models. We evaluated the Neural Network model on a dataset containing over 60,000 entries from July 17th, 2024, which contained no actual attacks. The model demonstrated a 79% accuracy, with a 21% rate of false alarms.

Further analysis of traffic pattern variations on a typical day (June 15th, 2024) highlighted key features such as flow count, octet count, and Packet-Capture Ratio (PCR), which are critical in identifying traffic abnormalities. Data collected from midnight (00:00) to 10:12 AM showed that traffic patterns deviating outside the Bollinger Bands should serve as alerts for users. We also cross-referenced these significant spikes with the alarming IP addresses identified by our machine learning models to further verify network security.

4. Skills and Knowledge Gained

Technical Skills:

- **Machine Learning:** Developing, training, and evaluating machine learning models, specifically Logistic Regression, Neural Networks, and experimenting with models like Random Forest, Support Vector Machine, and Isolation Forest.
- **Data Analysis:** Conducting Exploratory Data Analysis (EDA) using Python (NumPy, Pandas) to identify trends, remove anomalies, and enrich datasets with synthetic data.
- **Network Security:** Understanding and detecting network anomalies, particularly related to port scanning activities, and protecting sensitive patient data in a biomedical research environment.
- **Splunk Cloud Platform:** Utilizing Splunk for monitoring and analyzing network traffic in real-time.
- **Model Evaluation:** Assessing the performance of different machine learning models in various scenarios, focusing on accuracy, false alarms, and practical applicability.

Soft Skills:

- **Collaboration:** Working closely with IT Security teams, Red Teams, and other project stakeholders to achieve common objectives.
- **Problem-Solving:** Addressing complex challenges related to network security, data integrity, and model accuracy through systematic approaches.
- **Project Management:** Managing time, resources, and objectives to ensure the successful completion of the project, from data collection to model deployment.
- **Communication:** Effectively communicating technical findings and results to both technical and non-technical audiences, including through presentations and reports.

5. Challenges and Solutions

Challenge 1: Data Quality and Anomalies

- **Solution:** We used Exploratory Data Analysis (EDA) to clean the data, removing anomalies to ensure accurate representation of both normal and attack patterns.

Challenge 2 : Limited Synthetic Attack Data

- **Solution:** We augmented the dataset with diverse synthetic attack patterns and adjusted model parameters, employing cross-validation to enhance performance.

Challenge 3 : Model Performance Variability

- **Solution:** We compared various models, including Logistic Regression and Neural Networks, fine-tuned their parameters, and implemented techniques to reduce false alarm rates.

Challenge 4 : Real-Time Analysis and Scalability

- **Solution:** We utilized the Splunk Cloud Platform for efficient real-time monitoring and analysis, ensuring high system performance and scalability.

Challenge 5 : Communicating Technical Findings

- **Solution:** We created clear reports and presentations, and held regular meetings with stakeholders to ensure effective communication of findings and recommendations

6. Conclusion

In conclusion, we have created two machine learning models in order to automate the detection process of identifying malicious traffic data, particularly for port scanning attacks. Two machine learning models achieved over 96% precision on abnormality detection. Traffic visualizations further assist users to observe fluctuations which potentially lead to port scanning attacks. This research is valuable for protecting sensitive patient data in biomedical research environments, ensuring the security and privacy of critical information.

7. Acknowledgments

We want to thank the Duke ITSO staff and Chloe Poroslay, Alexander Merck, and Eric Hope in particular for their invaluable assistance. This project was built on their guidance. Additionally, we acknowledge the KAUST Academy, Code+ MISTRAL team and Honeypots team for their contributions. Finally, we thank the Data+ management for making this project possible.

Code Used:

Neural Network:

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from  sklearn.metrics  import  confusion_matrix,  accuracy_score,  roc_curve,
roc_auc_score

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.optimizers import Adam

import seaborn as sns

import matplotlib.pyplot as plt

from imblearn.over_sampling import SMOTE


# Load the data

file_path = 'combined_updated.csv'

df = pd.read_csv(file_path)


# Selecting relevant features

columns_to_keep = [

    'packetTotalCount',

    'octetTotalCount',

    'reversePacketTotalCount',

    'reverseOctetTotalCount',

    'flowDurationMilliseconds',
```

```
    # 'sourceTransportPort',   because all of our sus might have the same
rn
    'destinationTransportPort' ,
    'suspicious'
    ]
model_df = df[columns_to_keep]

# Feature Engineering
model_df["PCR"]          =          model_df["octetTotalCount"]          /
model_df["reverseOctetTotalCount"]
model_df["POR"]          =          model_df["packetTotalCount"]          /
model_df["octetTotalCount"]

# Replace infinite values with -1
model_df['POR'] = model_df['POR'].replace([np.inf, -np.inf], -1)
model_df['PCR'] = model_df['PCR'].replace([np.inf, -np.inf], -1)
# Handle missing values if any
model_df = model_df.fillna(-1)


# Ensure the target variable is of integer type
model_df['suspicious'] = model_df['suspicious'].astype(int)

# Splitting data into features and target
X = model_df.drop('suspicious', axis=1)
y = model_df['suspicious']
```

```
# Apply SMOTE

smote = SMOTE(random_state=42)

X_resampled, y_resampled = smote.fit_resample(X, y)


# Create a balanced dataframe

balanced_df = pd.DataFrame(X_resampled, columns=X.columns)

balanced_df['suspicious'] = y_resampled


# Splitting the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size=0.2, random_state=42)




#     non_suspicious_sample     =     model_df[model_df['suspicious']     ==
0].sample(n=2308, random_state=1)

# suspicious_entries = model_df[model_df['suspicious'] == 1]


# # Concatenate the sampled non-suspicious entries and suspicious entries

# balanced_df = pd.concat([non_suspicious_sample, suspicious_entries])

# balanced_df = balanced_df.reset_index(drop=True)


# # Splitting data into features and target

# X = balanced_df.drop('suspicious', axis=1)

# y = balanced_df['suspicious']


# # Splitting the dataset into training and testing sets
```

```
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Building the Neural Network

model = Sequential()

model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(32, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(16, activation='relu'))

model.add(Dense(1, activation='sigmoid'))


# Compiling the model

model.compile(loss='binary_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])


# Training the model

history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2)


# Predicting the test set results

y_pred = (model.predict(X_test) > 0.5).astype("int32")


# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Predicted labels")
```

```
plt.ylabel("True labels")

plt.title("Confusion Matrix")

plt.show()


# Calculating the Accuracy Score

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")


# ROC Curve and AUC

y_pred_proba = model.predict(X_test)

fpr, tpr, _ = roc_curve(y_test, y_pred_proba)

roc_auc = roc_auc_score(y_test, y_pred_proba)


plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area =
%0.2f)' % roc_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()


print(f"ROC AUC: {roc_auc}")
```