CSDS 493 Software Designation File

Haolai Che
hxc859@case.edu
Tiantian Pu
txp317@case.edu

03/29/2022

1 Development Theme

This Research focus on using LSTM(Long-Short Term Memory) Neural Network to achieve faster detection of new smart contract attack trends effectively.

2 BackGround Introduction

A smart contract is a computer protocol designed to disseminate, validate or enforce contracts in an informational manner. Smart contracts allow trusted transactions to be made without a third party, and these transactions are traceable and irreversible. Blockchain-based smart contracts are visible to all users on the blockchain. However, this can result in all vulnerabilities, including security bugs, being visible and potentially impossible to fix quickly. Ethereum, as the most popular platform for smart contracts holds the value of market capitalization up to \$21$billionUSdollars$, however, due to the nature of smart contracts being turing-complete and fully autonomous, the attacks on smart contracts can be extremely damaging as they are immutable on the blockchain. The notorious DAO attack in June 2016, caused over \$150$millionUSdollars$ losses. The detection of smart contracts vulnerabilities are urgently needed and has raised the world's attention.

3 Research Motivation

In 2017, Atzei et al analyzed the security loopholes in Ethereum smart contracts, and for the first time revealed that the security of the smart contract life cycle is closely related to the programming language level, virtual machine level and blockchain level individually, and different levels of characteristics will lead to different vulnerabilities. Ethereum smart contracts are deployed and run on the blockchain, and the non-tamperable feature of the blockchain is also inherited to the contract. This feature brings credibility to the contract, but also brings great security risks, and the contract, compared with traditional software storage, has greater value, so the vulnerability detection of smart contracts is particularly important. From the perspective of whether to run the program, the vulnerability detection technology can be divided into two categories: static detection and dynamic detection. For the detection of contract vulnerabilities, from whether to execute the contract classification, the contract vulnerability detection tools can also be divided into static detection tools and dynamic detection tools. At present, there are many detection tools, but most of them are difficult to use, and the detection efficiency is not high, and there are few vulnerabilities that can be detected. Most of the detection tools perform semi-dynamic detection based on the control flow graph of the contract, and do not take the time of multiple in-contract call into consider, resulting in low efficiency of vulnerability detection. Most of the existing detection tools are based on more traditional detection and analysis methods, which require expert knowledge to pre-define detection models, resulting in problems such as high limitations and poor scalability. Combining traditional detection methods with machine learning can improve the generality of the detection tools to a certain extent.

4 Designation Detail

4.1 Dataset Obtain Complie Automation Tool Implementation Based on web3.py MAIAN

In addition to using the data set from Prof.Xiao in the contract collection part, we also use the automatic download tool, Etherscan's api to obtain the contract source code on the chain. After registering in Etherscan, we can apply for free ApiKey for downloading the smart contract. Before obtaining the contract source code, you need to arrange the verified contract deployment address. Afte downloading contracts, save them in the data folder, and save the contract address in the contract_address file.

4.2 Dataset Collectioin

We used the Ethereum Dataset from Google BigQuery, we then parsed the EVM bytecodes into opcodes according to EVM instruction lists. By running our contracts through MAIAN tool, we collected the sequential opcodes, which are instructions found in the EVM list of execution code, as the input dataset.In our data set we have 892913 lines of smart contracts of different addresses, and we have categorized them into 5 levels of vulnerabilities which are suicidal contracts, prodigal contracts, greedy contracts, suicidal greedy contracts, last but not least, not-vulnerable contracts.

4.3 Data Preprocessing

The dataset we have is very imbalanced, amongst 892913 contracts there are only 8640 vulnerable contracts, which is 0.97% of the total number, this situation is not what we want for a valid training input. We have to resample the data from the not-vulnerable class set to create an equal number of synthetic vulnerable class, the oversampling technique we use here is SMOTE(Synthetic Minority Oversampling Technique). After oversampling towards vulnerable class and under-sampling towards not-vulnerable class, we spilt the dataset into train, validation and test set in proportion of 0.64, 0.16, 0.2.

4.4 LSTM Neural Network Model

Numerous tasks with sequential inputs and/or sequential outputs can be modeled with LSTM, which is a special type of RNN.The objective of our LSTM learning model is to perform a two-class classification, in order to detect if any given smart contract contains security threats. Motivated by the concepts in optimization, the objective in LSTM learning is to minimize the detection loss function, in order to maximize classification accuracy.
Our LSTM Model_V1 structure:
========Embedding Layer========.

========DropOut Layer 0.55========.

========LSTM Layer 128 dropout 0.2========.

========Dense64========.

========LSTM 64========.

