VeriSmart
스마트 컨트랙트 안전성 검증기

오학주

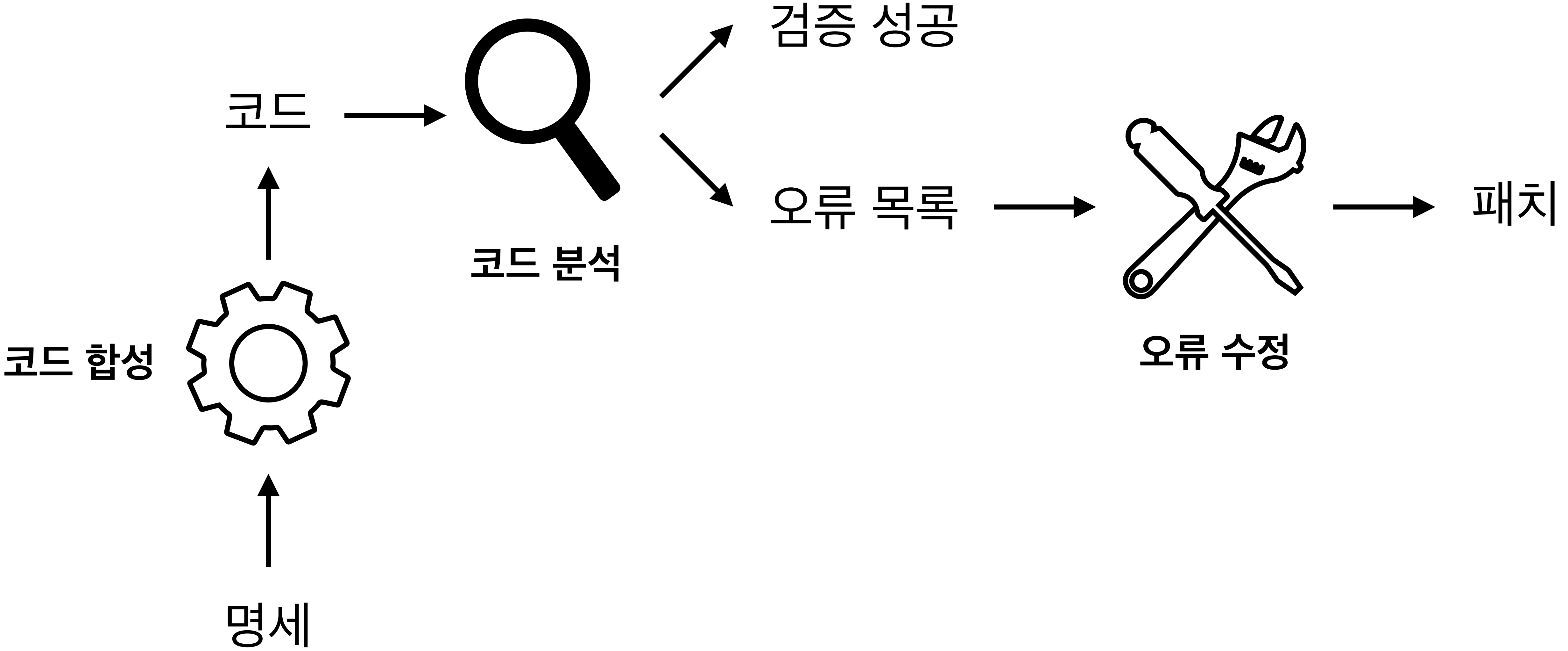
고려대학교 정보대학 컴퓨터학과

10 Dec 2019 @KAIST 정보보호대학원

연구 분야

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?

- A) 소프트웨어 **자동 분석**, **패치**, **합성** 기술



명세

코드 합성

코드

코드 분석

검증 성공

오류 목록

오류 수정

패치

스마트 컨트랙트

블록체인 1.0

코인 거래만 가능

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

코인 거래만 가능

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

코인 거래만 가능

vs.

블록체인 2.0

임의의 거래가 가능

스마트 컨트랙트

블록체인 1.0

코인 거래만 가능

vs.

블록체인 2.0

임의의 거래가 가능

Key: 스마트 컨트랙트

스마트 컨트랙트 생김새

```
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;
  }
}
```

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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;
  }
}
```

사용자의 계좌 정보

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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;
  }
}
```

사용자의 계좌 정보

데이터

생성자

송금

함수

함수

스마트 컨트랙트 생김새

```
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;
  }
}
```

사용자의 계좌 정보

송금

잔고가 충분하면

데이터

생성자

함수

함수

스마트 컨트랙트 생김새

```
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;
  }
}
```

