Frontrunning in Ethereum

•••

Evaluation of vulnerability detection tools

Supervisor: Ass.Prof.in Dipl.-Ing.in Mag.a rer.soc.oec. Dr.in techn. Monika di Angelo

What do these tools have in common?

- Conkas
- Ethracer
- Mythril
- Oyente
- Securify

What do these tools have in common?

- Conkas
- Ethracer
- Mythril
- Oyente
- Securify

1. They can detect frontrunning vulnerabilities in programs

What do these tools have in common?

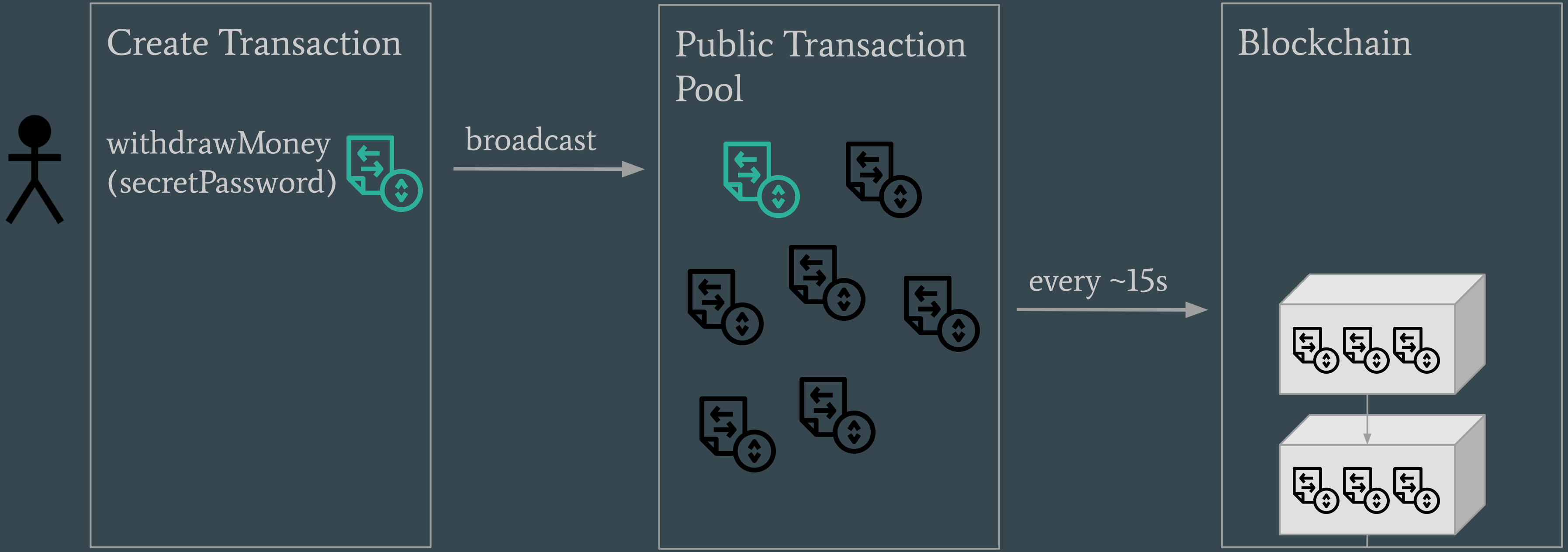
- Conkas
- Ethracer
- Mythril
- Oyente
- Securify

