

Configuration Manual

MSc Research Project
Master of Science in Cyber Security

Shashank Basavaraju
Student ID: 22241817

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Shashank Basavaraju
Student ID: 22241817
Programme: Master Of Science in Cybersecurity **Year:** 2023-2024
Module: Practicum Part2
Lecturer: Michael Prior
Submission Due Date: 12th August 2024
Project Title: MSc Research Project Part 2
Word Count: 695 **Page Count:** 7

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: Shashank Basavaraju

Date: 11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Shashank Basavaraju

Student ID: 22241817

Table of Contents

Table of Contents	1
1 Introduction	2
2 Experimental Setup.....	2
3 Technologies and Software used for Implementation	2
4 Implementation	3
References.....	7

1 Introduction

This configuration manual provides all the details about the tools and technologies used in this project. Section 2 contains an overview of the experimental setup. Section 3 covers the various technologies and software tools used. Section 4 is a step-by-step guide for setting up packet sniffing, training the deep learning model, and setting up the honeypot. Finally, the last section includes references for the software guide.

2 Experimental Setup

The experiment was conducted on a personal computer configured specifically for this purpose. The setup was chosen to ensure optimal performance and smooth execution of all required tasks.

- **Hardware Specifications:** The computer is equipped with an Intel i5 processor, 16GB of RAM, and a 256GB SSD. These specifications provide a balance of performance and efficiency, suitable for running complex simulations and data processing tasks involved in the project.
- **Operating System:** Windows 10, Kali Linux
- **Experimental Setup:** Anaconda 3.1 with Python 3.5, Visual Studio Code, VirtualBox, Kali Linux, XCTU, Packet Sniffer (Zigbee), Arduino IDE

3 Technologies and Software used for Implementation

- **Software Used:** Anaconda 3.1 with Python 3.9.13, Visual Studio Code, VirtualBox, Ubuntu, XCTU, Packet Sniffer (Zigbee), Arduino IDE
- **Anaconda:** Anaconda is a widely used open-source distribution for Python and R programming languages. It provides a comprehensive platform that includes package management and environment management, simplifying the process of setting up and managing different development environments. Anaconda comes pre-loaded with numerous libraries and tools essential for data science, machine learning, and scientific computing (Kamingu, 2023)
- **Visual Studio Code:** Visual Studio Code is a powerful and user-friendly IDE used for coding, debugging, and running experiments. Its extensive range of extensions and customization options makes it an ideal choice for developing complex projects.
- **VirtualBox:** VirtualBox is a virtualization software that allows the creation of virtual machines. It was used to set up a virtual Ubuntu environment, providing a flexible and isolated development environment for testing and implementation.

- **Kali Linux:** Kali Linux is a Debian-based Linux distribution widely used for penetration testing and security auditing. It includes numerous pre-installed tools for various aspects of cybersecurity, making it ideal for testing and analyzing security vulnerabilities in the project (L *et al.*, 2022).
- **XCTU:** XCTU is a software platform used for configuring and testing ZigBee networks. It provides tools for network management, node discovery, and configuration, ensuring reliable communication between devices.
- **Packet Sniffer (Zigbee):** The ZigBee packet sniffer is a tool used to capture and analyse ZigBee communication packets. It helps in monitoring network traffic and identifying potential security issues.
- **Arduino IDE:** The Arduino Integrated Development Environment (IDE) is used for programming and interfacing with ESP32 microcontrollers. It provides a simple and straightforward platform for writing code and uploading it to the hardware (Louis, 2016)

4 Implementation

Step 1: Transmitting data from the DHT11 sensor within the Zigbee environment



```

DHT11_ESP32 | Arduino 1.8.12
File Edit Sketch Tools Help
DHT11_ESP32
1 #include <DHT.h>
2
3 // Define the type of sensor and pin number
4 #define DHTPIN 4 // Digital pin connected to the DHT sensor
5 #define DHTTYPE DHT11 // DHT 11
6
7 // Initialize DHT sensor for normal 16 MHz Arduino
8 DHT dht(DHTPIN, DHTTYPE);
9
10 // Packet number
11 static int packetNumber = 0;
12
13 void setup() {
14   Serial.begin(9600); // Initialize primary serial communication for debugging
15   Serial2.begin(9600); // Initialize secondary serial communication
16
17   // Initialize DHT sensor
18   dht.begin();
19 }
20
21 void loop() {
22   // Wait a few seconds between measurements

```

Figure (1): Sending data

Step 2: Receiving and displaying the data on an OLED screen.

```

OLED$
File Edit Sketch Tools Help

1 #include <Wire.h>
2 #include <Adafruit_GFX.h>
3 #include <Adafruit_SSD1306.h>
4
5 // OLED display settings
6 #define SCREEN_WIDTH 128 // OLED display width, in pixels
7 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
8
9 // Create an instance of the SSD1306 display
10 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
11
12 void setup() {
13   // Initialize serial communication
14   Serial.begin(9600);
15   Serial2.begin(9600);
16   Serial.print("OK");
17
18   // Initialize the OLED display
19   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
20     Serial.println("SSD1306 allocation failed");
21     for (;;)
22   }

```

Figure 2: Receiving data

Step 3: Capturing packets within the Zigbee network.