사용자의 계좌 정보

데이터

생성자

송금

잔고가 충분하면

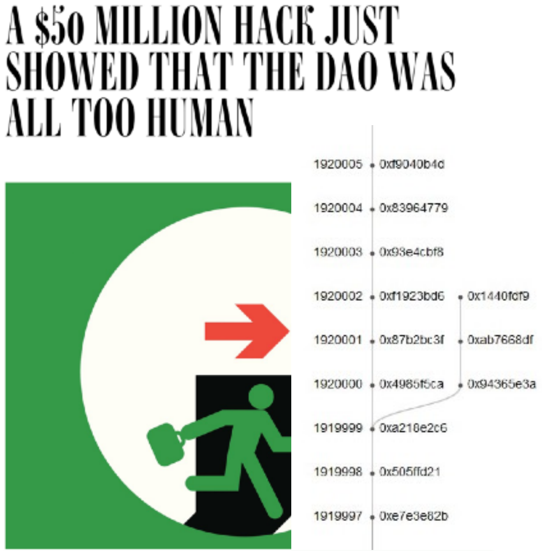
거래를 실행

함수

함수

스마트 컨트랙트의 위험성

- 스마트 컨트랙트는 매우 엄밀한 수준의 안전성 검증이 필요
    - 공격에 성공하면 막대한 금전적 피해가 발생
    - 누구나 온라인에서 소스코드 열람 가능하지만 수정 불가



A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN

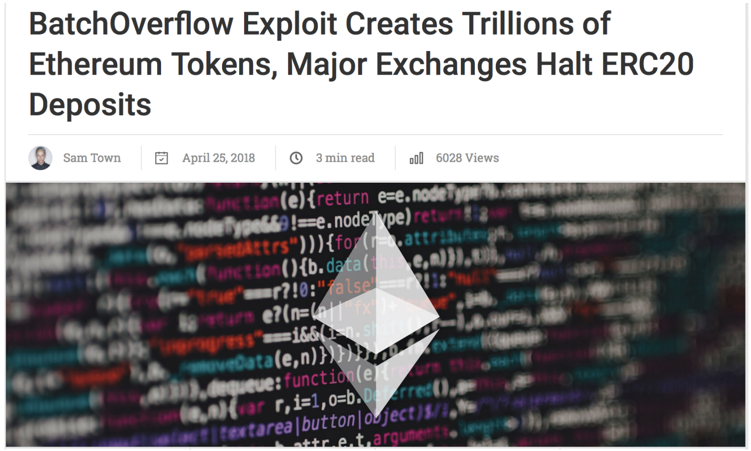
The DAO (2016)
750억원

Altcoin News Blockchain News Ethereum News

Hackers Seize $32 Million in Ethereum in Parity Wallet Bre

Josiah Wilmoth on 20/07/2017

Parity Wallet (2017)
350억원



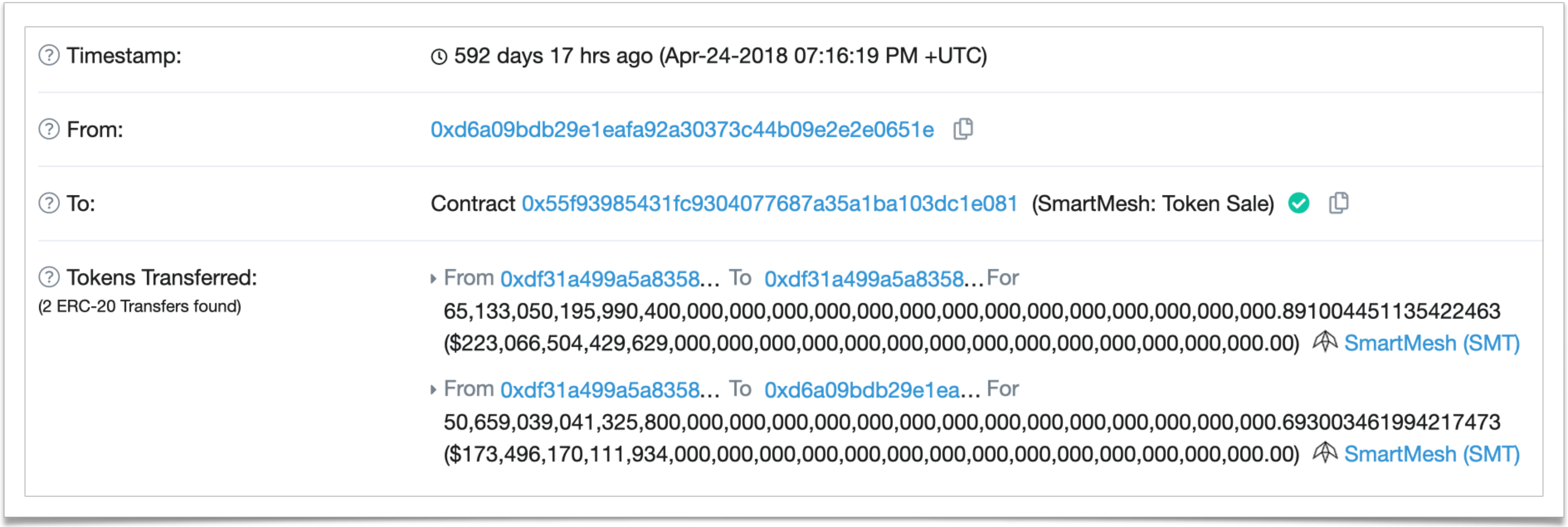
BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