1. They can detect frontrunning vulnerabilities in programs
2. They all miss > 94% of vulnerable programs[1]

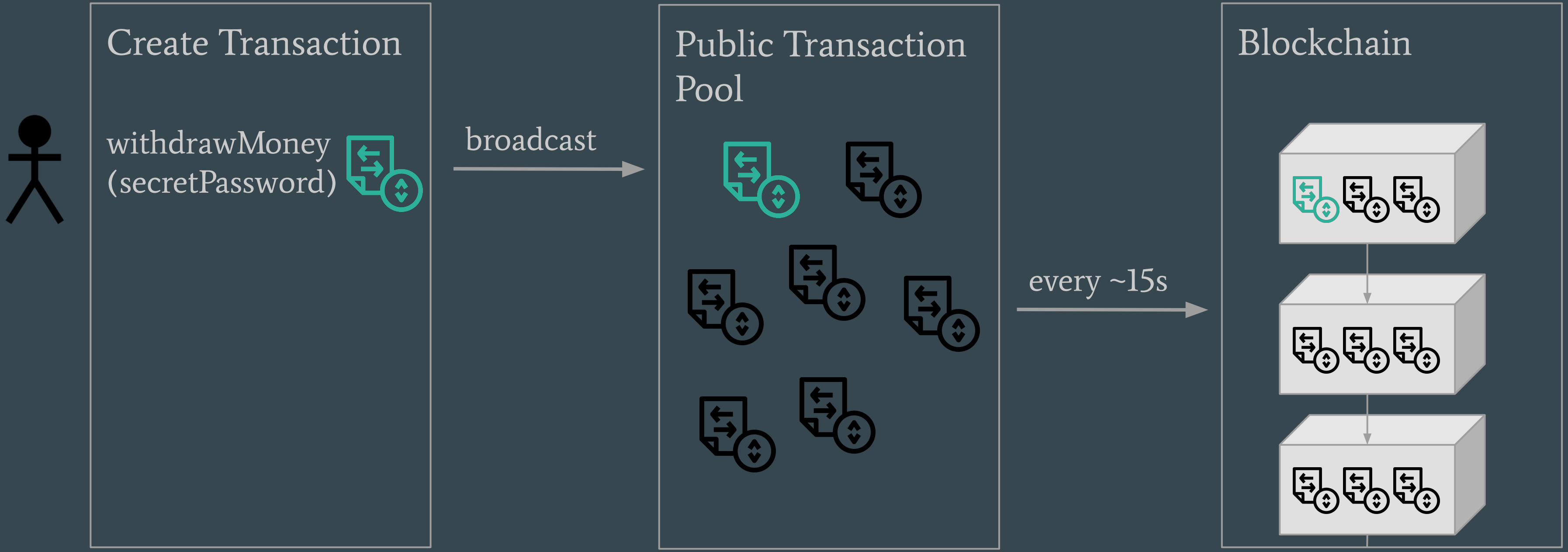
[1] Zhang et al. (2023) Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation.

Why?

