National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cybersecurity

Bhaskar Guttikonda
Student ID: x23243791

School of Computing
National College of Ireland

Supervisor: Imran khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Bhaskar Guttikonda
Student ID:	X23243791
Programme:	MSc in Cybersecurity **Year:** 2024-2025
Module:	MSc Practicum
Lecturer:	Imran Khan
Submission Due Date:	29-01-2025
Project Title:	Adaptive Detection of Advanced Persistent Threats (APT) in Multi Layered Network Environments
Word Count:	974 **Page Count:** 9

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:	Bhaskar Guttikonda
Date:	29-01-2025

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

Bhaskar Guttikonda
x23243791

1 System Overview

1.1 Introduction

This configuration manual provides detailed guidance for setting up and configuring the Advanced Persistent Threat (APT) detection system. The system incorporates three primary detection modules: network traffic analysis, phishing URL detection, and keylogger detection.

1.2 System Requirements

- Python 3.11.7 or higher
- Required libraries: TensorFlow, Keras, scikit-learn, pandas, numpy
- Minimum 8GB RAM recommended
- Storage space for model files and datasets

2 Network Traffic Analysis Module

2.1 Component Overview

The network traffic analysis module utilizes Sequential Neural Network architecture to detect various network-based threats including backdoor, DoS, and password attacks.

2.2 Configuration Parameters

- Input dimensions: Network flow features
- Hidden layers: Two dense layers (64 and 32 neurons)
- Dropout rate: 0.3
- Learning rate: 0.001
- Batch size: 32

2.3 Performance Metrics

- Accuracy: 99.30%
- Loss value: 0.0187
- Real-time processing capability

3 Phishing URL Detection Module

3.1 Component Setup

The phishing detection system implements NLP-driven approaches with multiple classification methods.

3.2 Configuration Settings

- Text processing: RegexpTokenizer with pattern '[A-Za-z]+'
- Stemming: SnowballStemmer (English configuration)
- Feature extraction: CountVectorizer
- Dataset size: 549,346 labeled URLs

3.3 Performance Parameters

- Training accuracy: 97.81%
- Testing accuracy: 96.45%
- Precision for legitimate URLs: 99%
- Precision for malicious URLs: 91%

4 Keylogger Detection Module

4.1 System Configuration

The keylogger detection framework processes network flow data using multiple machine learning models.

4.2 Feature Configuration

- Flow metrics processing
- Statistical features analysis
- TCP flag information processing
- Custom behavioral metrics calculation

4.3 Model Parameters

- Random Forest configuration: 20 estimators
- Decision Tree parameters: Default settings
- Logistic Regression: Maximum iterations 1000

4.4 Performance Metrics

- Overall accuracy: 96%
- Feature importance tracking enabled
- Real-time behavioural analysis

5 Inference Setup and Configuration

5.1 Environment Requirements

5.1.1 Package Dependencies

- joblib
- numpy
- scikit-learn
- tensorflow.keras
- sklearn.metrics

5.2 Import Statements

import joblib

import numpy as np

from sklearn.metrics import accuracy_score

from tensorflow.keras.models import load_model

5.3 Model Path Configuration

5.3.1 7.2.1 Base Directory

- Root Path: Anywhere in the pc {for example downloads}{ C:\Users\User\Downloads>}
- Place the model in the directory (create a sub directory for example APT) and place it in the directory.

7.2.2 Model Files

1. Phishing URL Model
    - File: phishing_url_model.pkl
    - Type: Pickle file
    - Full path: C:\Users\User\Downloads \phishing_url_model.pkl
2. Keylogger Model
    - File: key_logger_model.pkl
    - Type: Pickle file
    - Full path: C:\Users\User\Downloads \APT\key_logger_model.pkl
3. Network Traffic Model
    - File: ann_model.h5

- Type: HDF5 file
- Full path: C:\Users\User\Downloads \APT\ann_model.h5

5.4 Data Loading Configuration

And place the training data in the same directory i.e C:\Users\User\Downloads\APT

5.4.1 Phishing URL Data

- Training data: phishing_url_train_data.pkl
- Testing data: phishing_url_test_data.pkl
- Data format: Split into trainX, trainY, testX, testY

5.4.2 Keylogger Data

- Combined file: key_logger_data.pkl
- Contains: X_train, y_train, X_test, y_test
- Format: Pickle file with multiple arrays

5.4.3 Network Traffic Data

- File: ann_data.npz
- Format: NumPy compressed archive
- Contains: X_train, y_train, X_test, y_test arrays

5.5 Model Loading Procedures

5.5.1 Phishing URL Model Loading

phishing_url_model = joblib.load([model_path])

trainX, trainY = joblib.load([train_data_path])

testX, testY = joblib.load([test_data_path])

5.5.2 Keylogger Model Loading

keylogger_model = joblib.load([model_path])

X_train, y_train, X_test, y_test = joblib.load([data_path])

5.5.3 Network Traffic Model Loading

loaded_model = load_model([model_path])

loaded_data = np.load([data_path])

5.6 Evaluation Procedures

5.6.1 Phishing URL Model Evaluation

- Method: model.score(testX, testY)

- Output: Accuracy score

5.6.2 Keylogger Model Evaluation

- Method: accuracy_score(y_test, model.predict(X_test))
- Output: Classification accuracy

