VeriSmart:
A Highly Precise Safety Verifier for Ethereum Smart Contracts

Sunbeom So Myungho Lee Jisu Park Heejo Lee Hakjoo Oh

Korea University

May 20, 2020 @ IEEE S&P

Smart Contract

- Digital contract written in programming languages.

```
1  contract Netkoin {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor (uint initialSupply) {
6      totalSupply = initialSupply;
7      balance[msg.sender] = totalSupply;
8    }
9
10   function transfer (address to, uint value) public
11   returns (bool) {
12     require (balance[msg.sender] >= value);
13     balance[msg.sender] -= value;
14     balance[to] += value;
15     return true;
16   }
17
18   function burn(uint value) public returns (bool) {
19     require (balance[msg.sender] >= value);
20     balance[msg.sender] -= value;
21     totalSupply -= value;
22     return true;
23   }
24 }
```

Solidity Contract

Smart Contract

- Digital contract written in programming languages.

```
1  contract Netkoin {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor (uint initialSupply) {
6      totalSupply = initialSupply;
7      balance[msg.sender] = totalSupply;
8    }
9
10   function transfer (address to, uint value) public
11   returns (bool) {
12     require (balance[msg.sender] >= value);
13     balance[msg.sender] -= value;
14     balance[to] += value;
15     return true;
16   }
17
18   function burn(uint value) public returns (bool) {
19     require (balance[msg.sender] >= value);
20     balance[msg.sender] -= value;
21     totalSupply -= value;
22     return true;
23   }
24 }
```

Global state variables

Solidity Contract

Smart Contract

- Digital contract written in programming languages.

```
1  contract Netkoin {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor (uint initialSupply) {
6      totalSupply = initialSupply;
7      balance[msg.sender] = totalSupply;
8    }
9
10   function transfer (address to, uint value) public
11   returns (bool) {
12     require (balance[msg.sender] >= value);
13     balance[msg.sender] -= value;
14     balance[to] += value;
15     return true;
16   }
17
18   function burn(uint value) public returns (bool) {
19     require (balance[msg.sender] >= value);
20     balance[msg.sender] -= value;
21     totalSupply -= value;
22     return true;
23   }
24 }
```

Global state variables

Constructor

Solidity Contract

Smart Contract

- Digital contract written in programming languages.

```
1  contract Netkoin {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor (uint initialSupply) {
6      totalSupply = initialSupply;
7      balance[msg.sender] = totalSupply;
8    }
9
10   function transfer (address to, uint value) public
11   returns (bool) {
12     require (balance[msg.sender] >= value);
13     balance[msg.sender] -= value;
14     balance[to] += value;
15     return true;
16   }
17
18   function burn(uint value) public returns (bool) {
19     require (balance[msg.sender] >= value);
20     balance[msg.sender] -= value;
21     totalSupply -= value;
22     return true;
23   }
24 }
```

Global state variables

Constructor

Function

Function

Solidity Contract

Smart Contract

- Digital contract written in programming languages.

```
1  contract Netkoin {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor (uint initialSupply) {
6      totalSupply = initialSupply;
7      balance[msg.sender] = initialSupply;
8    }
9
10   function transfer (address to, uint value) public
11   returns (bool) {
12     require (balance[msg.sender] >= value);
13     balance[msg.sender] -= value;
14     balance[to] += value;
15     return true;
16   }
17
18   function burn(uint value) public returns (bool) {
19     require (balance[msg.sender] >= value);
20     balance[msg.sender] -= value;
21     totalSupply -= value;
22     return true;
23   }
24 }
```