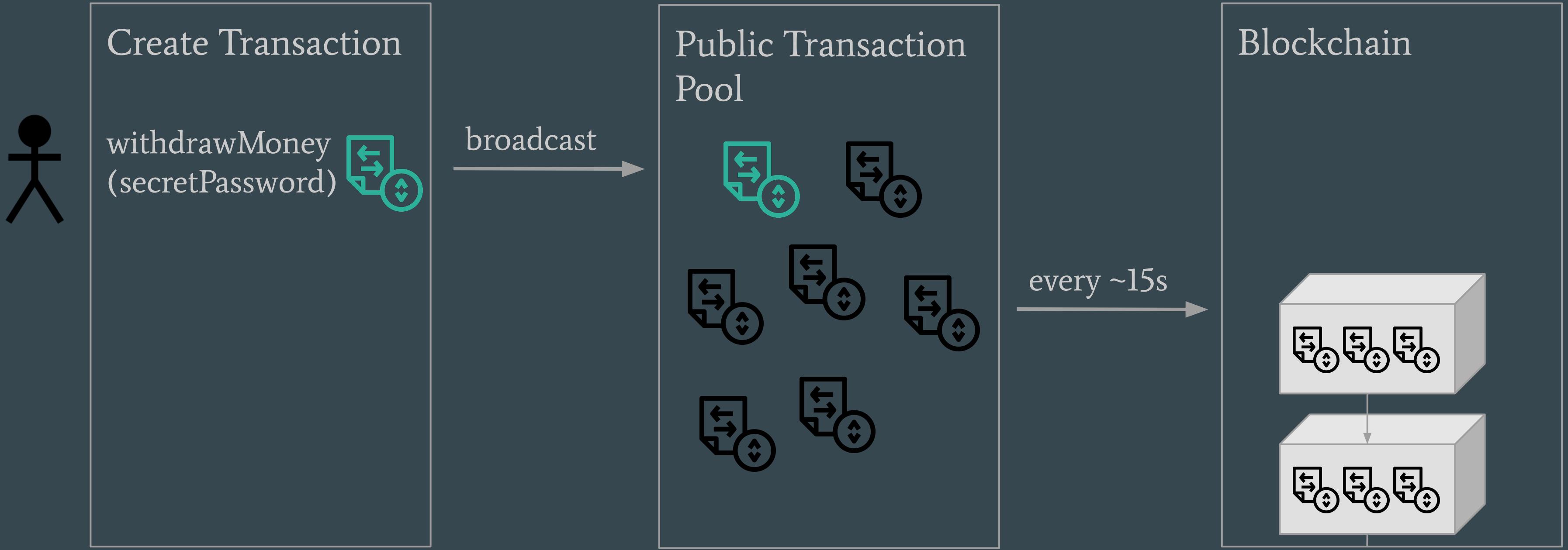
Ethereum Transactions



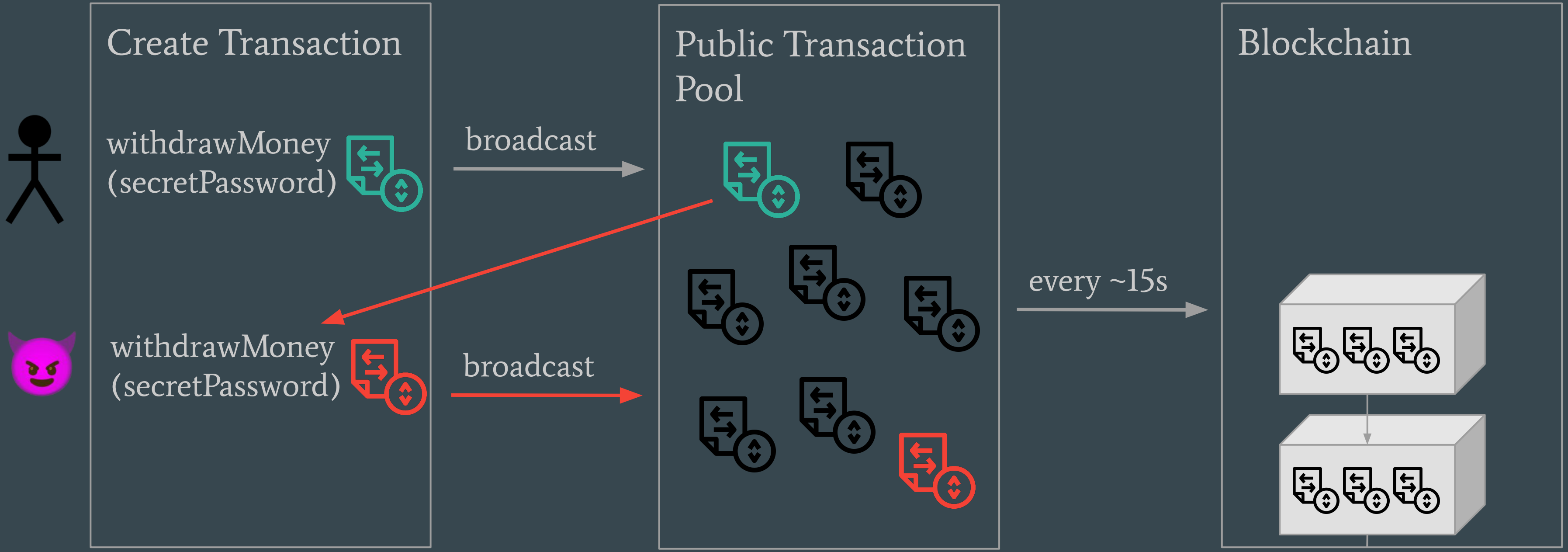
Ethereum Transactions



Frontrunning



Frontrunning



Create Transaction

withdrawMoney (secretPassword)