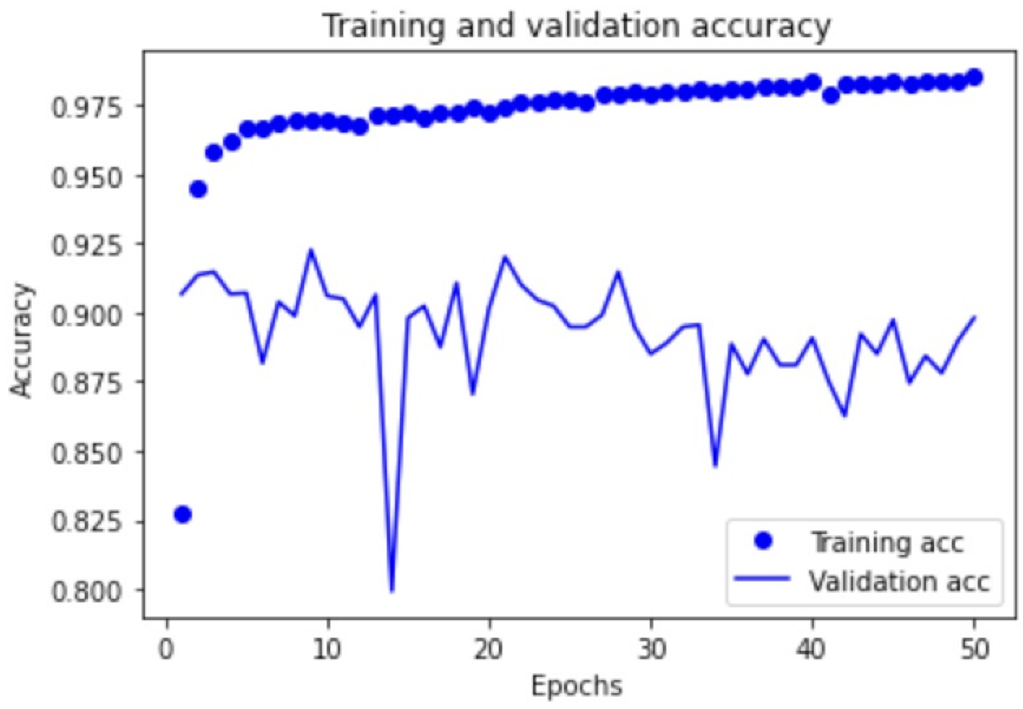
========Dense 2========.

========Optimizer adam========

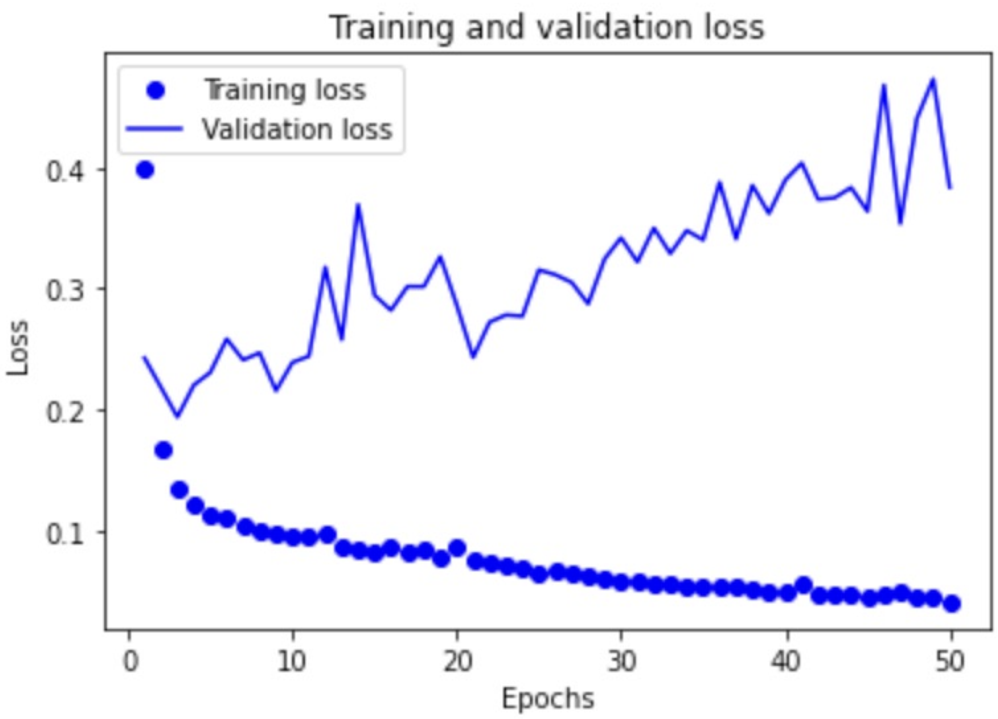
5 Preliminary results

We conduct code editing and tests locally with Jupyter lab, virtual environment version control with conda, model training with Google Colab. Our current preliminary data is as follows:

```
<matplotlib.legend.Legend at 0x7fa3f30c1fd0>
```



(a) Training and Validation Acc



(b) Training and Validation Loss

Figure 1: Current data