Configuration Manual

MSc Research Project
Cyber Security

Felipe Sá
Student ID: x19132352

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Felipe Tavares de Sá

Student ID: X19132352

Programme: Cyber Security **Year:** 2022

Module: MSc Internship

Lecturer: Michael Pantridge

Submission Due Date: 15/08/2022

Project Title: Configuration Manual

Word Count: 918 **Page Count:** 10

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: Felipe Tavares de Sá

Date: 14th August 2022

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

Felipe Sá
X19132352

1 Introduction

This configuration manual outlines the procedures for setting up the lab necessary to conduct the research.

Additionally employed for comparison purposes in this study are the XGBoost, Random Forest, Decision Tree and Logistic Regression algorithms. Python libraries were used to implement the entire project: Introduction, System Configuration, Network Diagram, Software Requirements, Environmental Setup, Data Selection and Collection, Data Cleaning, Model Training and Evaluation metrics for Classifiers are the sections of this configuration manual.

2 System Configuration

- **Raspberry Pi 4 Model B with Alfa Network AWUS036NHA**

Processor : Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory(RAM) Installed : 8 GB LPDDR4-3200 SDRAM
System Type : Kali Linux 2022.3
Storage: 64 GB Micro-SD

- **Raspberry Pi 3 Model B+ with Alfa Network AWUS036NHA**

Processor : BCM2837B0, Cortex-A53, 64-bit, quad-core SoC @ 1.4GHz
Memory(RAM) Installed : 1GB LPDDR2 SDRAM
System Type : Kali Linux 2022.3
Storage: 64 GB Micro-SD

Figure 1: Portable attacking machine Raspberry Pi with Alfa AWUS036NHA

- **Router (Access Point) Asus RT-AX82U**

Processor : 1.5 GHz tri-core processor
Memory : 256 MB Flash | 512 MB RAM

- **Desktop**

Processor : AMD® Ryzen™ 9 3900X 12-Core Processor
Memory(RAM) Installed : 64 GB DDR4 3200 MHz
System Type : Windows 10 21H2, 64 Bit Operating System with x64-based processor
Storage: Sabrent Rocket 4.0 1TB
GPU : AMD Radeon RX 5700 XT, 8 GB GDDR64

- **Various IoT Devices:** TP-Link KASA, TuyaSmart, Google Nest Mini & Hub, Coredy Robot Vacuum, among others.

3 Network Diagram

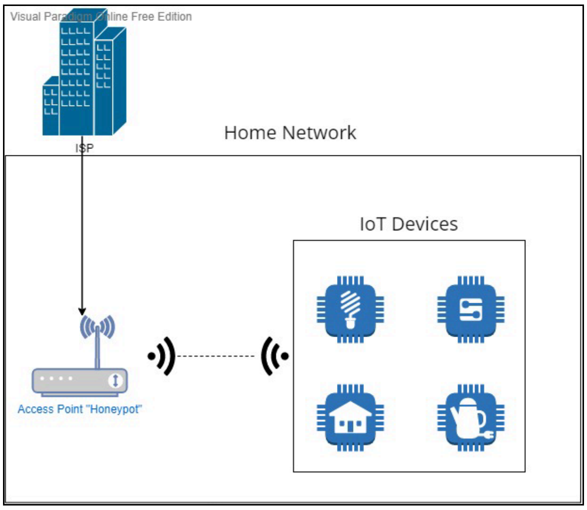

Figure 2: Network diagram

4 Software Requirements

The implementation needed three key components:

- **Google Colab**: Colaboratory, sometimes known as "Colab," is a data analysis and machine learning application that enables you to integrate rich text, charts, photos, executable Python code, HTML, LaTeX, and more into a single Google Drive document. You may quickly share your work and work together with others because to its connection to the robust Google Cloud Platform runtimes.

- **Wireshark:** The most famous and commonly used network protocol analyser in the world is called Wireshark. It is the de facto (and frequently de jure) standard across many commercial and non-profit firms, governmental organisations, and educational institutions because it enables you to observe what's occurring on your network at a microscopic level.

- **Aircrack-ng:** A full set of tools to evaluate Wi-Fi network security is called Aircrack- ng. It focuses on several aspects of Wi-Fi security, including Observation: Packet capture and data export to text files for processing by external tools. Attacking: Via packet injection, replay attacks, deauthentication, bogus access points, and other methods.

