CS529 Security Analytics Fall 2023

Homework 2

Due: **October 26, 2023, 11:59 PM**

Descriptions & Instructions:

The goal of this homework is to help you become familiar with the implementation of the neural network models covered in the lectures and understand their performances in different security classification tasks. You are provided with the Jupyter Notebook, **HW2.ipynb,** and you are required to complete the coding corresponding to the following questions. After completing the code segments, please ensure that the entire code in the notebook executes without any errors. The notebook walks you through building a MLP model on a malware detection task, and building an RNN model on a binary function boundary detection task. It covers the two most widely used neural network models on two popular security datasets.

The dataset for the malware detection task is the Drebin dataset, and the dataset for the function boundary detection is from this paper. You can read this paper for further details. We have included the datasets in the released files of HW2. The dataset for assignment 1 is *drebin_data.npz*, and the dataset for assignment 2 is *elf_x86_32_gcc_O1_train.pkl* and *elf_x86_32_gcc_O1_test.pkl*. We also include a brief description of the two datasets within HW2.ipynb.

Submit format:

You will compress your Jupyter Notebooks source code into a **zip file** called HW2.zip and submit them on **Brightspace**.

Then you need to submit a report including your solutions to the coding problems. You can choose to convert your Jupyter Notebook to a pdf report or take screenshots of your code and results. The report should be submitted on **Gradescope**.

Please mark your solutions to each question correctly while submitting the report on Gradescope.

Failure to follow the instructions will lead to a deduction in points!

Assignment 1: Malware Detection using MLP and Drebin Dataset [50 pt]

Dataset: Drebin
Task: Train a Multi-layer Perceptron (MLP) to detect Android malware using the Drebin dataset.

To accomplish this assignment, please complete the following questions in HW2.ipynb:

1. Data Preparation:
    - Load the Drebin dataset.
    - Split the dataset into training and testing sets. Ensure that you have an 80-20 split (80% for training, 20% for testing). [5 pt.]
2. Model Design:
    - Design a MLP with the following architecture: [10 pt.]
        i. Input layer corresponding to the number of features in the Drebin dataset.
        ii. You may explore your own design of the middle layers.
        iii. Output layer with 7 neurons (it indicates the number of classes in Drebin dataset) and a softmax activation function.
    - Define loss, optimizer, batch size, number of training epochs and other hyper-parameters you may need for later training. [5 pt.]
3. Training: [25 pt.]
    - Train the MLP model using the training data. [10 pt.]
    - Use a batch size defined by yourself and train for several epochs. [5 pt.]
    - Utilize the test split of 0.2 during training to monitor the accuracy and loss on unseen data. [10 pt.]
    - ***We expect the testing accuracy to be higher than at least 90%***.
4. Using you final well-trained model to calculate and report the per-class (for each class (from 0 to 6) precision, recall, and F1 score of the testing dataset. [5 pt]
    - You need to submit a 3*7 table with the precision, recall, and F1 for each class.

Assignment 2: Recognizing Functions in Binaries using RNN [50 pt]

Dataset: Bao et al. [1]
Task: Detecting function boundaries in sequences of bytes that are extracted from binary files using RNN.

To accomplish this assignment, please complete the following questions in HW2.ipynb:

1. Preprocess the dataset: [10 pt.]
    - Ensure that the sequences are appropriately padded or truncated to a fixed length suitable for RNN training.
2. Model Design:
    - Design an RNN model to classify whether a given byte sequence represents the start of a function or not. Use a combination of embedding layers, RNN layers (SimpleRNN, LSTM, or GRU), and Dense layers as needed. You are free to explore your own design of model architecture. [10 pt.]
    - Define loss, optimizer, batch size, number of training epochs and other hyper-parameters you may need for later training. [5 pt.]
3. Training:
    - Train the RNN model using the training data. [10 pt.]
    - Use a batch size defined by yourself and train for several epochs. [5 pt.]
4. Evaluation:
    - Evaluate the performance of your model on a held-out test set using accuracy, precision, and recall. [10 pt.]
    - ***We expect the testing accuracy to be higher than at least 88%***.

[1] ByteWeight: Learning to Recognize Functions in Binary Code, USENIX 2014.