Solidity Contract

Send transactions by invoking functions

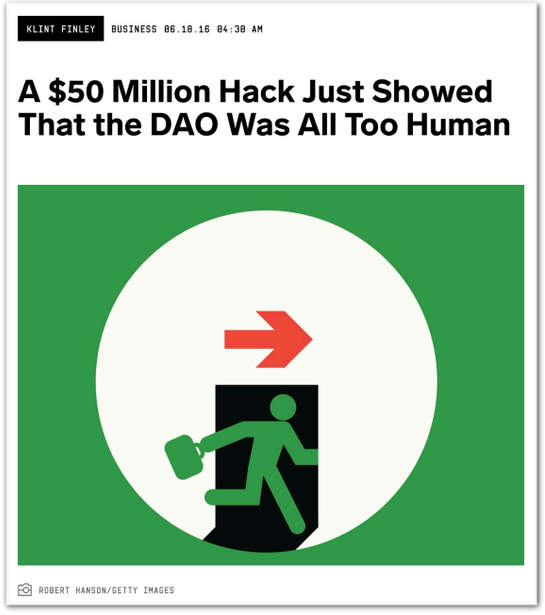
balance[X] = 15,
balance[Y] = 0

transfer(Y, 5)
where X=msg.sender

balance[X] = 10,
balance[Y] = 5

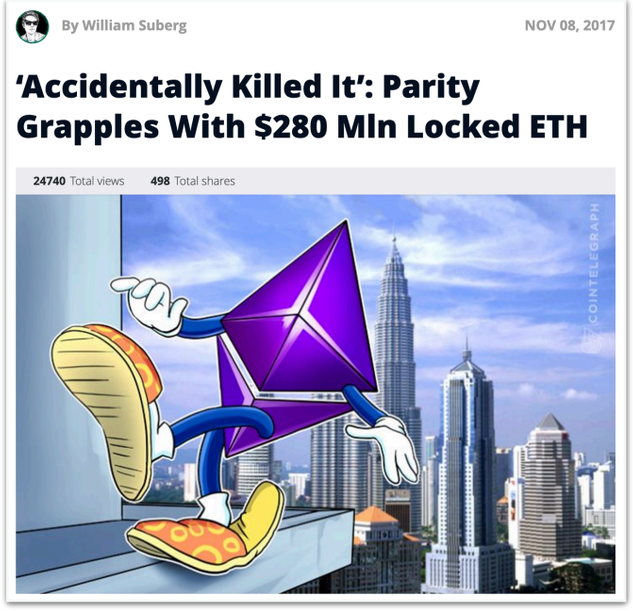
Pressing Issue:
Ensuring Safety of Smart Contracts

- Bugs cannot be fixed after deployed.
- Huge financial damage once exploited.


KLINT FINLEY BUSINESS 06.18.16 04:30 AM

A $50 Million Hack Just Showed That the DAO Was All Too Human

ROBERT HANSON/GETTY IMAGES


By William Suberg NOV 08, 2017

'Accidentally Killed It': Parity Grapples With $280 Mln Locked ETH

24740 Total views 498 Total shares


ETHEREUM

BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

Sam Town · April 25, 2018 at 10:38 pm UTC · 3 min read

The images were captured from:
https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/
https://cointelegraph.com/news/accidentally-killed-it-parity-grapples-with-280-mln-locked-eth
https://cryptoslate.com/batchoverflow-exploit-creates-trillions-of-ethereum-tokens

Goal: Automatic Safety Verification of Smart Contracts

- Focus: verifying safety of arithmetic operations.
  - Smart contracts involve lots of arithmetic operations.
  - Major sources of security vulnerabilities.

Arithmetic Over/underflow	Bad Randomness	Access Control	Unsafe Input Dependency	Others	Total
487 (95.7 %)	10 (1.9 %)	4 (0.8 %)	4 (0.8 %)	4 (0.8%)	509

Statistics on CVE-reported vulnerabilities of Ethereum smart contracts (as of May. 31, 2019)

SmartMesh (CVE-2018-10376)

CVE-ID	
CVE-2018-10376	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
An integer overflow in the transferProxy function of a smart contract implementation for SmartMesh (aka SMT), an Ethereum ERC20 token, allows attackers to accomplish an unauthorized increase of digital assets via crafted _fee and _value parameters, as exploited in the wild in April 2018, aka the "proxyOverflow" issue.	

Timestamp: 742 days 8 hrs ago (Apr-24-2018 07:16:19 PM +UTC)

From: 0xd6a09bdb29e1eafa92a30373c44b09e2e2e0651e

To: Contract 0x55f93985431fc9304077687a35a1ba103dc1e081 (SmartMesh: Token Sale)

Tokens Transferred: 2

From 0xdf31a499a5a8358… To 0xdf31a499a5a8358… For
65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463
($208,723,106,027,924,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) SmartMesh (SMT)

From 0xdf31a499a5a8358… To 0xd6a09bdb29e1ea… For
50,659,039,041,325,800,000,000,000,000,000,000,000,000,000,000,000,000,000,000.693003461994217473
($162,340,193,577,274,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) SmartMesh (SMT)

https://etherscan.io/tx/0x1abab4c8db9a30e703114528e31dee129a3a758f7f8abc3b6494aad3d304e43f

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

Send money to a receiver

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

Send money to a receiver

Pay fee to the proxy (msg.sender)

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

Send money to a receiver

Pay fee to the proxy (msg.sender)

Deduct money from the sender

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

To prevent underflows in token sender's balance

Send money to a receiver

Pay fee to the proxy (msg.sender)

Deduct money from the sender

SmartMesh (CVE-2018-10376)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

To prevent underflows in token sender's balance

To prevent overflows in token recipients' balances

Send money to a receiver

Pay fee to the proxy (msg.sender)

Deduct money from the sender

SmartMesh (CVE-2018-10376)

fee+value may overflow to 0!

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

To prevent underflows in token sender's balance

To prevent overflows in token recipients' balances

Send money to a receiver

Pay fee to the proxy (msg.sender)

Deduct money from the sender

SmartMesh (CVE-2018-10376)

```
balance[from] = balance [to] = balance[msg.sender] = 0
value   = 0x8fff…fff
fee     = 0x7000…001
```

256-bit unsigned integers in hexadecimal numbers (64 digits)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

SmartMesh (CVE-2018-10376)

```
balance[from] = balance [to] = balance[msg.sender] = 0
value   = 0x8fff…fff
fee     = 0x7000…001
```

256-bit unsigned integers in hexadecimal numbers (64 digits)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

0!

