National College of Ireland

Evaluating Smart Contract Vulnerabilities through the Comparative Analysis of CNN, EfficientNet B2, and Xception Algorithms

MSc Research Project

Mohammed Shahimshah
Student ID: x21227012

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Mohammed Shahimshah

Student ID: x21227012

Programme: Msc in Cybersecurity **Year:** 2023 - 2024

Module: Msc Research Project

Supervisor: Diego Lugones

Submission Due Date: 31/01/2024

Project Title: Evaluating Smart Contract Vulnerabilities through the Comparative Analysis of CNN, EfficientNet B2, and Xception Algorithms

Word Count: ……1026…… **Page Count**……8……..……..

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: Mohammed Shahimshah

Date: 31 January, 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

Introduction

This handbook outlines the setup for the Vulnerability Detection Web App using Deep Learning, aimed at improving data privacy and security. It guides you through the essential software and hardware requirements, offering detailed instructions for a successful project setup. This resource is key for efficiently navigating the project's development stages and achieving our goal.

Hardware Requirements

Operating System: Windows 10/11
RAM: 20.0 GB
Processor: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
Storage: 256GB SSD
System Type: 64-bit operating system, x64-based processor

Software Requirements

This section details the software prerequisites for the development of our Vulnerability Detection Web Application using Deep Learning. The following list includes specific versions of libraries and frameworks essential for the project:

- **py-solc-x (2.0.2)**: This is a vital Python library for compiling Solidity contracts, crucial for blockchain-related functionalities in our project.
- **wheel**: A key package for building, distributing, and installing Python distributions - essential for package management.
- **pandas (2.1.4)**: Used for data manipulation and analysis, pandas are critical for handling large datasets efficiently.
- **numpy (1.25.2)**: A fundamental package for scientific computing, numpy supports large, multi-dimensional arrays and matrices.
- **pillow (10.2.0)**: This Python Imaging Library adds image processing capabilities, essential for handling visual data.
- **hexbytes (1.0.0)**: This library is used for encoding and decoding hexadecimal strings, an essential part of data handling.
- **pyevmasm (0.2.3)**: A Python module for assembling and disassembling EVM code, supporting blockchain development.
- **efficientnet (1.1.1)**: This library is used for implementing EfficientNet models, enhancing our deep learning capabilities.
- **keras (2.15.0)** & **tensorflow (2.15.0)**: These are core deep learning libraries, with Keras providing a high-level neural networks API and TensorFlow serving as the backend.
- Flask (3.0.0): Flask is employed to develop API services for the project, handling user requests and hosting the application locally.
- **opencv-contrib-python (4.9.0.80)** & **opencv-python (4.9.0.80)**: These libraries provide OpenCV's functionality, which is crucial for image processing tasks.
- **scikit-learn (1.3.2)**: An essential tool for machine learning for data mining and analysis.

Installing Dependencies

Installing the Python dependencies using the following command
pip install -r requirements.txt

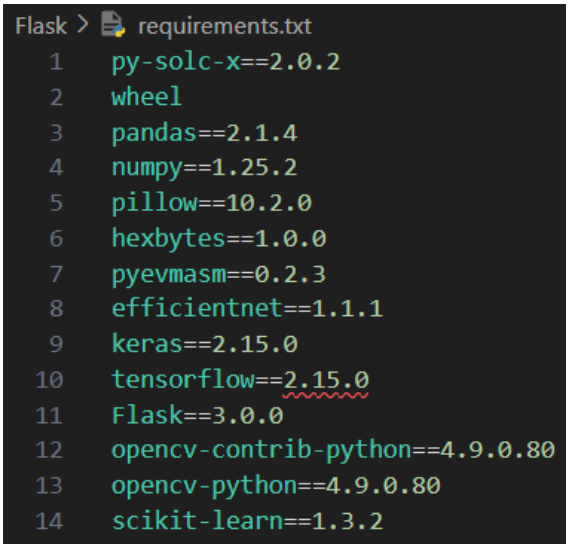

Figure 1: requirements.txt

The requirements.txt file is located in the path *Flask/requirements.txt.*

File Structure Overview

The artifacts for the Vulnerability Detection Web Application are organized in a structured manner for ease of navigation and use. Below is an overview of the key components:

