스마트 컨트랙트 검증, 테스팅, 자동 수정

소순범, 오학주

고려대학교 정보대학 컴퓨터학과

2022.07.14 @ERC 워크샵, 포항

블록체인

소비자

전자화폐 지불

상품

상점 A

블록체인

소비자

전자화폐 지불

상품

상점 A

상품

중복사용

상점 B

블록체인

소비자

전자화폐 지불

상품

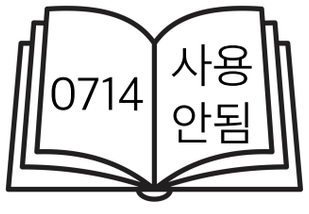
상점 A

상품

중복사용

상점 B

유효성 검증



신뢰가능한 중개인

블록체인

소비자

전자화폐 지불

상품

중복사용

상점 B

유효성 검증

상점 A

신뢰가능한 중개인

스마트 컨트랙트

블록체인 1.0

중개인이 필요 없는
전자화폐 거래 알고리즘

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

중개인이 필요 없는
전자화폐 거래 알고리즘

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

중개인이 필요 없는
전자화폐 거래 알고리즘

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

중개인이 필요 없는
전자화폐 거래 알고리즘

vs.

블록체인 2.0

임의의 거래가 가능

Key: 스마트 컨트랙트

스마트 컨트랙트 생김새

```
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  }
```

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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  }
```

사용자의 계좌 정보

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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  }
```

사용자의 계좌 정보

송금

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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  }
```

사용자의 계좌 정보

송금

잔고가 충분하면

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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  }
```

사용자의 계좌 정보

송금

잔고가 충분하면
거래를 실행

데이터

생성자

함수

함수

스마트 컨트랙트 분석 연구

Google 학술검색 | smart contract analysis

학술자료 | 검색결과 약 1,290,000개 중 12페이지 (**0.04**초)

모든 날짜
2022 년부터
2021 년부터
2018 년부터
기간 설정...

관련도별 정렬
날짜별 정렬

모든 언어
한국어 웹

모든 유형
검토 자료

☐ 특허 포함
☑ 서지정보 포함

Smart contract relations in e-commerce: legal implications of exchanges conducted on the blockchain
PA Ryan - Technology Innovation Management Review, 2017 - opus.lib.uts.edu.au
… This article will apply Macaulay's behavioural **analysis** of business exchanges to **smart** …
Once the management of performance of a **smart contract** is explained and understood, it is …
☆ 저장 99 인용 73회 인용 관련 학술자료 전체 11개의 버전 »

Defining **smart contract** defects on ethereum
J Chen, X Xia, D Lo, J Grundy, X Luo… - IEEE Transactions on …, 2020 - ieeexplore.ieee.org
… **contract** defects in 587 real world **smart contract** and publicly released our dataset. Finally, we summarized 5 impacts caused by **contract** … studies on **smart contract analysis** and testing, …
☆ 저장 99 인용 51회 인용 관련 학술자료 전체 14개의 버전

A study of inequality in the ethereum **smart contract** ecosystem
BC Gupta, SK Shukla - 2019 Sixth International Conference on …, 2019 - ieeexplore.ieee.org
… To tackle the problem of finding the **smart contract** addresses, we … find whether address contained a **smart contract** or not. Listing … Next we move our **analysis** to the **contract** creation trans…
☆ 저장 99 인용 5회 인용 관련 학술자료

스마트 컨트랙트 분석 연구

Since 2016, more than 100 tools have been developed

Table 6.1: Identified security analysis tools – continued from previous page

PS Ref.	Tool Name or *Identifier*	Publ. Date	Open Source	Short Description	Mapped SLRs, Surveys and *SmartBugs* [268]
[10]•	Syrup	2020-05	✓	Gas super-optimization of smart contracts, based on symbolic execution and Max-SMT encoding, which finds bytecode blocks with minimal gas cost	1: [141]
[164]	teEther	2019-02	✓	Automated vulnerability detection and exploit generation on smart contract bytecode, based on CFG generation, symbolic execution and SMT solving	11: [141, 160, 296, 304, 49, 74, 81, 145, 157, 229, 331]
[142]	*Transaction-based analysis with LSTM network*	2021-03	✗	Transaction-based classification of Ethereum smart contracts for anomaly detection and malicious contract identification, using a long short-term memory (LSTM) network machine learning approach	0: –
[36]	Vandal	2016-08	✓	Smart contract bytecode security analysis framework for vulnerability detection, based on semantic logic relation conversion, CFG and data flow analysis, user generated security specifications and vulnerability queries	14: [141, 160, 296, 15, 49, 74, 81, 105, 157, 189, 229, 233, 246, 253]
[270]	VeriSmart	2020-04	✓	Arithmetic safety verification for Solidity smart contracts, based on CEGIS-style verification, automatically inferring transaction invariants, providing safety property guarantees, such as freedom from integer overflow	3: [141, 160, 296]
[140]•	Vertigo	2019-08	✓	Mutation testing framework for Solidity smart contracts and Truffle, implementing a range of mutation operators	0: –
[231]	VerX	2020-05	✗	Automatic verification to prove functional properties and temporal safety properties, based on reachability checking, code instrumentation, symbolic execution and delayed predicate abstraction `https://verx.ch/`	2: [141, 304]
[266]	Visualgas	2018-11	✗	Gas cost analysis for Solidity smart contracts, based on AST generation, fuzz testing and transaction trace analysis to support gas-efficient smart contract development	0: –
[236]	VulDeeSmartContract	2020-01	✓	Reentrancy vulnerability detection in Solidity smart contracts, based on an extension of the VulDeePecker framework implementing the bidirectional long-short term memory with attention mechanism (BLSTM-ATT) deep learning approach	0: –
[156]•	Zeus	2018-02	✗	Smart contract security verification framework and vulnerability detection, based on Solidity to LLVM translation, abstract interpretation and symbolic model checking via constrained horn clauses	19: [141, 160, 267, 296, 304, 15, 14, 49, 74, 81, 105, 145, 157, 203, 206, 229, 233, 246, 253]
TOTALS	**140**		**83**		**>0: 101, 0: 39**

•: Additional primary study identified during data extraction. ✓: See table 6.2 for open source information

Images captured from:

Systematic Review of Ethereum Smart Contract Security Vulnerabilities, Analysis Methods and Tools (Heidelinde Rameder's BSc Thesis, 2021)

모든 유형

검토 자료

☐ 특허 포함

☑ 서지정보 포함

A study of inequality in the ethereum **smart contract** ecosystem

BC Gupta, SK Shukla - 2019 Sixth International Conference on …, 2019 - ieeexplore.ieee.org

… To tackle the problem of finding the **smart contract** addresses, we … find whether address contained a **smart contract** or not. Listing … Next we move our **analysis** to the **contract** creation trans…

☆ 저장 99 인용 5회 인용 관련 학술자료

스마트 컨트랙트는 안전성 검증이 필수

- 한번 배포되면 코드 수정 불가능
- 공격에 성공하면 막대한 금전적 피해 발생

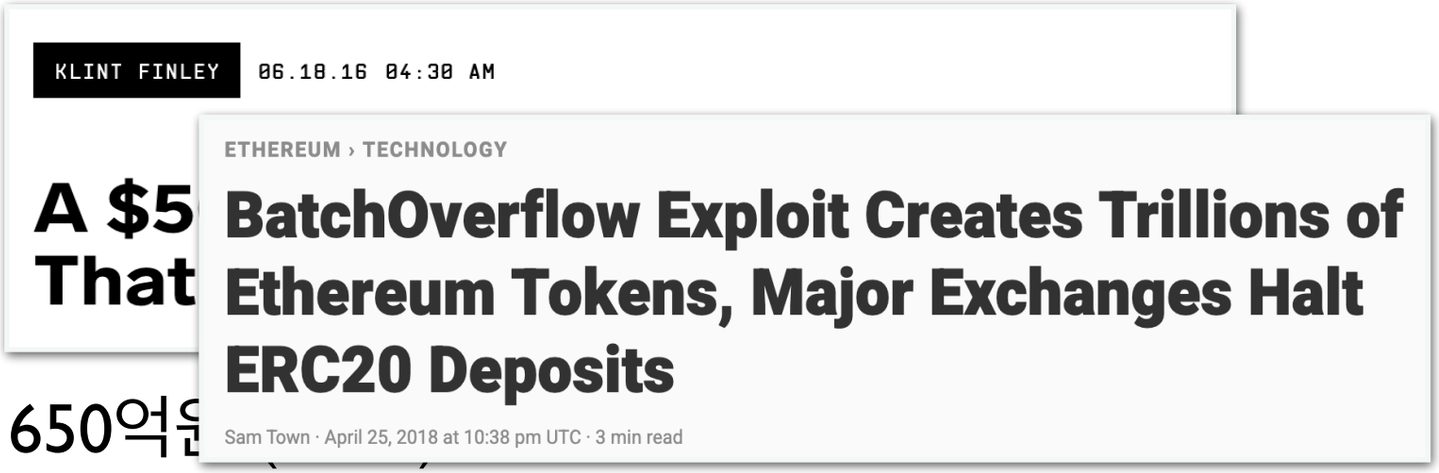

KLINT FINLEY 06.18.16 04:30 AM

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

650억원 (2016)