False

0!

False

0!

SmartMesh (CVE-2018-10376)

```
balance[from] = balance [to] = balance[msg.sender] = 0
value   = 0x8fff…fff
fee     = 0x7000…001
```

256-bit unsigned integers in hexadecimal numbers (64 digits)

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

False

0!

0!

False

0!

0x8fff…fff

0x7000…001

0

Shortcomings of Existing Approaches

- Bug-finders may miss critical bugs.
  - E.g., Osiris [ACSAC '18], Oyente [CCS '16], Mythril, Manticore

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

Only Osiris detects this vulnerability

Shortcomings of Existing Approaches

- Bug-finders may miss critical bugs.
  - E.g., Osiris [ACSAC '18], Oyente [CCS '16], Mythril, Manticore

```
1  function multipleTransfer (address [] to, uint value) {
2    require (value * to.length > 0);
3    require (balances[msg.sender] >= value * to.length);
4    balances[msg.sender] -= value * to.length;
5
6    for (uint i = 0; i < to.length; ++i) {
7      balances[to[i]] += value;
8    }
9  }
```

To prevent underflows in token sender's balance

Deduct money from the sender

CVE-2018-14006

Shortcomings of Existing Approaches

- Bug-finders may miss critical bugs.
  - E.g., Osiris [ACSAC '18], Oyente [ore

Despite the similarity, Osiris (and Mythril) fails

```
1  function multipleTransfer (address [] to, uint value) {
2    require (value * to.length > 0);
3    require (balances[msg.sender] >= value * to.length);
4    balances[msg.sender] -= value * to.length;
5
6    for (uint i = 0; i < to.length; ++i) {
7      balances[to[i]] += value;
8    }
9  }
```

To prevent underflows in token sender's balance

Deduct money from the sender

CVE-2018-14006

Shortcomings of Existing Approaches

- Existing verifiers are imprecise.
  - E.g., Zeus [NDSS '18], SMTChecker [ISoLA '18]

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Safe!

Safe!

Shortcomings of Existing Approaches

- Existing verifiers are imprecise.
  - E.g., Zeus [NDSS '18], SMTChecker [ISoLA '18]

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

False alarm by Zeus, SMTChecker

False alarm by Zeus, SMTChecker

Shortcomings of Existing Approaches

- Bug-finders may miss critical vulnerabilities.
  - Consider subset of behaviors.
  - E.g., Osiris [ACSAC '18], Oyente[CCS '16], Mythril, Manticore
- Exiting verifiers are not precise.
  - Compromise precision to detect all vulnerabilities.
  - E.g., Zeus [NDSS' 18], SMTChecker [ISoLA '18]

Shortcomings of Existing Approaches

- Bug-finders may miss critical vulnerabilities.

  - Consider subset of behaviors.

  - E.g., Osiris [ACSAC '18], Oyente[CCS '16], Mythril, Manticore

False negatives

- Exiting verifiers are not precise.

  - Compromise precision to detect all vulnerabilities.

  - E.g., Zeus [NDSS' 18], SMTChecker [ISoLA '18]

False positives

VeriSmart: Exhaustive, Precise, Fully Automated Smart Contract Safety Verifier

Exhaustive: detect all vulnerabilities

Precise: very few false positives

CVE-2018-10376

```
1   function transferProxy (address from, address to, uint value, uint fee)
2   public returns (bool) {
3     if (balance[from] < fee + value)
4       revert ();
5
6     if (balance[to] + value < balance[to] ||
7         balance[msg.sender] + fee < balance[msg.sender])
8       revert ();
9
10    balance[to] += value;
11    balance[msg.sender] += fee;
12    balance[from] -= value + fee;
13    return true;
14  }
```