withdrawMoney (secretPassword)

broadcast

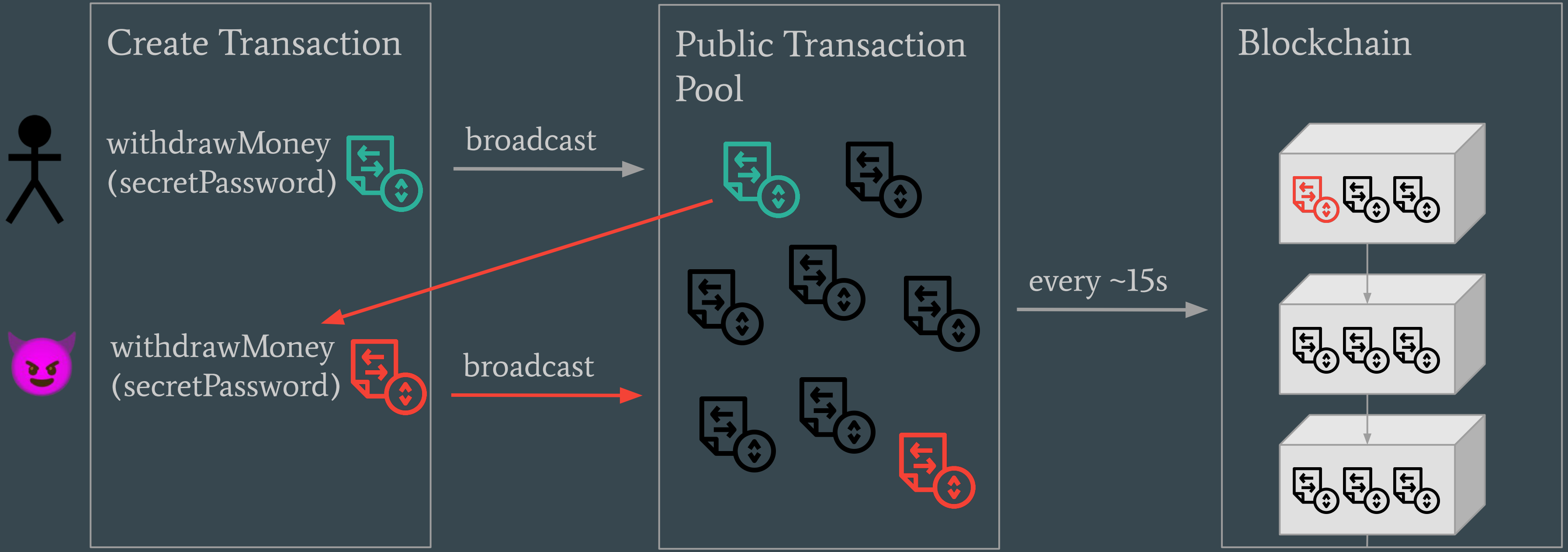
broadcast

Public Transaction Pool

every ~15s

Blockchain

Frontrunning



Create Transaction

withdrawMoney (secretPassword)

withdrawMoney (secretPassword)