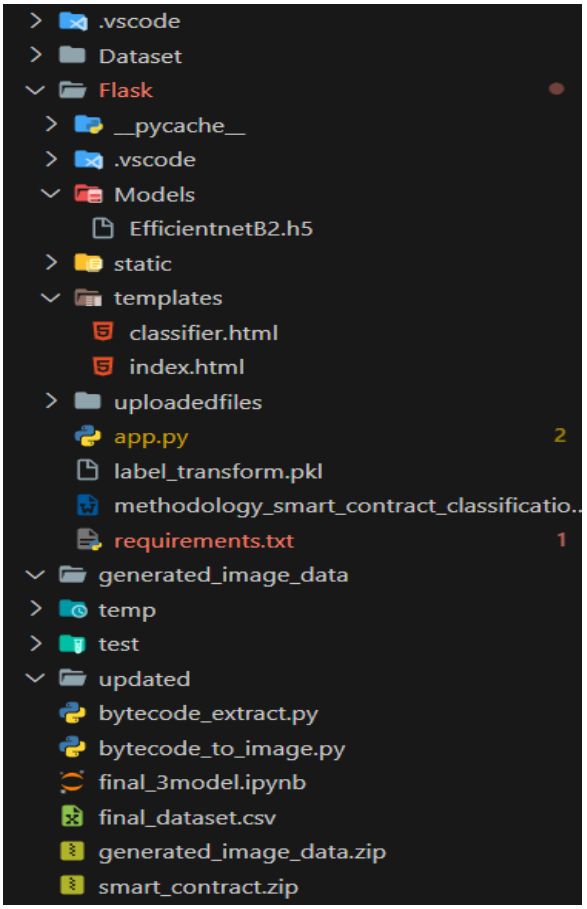

Figure 2 : Project Directory

- **Flask/Models/EfficientnetB2.h5**: The trained deep learning model, crucial for the application's predictive analysis.
- **Flask/templates**: Contains HTML templates for the web application's interface.
- **Flask/uploadedfiles**: Stores .sol files uploaded via the web application, serving as a data repository.
- **Flask/app.py**: The main file of the Flask Web Application, integrating all modules and functionalities.

- **Flask/label_transform.pkl**: A pickle file for label transformations, essential for data processing in the application.
- **Flask/requirements.txt**: Lists all Python dependencies required for the project, ensuring a consistent setup.
- **Flask/bytecode_extract.py**: Extracts bytecode from .sol files and saves it in final_dataset.csv, a critical step for dataset preparation.
- **Flask/bytecode_to_image.py**: Converts bytecode from final_dataset.csv into images, aiding in visual data analysis.
- **Flask/generated_image_data**: Stores image datasets generated by bytecode_to_image.py, used for model training and analysis.
- **Flask/final_3model.ipynb**: A Jupyter notebook for Deep Learning model analysis, containing code and insights for model evaluation.

Building the Dataset

https://github.com/Messi-Q/Smart-Contract-Dataset

Dataset Description

This dataset contains over 12K Ethereum smart contracts (where inherited contracts are also included) and concerns eight types of vulnerabilities.

Download this resource at Dataset.

ReadMe File of the Dataset

Ethereum smart contract dataset

We obtain our dataset by crawling Etherscan verified contracts, which are real-world smart contracts deployed on Ethereum Mainnet.

Our final dataset contains a total 12,515 smart contacts that have source code and concentrates on eight types of vulnerabilities, namely:

1. Timestamp dependency

2. Block number dependency

3. Dangerous delegatecall

4. Unchecked external call

5. Reentrancy

6. Integer overflow/underflow

7. Dangerous Ether strict equality

The ground truth labels (in the file `ground truth label`) of smart contracts in the dataset are confirmed based on defined vulnerability-specific patterns and further manual inspection.

Template for the web application

https://templatemo.com/tm-578-first-portfolio

Running the program

Open the terminal in the project folder.