스마트 컨트랙트는 안전성 검증이 필수

- 한번 배포되면 코드 수정 불가능
- 공격에 성공하면 막대한 금전적 피해 발생


KLINT FINLEY 06.18.16 04:30 AM

A $5
That

ETHEREUM › TECHNOLOGY

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

650억원

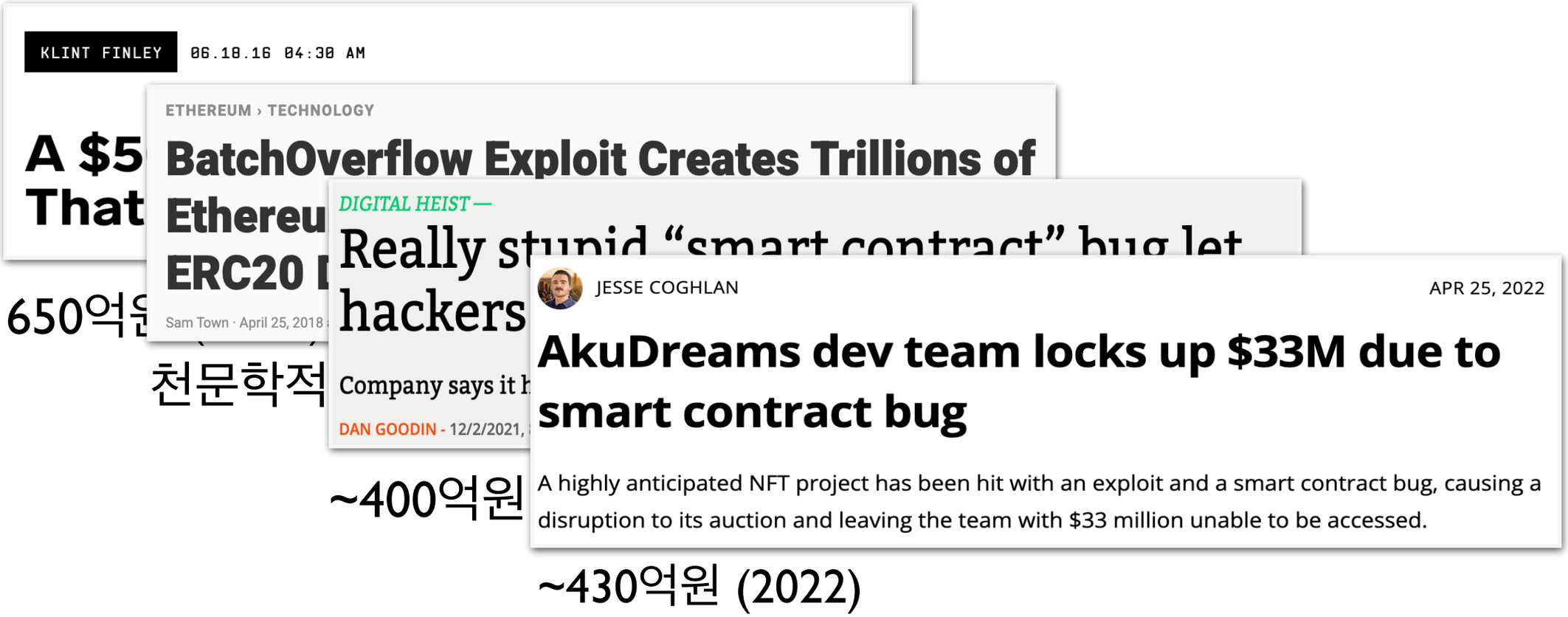
천문학적 (2018)

스마트 컨트랙트는 안전성 검증이 필수

- 한번 배포되면 코드 수정 불가능
- 공격에 성공하면 막대한 금전적 피해 발생

스마트 컨트랙트는 안전성 검증이 필수

- 한번 배포되면 코드 수정 불가능
- 공격에 성공하면 막대한 금전적 피해 발생



SmartMesh 사례 (2018)

- 정수 오버플로우 취약점을 이용하여 천문학적 금액의 토큰을 생성 (CVE-2018-10376)

0xd6a09bdb29e1eafa92a30373c44b09e2e2e0651e

Contract 0x55f93985431fc9304077687a35a1ba103dc1e081 (SmartMesh: Token Sale)

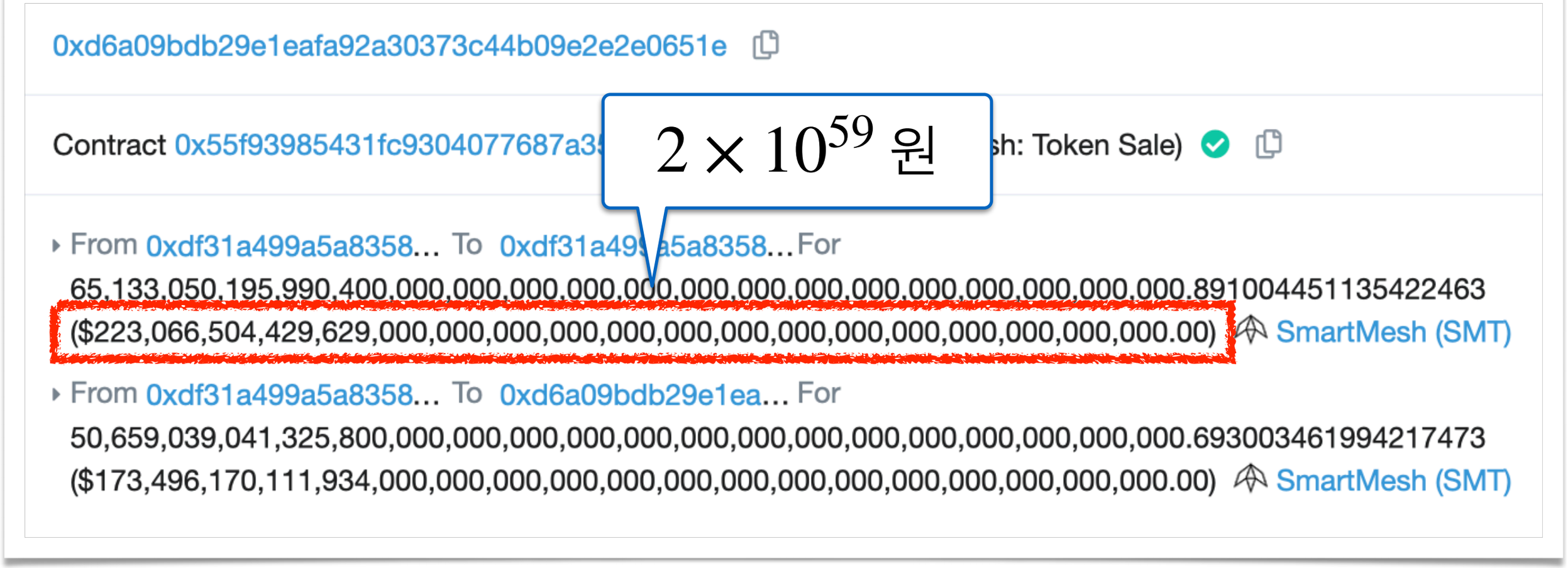
▸ From 0xdf31a499a5a8358... To 0xdf31a499a5a8358...For
65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463
($223,066,504,429,629,000,000,000,000,000,000,000,000,000,000,000,000,000.00) SmartMesh (SMT)

▸ 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
($173,496,170,111,934,000,000,000,000,000,000,000,000,000,000,000,000,000.00) SmartMesh (SMT)

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

SmartMesh 사례 (2018)

- 정수 오버플로우 취약점을 이용하여 천문학적 금액의 토큰을 생성 (CVE-2018-10376)



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

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

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

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

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

송금

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

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

보내는 사람의 잔고가 충분한지 체크

송금

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

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

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

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

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

(실질적) 오버플로우/언더플로우 발생하지 않음

SmartMesh 사례 (2018)

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

SmartMesh 사례 (2018)

`balance[from] = balance[to] = balance[msg.sender] = 0`

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

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

O!

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

0!

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

8fffff…ff

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

8fffff…ff

700…00

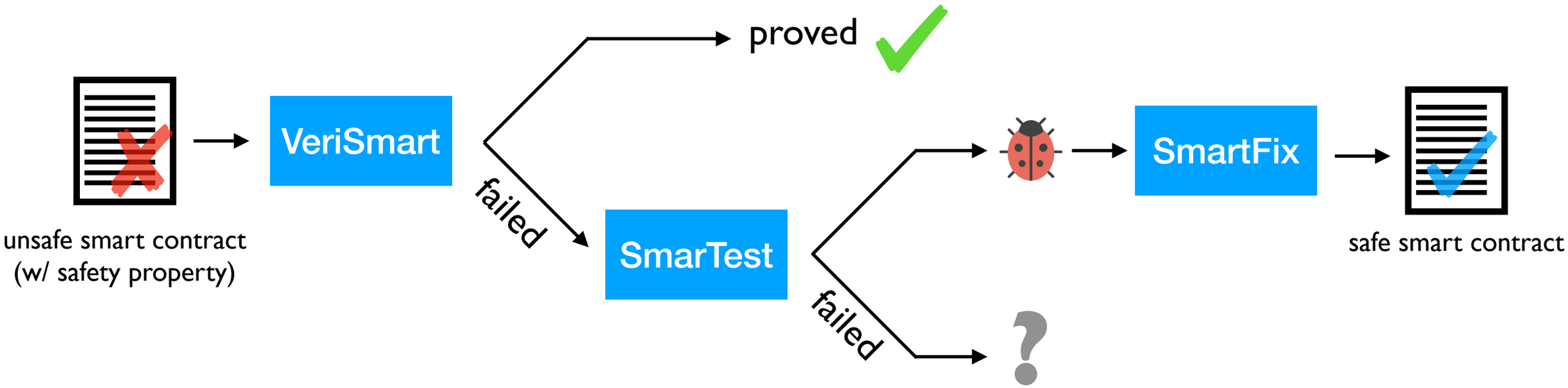
SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
1   function transferProxy (address from, address to, uint
          value, uint fee) public returns (bool) {
2       if (balance[from] < fee + value) // false, 0!
3           revert();
4       if (balance[to] + value < balance[to] ||
5           balance[msg.sender] + fee < balance[msg.sender]) // false
6           revert();
7       balance[to] += value; // 8fffff…ff
8       balance[msg.sender] += fee; // 700…00
9       balance[from] -= value + fee; // 0!
10      return true;
11  }
```

스마트 컨트랙트 자동 분석 기술

- **VeriSmart** (IEEE S&P 2020): 오류 검증기 (정적 분석)
- **SmarTest** (USENIX Security 2021): 오류 검출기 (동적 분석)
- **SmartFix** (in progress): 오류 수정기



VeriSmart: 오류 검증기

```
1  contract SocialChain {
2    uint totalSupply;
3    mapping(address=>uint) balance;
4    mapping(address=>mapping(address=>uint)) allowance;
5
6    constructor (uint initialSupply) {
7      totalSupply = initialSupply;
8      balance[msg.sender] = initialSupply;
9    }
10
11   function transfer (address to, uint value)
12   public returns(bool) {
13     require (balance[msg.sender] >= value);
14     balance[msg.sender] -= value;
15     balance[to] += value;
16     return true;
17   }
18
19   function approve (address spender, uint value)
20   public returns(bool) {
21     allowance[msg.sender][spender] = value;
22     return true;
23   }
24
25   function transferFrom (address from, address to,
26         uint value) public returns (bool) {
27     require (balance[from] >= value);
28     require (balance[to] + value > balance[to]);
29     require (allowance[from][msg.sender] >= value);
30     balance[from] -= value;
31     balance[to] += value;
32     allowance[from][msg.sender] += value;
33     return true;
34   }
35 }
```

VeriSmart: 오류 검증기