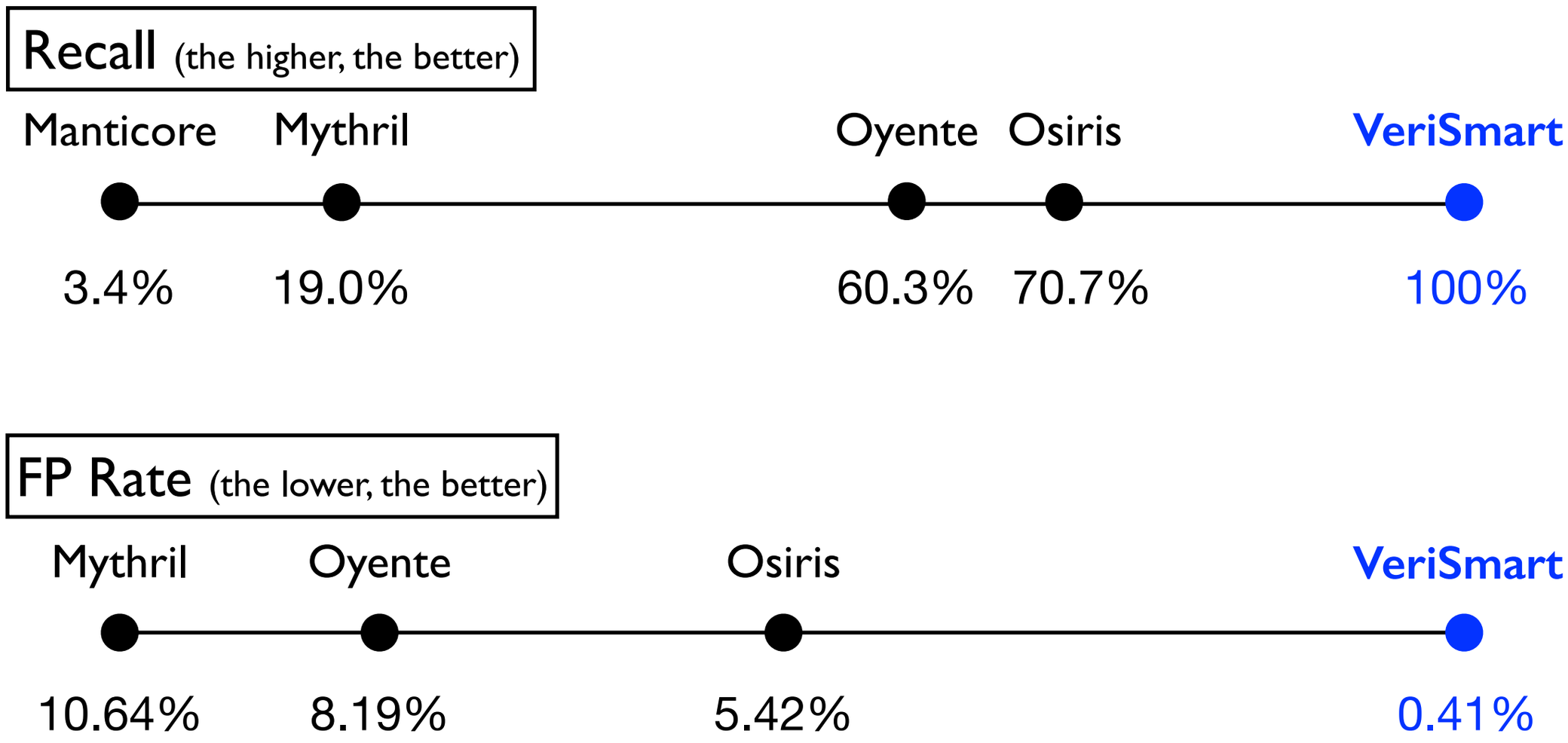
CVE-2018-14006

```
1   function multipleTransfer (address [] to, uint value) {
2     require (value * to.length >= 0);
3     require (balances[msg.sender] >= value * to.length);
4     balances[msg.sender] -= value * to.length;
5
6     for (uint i = 0; i < to.length; ++i)
7       balances[to[i]] += value;
8     }
9   }
```

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

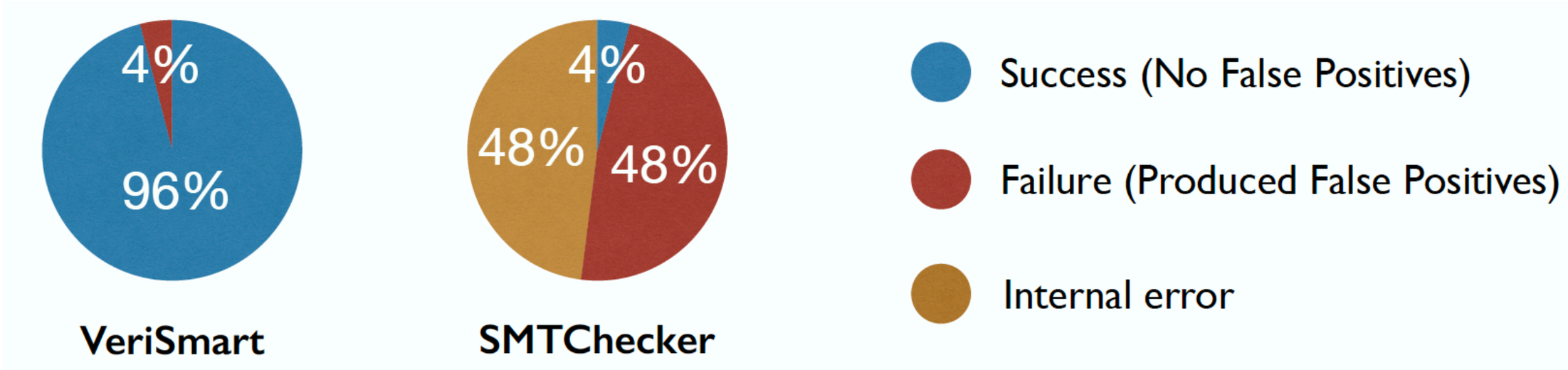
Result Highlight: vs. Bug-finders

- On 60 smart contracts with CVE vulnerabilities.