broadcast

broadcast

Public Transaction Pool

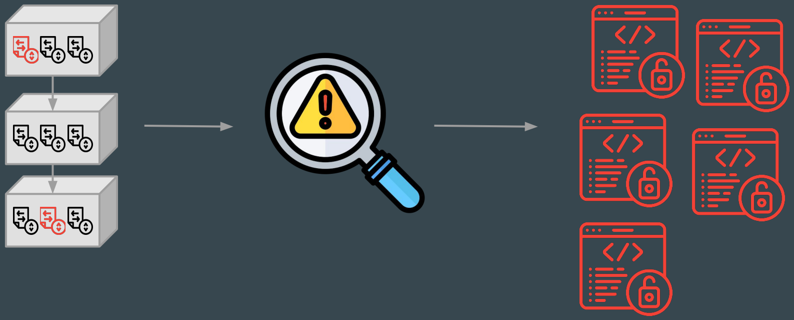
every ~15s

Blockchain

State of the Art

Why are frontrunning vulnerability detectors so bad?

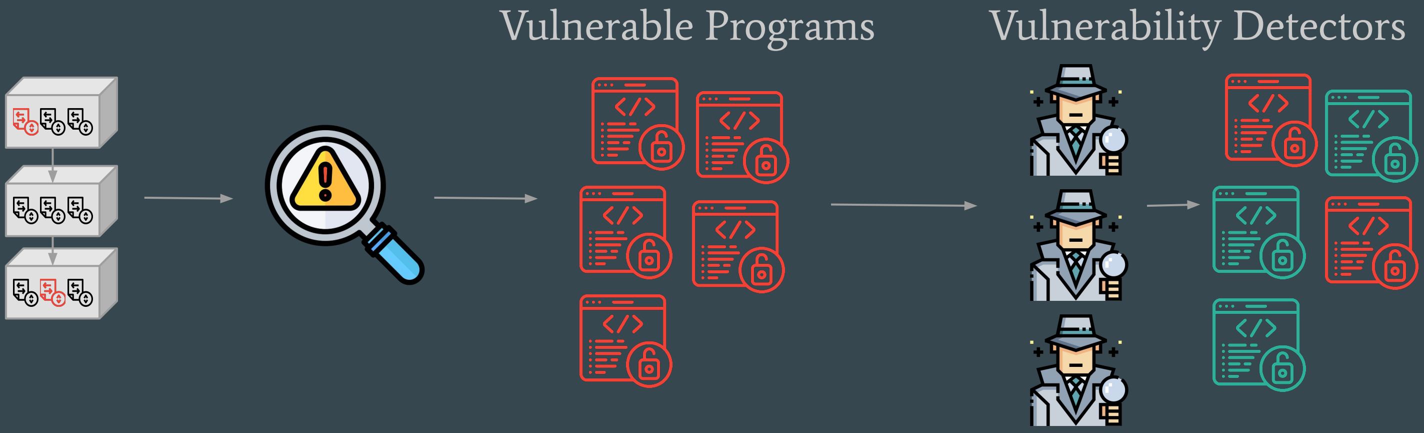
State of the Art



Vulnerable Programs

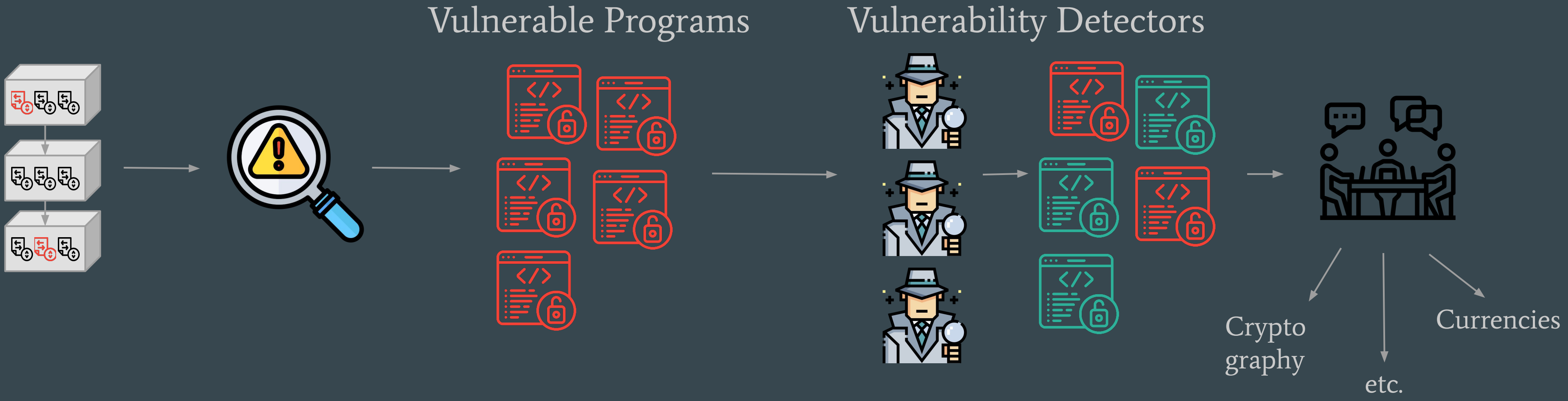
Zhang et al. (2023) Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation.

