National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Ans Maria Mathew
Student ID: 23173661

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ans Maria Mathew		
Student ID:	23173661		
Programme:	MSc Cybersecurity	**Year:**	2024
Module:	MSc Research Practicum Part 2		
Lecturer:	Jawad Salahuddin		
Submission Due Date:	12th December 2024		
Project Title:	ML- Based Zero Day Attack Detection		
Word Count: 740	**Page Count:** 3		

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: Ans Maria Mathew

Date: 12th December 2024

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

Ans Maria Mathew
Student ID: 23173661

1. Overview

The configuration manual details the setup, configuration, and the use of a Machine Learning (ML)-based Intrusion Detection and Prevention System (IDPS). By following this manual we can configure, train, and deploy an ML-based IDPS effectively to identify anomalies and mitigate zero-day vulnerabilities in a controlled IoT environment.

2. Prerequisites

a. Hardware Requirements
   - Minimum 8 GB RAM
   - Multi-core processor with support for vectorized computations
   - SSD storage for faster data I/O operations

b. Software Requirements
   - Python 3.11.9
   - VSCode
   - Libraries: `pandas`, `numpy`, `sklearn`, `matplotlib`
   - Jupyter Notebook

c. Dataset Requirements

The dataset used for training must consist of network communication records with features such as packet size, response time, source/destination ports, and IP bytes.

3. Jupiter Notebook

Jupyter Notebook is an interactive computing environment that allows users to write and execute Python code, visualize data, and document their workflows seamlessly. If you have a Python file (.py) and wish to open it in Jupyter Notebook, follow the steps below.

Step 1: Install Jupyter Notebook

Before proceeding, ensure Jupyter Notebook is installed on your system. Use the following command to install it: pip install notebook

If you are using Anaconda, Jupyter Notebook comes pre-installed.

Step 2: Launch Jupyter Notebook

a. Open a terminal (or command prompt): Navigate to the directory containing your Python file using the cd command. For example: cd path/to/your/python/file
b. Launch Jupyter Notebook by typing: jupyter notebook
c. This command will open the Jupyter Notebook interface in your default web browser.

Step 3: Create a Notebook

If you want to work with the Python file in notebook format:

a. In the Jupyter Notebook interface, click New (on the right-hand side) and select Python 3 (ipykernel). This will create a new notebook file.

b. Name your notebook by clicking the title (e.g., Untitled) at the top of the page and typing a new name.

Step 4: Open or Import a Python File

Copy and Paste Code

a. Open the Python file in a text editor (e.g., Notepad, VSCode, PyCharm).
b. Copy the code.
c. Paste the code into a cell in the Jupyter Notebook.

Step 5: Save and Run the Notebook

Once your code is in the notebook, you can execute it cell by cell using Shift + Enter. Save your work by clicking the Save icon or pressing Ctrl + S.

By following these steps, you can easily open and work with Python files in Jupyter Notebook for better code management and interactivity.

4. Installation

a. Install Python and Libraries
   pip install pandas numpy scikit-learn matplotlib seaborn
b. Set Up the Dataset
   i. Ensure the dataset is in `.csv` format.
   ii. Place the dataset file in a directory accessible to the script, e.g., `/path/to/iot23_combined_new.csv`.

5. Configuration Steps

a. Data Preparation
   i. Load the dataset using pandas:
   data = pd.read_csv('/path/to/iot23_combined_new.csv')
   ii. Replace invalid or missing values with numerical equivalents:
   data.replace('-', np.nan, inplace=True)
   data = data.apply(pd.to_numeric, errors='coerce')
   data.fillna(0, inplace=True)
b. Feature Selection and Scaling
   i. Select relevant features and target labels:
   X = data.iloc[:, 1:-1]
   y = data.iloc[:, -1]
   ii. Standardize features for model compatibility:
   scaler = StandardScaler()
   X_scaled = scaler.fit_transform(X)
c. Train-Test Split
   i. Split the dataset into training and testing subsets:
   from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

6. Model Training and Evaluation

a. Random Forest Classifier
    i. Train the model:
    ```
    from sklearn.ensemble import RandomForestClassifier
    rf = RandomForestClassifier(random_state=42)
    rf.fit(X_train, y_train)
    ```
    ii. Evaluate accuracy and feature importance:
    ```
    y_pred_rf = rf.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred_rf))
    print("Feature Importance:", rf.feature_importances_)
    ```
b. Support Vector Machine (SVM)
    i. Train the model:
    ```
    from sklearn.svm import SVC
    svm = SVC(kernel='rbf', probability=True, random_state=42)
    svm.fit(X_train, y_train)
    ```
    ii. Evaluate performance:
    ```
    y_pred_svm = svm.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred_svm))
    ```
c. Neural Network
    i. Train the model:
    ```
    from sklearn.neural_network import MLPClassifier
    nn = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
    nn.fit(X_train, y_train)
    ```
    ii. Evaluate performance:
    ```
    y_pred_nn = nn.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred_nn))
    ```
d. Isolation Forest for Anomaly Detection
    i. Train the anomaly detection model:
    ```
    from sklearn.ensemble import IsolationForest
    iso_forest = IsolationForest(random_state=42)
    iso_forest.fit(X_train)
    ```
    ii. Identify anomalies:
    ```
    anomaly_predictions = iso_forest.predict(X_test)
    print("Normal Count:", np.sum(anomaly_predictions == 1))
    print("Anomaly Count:", np.sum(anomaly_predictions == -1))
    ```

References

Guo, Y., 2023. A review of Machine Learning-based zero-day attack detection: Challenges and future directions. *Computer communications*, *198*, pp.175-185.

Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J.-N., Bayne, E., & Bellekens, X., 2020. Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. *Electronics*, 9(10), p.1684.