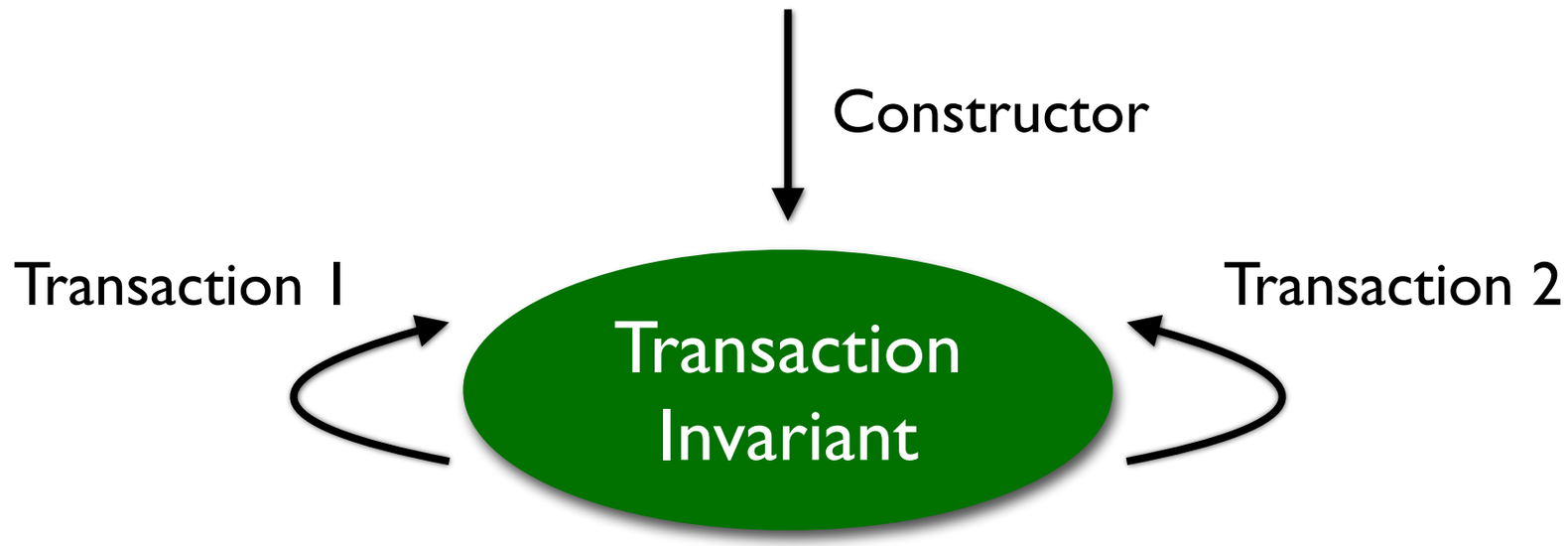


Result Highlight: vs. Verifiers

- On 25 smart contracts where Zeus produced false positives.
  - VeriSmart produced false positives on 1 contract.
  - SMTChecker produced false positives on 12 contracts.



Key Feature of VeriSmart: Inference and Use of Transaction Invariant



- Global invariant that holds under arbitrary interleaving of transactions.
  - Valid at the end of the constructor.
  - Validity preserved by executions of transactions.
- Precise verification with inferred invariants.

Transaction Invariant

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Transaction Invariant:
totalSupply = $\Sigma_i$balance[i]

Transaction Invariant

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Transaction Invariant:
totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

Transaction Invariant

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Transaction Invariant:
totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

Transaction Invariant

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Transaction Invariant:
totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