State of the Art



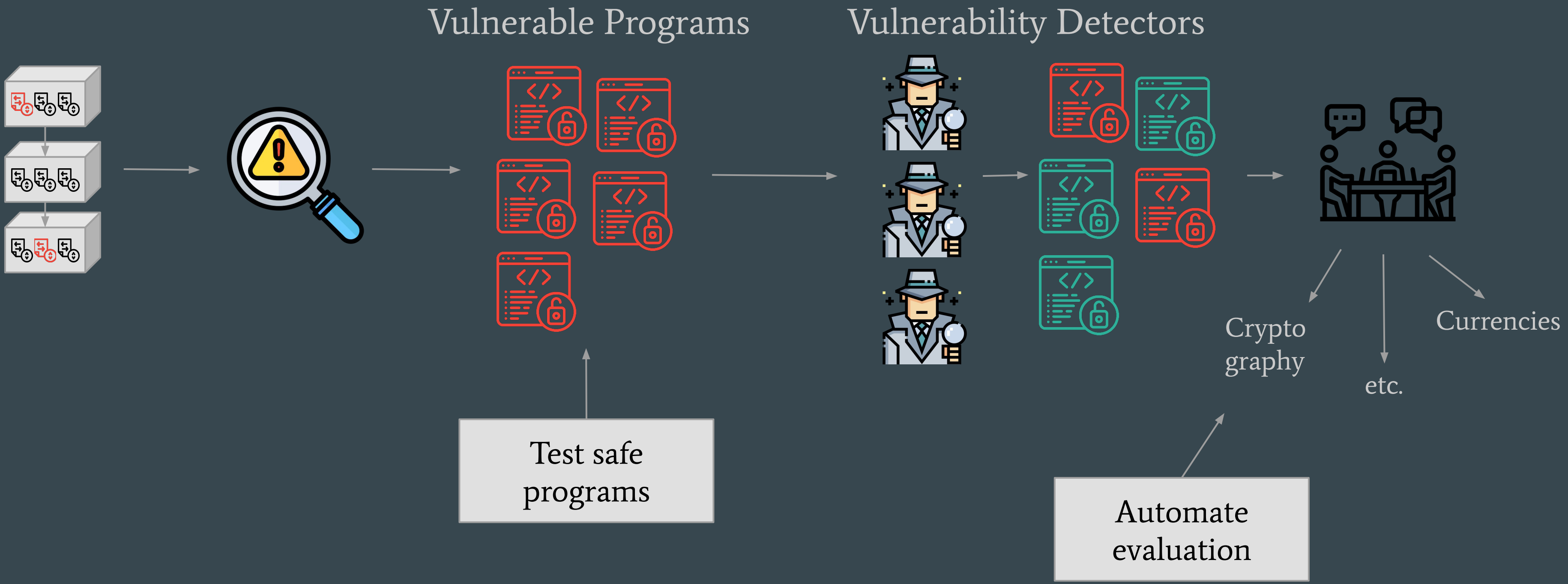
Zhang et al. (2023) Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation.

