National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cyber Security

Md Masudur Rahman
Student ID: X23218291

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	MD MASUDUR RAHMAN
Student ID:	X23218291
Programme:	MSc in Cybersecurity **Year:** 2024
Module:	MSc Research Project
Lecturer:	Kamil Mahajan
Submission Due Date:	29-01-2025
Project Title:	Implement a System that can Detect Ransomware Attacks in Real-Time using Behaviour Analysis
Word Count:	383 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	MD MASUDUR RAHMAN
Date:	28-01-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

MD MASUDUR RAHMAN
Student ID: X23218291

1 Introduction

This configuration manual is targeted at setting up, running, and understanding the ransomware detection system using behavior analysis. The system will make use of machine learning models in order to identify ransomware in real time, with strong detection based on feature engineering and exploratory data analysis. This manual will lead the user through the configuration of the system in an efficient way, the preparation of datasets, the execution of model training, and the visualization of the outcomes.

2 Section 1: Prerequisites

Software Requirements:

- Python 3.x installed (recommend Python 3.12+).
- Required Python libraries: pandas, numpy, matplotlib, seaborn, sklearn, scipy.
- IDE or text editor (Jupyter Notebook).

Hardware Requirement:

To ensure optimal performance of the ransomware detection system, the following hardware specifications are recommended:

- **Processor**: Intel i5
- **RAM**: Minimum 8Gb or 16GB.
- **Storage**: At least 10GB free space for smooth and faster execution of code
- **Operating System**: Windows 10/11.

Data Requirements:

- Input dataset in CSV format (data_file.csv).
- Ensure columns align with the described structure in the project (e.g., FileName, DebugSize, Benign, etc.).

3 Section 2: Dataset Preprocessing

Loading Data:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

# Load the dataset
file_path = 'data_file.csv'
df = pd.read_csv(file_path)
```

- Place the dataset file in the same directory as the script.
- Use pd.read_csv('data_file.csv') to load the dataset into a pandas DataFrame.

Cleaning:

```
# Step 2: Checking for Missing Values
missing_values = df.isnull().sum()
print("\nMissing Values in Each Column:")
print(missing_values[missing_values > 0])
```

```
Missing Values in Each Column:
Series([], dtype: int64)
```

```
# Step 3: Removing Irrelevant Columns
irrelevant_columns = ['ID', 'Timestamp'] if 'ID' in df.columns and 'Timestamp' in df.columns else []
df.drop(columns=irrelevant_columns, inplace=True)
```

```
# Step 4: Handling Categorical Variables
# Encoding categorical variables if present
categorical_columns = df.select_dtypes(include=['object']).columns
for col in categorical_columns:
    if df[col].nunique() < 10:  # Only encode if there are few unique categories
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
    else:
        df.drop(columns=[col], inplace=True)  # Drop high-cardinality columns
```

```
# Step 5: Data Preprocessing (Outlier Detection and Removal)
# Detecting outliers based on Z-score
from scipy.stats import zscore

numeric_columns = df.select_dtypes(include=[np.number]).columns
z_scores = np.abs(zscore(df[numeric_columns]))
df = df[(z_scores < 3).all(axis=1)]
```

- Check for missing values using df.isnull().sum().
- Remove or encode high-cardinality categorical columns.
- Ensure all features are appropriately scaled using StandardScaler.

Feature Engineering:

```
# Additional imports for machine learning
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
```

```
# Step 1: Feature Engineering
# Feature: Ratio of DebugSize to ExportSize (to identify unusual file characteristics)
df['Debug_Export_Ratio'] = df['DebugSize'] / (df['ExportSize'] + 1)  # Avoid division by zero

# Feature: SizeOfStackReserve to ResourceSize ratio (anomalies may indicate unusual reserve sizes)
df['Stack_Resource_Ratio'] = df['SizeOfStackReserve'] / (df['ResourceSize'] + 1)

# Feature: Check for any missing data
df.fillna(0, inplace=True)
```

- Generate derived features such as Debug_Export_Ratio and Stack_Resource_Ratio to enhance model prediction.

4 Section 3: Exploratory Data Analysis (EDA)

- Use histograms (sns.histplot) to explore feature distributions.

```
# Step 6: Exploratory Data Analysis (EDA)
# Plotting distributions of numerical features
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(4, 4, i)
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```

- Create box plots (sns.boxplot) to detect and visualize outliers.

```
# Box plots for each feature to identify spread and potential outliers
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(4, 4, i)
    sns.boxplot(y=df[col])
    plt.title(f'Box Plot of {col}')
plt.tight_layout()
plt.show()
```

- Generate a correlation heatmap (sns.heatmap) to identify feature relationships.

```
# Correlation Heatmap
plt.figure(figsize=(12, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

5 Section 4: Model Configuration

Data Split:

```
# Step 2: Prepare Data for Model Training
# Separating features (X) and target (y)
X = df.drop(columns=['Benign'])  # Drop target variable
y = df['Benign']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 3: Data Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Split the dataset into training and testing sets using train_test_split with test_size=0.3.

Scaling:

- Scale features using StandardScaler.

Models:

```python
# Step 4: Model Training
# Train and evaluate multiple models as per architecture

# Model 1: Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)

# Model 2: Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

# Model 3: Support Vector Machine
svm_model = SVC(probability=True, random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
```

- Logistic Regression: Suitable for quick baseline performance.
- Random Forest Classifier: Use for robust predictions with tunable hyperparameters.
- Support Vector Machine (SVM): Employ for handling complex decision boundaries.