```
1   contract SocialChain {
2     uint totalSupply;
3     mapping(address=>uint) balance;
4     mapping(address=>mapping(address=>uint)) allowance;
5
6     constructor (uint initialSupply) {
7       totalSupply = initialSupply;
8       balance[msg.sender] = initialSupply;
9     }
10
11    function transfer (address to, uint value)
12    public returns(bool) {
13      require (balance[msg.sender] >= value);
14      balance[msg.sender] -= value;
15      balance[to] += value;
16      return true;
17    }
18
19    function approve (address spender, uint value)
20    public returns(bool) {
21      allowance[msg.sender][spender] = value;
22      return true;
23    }
24
25    function transferFrom (address from, address to,
26          uint value) public returns (bool) {
27      require (balance[from] >= value);
28      require (balance[to] + value > balance[to]);
29      require (allowance[from][msg.sender] >= value);
30      balance[from] -= value;
31      balance[to] += value;
32      allowance[from][msg.sender] += value;
33      return true;
34    }
35  }
```

VeriSmart Output

safe

safe

unsafe (may overflow)

기존 정적 분석 도구들

- Bug-finders

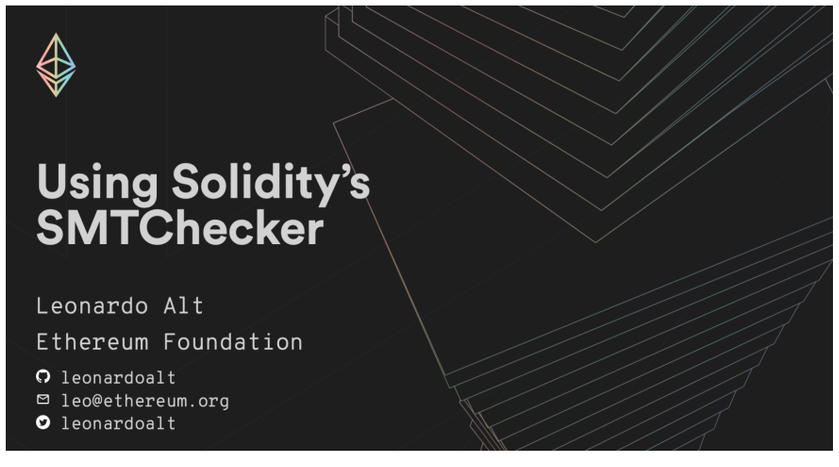
Oyente

Mythril

Manticore

Osiris

- Verifiers



SMT Checker

Zeus

기존 자동 분석 기술의 한계 (I)

- 오류 검출기(e.g., Mythril, Osiris, Oyente): 놓치는 취약점이 존재

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

Osiris만 검출 가능

CVE-2018-10376

기존 자동 분석 기술의 한계 (I)

- 오류 검출기(e.g., Mythril, Osiris, Oyente): 놓치는 취약점이 존재

```
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    for (uint i = 0; i < to.length; ++i) {
6      balances[to[i]] += value;
7    }
8    return true;
9  }
```

앞의 경우와 비슷한 오류이지만 검출 모두 실패

CVE-2018-14006

기존 자동 분석 기술의 한계 (2)

- 오류 검증기(SMTChecker, Zeus): 허위경보 존재

```
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 }
```

허위 경보 (False alarm)

허위 경보 (False alarm)

41st IEEE Symposium on Security and Privacy

VeriSmart

- 안전하면서 정확한 스마트 컨트랙트 정적 분석기

CVE-2018-10376

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

CVE-2018-14006

```
function multipleTransfer(address[] to, uint value) {
 require(value * to.length > 0);
 require(balances[msg.sender] >= value * to.length);
 balances[msg.sender] -= value * to.length;
 for (uint i = 0; i < to.length; ++i) {
   balances[to[i]] += value;
 }
 return true;
}
```

오류가 있다면 검출

```
contract Netkoin {
  mapping (address => uint) public balance;
  uint public totalSupply;

  constructor (uint initialSupply) {
    totalSupply = initialSupply;
    balance[msg.sender] = totalSupply;
  }

  function transfer (address to, uint value) public
  returns (bool) {
    require (balance[msg.sender] >= value);
    balance[msg.sender] -= value;
    balance[to] += value;
    return true;
  }