State of the Art



Zhang et al. (2023) Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation.

State of the Art



Vulnerable Programs

Vulnerability Detectors

Test safe programs

Automate evaluation

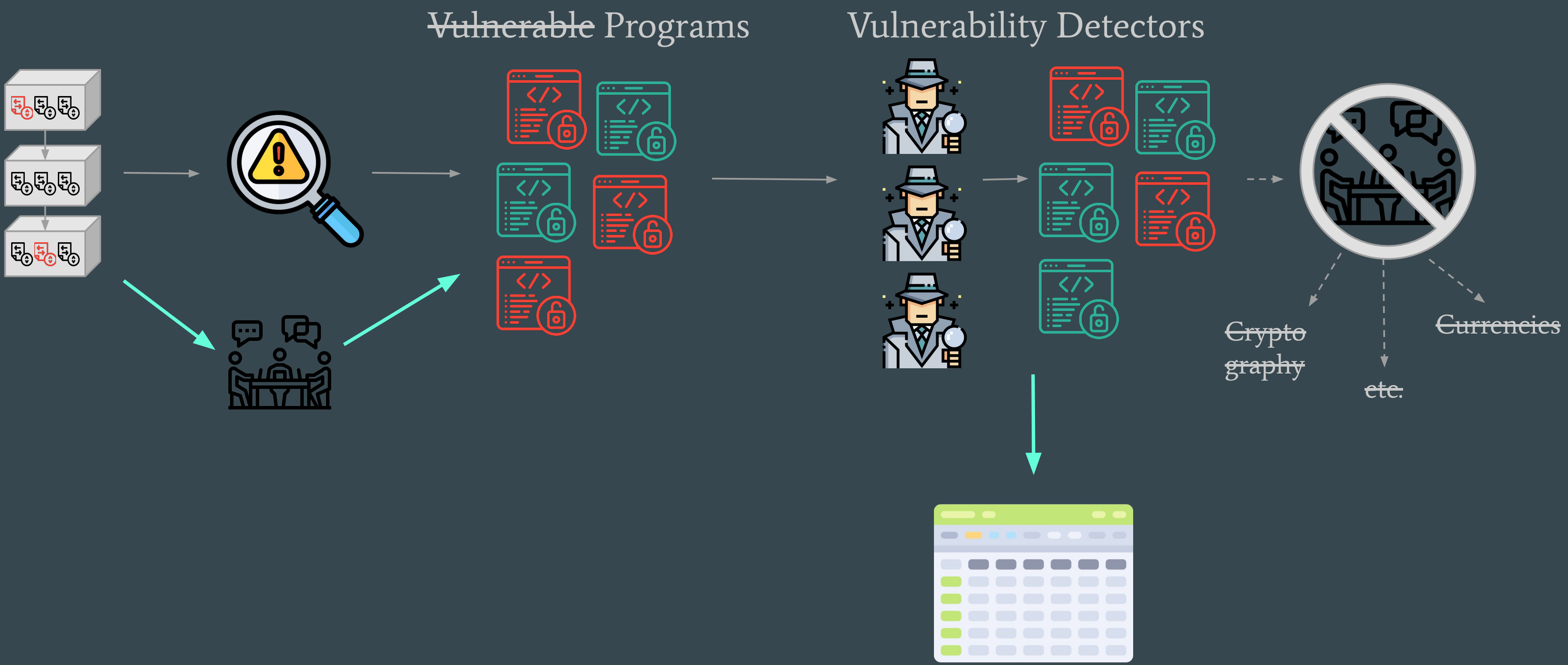
Crypto graphy

etc.

Currencies

Zhang et al. (2023) Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation.