5 Environmental Setup

We conducted this investigation using Google Colab, as previously mentioned, which does not need any environmental setup on a local PC. To set up and run the code files necessary for this research, follow these instructions below:

Select "Upload" and then click on "Choose File":

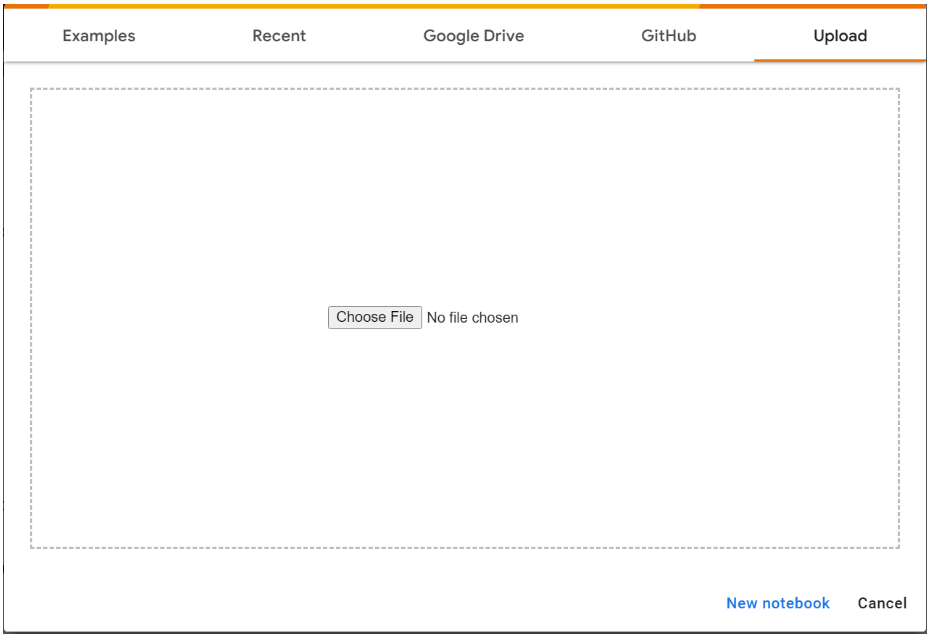

Figure 3: Uploading the Jupyter notebook file

Select the file "**Classification_of_Deauth_Attack_on_WiFi_Using_RaspberryPI.ipynb**":

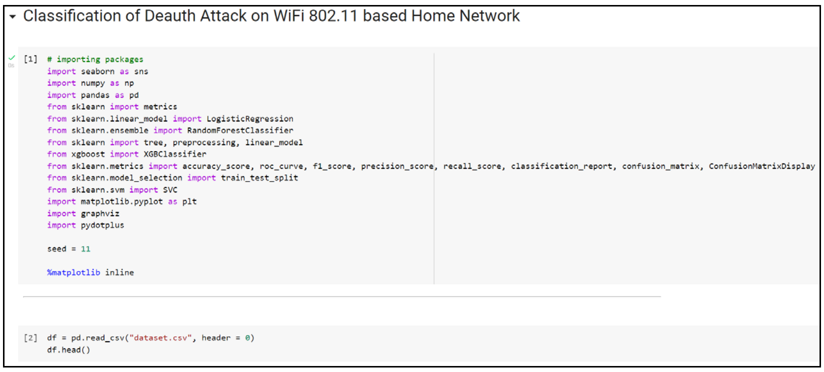

Figure 4: Classification_of_Deauth_Attack_on_WiFi_Using_RaspberryPI.ipynb loaded on Google Colab

6 Data Selection and Collection

The data set used in this study has been capured in-house from author's personal Wi-Fi environment using Wireshark tool and the raspberry pi with an Alfa antenna in monitor mode.

6.1 Data Cleaning

Before moving on to the model building stage, the data was cleaned up and we made improvements to ensure that the model performs at its best. The information in the "Dest Name" and "SrcName" columns were standardized, and empty fields were filled in as "blank".

6.2 Data Collection using Wireshark

This image describes the network traffic that was captured. Alfa AWUS036NHA has been put in "monitor mode" to sniff 802.11 traffic on channel number 9 for two uninterrupted hours.