  function burn (uint value) public returns (bool) {
    require (balance[msg.sender] >= value);
    balance[msg.sender] -= value;
    totalSupply -= value;
    return true;
  }
}
```

Verified!

오류가 없다면 없다고 검증

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart			Osiris [7]			Oyente [9], [26]			Mythril [8]			MantiCore [10]		
					#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL	165	9	4	0	✓	3	0	✓	2	0	△	0	0	✗	0	0	✗
#8	2018-11446	GRX	434	39	24	2	✓	8	2	✗	12	4	✗	4	2	✗	internal error		
#9	2018-11561	EETHER	146	10	5	0	✓	4	0	✓	2	0	△	2	0	✓	0	0	✗
#10	2018-11687	BTCR	99	20	4	0	✓	2	0	✓	2	0	△	3	2	✗	0	0	✗
#11	2018-12070	SEC	269	40	8	0	✓	6	0	✓	4	0	✗	3	1	✗	0	0	✗
#12	2018-12230	RMC	161	9	5	0	✓	3	0	✓	5	0	✓	0	0	✗	0	0	✗
#13	2018-13113	ETT	142	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#14	2018-13126	MoxyOnePresale	301	5	3	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#15	2018-13127	DSPX	238	6	4	0	✓	3	0	✓	3	0	△	1	0	✗	0	0	✗
#16	2018-13128	ETY	193	10	4	0	✓	3	0	✓	3	0	△	0	0	✗	0	0	✗
#17	2018-13129	SPX	276	9	6	0	✓	5	0	✓	3	0	△	1	0	✗	internal error		
#18	2018-13131	SpadePreSale	312	4	3	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		
#19	2018-13132	SpadeIco	403	9	6	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		
#20	2018-13144	PDX	103	5	2	0	✓	2	1	✓	2	1	✓	internal error			0	0	✗
#21	2018-13189	UNLB	335	4	3	0	✓	2	0	✓	3	0	✓	1	0	✗	0	0	✗
#22	2018-13202	MyBO	183	17	11	0	✓	5	0	✓	3	0	✗	1	0	✗	internal error		
#23	2018-13208	MoneyTree	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#24	2018-13220	MAVCash	171	15	10	0	✓	4	0	✓	2	0	✗	1	0	✗	0	0	✗
#25	2018-13221	XT	186	15	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#26	2018-13225	MyYLCToken	181	17	11	0	✓	5	0	✓	6	0	✗	0	0	✗	0	0	✗
#27	2018-13227	MCN	172	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#28	2018-13228	CNX	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	2	0	✓	4	2	✓	1	1	✗	0	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#32	2018-13327	CCLAG	92	5	2	0	✓	2	1	✓	2	1	✓	0	0	✗	0	0	✗
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	3	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	13	0	✓	8	0	✓	2	0	✓	1	0	✗	1	0	✗
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	6	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	8	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	5	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	2	1	✗	0	0	✗
#40	2018-13722	HYIPToken	410	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	internal error		
#41	2018-13777	RRToken	166	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	0	0	✗
#42	2018-13778	CGCToken	224	13	6	0	✓	4	0	✓	4	0	✓	1	0	✗	1	0	✗
#43	2018-13779	YLCToken	180	17	11	0	✓	5	0	✓	6	0	✓	0	0	✗	0	0	✗
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	2	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	11	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	3	1	✓	timeout (> 3 days)		
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✗	3	0	△	3	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	2	1	✗	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✗	2	0	△	3	0	✓	1	0	✗
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	0	0	✗
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	3	0	△	0	0	✗
#52	2018-14006	NGT	249	27	13	0	✓	1	0	✗	5	0	△	0	0	✗	timeout (> 3 days)		
#53	2018-14063	TRCT	178	9	1	0	✓	1	0	✓	1	0	✓	4	2	✓	0	0	✗
#54	2018-14084	MKCB	273	17	10	0	✓	5	0	✓	4	0	✗	2	0	✗	1	0	✗
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	0	0	✗
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	0	0	✗	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	14	0	✓	0	0	✗
#58	2018-14576	SunContract	194	12	4	0	✓	1	0	✓	0	0	✗	0	0	✗	0	0	✗
#59	2018-17050	AI	141	8	3	0	✓	1	0	✓	1	0	✓	0	0	✗	0	0	✗
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	4	0	✓	0	0	✗	0	0	✗
Total			12493	976	492	2	✓:58 △: 0 ✗ : 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9], [26] #Alarm	Oyente [9], [26] #FP	Oyente [9], [26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL	165	9	4	0	✓	3	0	✓	2	0	△	0	0	✗	0	0	✗
#8	2018-11446	GRX	434	39	24	2	✓	8	2	✗	12	4	✗	4	2	✗	internal error		
#9	2018-11561	EETHER	146	10	5	0	✓	4	0	✓	2	0	△	2	0	✓	0	0	✗
#10	2018-11687	BTCR	99	20	4	0	✓	2	0	✓	2	0	△	3	2	✗	0	0	✗
#11	2018-12070	SEC	269	40	8	0	✓	6	0	✓	4	0	✗	3	1	✗	0	0	✗
#12	2018-12230	RMC	161	9	5	0	✓	3	0	✓	5	0	✓	0	0	✗	0	0	✗
#13	2018-13113	ETT	142	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#14	2018-13126	MoxyOnePresale	301	5	3	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#15	2018-13127	DSPX	238	6	4	0	✓	3	0	✓	3	0	△	1	0	✗	0	0	✗
#16	2018-13128	ETY	193	10	4	0	✓	3	0	✓	3	0	△	0	0	✗	0	0	✗
#17	2018-13129	SPX	276	9	6	0	✓	5	0	✓	3	0	△	1	0	✗	internal error		
#18	2018-13131	SpadePreSale	312	4	3	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	2	0	✓	4	2	✓	1	1	✗	0	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#32	2018-13327	CCLAG	92	5	2	0	✓	2	1	✓	2	1	✓	0	0	✗	0	0	✗
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	3	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	13	0	✓	8	0	✓	2	0	✓	1	0	✗	1	0	✗
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	6	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	8	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	5	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	2	1	✗	0	0	✗
#40	2018-13722	HYIPToken	410	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	internal error		
#41	2018-13777	RRToken	166	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	0	0	✗
#42	2018-13778	CGCToken	224	13	6	0	✓	4	0	✓	4	0	✓	1	0	✗	1	0	✗
#43	2018-13779	YLCToken	180	17	11	0	✓	5	0	✓	6	0	✓	0	0	✗	0	0	✗
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	2	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	11	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	3	1	✓	timeout (> 3 days)		
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✗	3	0	△	3	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	2	1	✗	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✗	2	0	△	3	0	✓	1	0	✗
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	0	0	✗
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	3	0	△	0	0	✗
#52	2018-14006	NGT	249	27	13	0	✓	1	0	✗	5	0	△	0	0	✗	timeout (> 3 days)		
#53	2018-14063	TRCT	178	9	1	0	✓	1	0	✓	1	0	✓	4	2	✓	0	0	✗
#54	2018-14084	MKCB	273	17	10	0	✓	5	0	✓	4	0	✗	2	0	✗	1	0	✗
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	0	0	✗
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	0	0	✗	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	14	0	✓	0	0	✗
#58	2018-14576	SunContract	194	12	4	0	✓	1	0	✓	0	0	✗	0	0	✗	0	0	✗
#59	2018-17050	AI	141	8	3	0	✓	1	0	✓	1	0	✓	0	0	✗	0	0	✗
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	4	0	✓	0	0	✗	0	0	✗
Total			12493	976	492	2	✓:58 △: 0 ✗ : 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

			VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	MantiCore [2] #Alarm	MantiCore [2] #FP	MantiCore [2] CVE
Total	12493	976	492	2	✓: 58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9], [26] #Alarm	Oyente [9], [26] #FP	Oyente [9], [26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL						3	0	✓	2	0	△	0	0	✗	0	0	✗
#8	2018-11446	GRX						8	2	✗	12	4	✗	4	2	✗	internal error		
#9	2018-11561	EET						4	0	✓	2	0	△	2	0	✓	0	0	✗
#10	2018-11687	BTC						2	0	✓	2	0	△	3	2	✗	0	0	✗
#11	2018-12070	SEC						6	0	✓	4	0	✗	3	1	✗	0	0	✗
#12	2018-12230	RMC						3	0	✓	5	0	✓	0	0	✗	0	0	✗
#13	2018-13113	ETT						4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#14	2018-13126	Mox						0	0	✗	0	0	✗	0	0	✗	0	0	✗
#15	2018-13127	DSP						3	0	✓	3	0	△	1	0	✗	0	0	✗
#16	2018-13128	ETY						3	0	✓	3	0	△	0	0	✗	0	0	✗
#17	2018-13129	SPX						5	0	✓	3	0	△	1	0	✗	internal error		
#18	2018-13131	SpadePreSale	312	4	3	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	2	0	✓	4	2	✓	1	1	✗	0	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#32	2018-13327	CCLAG	92	5	2	0	✓	2	1	✓	2	1	✓	0	0	✗	0	0	✗
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	3	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	13	0	✓	8	0	✓	2	0	✓	1	0	✗	1	0	✗
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	6	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	8	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	5	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	2	1	✗	0	0	✗
#40	2018-13722	HYIPToken	410	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	internal error		
#41	2018-13777	RRToken	166	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	0	0	✗
#42	2018-13778	CGCToken	224	13	6	0	✓	4	0	✓	4	0	✓	1	0	✗	1	0	✗
#43	2018-13779	YLCToken	180	17	11	0	✓	5	0	✓	6	0	✓	0	0	✗	0	0	✗
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	2	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	11	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	3	1	✓	timeout (> 3 days)		
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✗	3	0	△	3	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	2	1	✗	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✗	2	0	△	3	0	✓	1	0	✗
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	0	0	✗
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	3	0	△	0	0	✗
#52	2018-14006	NGT	249	27	13	0	✓	1	0	✗	5	0	△	0	0	✗	timeout (> 3 days)		
#53	2018-14063	TRCT	178	9	1	0	✓	1	0	✓	1	0	✓	4	2	✓	0	0	✗
#54	2018-14084	MKCB	273	17	10	0	✓	5	0	✓	4	0	✗	2	0	✗	1	0	✗
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	0	0	✗
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	0	0	✗	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	14	0	✓	0	0	✗
#58	2018-14576	SunContract	194	12	4	0	✓	1	0	✓	0	0	✗	0	0	✗	0	0	✗
#59	2018-17050	AI	141	8	3	0	✓	1	0	✓	1	0	✓	0	0	✗	0	0	✗
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	4	0	✓	0	0	✗	0	0	✗
Total			12493	976	492	2	✓:58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

정확도: 99.5%
검출률: 100%

			VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	MantiCore [2] #Alarm	MantiCore [2] #FP	MantiCore [2] CVE
Total	12493	976	492	2	✓: 58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart			Osiris [7]			Oyente [9], [26]			Mythril [8]			ManticCore [10]		
					#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL						3	0	✓	2	0	∠						
#8	2018-11446	GRX						8	2	✗	12	4	✗						
#9	2018-11561	EET						4	0	✓	2	0	∠						
#10	2018-11687	BTC						2	0	✓	2	0	∠						
#11	2018-12070	SEC						6	0	✓	4	0	✗						
#12	2018-12230	RMC						3	0	✓	5	0	✓						
#13	2018-13113	ETT						4	2	N/A	2	2	N/						
#14	2018-13126	Mox						0	0	✗	0	0	✗						
#15	2018-13127	DSP						3	0	✓	3	0	∠						
#16	2018-13128	ETY						3	0	✓	3	0	∠						
#17	2018-13129	SPX						5	0	✓	3	0	∠						
#18	2018-13131	SpadePreSale	312	4	3	0	✓	0	0	✗	0	0	✗	0	0	✗	internal error		
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	2	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	2	0	✓	4	2	✓	1	1	✗	0	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	4	2	N/A	2	2	N/A	0	0	N/A	0	0	N/A
#32	2018-13327	CCLAG	92	5	2	0	✓	2	1	✓	2	1	✓	0	0	✗	0	0	✗
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	3	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	13	0	✓	8	0	✓	2	0	✓	1	0	✗	1	0	✗
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	6	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	8	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	5	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	2	1	✗	0	0	✗
#40	2018-13722	HYIPToken	410	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	internal error		
#41	2018-13777	RRToken	166	8	3	0	✓	2	0	✓	2	0	✓	0	0	✗	0	0	✗
#42	2018-13778	CGCToken	224	13	6	0	✓	4	0	✓	4	0	✓	1	0	✗	1	0	✗
#43	2018-13779	YLCToken	180	17	11	0	✓	5	0	✓	6	0	✓	0	0	✗	0	0	✗
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	2	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	11	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	3	1	✓	timeout (> 3 days)		
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✗	3	0	△	3	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	2	1	✗	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✗	2	0	△	3	0	✓	1	0	✗
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	0	0	✗
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	3	0	△	0	0	✗
#52	2018-14006	NGT	249	27	13	0	✓	1	0	✗	5	0	△	0	0	✗	timeout (> 3 days)		
#53	2018-14063	TRCT	178	9	1	0	✓	1	0	✓	1	0	✓	4	2	✓	0	0	✗
#54	2018-14084	MKCB	273	17	10	0	✓	5	0	✓	4	0	✗	2	0	✗	1	0	✗
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	0	0	✗
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	0	0	✗	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	14	0	✓	0	0	✗
#58	2018-14576	SunContract	194	12	4	0	✓	1	0	✓	0	0	✗	0	0	✗	0	0	✗
#59	2018-17050	AI	141	8	3	0	✓	1	0	✓	1	0	✓	0	0	✗	0	0	✗
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	4	0	✓	0	0	✗	0	0	✗
	Total		12493	976	492	2	✓:58 △: 0 ✗ : 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

정확도: 99.5%
검출률: 100%

정확도: < 94.6%
검출률: < 70.7%

			VeriSmart			Osiris [43]			Oyente [9, 34]			Mythril [7]			ManticCore [2]		
			#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE
Total	12493	976	492	2	✓: 58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

기존 취약점 검증기와 성능 비교

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

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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 }
```

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션의 불변 성질을 이용한 안전성 증명

```
require (balance[msg.sender] >= value);
balance[msg.sender] -= value;
totalSupply -= value;
```

assert (totalSupply >= value)

totalSupply = ∑balance … transaction invariant

≥ balance[msg.sender] … def. of ∑balance

≥ value … assumption (require)

VeriSmart 핵심 차별점

- 트랜잭션의 불변 성질을 이용한 안전성 증명

```
require (balance[msg.sender] >= value);
balance[msg.sender] -= value;
totalSupply -= value;
```

assert (totalSupply >= value)

