Smart Contracts Vulnerability Classification Through Deep Learning

Martina Rossini
University of Bologna
Bologna, Italy
martina.rossini3@studio.unibo.it

Mirko Zichichi
Universidad Politécnica de Madrid
Madrid, Spain
mirko.zichichi@upm.es

Stefano Ferretti
University of Urbino "Carlo Bo"
Urbino, Italy
stefano.ferretti@uniurb.it

ABSTRACT

We investigate the use of deep learning to classify smart contract code vulnerabilities. We use different variants of Convolutional Neural Networks (CNNs) and a Long Short-Term Memory (LSTM) neural network. Five classes of vulnerabilities were employed. Our results suggest that the CNNs are able to provide a good level of accuracy, thus showing the viability of the proposed approach.

ACM Reference Format:
Martina Rossini, Mirko Zichichi, and Stefano Ferretti. 2022. Smart Contracts Vulnerability Classification Through Deep Learning. In *Fourth ACM International Workshop on Blockchain-enabled Networked Sensor Systems (SenSys '22), November 6–9, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3560905.3568175

1 INTRODUCTION

Smart contracts have gained momentum in recent years thanks to their unique immutability and automatic enforceability features. They are used in various applications, ranging from decentralized finance up to traceability, service monitoring, decentralized social applications, and personal data distribution management [2, 10–12]. The most popular blockchain platform able to support smart contracts is Ethereum. Ethereum's smart contracts are written in Solidity, a Turing complete programming language. It is a powerful language, but, at the same time, it paves the way for bugs and code vulnerabilities.

This paper explores deep learning techniques for detecting and classifying vulnerabilities in smart contracts deployed on the Ethereum main net. Our study compares four different types of neural networks, i.e., a baseline Long Short-Term Memory (LSTM), a ResNet 1D Convolutional Neural Network (CNN), a 2D ResNet-18 CNN, and a 2D Inception v3 CNN.

Results show the viability of the proposal. Indeed, the ResNet 1D CNN, working directly on the smart contract byte code, seems to offer the best results in terms of classification capabilities.

This work has received funding from the EU H2020 research and innovation programme under the MSCA ITN grant agreement No 814177 LAST-JD-RIoE..

SenSys '22, November 6–9, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9886-2/22/11…$15.00
https://doi.org/10.1145/3560905.3568175

2 METHODOLOGY

We employed two available datasets of labeled smart contracts, i.e. SmartBugs [3] and ScrawlID [9]. Moreover, a list of verified smart contracts was retrieved from Smart Contract Sanctuary [7]. The final dataset comprises more than 100k smart contracts labeled using the Slither static analyzer.

This tool passes the code through several rule-based detectors and returns a JSON file containing details about where those detectors found a vulnerability. The 38 detectors that found a match in our dataset were then mapped to the following five classes: i) **Access-control**, i.e. if the visibility of some fields/functions is not correctly set to private, malicious users could have access to them; ii) **Arithmetic**, i.e. related to integer underflow and overflow errors; iii) **Reentrancy**, i.e. when a call to an external contract is allowed to make new calls to the calling contract before the initial execution is complete; iv) **Unchecked-calls** i.e. Solidity offers some low-level functions, which do not propagate errors and return false without ending the execution; v) **Others** i.e. a class that groups all other detectors. The dataset was separated into 79.6k, 10.8k, and 15.9k for training, validation, and test sets, respectively. It is important to note that a single contract may have more than one vulnerability. Table 1 shows the number of contracts our training set has per vulnerability class and the percentage of the contracts in that class that also have one or more other vulnerabilities. It is possible to notice how the classes are strongly unbalanced.

Table 1: Number of smart contracts per class (training set), along with the percentage of contracts of that class that have other vulnerabilities

Vulnerability	Contracts	Multi-label Contracts (%)
`unchecked-calls`	36353	72.32 %
`safe`	27036	00.00 %
`reentrancy`	24161	91.51 %
`other`	20993	84.17 %
`arithmetic`	13530	77.05 %
`access-control`	11704	87.27 %

Given the dataset, the smart contract bytecodes were transformed into RGB images following the procedure proposed in [5].

Then, smart contracts were classified using four deep neural networks architectures: two traditional 2D Convolutional Neural Networks (CNNs), namely ResNet-18 [4] and Inception v3 [8], both working on built RGB images; a 1D CNN applied directly to the contract bytecode, which was treated as a signal and normalized to be between -1 and 1 [6]; a baseline Long Short-Term Memory

(LSTM) model was again trained only on the sequences of opcodes in the contract bytecode.

3 RESULTS

We tried different training configurations for every model type. The ResNet-inspired 1D convolutional neural network achieved the best performances. The model was trained using the SGD optimizer with a learning rate set to 0.001, and an L2 penalty of 0.0001 applied only on convolutional and dense layer weights [1]. Finally, the optimized loss for this model was binary-cross-entropy, and in our case, we found the performance to be better without them.

Table 2 reports the best results we achieved on the validation set for every architecture. In this context, where every element can be in more than one class, and the class labels are not balanced, accuracy is not an ideal metric. Indeed, we also consider a micro-averaged version of the F1 score, which aggregates the contributions of all classes to compute the average metric and thus treats the examples of each class with equal weight.