5.6.3 Network Traffic Model Evaluation

- Method: model.evaluate(X_test, y_test)
- Outputs: Loss and accuracy metrics

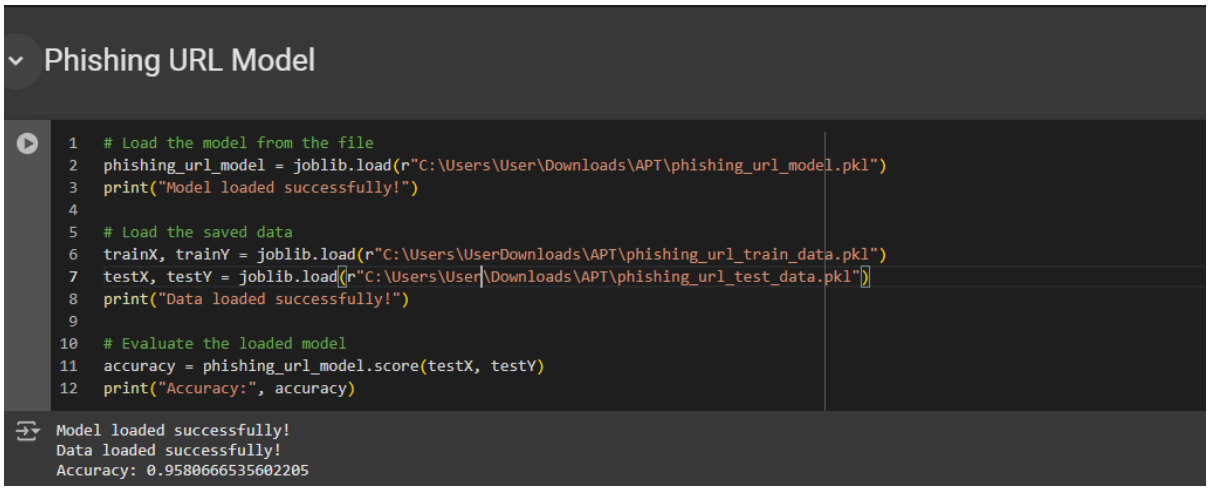
5.7 Expected Outputs

5.7.1 Success Messages

- "Model loaded successfully!"
- "Data loaded successfully!"
- Accuracy scores in decimal format

5.7.2 Performance Metrics

- Accuracy scores for each model
- Loss values for neural network model
- Prediction arrays for validation



Keylogger Model

```
model_path = r"C:\Users\User\Downloads\APT\key_logger_model.pkl"


# Load the model
keylogger_model = joblib.load(model_path)
print("Model loaded successfully!")

# Load the data
loaded_data_path = r"C:\Users\User\Downloads\APT\key_logger_data.pkl"
X_train, y_train, X_test, y_test = joblib.load(loaded_data_path)
print("Data loaded successfully!")
```

```
Model loaded successfully!
Data loaded successfully!
```

```
# Perform predictions
loaded_y_pred_rf = keylogger_model.predict(X_test)

# Evaluate the loaded model
loaded_key_logger_accuracy = accuracy_score(y_test, loaded_y_pred_rf)
print(f"Accuracy of the loaded model: {loaded_key_logger_accuracy:.2f}")

```

```
Accuracy of the loaded model: 0.96
```

Network Traffic Data

```
model_path = r"C:\Users\User\Downloads\APT\ann_model.h5"
data_path = r"C:\Users\User\Downloads\APT\ann_data.npz"
```

```
[ ]
from tensorflow.keras.models import load_model

# Load the model
loaded_model = load_model(model_path)
print("Model loaded successfully!")
```

```
Model loaded successfully!
```

```
[ ]
# Load the data
loaded_data = np.load(data_path)

# Extract training and testing datasets
X_train = loaded_data['X_train']
y_train = loaded_data['y_train']
X_test = loaded_data['X_test']
y_test = loaded_data['y_test']
print("Data loaded successfully!")
```

```
Data loaded successfully!
```

```
[ ]
# Make predictions with the loaded model
y_pred = loaded_model.predict(X_test)

# Evaluate the loaded model
loss, accuracy = loaded_model.evaluate(X_test, y_test, verbose=1)
print(f"Loaded Model Accuracy: {accuracy:.2f}")

```

```
647/647 [==============================] - 2s 2ms/step
647/647 [==============================] - 2s 2ms/step - loss: 0.0199 - accuracy: 0.9931
Loaded Model Accuracy: 0.99
```

5.8 Troubleshooting Guide

5.8.1 Common Issues

1. File not found errors

- Verify path existence
- Check file permissions
- Confirm path formatting

2. Memory issues
    - Monitor RAM usage
    - Clear variables when possible
    - Use appropriate batch sizes
3. Model loading errors
    - Verify model format
    - Check compatibility
    - Confirm dependencies

5.8.2 Best Practices

1. Always use raw strings (r"path") for Windows paths
2. Implement error handling for file operations
3. Verify model loading before proceeding with evaluation
4. Monitor system resources during large data loading
5. Keep consistent naming conventions for files and variables

6 References

https://www.solarwinds.com/security-event-manager/use-cases/apt-security-software

thatDot. (n.d.). Advanced Persistent Threat Detection. Retrieved from https://www.thatdot.com/use-cases/advanced-persistent-threat-apt-detection/

TechTarget. (n.d.). Advanced Persistent Threat (APT). Retrieved from https://www.techtarget.com/searchsecurity/definition/advanced-persistent-threat-APT

MarkovML. (n.d.). Guide to ML Model Deployment. Retrieved from https://www.markovml.com/blog/model-deployment