Run the flask app ‘python3 app.py’. Note that the working directory should be ‘smart contract/Flask’.

```
2024-01-05 12:18:14.025362: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due
to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please
 use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Pleas
e use tf.compat.v1.executing_eagerly_outside_functions instead.

2024-01-05 12:18:20.023366: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions
 in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate comp
iler flags.
WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\layers\normalization\batch_normalization.py:979: The name tf.nn.fused_batch_norm is d
eprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2024-01-05 12:18:23.456225: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due
to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please
 use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Pleas
e use tf.compat.v1.executing_eagerly_outside_functions instead.

2024-01-05 12:18:26.580294: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions
 in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate comp
iler flags.
WARNING:tensorflow:From C:\Python311\Lib\site-packages\keras\src\layers\normalization\batch_normalization.py:979: The name tf.nn.fused_batch_norm is d
eprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

 * Debugger is active!
 * Debugger PIN: 108-442-364
```

Figure 3: Execution of python file app.py

Note the Local Server IP on which the app is running. Here it is ‘*127.0.0.1/5000’*.

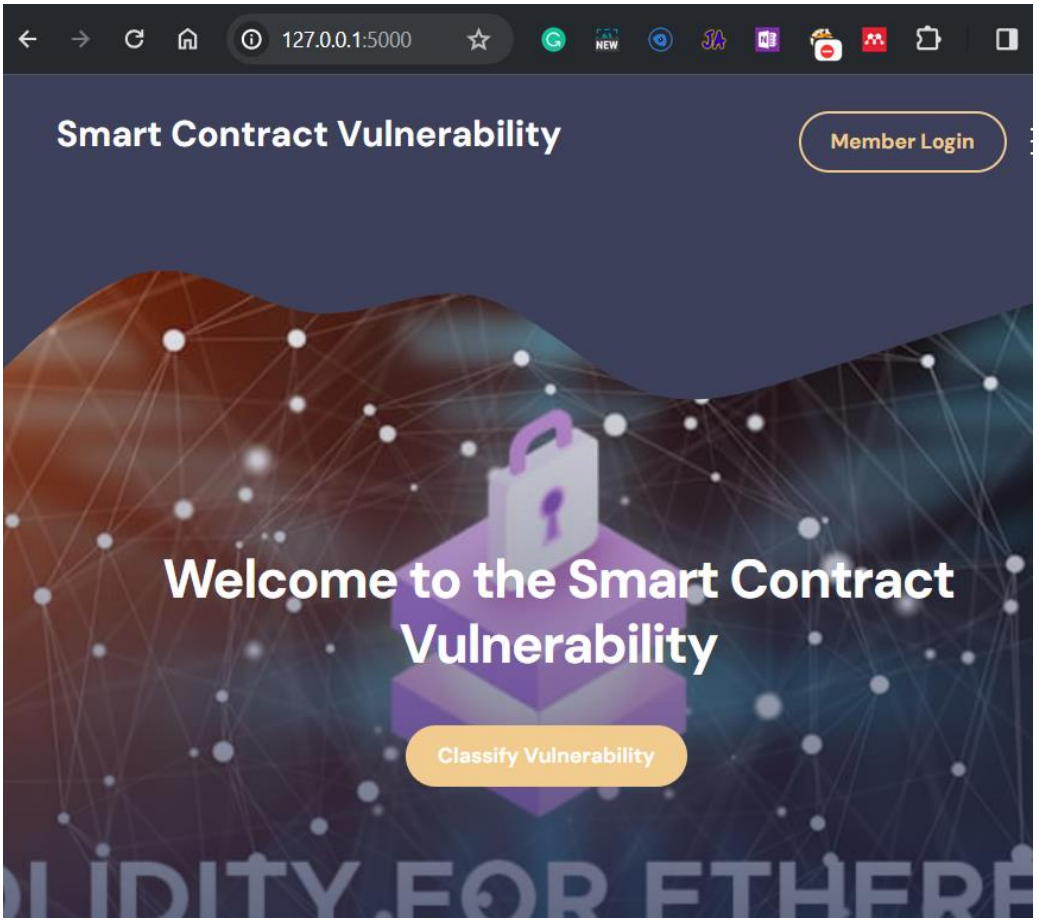

Figure 4: Home page for the smart contract vulnerability detection page