totalSupply = ∑balance … transaction invariant

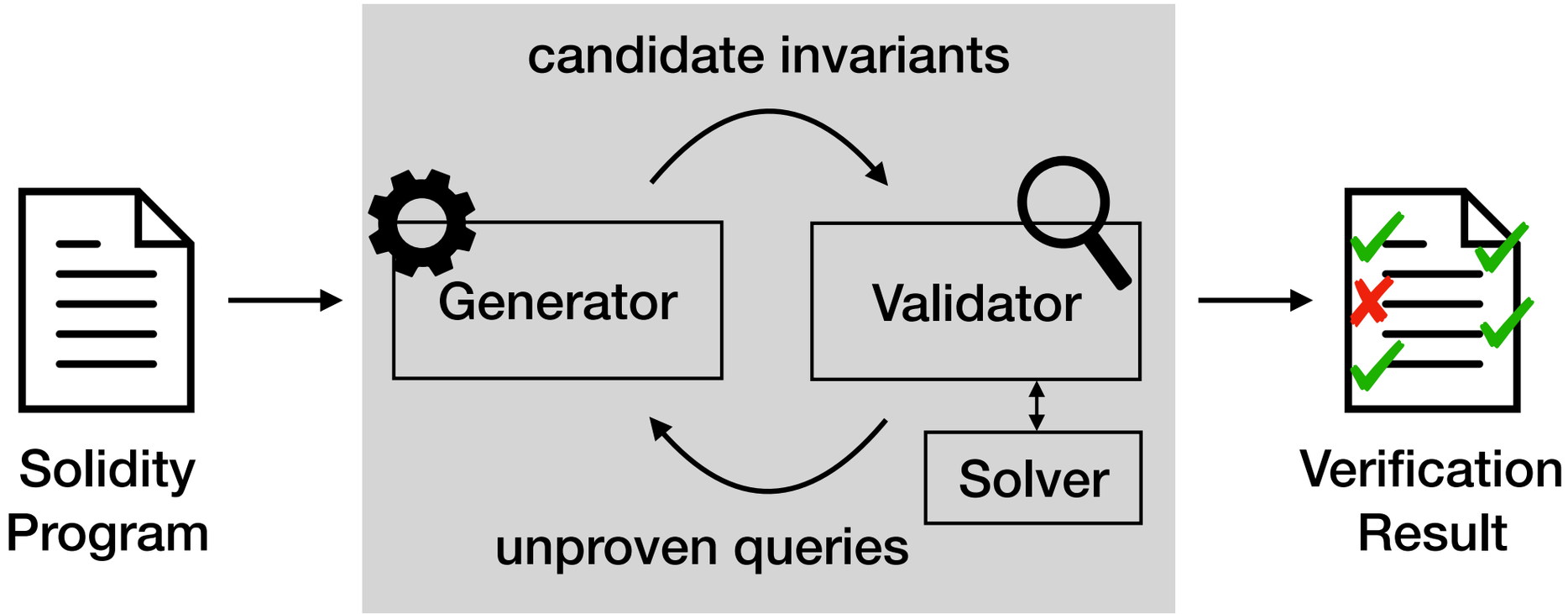
≥ balance[msg.sender] … def. of ∑balance

≥ value … assumption (require)

기존 취약점 검출기 / 검증기들은 불변 성질 추론을 못하여 FN / FP 발생

VeriSmart 검증 알고리즘

- Invariant Generator: 트랜잭션 불변 성질을 추론 시도
- Validator: 추론된 불변 성질을 이용하여 안전성 검증 시도



SmarTest: 오류 검출기

```
1   contract SocialChain {
2     uint totalSupply;
3     mapping(address=>uint) balance;
4     mapping(address=>mapping(address=>uint)) allowance;
5
6     constructor (uint initialSupply) {
7       totalSupply = initialSupply;
8       balance[msg.sender] = initialSupply;
9     }
10
11    function transfer (address to, uint value)
12    public returns(bool) {
13      require (balance[msg.sender] >= value);
14      balance[msg.sender] -= value;
15      balance[to] += value;
16      return true;
17    }
18
19    function approve (address spender, uint value)
20    public returns(bool) {
21      allowance[msg.sender][spender] = value;
22      return true;
23    }
24
25    function transferFrom (address from, address to,
26          uint value) public returns (bool) {
27      require (balance[from] >= value);
28      require (balance[to] + value > balance[to]);
29      require (allowance[from][msg.sender] >= value);
30      balance[from] -= value;
31      balance[to] += value;
32      allowance[from][msg.sender] += value;
33      return true;
34    }
35  }
```

safe

safe

unsafe (may overflow)

SmarTest: 오류 검출기

```
1   contract SocialChain {
2     uint totalSupply;
3     mapping(address=>uint) balance;
4     mapping(address=>mapping(address=>uint)) allowance;
5
6     constructor (uint initialSupply) {
7       totalSupply = initialSupply;
8       balance[msg.sender] = initialSupply;
9     }
10
11    function transfer (address to, uint value)
12    public returns(bool) {
13      require (balance[msg.sender] >= value);
14      balance[msg.sender] -= value;
15      balance[to] += value;
16      return true;
17    }
18
19    function approve (address spender, uint value)
20    public returns(bool) {
21      allowance[msg.sender][spender] = value;
22      return true;
23    }
24
25    function transferFrom (address from, address to,
26          uint value) public returns (bool) {
27      require (balance[from] >= value);
28      require (balance[to] + value > balance[to]);
29      require (allowance[from][msg.sender] >= value);
30      balance[from] -= value;
31      balance[to] += value;
32      allowance[from][msg.sender] += value;
33      return true;
34    }
35  }
```

safe

safe

unsafe (may overflow)

오류 시나리오

with msg.sender = A

constructor(0x8800…00)

with msg.sender = A

approve(B, 0x8100..00)

with msg.sender = B

transferFrom(A,C,0x7f00..00)

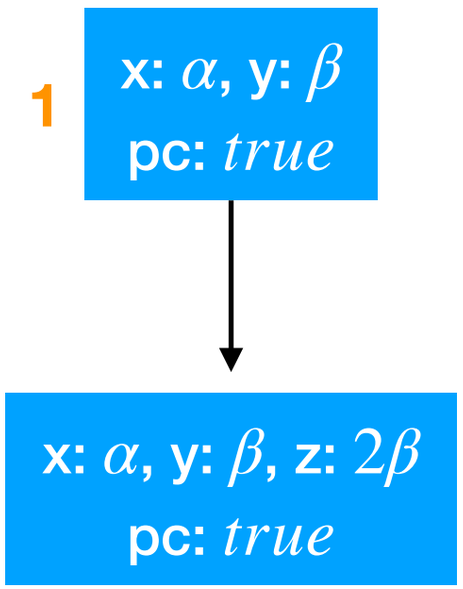
기호 실행 (Symbolic Execution)