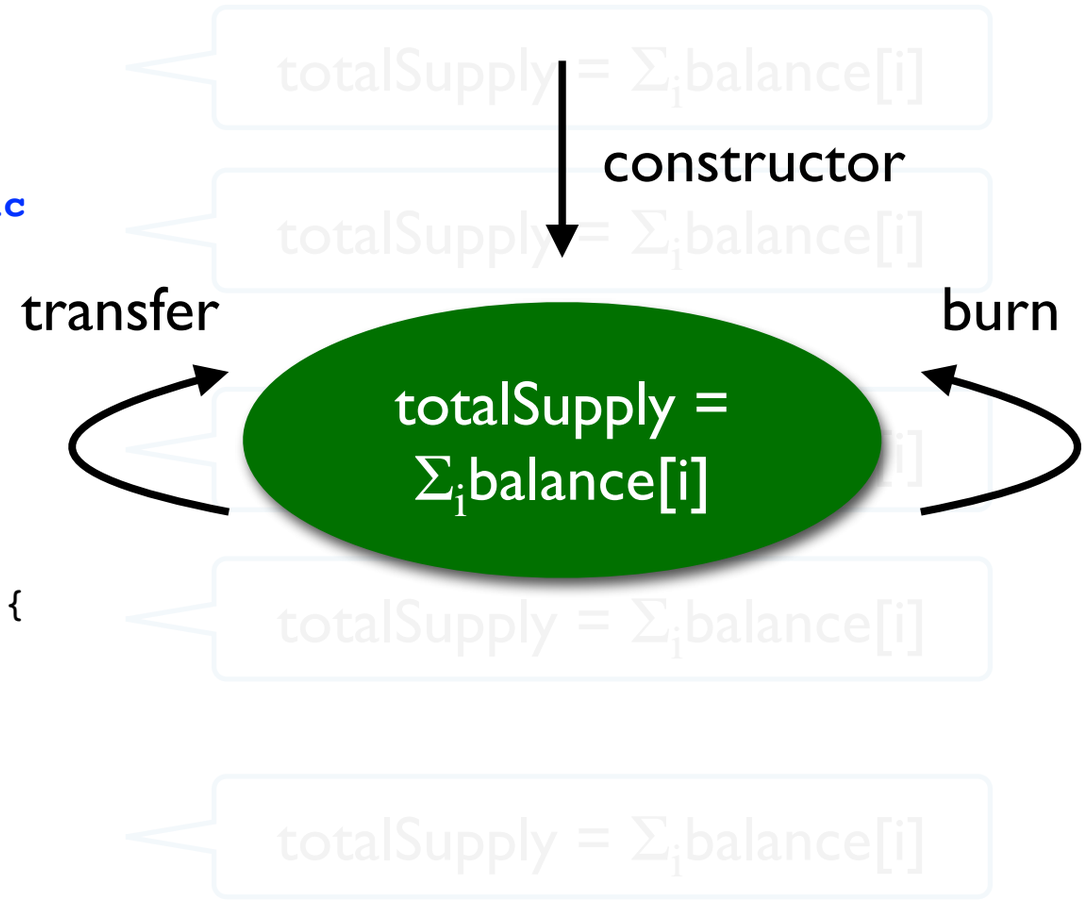
totalSupply = $\Sigma_i$balance[i]

totalSupply = $\Sigma_i$balance[i]

Transaction Invariant

```
1   contract Netkoin {
2     mapping (address => uint) public balance;
3     uint public totalSupply;
4
5     constructor (uint initialSupply) {
6       totalSupply = initialSupply;
7       balance[msg.sender] = totalSupply;
8     }
9
10    function transfer (address to, uint value) public
11    returns (bool) {
12      require (balance[msg.sender] >= value);
13      balance[msg.sender] -= value;
14      balance[to] += value;
15      return true;
16    }
17
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

Transaction Invariant:
totalSupply = $\Sigma_i$balance[i]



Verification with Transaction Invariants

```
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

- To show: prove totalSupply >= value at line 21.

Verification with Transaction Invariants

```
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

- To show: prove totalSupply >= value at line 21.

- Verification with the inferred transaction invariant:

totalSupply >= balance[msg.sender]

from the transaction invariant:
totalSupply=$\Sigma_i$balance[i]

Verification with Transaction Invariants