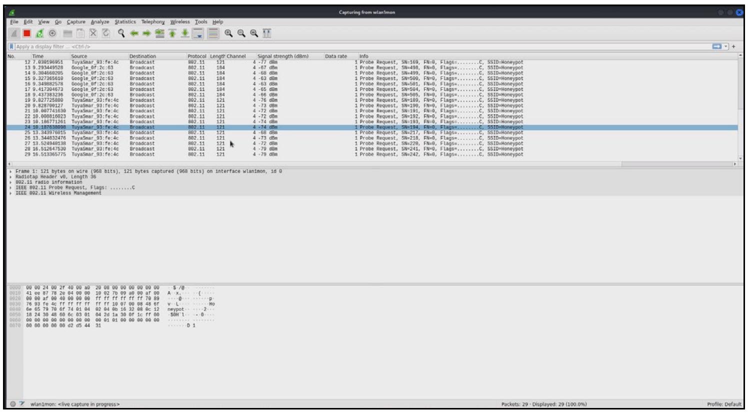

Figure 5: Data capture using Wireshark

To obtain the traffic considered "DoS", we performed a Deauth attack using aircrack-ng tool and then captured it again using Wireshark. Deauthentication attacks are a particular kind of attack that target router-to-device connectivity, effectively turning off the device's Wi-Fi.

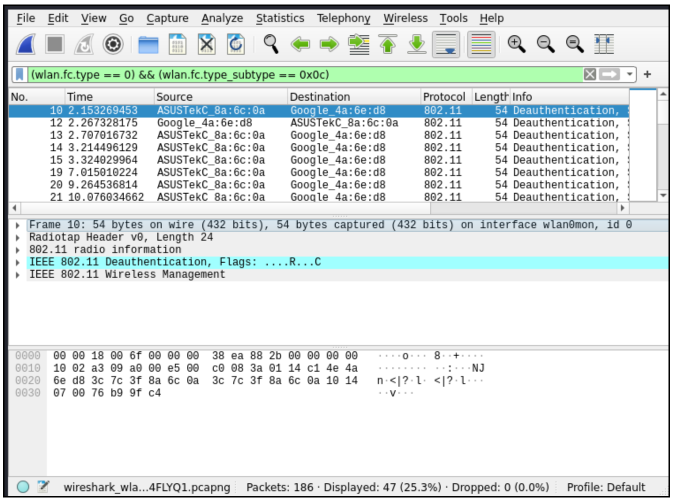

Figure 6: Deauth data capture using Wireshark

6.3 Uploading the dataset into Google Colab

1. Click on the menu > Folder icon
2. Upload the database into Google Colab
3. Select "**dataset.csv**" > Open

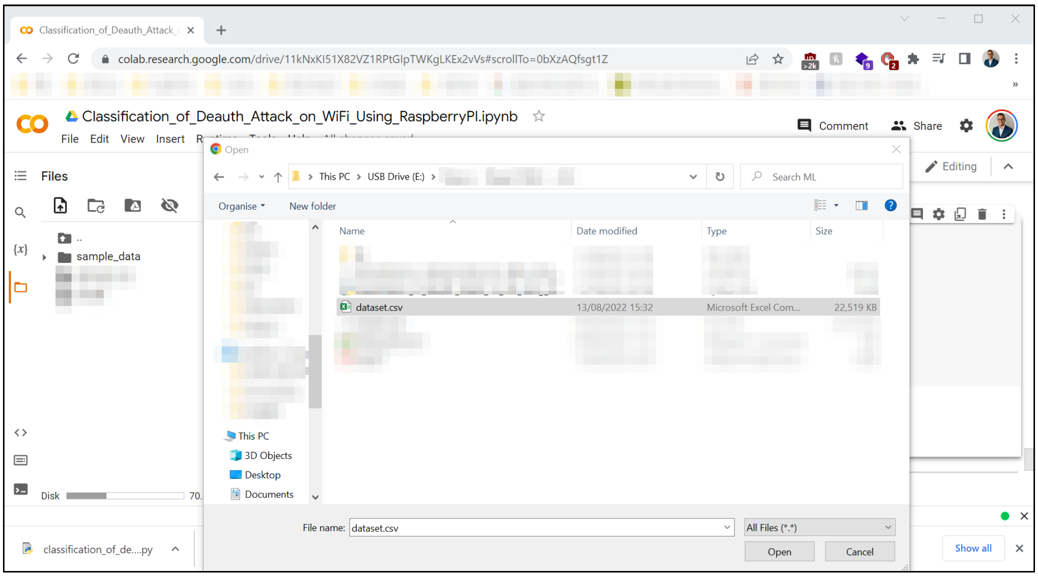

Figure 7: Importing the dataset.csv into Google Colab

The following is a discussion of the dataset's features:

```
[5]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414940 entries, 0 to 414939
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Protocol   414940 non-null  object
 1   Bytes      414940 non-null  int64
 2   TimeDelta  414940 non-null  float64
 3   DestName   414940 non-null  object
 4   SrcName    414940 non-null  object
 5   Class      414940 non-null  object
dtypes: float64(1), int64(1), object(4)
memory usage: 19.0+ MB
```

Figure 8: df.info() of the dataset

The chosen dataset comprises 6 columns. Our binary target variables are called "Bytes" and "Class."

