SmartCheck: Static Analysis of Ethereum Smart Contracts

Sergei Tikhomirov, Ekaterina Voskresenskaya,
Ivan Ivanitskiy, Ramil Takhaviev, Evgeny Marchenko,
Yaroslav Alexandrov

SnT, University of Luxembourg / SmartDec

27 May 2018
WETSEB, Gothenburg, Sweden

Outline

Outline

Goal: finding bugs that can cost you millions

- Smart contracts: a decentralized way to enforce digital agreements
- Ethereum: a blockchain-based Turing complete application platform
- Bugs can be (and have been) exploited: hundreds million USD lost
- We present SmartCheck – a static analyzer for Ethereum contracts

What Ethereum nodes do

- Store account balances, contract code and variables
- Execute smart contracts code on request
- Maintain a shared view of the global state

Ethereum security challenges

- Decentralized execution environment
- New software stack
- Very limited ability to patch contracts
- Anonymous financially motivated attackers
- Rapid pace of development
- Suboptimal high-level language

Outline

Classification of issues in Solidity code

- **Security**: directly lead to exploits
- **Functional**: violate the intended functionality[1]
- **Operational**: lead to run-time problems
- **Developmental**: make code hard to improve

[1] Though without a specification we only assume what the intended functionality is.

Typical issues in Solidity code

Let us focus on three examples of code issues:

- Re-entrancy (security)
- Locked money (functional)
- Costly loop (operational)

SmartCheck detects 21 types of issues.

Example 1/3: Re-entrancy

- Contract maintains internal list of balances
- If a user withdraws funds; their balance is set to zero
- Adversary requests withdrawal via malicious contract which calls the victim back before their balance is set to zero, depleting the victim contract's balance
- Real-world case: The DAO hack (June 2016): $50m lost

```
pragma solidity 0.4.19;
contract Fund {
  mapping(address => uint) balances;
  function withdraw() public {
    if (msg.sender.call.value(balances[msg.sender])())
    balances[msg.sender] = 0;
  }
}
```

Example 2/3: Locked money

- Contracts that receive ether should have a way to withdraw it: call `transfer`, `send`, or `call.value`
- Otherwise money is be stuck in contract forever

Example 3/3: Costly loop

- Ethereum users pay for contract execution with *gas*
- Tx's are atomic: if one step fails, whole tx fails
- Miners enforce a *block gas limit*
  (hence, a limit on computation in one tx)
- A costly function called inside a long enough loop exceeds block gas limit: tx is never confirmed
- Example: payouts for all winners in a game fail because of one (maliciously) failing payout

```
for (uint256 i = 0; i < array.length; i++) { costlyF(); }
```

Outline

Code analysis: dynamic vs static

Dynamic code analysis:

- black box
- no false positives
- some code execution paths are missed

Static code analysis:

- white box
- some false positives
- all the code is analyzed

Smart contracts code analysis

Smart contracts code compared to web applications code:

- immutable
- self-bug-bounty
- all the code is crucial

but

- less code (∼1,000 LOC vs ∼100,000 LOC)

Thus, **static** analysis is our choice.

Static code analysis

Static analysis usually includes three stages:

