Security Verification of Ethereum Smart Contracts with ML Taking a Free Ride on Static Analysis

Dalila Ressi
Carla Piazza
University of Udine
Italy

Sabina Rossi
Michele Bugliesi
Ca'Foscari University of Venice
Italy

Lorenzo Benetollo
Ca'Foscari University of Venice
University of Camerino
Italy

Silvia Crafa
University of Padova
Italy

Machine Learning for Blockchain

This proposal is the result of our survey about how ML/AI can enhance blockchain technology

Many application areas:

- Healthcare
- IoT/IoV/Smart Cities
- DeFi
- Cryptocurrency

Areas improved through AI:

- Security
- Consensus Algorithm
- Auction and Smart Grids Optimization
- Smart Contracts

Vulnerabilities in Ethereum Smart Contracts

Denial of Service
"tx.origin" usage
Integer Overflow/Underflow
Re-entrancy
Call to the unknown
Gassless "send"
…

→ Solidity Programming Language

Immutable bugs/mistakes
Ether lost in transfer
…

→ Ethereum Virtual Machine

Timestamp dependency
Transaction Ordering Dependency
…

→ Ethereum Blockchain Design

Vulnerabilities in Ethereum Smart Contracts

Denial of Service
“tx.origin” usage
Integer Overflow/Underflow
Re-entrancy
Call to the unknown
Gassless “send”
…

→ Solidity Programming Language

Immutable bugs/mistakes
Ether lost in transfer
…

→ Ethereum Virtual Machine

Timestamp dependency
Transaction Ordering Dependency
…

→ Ethereum Blockchain Design

⇨ More than 20 types of different vulnerabilities, we focus only on detection

Vulnerability Detectors: Formal Verification Techniques

Most of existing frameworks exploit static analysis, some examples are:

- Slither
- Mythril
- Oyente
- Securify
- SmartCheck
- SmartScan
- …

Usually specialized on specific security properties,
running multiple frameworks is computationally expensive

Vulnerability Detectors: ML Techniques

ML learning techniques take advantage of existing frameworks for dataset labeling, and then they simply perform classification with techniques such as:

- SVM
- **Boosting**
- Random Forest
- Decision Tree
- CNN
- GNN
- …

Machine Learning techniques claim to have higher performance than static analyzers with respect to <u>inference time</u> and <u>accuracy</u>

Limitations and Open Problems

FORMAL VERIFICATION

Lack of soundness guarantees

Restricted number of vulnerabilities

Different datasets:

Number/type of vulnerabilities

Number of contracts

Source code/Opcode/Bytecode

MACHINE LEARNING

Scalability

Easily deprecated

Based on possible mislabelling

Hard to compare detectors

Dataset Creation: Getting to the Root of the Problem

The availability of **benchmark dataset** would not only provide a solid base to develop new detection algorithms, but it would also allow us to evaluate and compare existing ones.

Key features:

- Large number of smart contracts
- Include all three representations of a contract (source code, opcode, bytecode)

Labeling process:

- Exploit only formal verification techniques providing soundness guarantees
- Represent as many as possible vulnerabilities (eventually augmenting the cardinality of heavily underrepresented classes)

Thank you