1 x: α, y: β
pc: $true$

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
   z := double (y);
2
   if (z==x) {
    3
     if (x>y+10) {
      4 Error;
     } else { 5 …}
   }
6 }
```

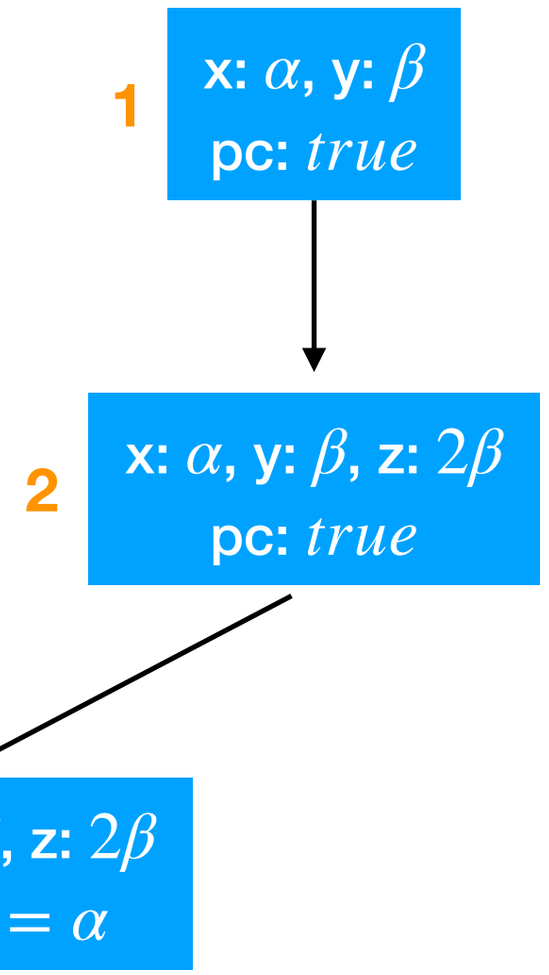
기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```



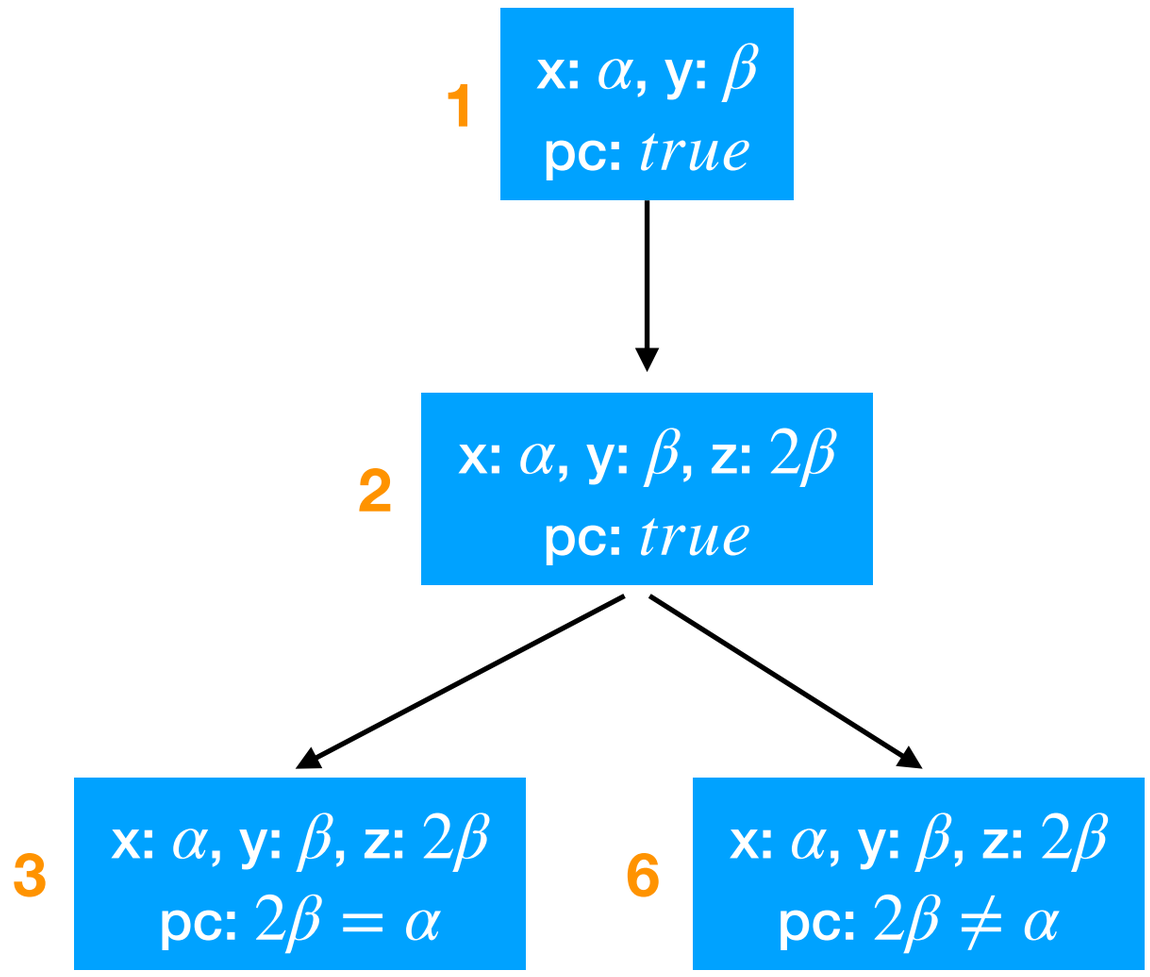
기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```



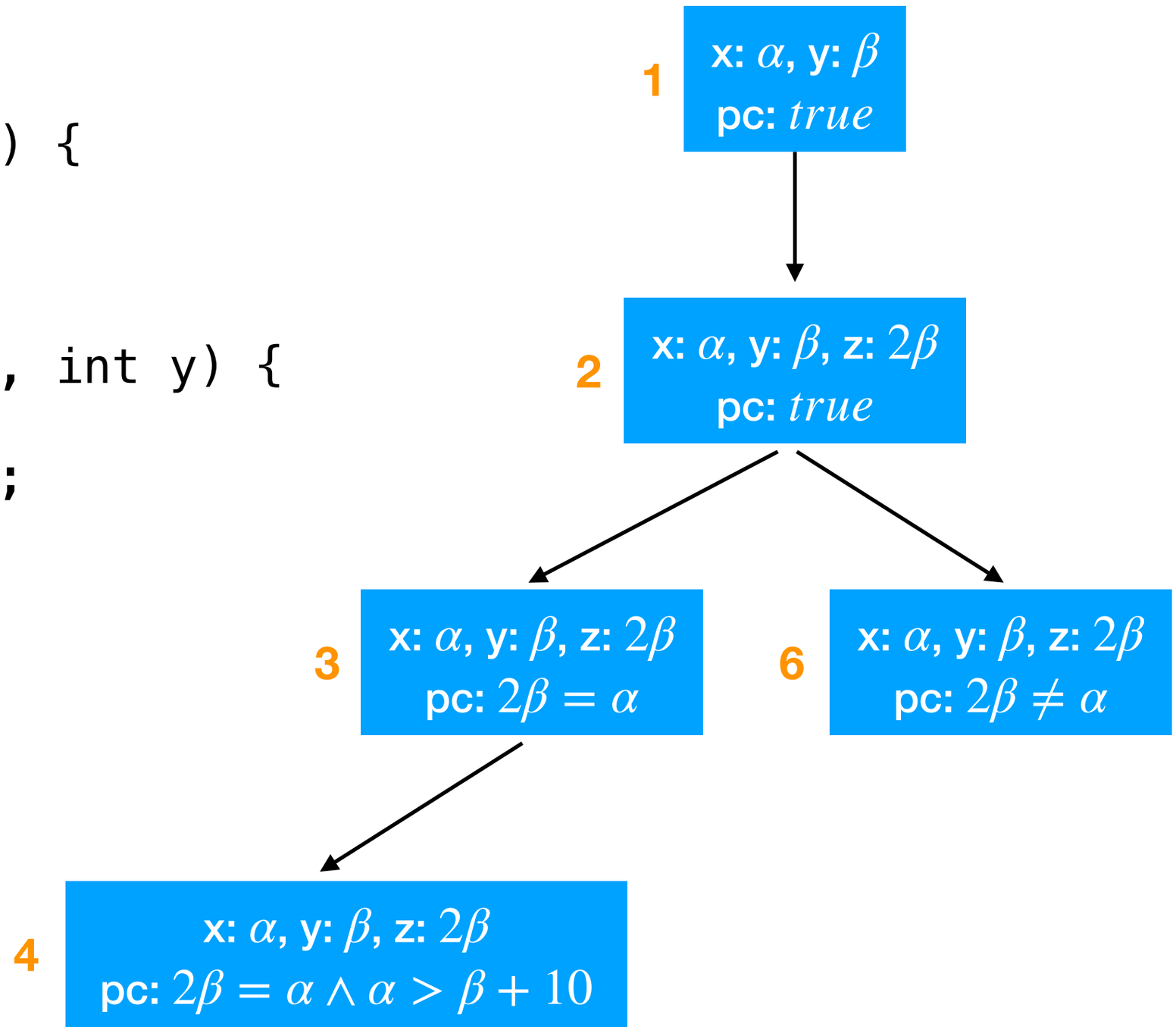
기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```



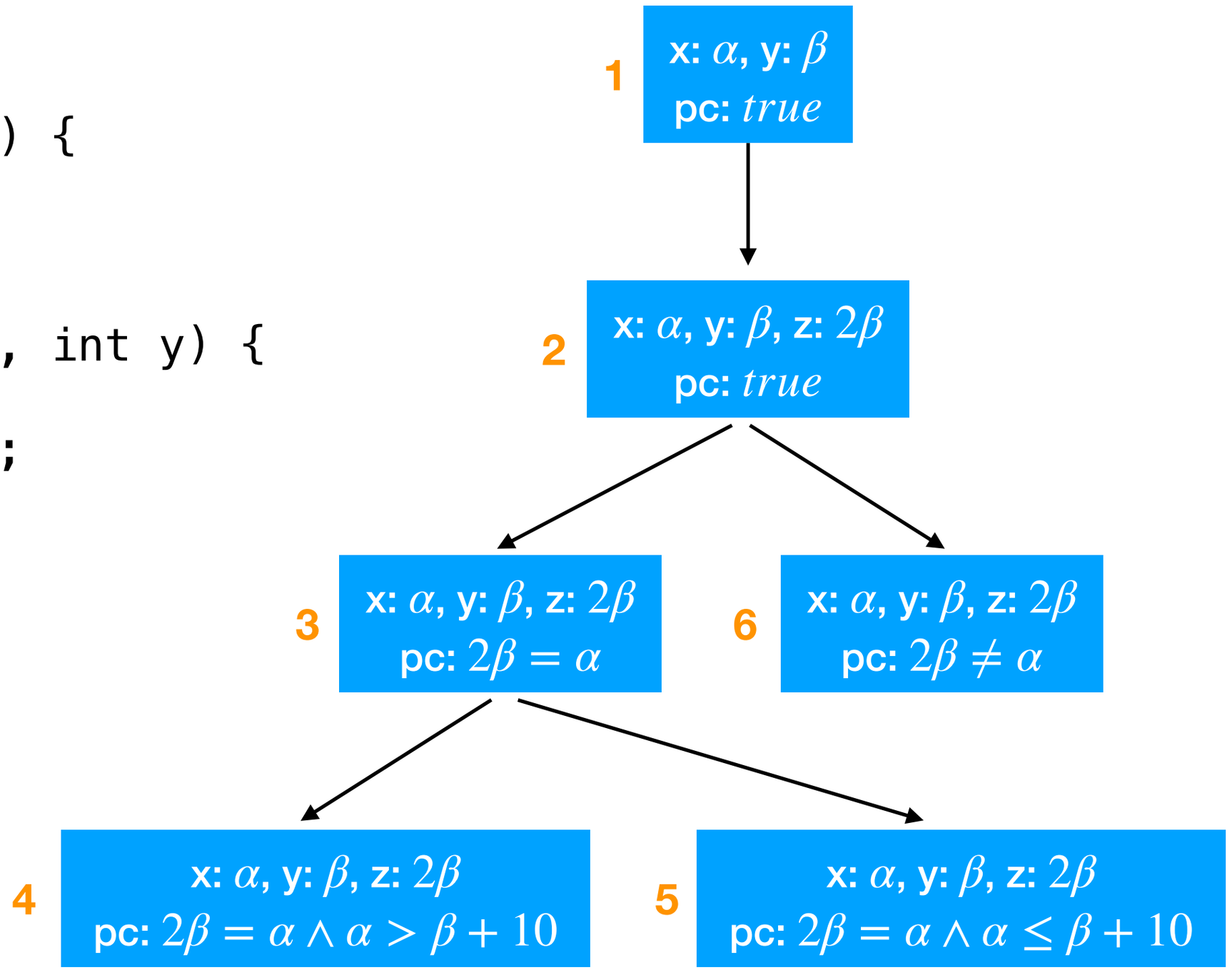
기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```



기호 실행 (Symbolic Execution)

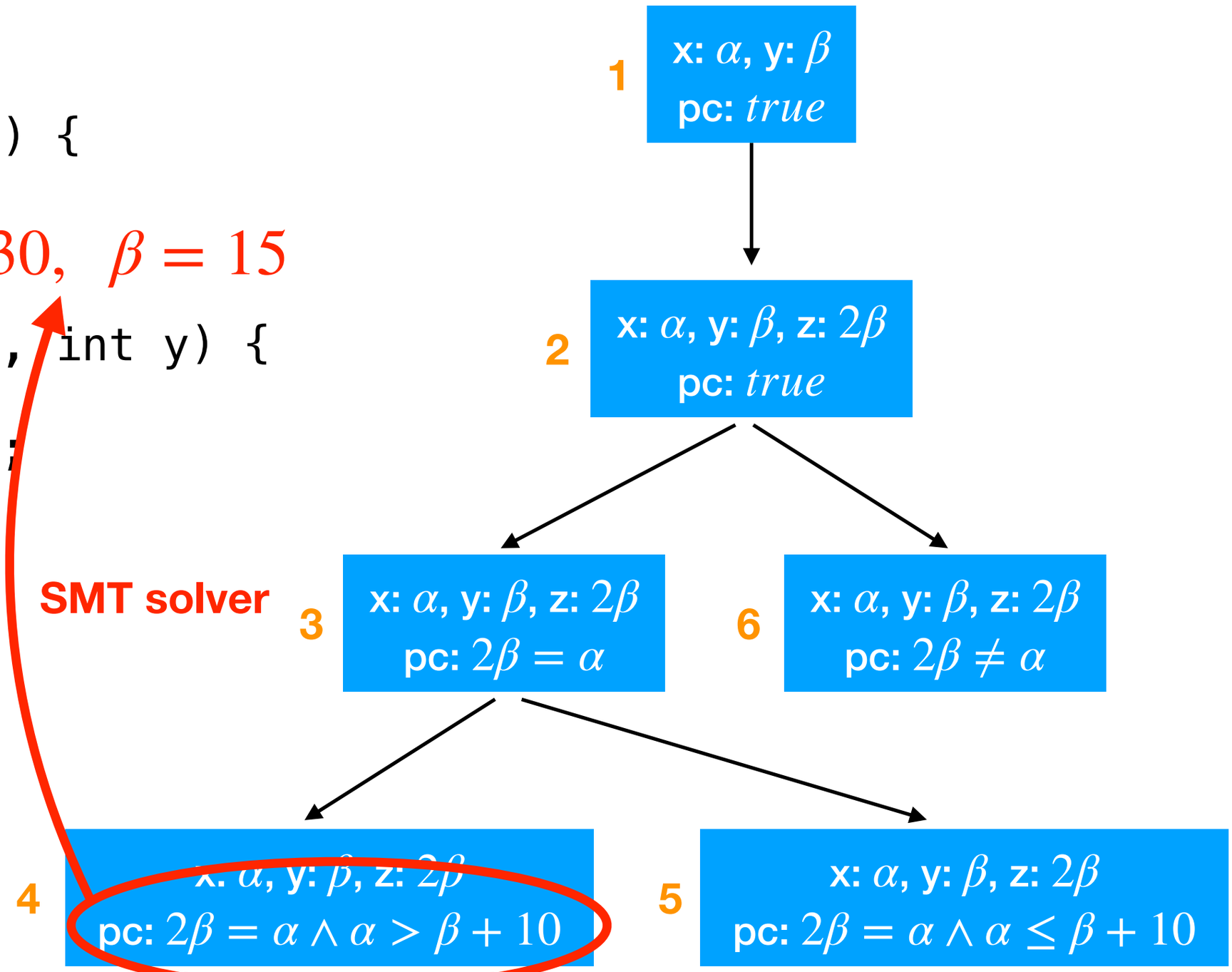
```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```



