National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Sai Hari Parthiban
Student ID: 22169148

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sai Hari Parthiban		
Student ID:	22169148		
Programme:	MSc in Cyber Security	**Year:**	2023
Module:	MSc Research Project		
Lecturer:	Jawad Salahuddin		
Submission Due Date:	05/04/2024		
Project Title:	Android Malware Detection using Machine Learning and Convolutional Neural Network		
Word Count:	415	**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: Sai Hari Parthiban

Date: 04/04/2024

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

Sai Hari Parthiban
Student ID: 22169148

1 Introduction

This document contains all the information and details about the software technology, tools that are used during this research project that is used in the idea of improving the Android Malware Detection using Machine Learning Algorithms and Convolutional Neural Network in the data preparation , feature extraction, implementation and evaluation phases.

2 Section 2

Code Editor
VS code studio 1.85.0
Google Colab
Program Language
python – version >3.7
Web Brower
Google chrome
Hardware Specification
Ram – 4GB
Disk Space – Minimum 2GB
OS – Windows 10 and above
NVIDIA GPU driver version: Windows 461.33 or higher

3 Package Details

Python libraries used are numpy, pandas, seaborn, scikit learn and etc

Numpy – Version 1.21
Used for numerical operations in python. Supports handling large number dimensional arrays and matrices.
Pandas – Version 1.3.5
Used for data manipulation and offer structured dataframes.
Seaborn – Version 0.12.0
Used for statistical data visualization in python like graphs.
Scikit learn – Version 0.22

This library contribute to various machine learning workflows.
Tkinter – Version 8.6
used to construct basic graphical user interface (GUI) applications.

4 Dataset

The dataset https://www.unb.ca/cic/datasets/invesandmal2019.html, more precisely the "invesandmal2019" dataset from the University of New Brunswick, comprises of gathered samples of Android applications (APK files) classified as benign or malicious (or "malign").

5 Implementation

Step 1: Importing the packages

```
import pandas as pd
import numpy as np
import pickle
import shutil
import os
import zipfile
from androguard.core.bytecodes.apk import APK
import os, sys, stat
import seaborn as sns
import numpy as np
import sklearn
import pickle
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
import six
import sys
sys.modules['sklearn.externals.six'] = six
from mlxtend.classifier import StackingClassifier
from sklearn.metrics import classification_report, confusion_matr

from PIL import ImageTk,Image
import tkinter as tk
from tkinter import *
from tkinter import filedialog
```

Required packages and libraries are imported

Step 2: Collecting the data

```
!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Adware.zip
!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/PremiumSMS.zip
!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Ransomware.zip
!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/SMS.zip
!wget http://205.174.165.80/CICDataset/CICInvesAndMal2019/Dataset/APKs/Scareware.zip
```

Step 3: Extracting features from Androguard

```
permissions =  []
countM = 0
countB = 0

for entry in os.scandir(benign):
  if entry.path.endswith(".apk") and entry.is_file():
          try:
            a = APK(entry.path)
            perm = a.get_permissions()
            countB += 1
            for per in perm:
              if per not in permissions and per.startswith('android.permission'):
                permissions.append(per)
          except:
            os.remove(entry.path)

for entry in os.scandir(malign):
  if entry.path.endswith(".apk") and entry.is_file():
          try:
            a = APK(entry.path)
            perm = a.get_permissions()
            countM += 1
            for per in perm:
              if per not in permissions and per.startswith('android.permission'):
                permissions.append(per)
          except:
            os.remove(entry.path)
```

The above code aims to collect all unique permissions requested by the APK files in the provided benign and malignant directories.

Step 4: Loading the data

```
#data loading
dataframe = pd.read_csv('/content/drive/MyDrive/android_malware/android_dataset.csv')
dataframe.head()
```

Dataset is loaded from the desired location file with .csv

Step 5: Data Visualization

```
df1 = dataframe[['class']].groupby(['class']).value_counts()
df1 = df1.reset_index()
df1.columns = ['class','counts']
fig = px.bar(df1, x='class', y='counts',color='class',title='value count of target class')
fig.show()
```

This code effectively visualizes the distribution of classes in the DataFrame, providing insights into the number of data points belonging to each class.