Model Evaluation:

```python
# Step 5: Model Evaluation
# Using evaluation metrics as per the architecture

def evaluate_model(y_test, y_pred, y_prob):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob[:, 1])  # AUC requires probability scores

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print(f"AUC: {auc:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

print("Logistic Regression Performance:")
evaluate_model(y_test, y_pred_lr, lr_model.predict_proba(X_test))

print("\nRandom Forest Performance:")
evaluate_model(y_test, y_pred_rf, rf_model.predict_proba(X_test))

print("\nSupport Vector Machine Performance:")
evaluate_model(y_test, y_pred_svm, svm_model.predict_proba(X_test))
```

- Evaluate using metrics like accuracy_score, precision_score, recall_score, f1_score, and roc_auc_score.

6 Section 5: Visualization and Validation

```python
# Import libraries for visualization and metrics
from sklearn.metrics import confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

# Helper function to plot confusion matrix
def plot_confusion_matrix(y_test, y_pred, title):
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
    plt.title(f'Confusion Matrix - {title}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# Helper function to plot ROC curve
def plot_roc_curve(y_test, y_prob, title):
    fpr, tpr, _ = roc_curve(y_test, y_prob[:, 1])
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'Receiver Operating Characteristic - {title}')
    plt.legend(loc="lower right")
    plt.show()
```

- Confusion Matrix: Use sns.heatmap to visualize prediction errors.
- ROC Curve: Generate using roc_curve and auc for model comparison.

7 Section 6: Prediction Functionality

- Use the predict_manual_input() function to manually input features and test the model predictions interactively.

```python
# Prediction Function
def predict_manual_input():
    print("\n--- Manual Prediction ---")
    # List the features needed for prediction (must match trained model features)
    feature_names = X.columns.tolist()
    input_features = []

    # Collect input for each feature
    for feature in feature_names:
        while True:
            try:
                value = float(input(f"Enter value for {feature}: "))
                input_features.append(value)
                break
            except ValueError:
                print("Invalid input. Please enter a numeric value.")

    # Convert to numpy array and reshape
    input_features = np.array(input_features).reshape(1, -1)

    # Scale the input features using the same scaler
    scaled_features = scaler.transform(input_features)

    # Make predictions with all models
    print("\nPredictions:")
    print("Logistic Regression Prediction:")
    lr_pred = lr_model.predict(scaled_features)
    lr_prob = lr_model.predict_proba(scaled_features)
    print(f"Predicted Class: {lr_pred[0]}")
    print(f"Class Probabilities: {lr_prob[0]}")

    print("\nRandom Forest Prediction:")
    rf_pred = rf_model.predict(scaled_features)
    rf_prob = rf_model.predict_proba(scaled_features)
    print(f"Predicted Class: {rf_pred[0]}")
    print(f"Class Probabilities: {rf_prob[0]}")

    print("\nSupport Vector Machine Prediction:")
    svm_pred = svm_model.predict(scaled_features)
    svm_prob = svm_model.predict_proba(scaled_features)
    print(f"Predicted Class: {svm_pred[0]}")
    print(f"Class Probabilities: {svm_prob[0]}")
```

Use Cases

Prediction: Use Case 1

```
# Call the prediction function
predict_manual_input()
```

```
--- Manual Prediction ---
Predictions:
Logistic Regression Prediction:
Predicted Class: 1
Class Probabilities: [0.43592468 0.56407532]

Random Forest Prediction:
Predicted Class: 1
Class Probabilities: [0.05713167 0.94286833]

Support Vector Machine Prediction:
Predicted Class: 1
Class Probabilities: [0.04337279 0.95662721]
C:\Users\dell\AppData\Local\Programs\Python\Python312\Lib\si
  warnings.warn(
```

Prediction: Use Case 2

```
# Call the prediction function
predict_manual_input()
```

```
--- Manual Prediction ---
Predictions:
Logistic Regression Prediction:
Predicted Class: 1
Class Probabilities: [6.90558721e-13 1.00000000e+00]

Random Forest Prediction:
Predicted Class: 1
Class Probabilities: [0. 1.]

Support Vector Machine Prediction:
Predicted Class: 1
Class Probabilities: [6.91393393e-12 1.00000000e+00]
C:\Users\dell\AppData\Local\Programs\Python\Python312\Lib\si
  warnings.warn(
```

References

www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set

Zahoora, U., Khan, A., Rajarajan, M., Khan, S.H., Asam, M. and Jamal, T., 2022. Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier. Scientific reports, 12(1), p.15647. https://www.nature.com/articles/s41598-022-19443-7.pdf

ISLAM, M.Z., 2024. Ransomware Detection Using Machine Learning: A Review, Research Limitations and Future Directions. https://researchoutput.csu.edu.au/ws/portalfiles/portal/480414522/480413277_Published_article.pdf

Urooj, U., Al-rimy, B.A.S., Zainal, A., Ghaleb, F.A. and Rassam, M.A., 2021. Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. Applied Sciences, 12(1), p.172. https://www.mdpi.com/2076-3417/12/1/172

Loco, P., Alonso, S., Hartmann, G., Whitmore, J. and McLaughlin, E., 2024. Adaptive behavior-based ransomware detection via dynamic flow signatures. https://assets-eu.researchsquare.com/files/rs-5317374/v1_covered_be9cdf81-73e4-4f2a-b398-db48ab0bf9f8.pdf?c=1729743725

Garter, L., Johnson, C., Brown, A., Miller, W., Davis, M. and Martin, D., 2024. A Novel Approach of Ransomware Detection Using Dynamic Behavior Modeling and Network Pattern Profiling. https://d197for5662m48.cloudfront.net/documents/publicationstatus/230911/preprint_pdf/601aed5b1c31ba5b87514f0281e1dd3f.pdf