```
18    function burn(uint value) public returns (bool) {
19      require (balance[msg.sender] >= value);
20      balance[msg.sender] -= value;
21      totalSupply -= value;
22      return true;
23    }
24  }
```

- To show: prove totalSupply >= value at line 21.

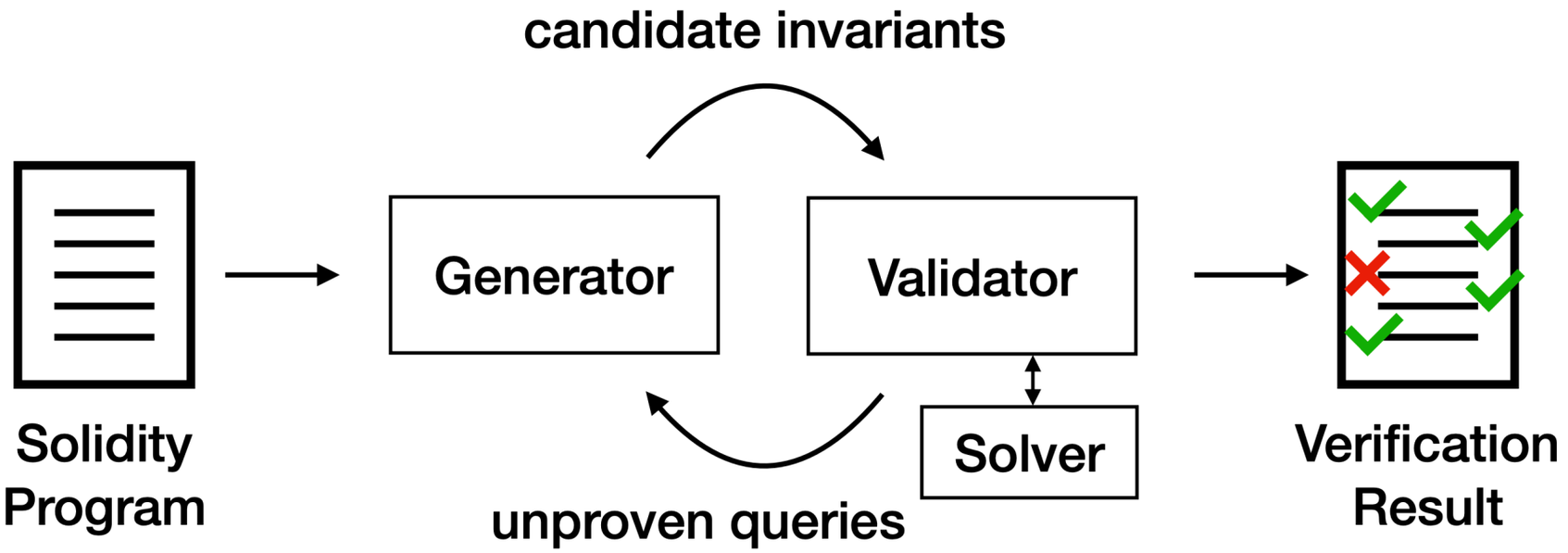
- Verification with the inferred transaction invariant:

totalSupply >= balance[msg.sender] >= value

from the transaction invariant:
totalSupply=$\Sigma_i$balance[i]

Line 19

VeriSmart Algorithm



- Generator: producing candidate transaction (and loop) invariants.
- Validator: verification with candidate invariants.

Experimental Setup

- Compared with 6 existing analyzers.
  - Bug-finder: Osiris [ACSAC '18], Oyente [CCS '16], Mythril, Manticore
  - Verifier: Zeus [NDSS '18], SMTChecker [ISoLA '18]
- Benchmarks (https://github.com/kupl/VeriSmart-benchmarks)
  - vs. Bug-finders: 60 Solidity smart contracts with CVE vulnerabilities
  - vs. Verifiers: 25 Solidity smart contracts from Zeus [NDSS '18]
- Target: arithmetic safety (integer over/underflows, division-by-zeros)

vs. Bug-finders

- Results on the CVE data set (60 contracts).

	VeriSmart	Osiris [ACSAC '18]	Oyente [CCS '16]	Mythril	Manticore
#Alarm	492	240	171	94	14
#False Positive	2	13	14	10	0
#Detected CVE	58	41	35	11	2
Recall (#valid CVE = 58)	100%	70.7%	60.3%	19.0%	3.4%
FP Rate (#FP/ #Alarm)	0.41%	5.42%	8.19%	10.64%	0% (low recall)

vs. Bug-finders

- Results on the CVE data set (60 contracts).

	VeriSmart	Osiris [ACSAC '18]	Oyente [CCS '16]	Mythril	Manticore
#Alarm	492	240	171	94	14
#False Positive	2	13	14	10	0
#Detected CVE	58	41	35	11	2
Recall (#valid CVE = 58)	100%	70.7%	60.3%	19.0%	3.4%
FP Rate (#FP/ #Alarm)	0.41%	5.42%	8.19%	10.64%	0% (low recall)

vs. Bug-finders

- Results on the CVE data set (60 contracts).

	VeriSmart	Osiris [ACSAC '18]	Oyente [CCS '16]	Mythril	Manticore
#Alarm	492	240	171	94	14
#False Positive	2	13	14	10	0
#Detected CVE	58	41	35	11	2
Recall (#valid CVE = 58)	100%	70.7%	60.3%	19.0%	3.4%
FP Rate (#FP/ #Alarm)	0.41%	5.42%	8.19%	10.64%	0% (low recall)

Incorrect CVE reports Found by VeriSmart

- VeriSmart found 6 incorrectly-reported CVE vulnerabilities.
  - Proved the safety thanks to the ability to infer transaction invariants.

CVE ID	Name	#Incorrect Queries	#FP		
			OSIRIS	OYENTE	VERISMART
2018-13113	ETT	2	2	2	0
2018-13144	PDX	1	1	1	0
2018-13326	BTX	2	2	2	0
2018-13327	CCLAG	1	1	1	0

vs. Verifiers

- Results on 25 contracts where Zeus produced false positives.
- Without transaction invariants, VeriSmart failed on 17 contracts.

No.	LOC	#Q	VeriSmart			SMTChecker [12]			Zeus [11]
			#Alarm	#FP	Verified	#Alarm	#FP	Verified	Verified
#1	42	3	0	0	✓	3	3	✗	✗
#2	78	2	1	0	✓	2	1	✗	✗
#3	75	7	2	0	✓	7	5	✗	✗
#4	70	7	0	0	✓	7	7	✗	✗
#5	103	8	0	0	✓	6	6	✗	✗
#6	141	5	2	0	✓	internal error			✗
#7	74	6	1	0	✓	6	5	✗	✗
#8	84	6	0	0	✓	4	4	✗	✗
#9	82	6	0	0	✓	6	6	✗	✗
#10	99	2	1	0	✓	internal error			✗
#11	171	15	9	0	✓	internal error			✗
#12	139	7	0	0	✓	internal error			✗
#13	139	7	0	0	✓	internal error			✗
#14	139	7	0	0	✓	internal error			✗
#15	139	7	0	0	✓	internal error			✗
#16	141	16	10	0	✓	internal error			✗
#17	153	5	0	0	✓	internal error			✗
#18	139	7	0	0	✓	internal error			✗
#19	113	4	0	0	✓	4	4	✗	✗
#20	40	3	0	0	✓	3	3	✗	✗
#21	59	3	0	0	✓	internal error			✗
#22	28	3	1	0	✓	1	0	✓	✗
#23	19	3	0	0	✓	3	3	✗	✗
#24	457	30	13	6	✗	internal error			✗
#25	17	3	0	0	✓	3	3	✗	✗
Total	2741	172	40	6	✓:24 ✗ : 1	55	50	✓: 1 ✗: 12	✓: 0 ✗:25

General Applicability of VeriSmart

- VeriSmart can be used to verify other safety properties.
- Case study: access control vulnerabilities.
  - Security-sensitive variables (e.g., ‘owner’) can be updated by anyone.

General Applicability of VeriSmart

CVE-2018-11329