기호 실행 (Symbolic Execution)

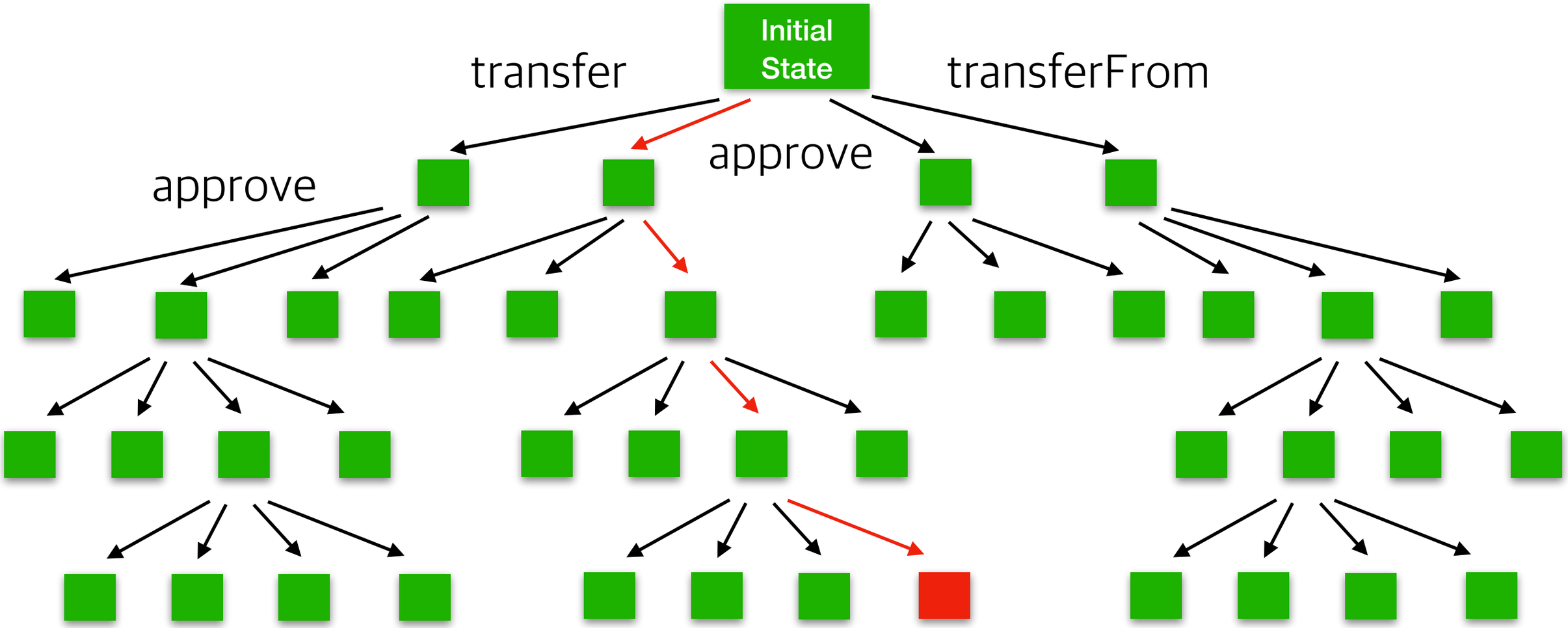
```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
1
  z := double (y);
2
  if (z==x) {
   3
    if (x>y+10) {
     4 Error;
    } else { 5 …}
  }
6 }
```

$\alpha = 30, \quad \beta = 15$



Path Explosion

- 트랜잭션들이 만들어내는 경우의 수가 폭발



Language Model-guided Symbolic Execution

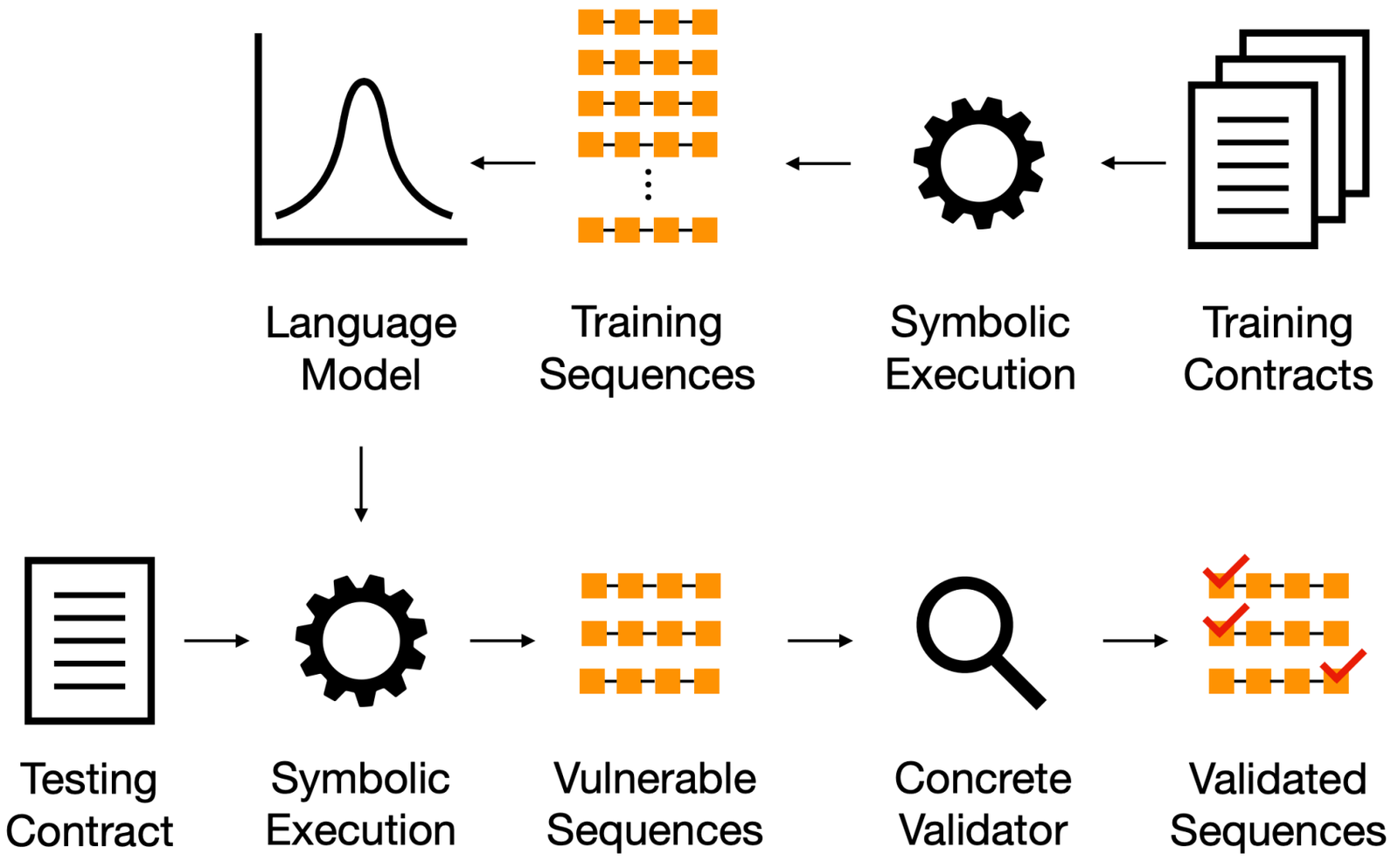
transfer ⟶ approve ⟶ transferFrom ⟶ ⋯ 80%

approve ⟶ transferFrom ⟶ transfer ⟶ ⋯ 20%

Language Model-guided Symbolic Execution

transfer ⟶ approve ⟶ transferFrom ⟶ ⋯ 80%

approve ⟶ transferFrom ⟶ transfer ⟶ ⋯ 20%



Performance

Tool	Integer Overflow	Division by Zero	Assertion Violation	ERC20 Violation	CVE Vulnerabilities
SmarTest	1982	203	77	654	219
Mythril (ConsenSys)	460	73	25	n/a	85
Manticore (Trail of Bits)	2	1	3	n/a	0

SmartFix: 오류 자동 수정기

```
1   contract SocialChain {
2     uint totalSupply;
3     mapping(address=>uint) balance;
4     mapping(address=>mapping(address=>uint)) allowance;
5
6     constructor (uint initialSupply) {
7       totalSupply = initialSupply;
8       balance[msg.sender] = initialSupply;
9     }
10
11    function transfer (address to, uint value)
12    public returns(bool) {
13      require (balance[msg.sender] >= value);
14      balance[msg.sender] -= value;
15      balance[to] += value;
16      return true;
17    }
18
19    function approve (address spender, uint value)
20    public returns(bool) {
21      allowance[msg.sender][spender] = value;
22      return true;
23    }
24
25    function transferFrom (address from, address to,
26          uint value) public returns (bool) {
27      require (balance[from] >= value);
28      require (balance[to] + value > balance[to]);
29      require (allowance[from][msg.sender] >= value);
30      balance[from] -= value;
31      balance[to] += value;
32      allowance[from][msg.sender] += value;
33      return true;
34    }
35  }
```

safe

safe

unsafe (may overflow)