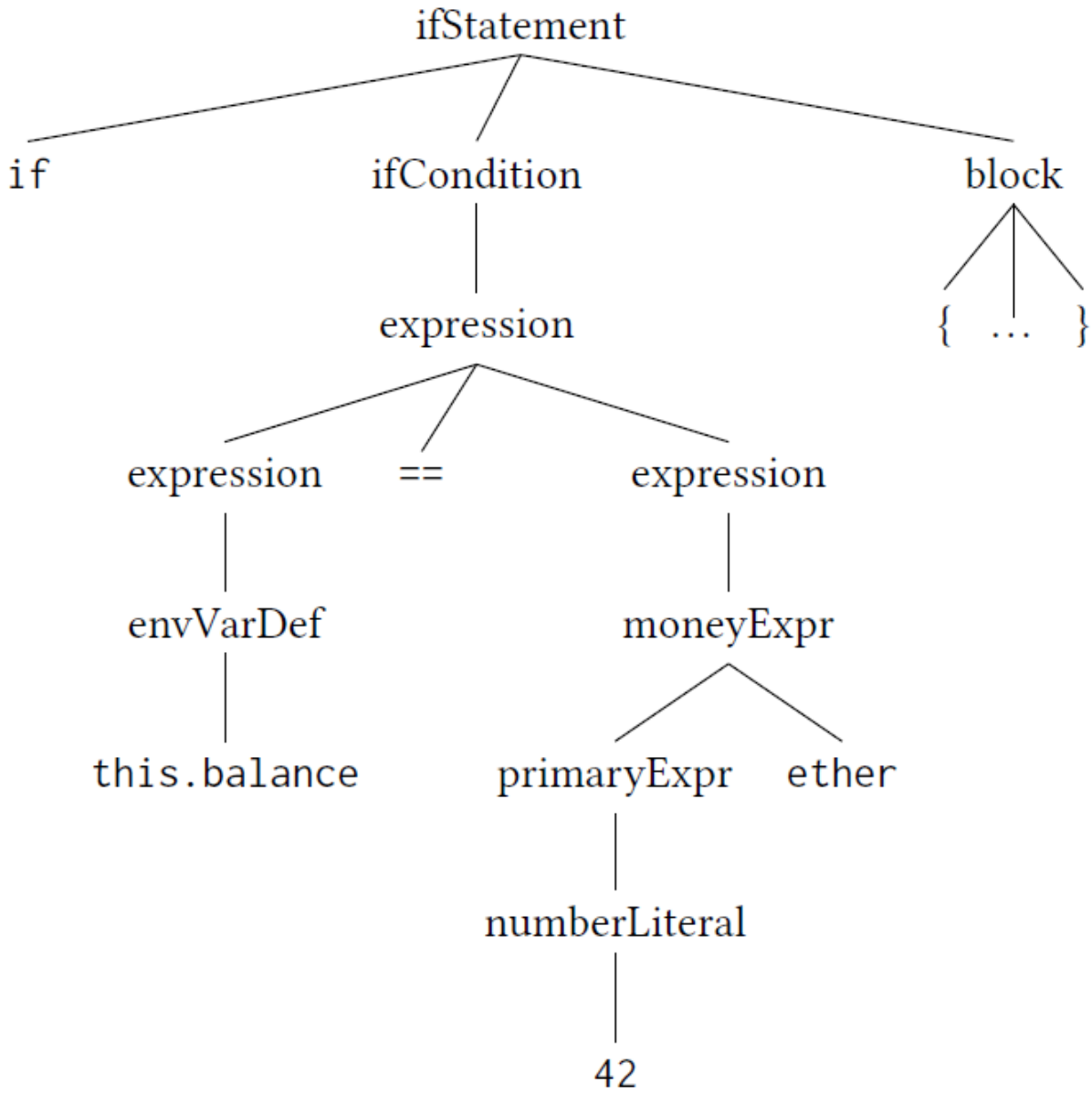
1. building an intermediate representation (IR)
2. enriching the IR with additional information
3. vulnerability detection