Step 6: Data balancing

```
#data balancing using smote over sampling
oversample = SMOTE()
X, Y = oversample.fit_resample(X,Y)
```

Smote is used to balance ht data over sampling

Step 7: Training the Models

Step 7.1: Random Forest Classifier

```
rfc_model = RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=4)
rfcmodel = rfc_model
rfcmodel.fit(X_train,y_train)
pred = rfcmodel.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test,pred)*100)
target_names = ['benign','malign']
```

Step 7.2: Extra-Tree Classifier

```
etc_model = ExtraTreesClassifier(n_estimators=12, max_depth=6,)
etcmodel = etc_model
etcmodel.fit(X_train,y_train)
pred = etcmodel.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test,pred)*100)
target_names = ['benign','malign']
```

Step 7.3: XGBoost Classifier

```
xgbc_model = XGBClassifier(n_estimators=6)
xgbcmodel = xgbc_model
xgbcmodel.fit(X_train,y_train)
pred = xgbcmodel.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test,pred)*100)
target_names = ['benign','malign']
```

Step 7.4: Stacking Classifier

```
classifier=[XGBClassifier(), RandomForestClassifier()]
stc_model = StackingClassifier(classifier, meta_classifier=ExtraTreesClassifier())
stcmodel = stc_model
stcmodel.fit(X_train,y_train)
pred = stcmodel.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test,pred)*100)
target_names = ['benign','malign']
```

Step 7.5: CNN Model

```
#cnn
model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(2, activation='sigmoid'))
model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

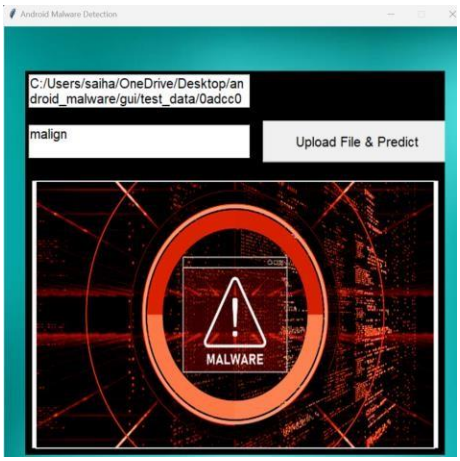
Step 8: Testing

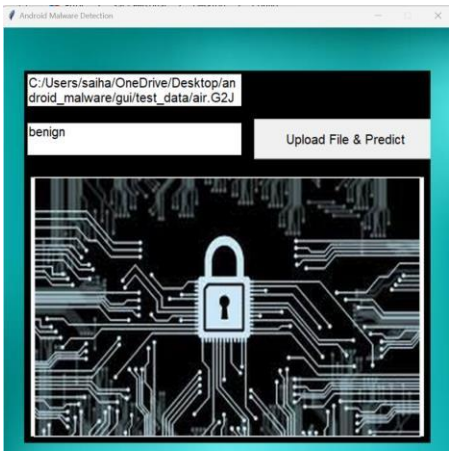
```
from androguard.core.bytecodes.apk import APK

def predictfile(apk):
  vector = {}
  a = APK(apk)
  perm = a.get_permissions()
  print(perm)
  for d in perms1:
    if d in perm:
      vector[d]=1
    else:
      vector[d]=0
  input = [ v for v in vector.values() ]
  print(input)
  ppred = sc1.predict([input])
  print(ppred)
  return ppred
```

This Python code analyzes APKs to predict requested permissions. It extracts permissions, creates a feature vector, and uses a pre-trained model to predict the most likely permission based on probability scores.

```
if(prediction == 1):
    textbox1.insert("end-1c",str('malign'))
else:
    textbox1.insert("end-1c",str('benign'))

if(prediction == 1):
        load = Image.open('malign.jpg')
        load = load.resize((600, 400), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(load)
        img = Label(image=render)
        img.image = render
        img.place(x=50, y=235)
else:
        load = Image.open('benign.jpg')
        load = load.resize((600, 400), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(load)
        img = Label(image=render)
        img.image = render
        img.place(x=50, y=235)
```

Step 9: Output





If the APK file is malicious, It is highlighted as “Malware Alert”
If the APK file is Benign, It is highlighted as "Benign”

References

1. Search UNB (no date) University of New Brunswick est.1785. Available at: https://www.unb.ca/cic/datasets/invesandmal2019.html (Accessed: 06 March 2024).