오류 시나리오

with msg.sender = A

constructor(0x8800…00)

with msg.sender = A

approve(B, 0x8100..00)

with msg.sender = B

transferFrom(A,C,0x7f00..00)

SmartFix: 오류 자동 수정기

```
1  contract SocialChain {
2    uint totalSupply;
3    mapping(address=>uint) balance;
4    mapping(address=>mapping(address=>uint)) allowance;
5
6    constructor (uint initialSupply) {
7      totalSupply = initialSupply;
8      balance[msg.sender] = initialSupply;
9    }
10
11   function transfer (address to, uint value)
12   public returns(bool) {
13     require (balance[msg.sender] >= value);
14     balance[msg.sender] -= value;
15     balance[to] += value;
16     return true;
17   }
18
19   function approve (address spender, uint value)
20   public returns(bool) {
21     allowance[msg.sender][spender] = value;
22     return true;
23   }
24
25   function transferFrom (address from, address to,
26         uint value) public returns (bool) {
27     require (balance[from] >= value);
28     require (balance[to] + value > balance[to]);
29     require (allowance[from][msg.sender] >= value);
30     balance[from] -= value;
31     balance[to] += value;
32     allowance[from][msg.sender] += value;
33     return true;
34   }
35 }
```

safe

safe

unsafe (may overflow)

+= should be −=

패치

오류 시나리오

with msg.sender = A

constructor(0x8800…00)

with msg.sender = A

approve(B, 0x8100..00)

with msg.sender = B

transferFrom(A,C,0x7f00..00)

SmartFix 목표: 취약점을 자동으로 안전하게 수정

CVE-2018-11411

```
1   function transferFrom (address from, address to, uint value) returns (bool success) {
2     if (value == 0) return false;
3     uint fromBalance = balance[from];
4     uint allowance = allowed[from][msg.sender];
5
6     bool sufficientFunds = fromBalance <= value;
7     bool sufficientAllowance = allowance <= value;
8     bool overflowed = balance[to] + value > balance[to];
9
10     if(sufficientFunds && sufficientAllowance && !overflowed) {
11      balance[to] += value;                    // overflow
12      balance[from] -= value;                  // underflow
13      allowed[from][msg.sender] -= value; // underflow
14      return true;
15    }
16    else {return false;}
17  }
```

SmartFix

Line 6 : Replace <= by >=
Line 7 : Replace <= by >=
Line 8 : Replace > by <

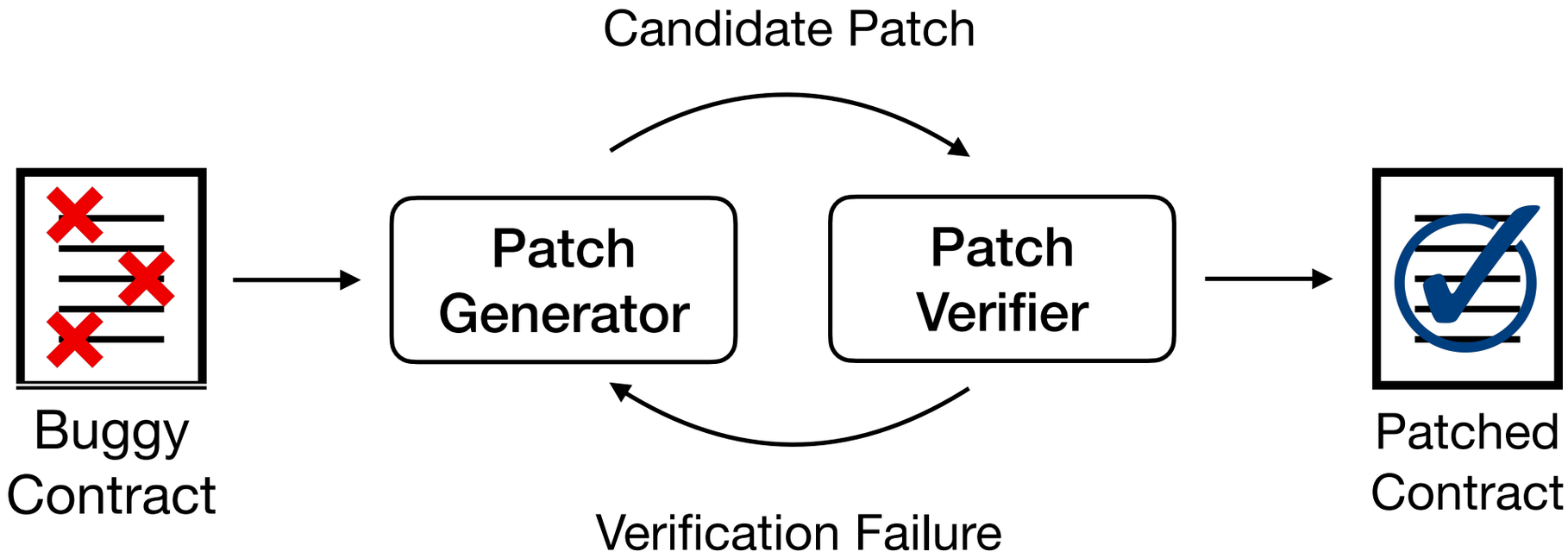
기존 기술: 단순한 패치만 가능

```
1   function transferFrom (address from, address to, uint value) returns (bool success) {
2     if (value == 0) return false;
3     uint fromBalance = balance[from];
4     uint allowance = allowed[from][msg.sender];
5
6     bool sufficientFunds = fromBalance <= value;
7     bool sufficientAllowance = allowance <= value;
8     bool overflowed = safeAdd(balance[to],value) > balance[to];
9
10     if(sufficientFunds && sufficientAllowance && !overflowed) {
12       balance[to] = safeAdd(balance[to],value);
11       balance[from] = safeSub(balance[from],value);
13       allowed[from][msg.sender] = safeSub(allowed[from][msg.sender],value);
14       return true;
15     }
16     else {return false;}
17  }
```

sGuard [IEEE S&P '21], SmartShield, Elysium:
Runtime check 삽입에만 의존

Approach

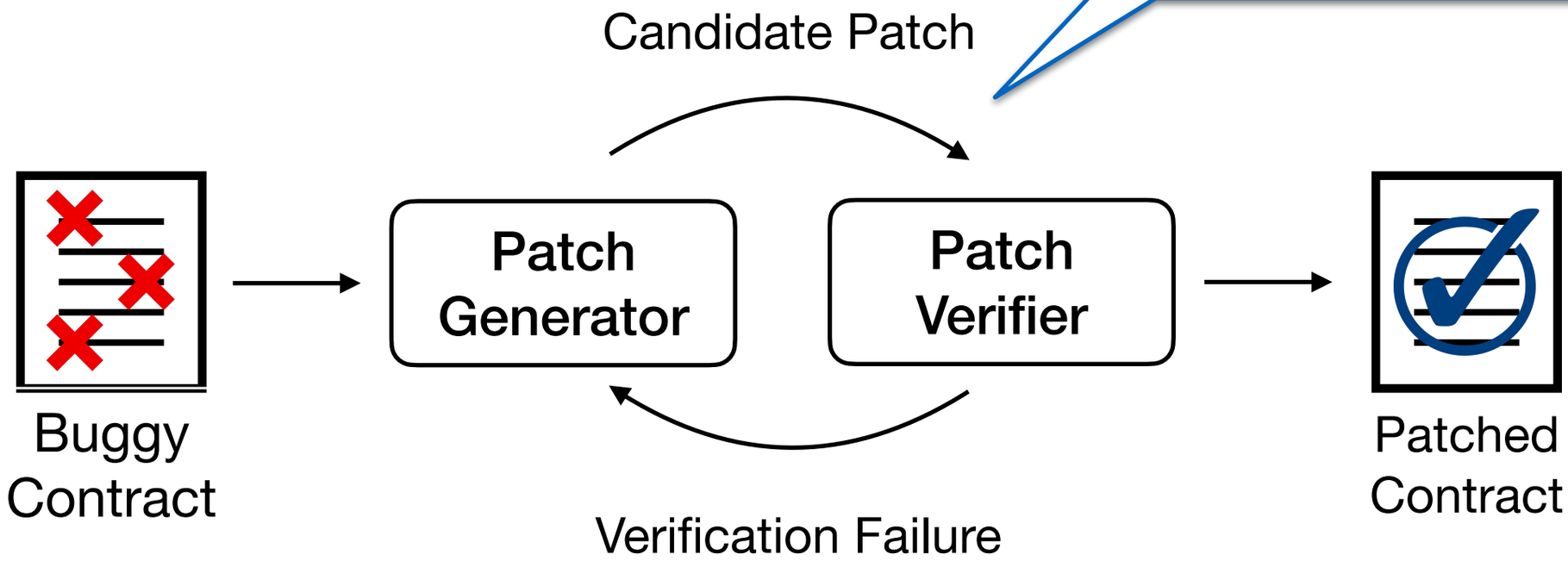
- (패치 후보 나열; 패치 후보 검증)*



Approach

- (패치 후보 나열; 패치 후보 검증)*

온/오프라인 학습을 통해 가속



Performance

Bug Type	#Bug	SmartFix		sGuard [IEEE S&P '21]	
		Succ. Rate	Accuracy	Succ. Rate	Accuracy
IO	229	93.9%	98.6%	60.8%	100%
RE	52	90.2%	100.0%	87.9%	87.9%
TX	12	100.0%	100.0%	100.0%	100.0%
EL	138	57.8%	94.0%	n/a	n/a
SU	53	70.6%	90.0%	n/a	n/a
IO+RE+TX	293	**93.5%**	**98.9%**	**65.5%**	**97.1%**

마무리

- 스마트 컨트랙트는 배포 전 안전성 검증이 필수
- **목표**: 안전한 스마트 컨트랙트 개발을 돕기 위한 **검증, 테스팅, 수정 기술**
  - **VeriSmart**: 스마트 컨트랙트 검증기
  - **SmarTest**: 스마트 컨트랙트 테스팅 도구
  - **SmartFix**: 스마트 컨트랙트 수정 도구

소스코드: https://github.com/kupl/VeriSmart-public
벤치마크: https://github.com/kupl/VeriSmart-benchmarks

감사합니다!