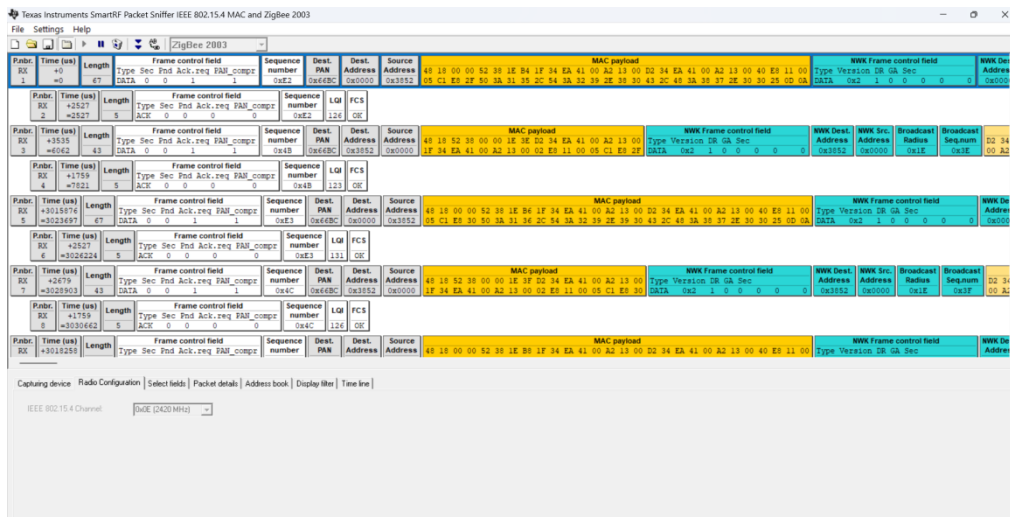


Figure 3: Packet capturing

Step 4 : Download and install Anaconda in windows

Step 5: Create a Django web application and integrate honeypot

```

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "tempdata",
    "admin_honeypot",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "tempdata.middleware.LogRequestMiddleware",
]

```

Figure (4): Integration of Django honeypot

Step 6: Configure sniffed data into the application.

Step 7: Develop a middleware to fetch the logs of the outside attacks.

```
def __call__(self, request):

    client_ip, client_port = self.get_client_ip(request)
    current_time = time.time()

    with self.lock:
        self.requests[client_ip].append((current_time, client_port))

        self.requests[client_ip] = [(t, p) for t, p in self.requests[client_ip] if current_time - t < 60]

    response = self.get_response(request)
    return response

def get_client_ip(self, request):
    """Get client IP address from request, accounting for reverse proxy headers."""
    x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
    if x_forwarded_for:
        ip = x_forwarded_for.split(',')[0]
        port = request.META.get('REMOTE_PORT')
    else:
        ip = request.META.get('REMOTE_ADDR')
        port = request.META.get('REMOTE_PORT')
    return ip, port
```

Figure(5): Log Middleware

Step 8: Enable middleware script and honeypot by adding into the settings.

Step 9: Configure URL paths to display various webpages of the application.

```
from django.contrib import admin
from django.urls import path, include
from tempdata.views import *

urlpatterns = [
    path("admin/", include("admin_honeypot.urls")),
    path("secret/", admin.site.urls),
    path('', login),
    path('login/', login, name='login'),
    path('index/', index, name='index'),
    path('prediction/', pred_page, name='prediction'),
]
```

Figure (6): URL paths of web application

Step 10: While the application runs, capture network traffic of TCP protocols.

```

cap = pyshark.LiveCapture(interface=interface, display_filter='tcp')
captured_packets = []

try:
    for packet in cap.sniff_continuously(packet_count=25):
        time = packet.sniff_time.strftime('%Y-%m-%d %H:%M:%S.%f')
        source = packet.ip.src if 'ip' in packet else 'N/A'
        destination = packet.ip.dst if 'ip' in packet else 'N/A'
        length = packet.length
        info = packet.info if 'info' in packet else 'N/A'
        protocol = packet.transport_layer if hasattr(packet, 'transport_layer') else 'N/A'

        captured_packets.append({
            'time': time,
            'source': source,
            'destination': destination,
            'length': length,
            'info': info,
            'protocol': protocol
        })

except KeyboardInterrupt:
    print('Capture stopped by user.')

finally:
    # Close the capture process
    cap.close()

    # Save captured data to a CSV file
    csv_filename = 'captured_packets.csv'

```

Figure (7): Capture network traffic

Step 11: Import libraries for developing a CNN-GRU model architecture to predict the attack.

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import chi2, SelectKBest
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, GRU, Flatten, MaxPooling1D, BatchNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.models import load_model, model_from_json

```

Figure (8): Libraries for machine learning

Step 12: Develop CNN- GRU architecture .

```

model = Sequential([
    Conv1D(64,3,activation='relu',padding='same',kernel_regularizer=l2(0.03),input_shape=(X_train_reshape.shape[1],1)),
    BatchNormalization(),
    Dropout(0.2),
    GRU(128,activation='relu',return_sequences=True),
    Flatten(),
    Dense(1,activation='sigmoid')
])

```

Figure (9): Model Architecture

Step 13: Save the trained model and utilize the file for prediction.

Step 14: Input the captured traffic into the model and get the prediction.

Step 15: Display the resultant log details on the admin page.

References

- Kamingu, G. (2023) *Python vs. R for Data Scientists: An introduction*. Available at: <https://doi.org/10.13140/RG.2.2.17253.37602>.
- L, G.H. *et al.* (2022) 'Analysis of Cyber Security Attacks using Kali Linux', in *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE). 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pp. 1–6. Available at: <https://doi.org/10.1109/ICDCECE53908.2022.9793164>.
- Louis, L. (2016) 'Working Principle of Arduino and Using it as a Tool for Study and Research', *International Journal of Control, Automation, Communication and Systems*, 1(2), pp. 21–29. Available at: <https://doi.org/10.5121/ijcacs.2016.1203>.