SmartCheck: static code analyser

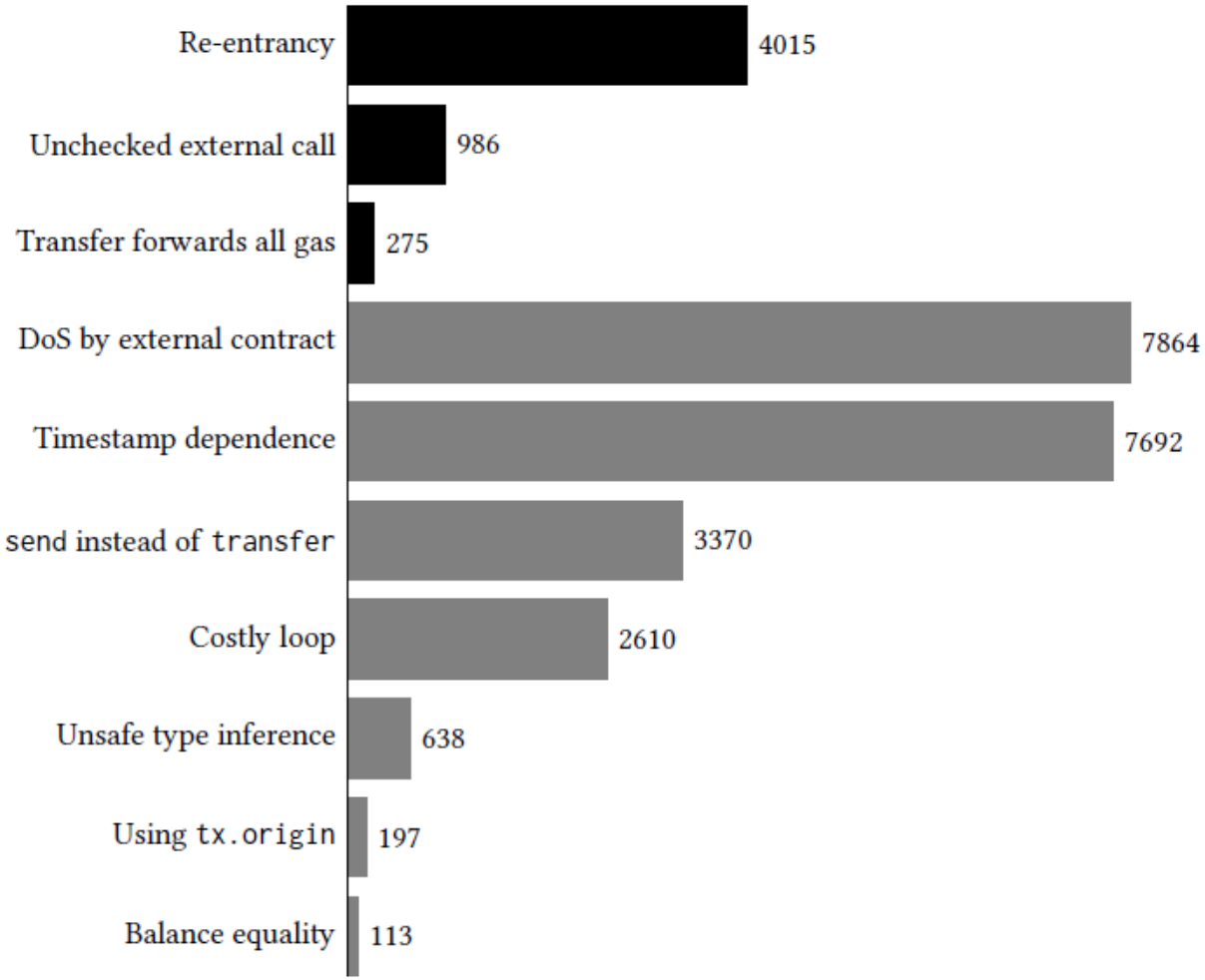
SmartCheck uses:

- ANTLR parser generator
- custom Solidity grammar
- XPath queries

Example parse tree



Vulnerabilities in 4,600 real contracts



Outline

Future work

- Improve the grammar
- Make patterns more precise
- Add new patterns
- Implement more sophisticated static analysis methods
- Add support for other languages

Current state of the project

- First version is open-sourced (GPL-3.0): `github.com/smartdec/smartcheck`
- Improved version is freely available as a service: `tool.smartdec.net`
- Currently 100 scans per day, 4212 scans in total

Questions?

- github.com/smartdec/smartcheck
- tool.smartdec.net
- cryptolux.org
- s-tikhomirov.github.io