```
1  function DrugDealer () public {
2    ceoAddress = msg.sender;
3  }
4
5  function buyDrugs () public payable {
6    ceoAddress.transfer(msg.value);
7    drugs[msg.sender] += .. ;
8    ..
9  }
```

Can be updated by anyone

send Ether to 'ceoAddress'

General Applicability of VeriSmart

CVE-2018-11329

```
1  function DrugDealer () public {
2    ceoAddress = msg.sender;
3  }
4
5  function buyDrugs () public payable {
6    ceoAddress.transfer(msg.value);
7    drugs[msg.sender] += .. ;
8    ..
9  }
```

Can be updated by anyone

send Ether to 'ceoAddress'

Transaction 1

DrugDealer () where
msg.sender=attacker

Transaction 2

buyDrugs () where
msg.sender=benign user
msg.value=some Ethers

General Applicability of VeriSmart

- Safety property:

```
function DrugDealer () public {
   assert (ceoAddress == msg.sender);
   ceoAddress = msg.sender;
}
```

General Applicability of VeriSmart

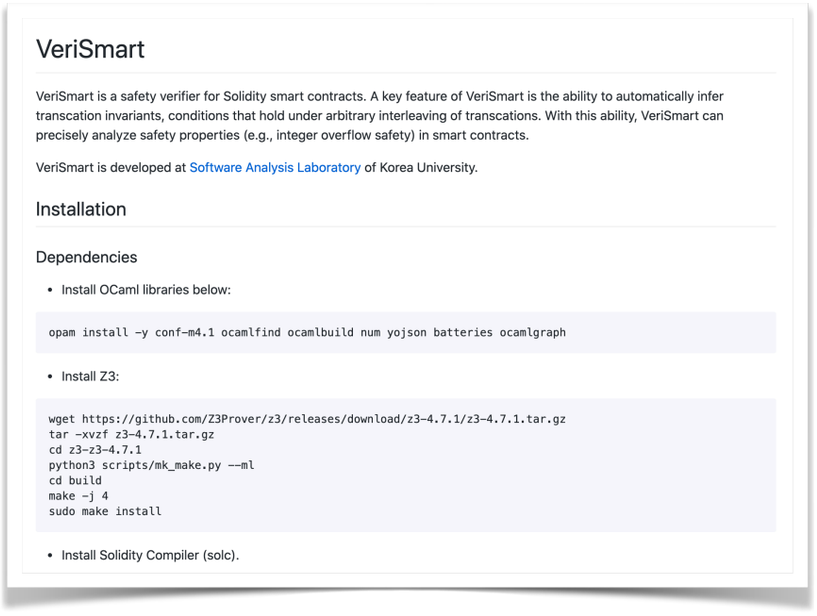
- Safety property:

```
function DrugDealer () public {
    assert (ceoAddress == msg.sender);
    ceoAddress = msg.sender;
}
```

- Detected all related access control CVE vulnerabilities:
  - CVE-2018-10666, CVE-2018-10705, CVE-2018-11329
- Proved the absence of the bugs for 55 out of 60 from the CVE data set.

Summary

- VeriSmart: Exhaustive, Precise, Fully Automated Safety Verifier
  - Key feature: automatic inference and use of transaction invariants
- VeriSmart is publicly available:
  - http://prl.korea.ac.kr/verismart
  - https://iotcube.korea.ac.kr/process/type/veris



Thank you for listening!