Add secure programs



~~Vulnerable~~ Programs

Vulnerability Detectors

~~Crypto graphy~~

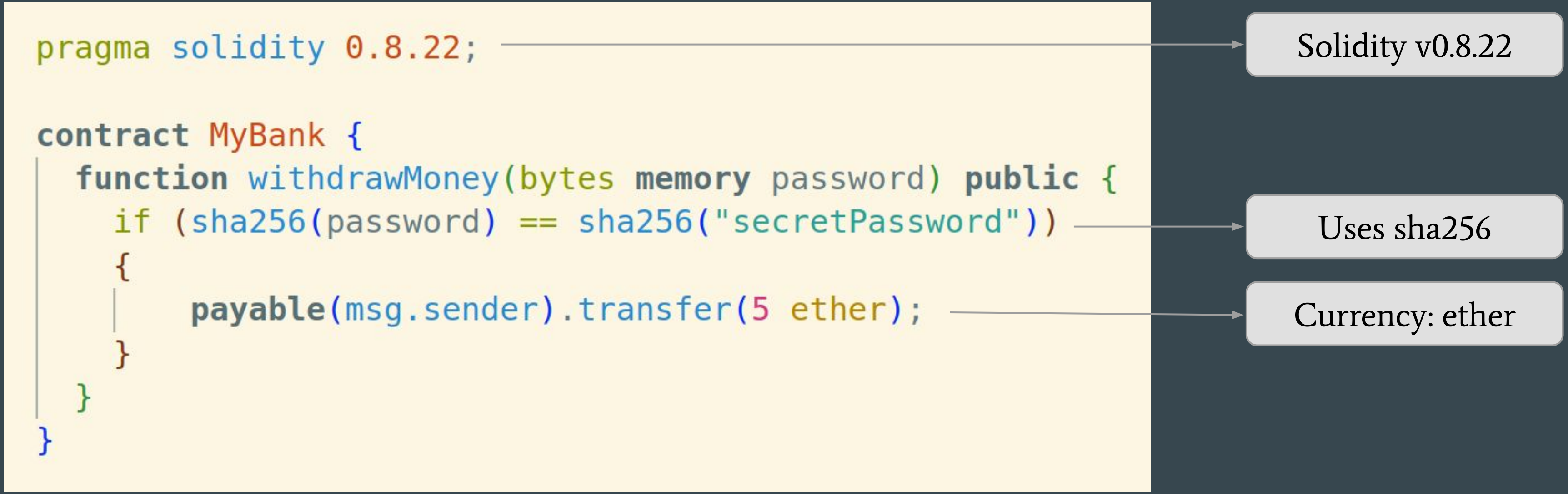
~~etc.~~

~~Currencies~~

Evaluation Example

Tool \ Secure?	Vulnerable Programs	Secure Programs
Conkas	20%	15%
Oyente	18%	8%
Securify	0%	20%

Label the programs!



```
pragma solidity 0.8.22;

contract MyBank {
  function withdrawMoney(bytes memory password) public {
    if (sha256(password) == sha256("secretPassword"))
    {
        payable(msg.sender).transfer(5 ether);
    }
  }
}
```

Solidity v0.8.22

Uses sha256

Currency: ether

Evaluation Example

Tool \ Label	Uses sha256	Currency: ether	Currency: Token
Conkas	0%	15%	0%
Oyente	18%	8%	14%
Securify	0%	20%	12%

Contributions

- Verification of previous results
- Detailed understanding of causes for missed vulnerabilities
- Dataset for reproducible & automatic tool evaluation
- First analysis of false positives

References & Credits

[1] Wuqi Zhang, Lili Wei, Shing-Chi Cheung, Yepang Liu, Shuqing Li, Lu Liu, and Michael R. Lyu. Combatting Front-Running in Smart Contracts: Attack Mining, Benchmark Construction and Vulnerability Detector Evaluation. In IEEE Transactions on Software Engineering, volume 49, pages 3630–3646, 2023.

Icons from Flaticon:

- Transaction
- Investigation
- Source Code
- Investigator
- Coworking
- Stats Table
- Check
- Nope Shield