7 Model Training

The most important aspect of our study is modelling. We are selecting our models, training them on imbalanced data, and comparing their performance against a validation dataset to assess our suggested methods.

7.1 Training Logistic Regression

It is a statistical parametric approach that takes into account a binomial classification issue, in our case, "deauth" or "normal" packets. The classification value is calculated using the likelihood of each feature.

Model Evaluation

```
train_lg_pred = LR.predict(X_train)
test_lg_pred = LR.predict(X_test)

score_lg_train = round(accuracy_score(y_train, train_lg_pred) * 100, 2)
score_lg_test = round(accuracy_score(y_test, test_lg_pred) * 100, 2)
print("Accuracy of Logistic Regression on training dataset: ", score_lg_train)
print("Logistic Regression Classifier Accuracy: ", score_lg_test)
print(classification_report(y_test, test_lg_pred,target_names=target_names))
```

```
Accuracy of Logistic Regression on training dataset:  91.15
Logistic Regression Classifier Accuracy:  91.16
              precision    recall  f1-score   support

      Normal       0.91      1.00      0.95    149576
      Deauth       1.00      0.10      0.19     16400

    accuracy                           0.91    165976
   macro avg       0.96      0.55      0.57    165976
weighted avg       0.92      0.91      0.88    165976
```

Figure 9: Logistic Regression Classifier Model

7.2 Training Decision Tree (DT)

The dataset is divided into sections depending on the contribution of each feature by the DT algorithm, which is based on information gain (entropy, Gini impurity), until the DT correctly classifies all training data.

Model Evaluation

```
[15] print ("Tree Classifier Accuracy: ")
     print (clf.score(X_test,y_test))
     y_pred_dt = clf.predict(X_test)
     print(classification_report(y_test, y_pred_dt, target_names=target_names))
```

```
Tree Classifier Accuracy:
1.0
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00    149577
      Deauth       1.00      1.00      1.00     12969

    accuracy                           1.00    162546
   macro avg       1.00      1.00      1.00    162546
weighted avg       1.00      1.00      1.00    162546
```

Figure 10: Tree Classifier Model

7.3 Training Random Forest

Machine learning code for the Random Forest Classifier that includes the accuracy, precision, f1-score, and recall assessment metrics. Using a k10 fold cross-validation accuracy and confusion matrix, the model's accuracy is assessed.

Random Forest Classifier

```
[97] from pandas.core.common import random_state
     RFC = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 0)
     RFC.fit(X_train, y_train)

     y_pred = RFC.predict(X_test)

     #evaluation and results
     print(classification_report(y_test,y_pred))

     #ROC curve
     fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    149576
           1       1.00      1.00      1.00     16400

    accuracy                           1.00    165976
   macro avg       1.00      1.00      1.00    165976
weighted avg       1.00      1.00      1.00    165976
```

Figure 11: Random Forest Classifier Model

7.4 Training XGBoost

In this section, we trained our XGBoost model using unbalanced data.

```
XGBoost

[95] xgb_model = XGBClassifier(learning_rate=1000, max_depth=3, min_child_weight=5, n_estimators=5, n_jobs=-1, gamma=10)
     xgb_model.fit(X_train, y_train)

     xgb_pred = xgb_model.predict(X_test)

     xgb_accuracy = accuracy_score(y_test, xgb_pred)
     xgb_accuracy

     print("Training Accuracy: ",xgb_model.score(X_train, y_train))
     print("Test Accuracy: ", xgb_accuracy)
     print(classification_report(y_test, xgb_pred,target_names=target_names))

     Training Accuracy:  1.0
     Test Accuracy:  1.0
                   precision    recall  f1-score   support

          Normal       1.00      1.00      1.00    149576
          Deauth       1.00      1.00      1.00     16400

        accuracy                           1.00    165976
       macro avg       1.00      1.00      1.00    165976
    weighted avg       1.00      1.00      1.00    165976
```

Figure 12: XGBoost Classifier Model

8 Evaluation metrics for Classifiers

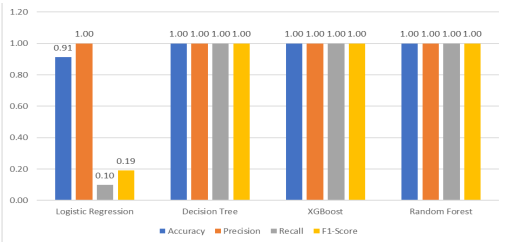

Figure 13: Evaluation metrics for Classifiers