Table 2: Models' results on the validation set

Model name	Accuracy	Micro F1
ResNet1D	0.7353	0.8381
ResNet	0.6841	0.7928
Inception	0.6988	0.8015
LSTM Baseline	0.6934	0.7953

We can immediately see that the LSTM baseline obtains poor results; this is probably because we cut the bytecode to just 512 opcodes due to the limited memory and computational resources. However, our data analysis shows that most bytecodes have a length of about 5000 opcodes. The portion we use probably corresponds to only a tiny first portion of the contract code.

2D CNNs (ResNet, Inception) do not require the input to be truncated in any way: we first create the images using all the bytecode and then resize them as needed. However, results suggest they do not represent an ideal choice in this application. Indeed, patterns for code vulnerability detection in Solidity may only be at the level of a small sequence of opcodes and thus may be missed when using stridden convolutions. Indeed, literature in malware classification shows that malicious code patterns in that domain are usually much more significant and easier to detect, even to the human eye [6].

Note that the ResNet 1D convolutional network again requires the input to be cut off. However, the nature of the network lets us use a larger maximum length of 16384 (corresponding to a flattened 128x128 image). As shown in the table above, this architecture is the one that achieves the best results.

We show in Table 3 the confusion matrices relative to the performance of our best model (i.e., ResNet1D) on the test set. They show that the class unchecked-calls have very few mis-classified examples, while the percentage of errors significantly increases when considering the other classes. This was predictable because unchecked-calls is a vulnerability in more than 35000 of the original train contracts, thus making it the majority class. Among the other classes, we notice that the two with fewer training examples (access-control and arithmetic) are where our classifier makes the most errors. Finally, classes others and reentrancy have more or less the same number of samples in the training set. However, the first one is mis-classified a lot more: this is probably due to the inherent nature of this class, which groups all the interesting vulnerabilities that are not part of the other four classes. This variety may indeed generate some confusion for the detector.

Table 3: Per-class confusion matrices obtained by ResNet1D on the test set

		Predicted Class									
		Access Control		Arithmetic		Reentrancy		Unchecked-calls		Other	
Actual Class		Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
	Yes	0.71	0.29	0.72	0.28	0.81	0.19	0.90	0.10	0.77	0.23
	No	0.02	0.98	0.04	0.96	0.04	0.96	0.05	0.95	0.07	0.93

4 CONCLUSIONS

This paper showed that deep learning techniques could be a viable tool for identifying and classifying smart contract code vulnerabilities. Based on the dataset we built, we have trained and tested four different neural network architectures. Using the aforementioned dataset, we thus approach the problem of vulnerability classification as a multi-label classification problem. Results show that, according to our configuration based on the available computational capabilities, the 1D ResNet CNN working on the smart contract bytecodes can provide the best results in terms of accuracy and Micro F1.

REFERENCES

[1] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-Performance Large-Scale Image Recognition Without Normalization. https://doi.org/10.48550/ARXIV.2102.06171

[2] Gabriele D'Angelo, Stefano Ferretti, and Moreno Marzolla. 2018. A Blockchain-Based Flight Data Recorder for Cloud Accountability. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems* (Munich, Germany) *(CryBlock'18)*. Association for Computing Machinery, New York, NY, USA, 93–98. https://doi.org/10.1145/3211933.3211950

[3] Thomas Durieux, João F Ferreira, Rui Abreu, and Pedro Cruz. 2020. Empirical review of automated analysis tools on 47,587 Ethereum smart contracts. In *Proceedings of the ACM/IEEE 42nd International conference on software engineering*. 530–541.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[5] TonTon Hsien-De Huang. 2018. Hunting the ethereum smart contract: Color-inspired inspection of potential attacks. *arXiv preprint arXiv:1807.01868* (2018).

[6] Seon-Jin Hwang, Seok-Hwan Choi, Jinmyeong Shin, and Yoon-Ho Choi. 2022. CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. *IEEE Access* 10 (2022), 32595–32607.

[7] Martin Ortner and Shayan Eskandari. 2022. Smart Contract Sanctuary. https://github.com/tintinweb/smart-contract-sanctuary

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567 [cs.CV]

[9] Chavhan Sujeet Yashavant, Saurabh Kumar, and Amey Karkare. 2022. ScrawlD: A Dataset of Real World Ethereum Smart Contracts Labelled with Vulnerabilities. *arXiv preprint arXiv:2202.11409* (2022).

[10] Mirko Zichichi, Michele Contu, Stefano Ferretti, and Gabriele D'Angelo. 2019. LikeStarter: a Smart-contract based Social DAO for Crowdfunding. In *Proc. of the 2nd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*.

[11] M. Zichichi, S. Ferretti, and G. D'Angelo. 2020. A Distributed Ledger Based Infrastructure for Smart Transportation System and Social Good. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*. 1–6.

[12] Mirko Zichichi, Stefano Ferretti, and Gabriele D'angelo. 2020. A Framework Based on Distributed Ledger Technologies for Data Management and Services in Intelligent Transportation Systems. *IEEE Access* 8 (2020), 100384–100402. https://doi.org/10.1109/ACCESS.2020.2998012