Sam Town April 25, 2018 3 min read 6028 Views

SmartMesh (2018)
천문학적 금액 인출 시도

SmartMesh 사례 (2018)

- SmartMesh 토큰 스마트 컨트랙트의 정수 오버플로우 취약점 (CVE-2018-10376)을 이용하여 천문학적 금액의 토큰을 생성

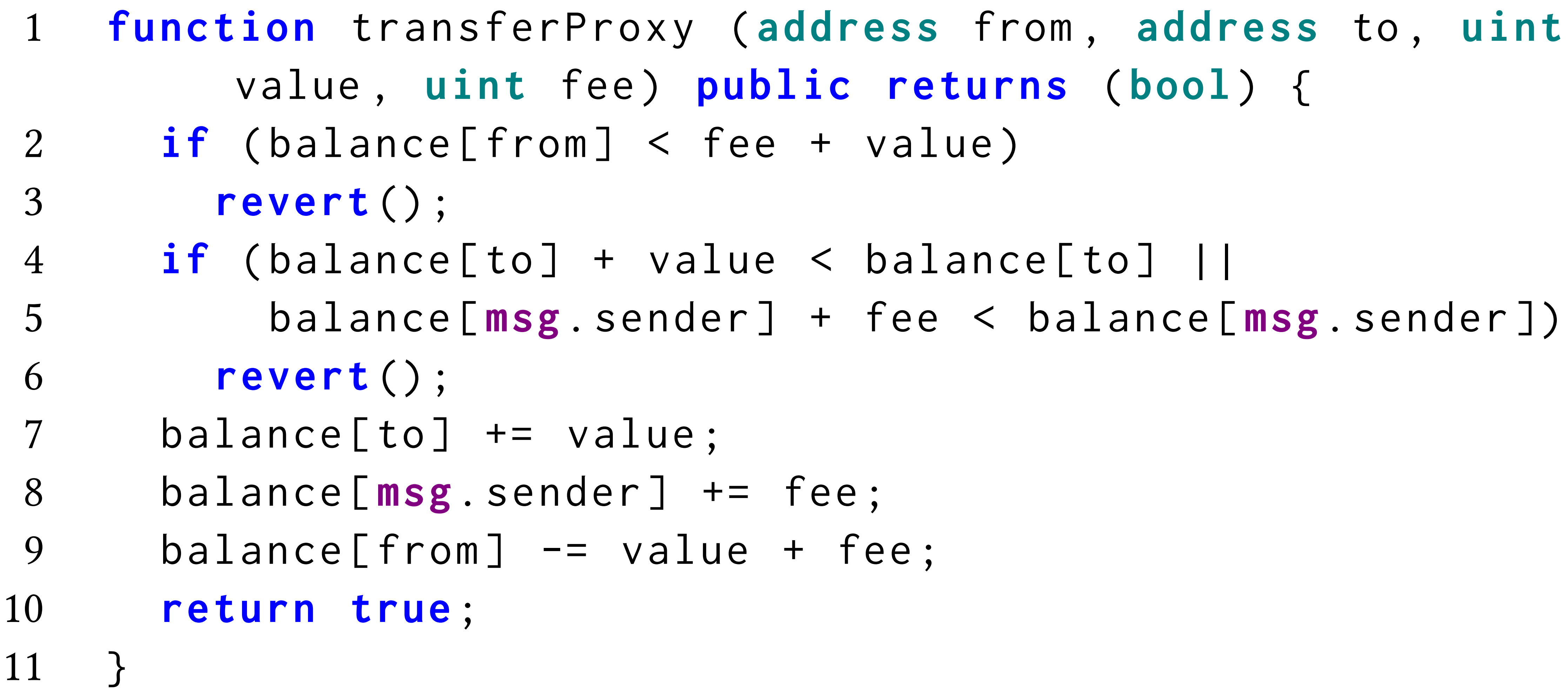


? Timestamp:	592 days 17 hrs ago (Apr-24-2018 07:16:19 PM +UTC)
? From:	0xd6a09bdb29e1eafa92a30373c44b09e2e2e0651e
? To:	Contract 0x55f93985431fc9304077687a35a1ba103dc1e081 (SmartMesh: Token Sale) ✓
? Tokens Transferred: (2 ERC-20 Transfers found)	‣ From 0xdf31a499a5a8358... To 0xdf31a499a5a8358...For 65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463 ($223,066,504,429,629,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) 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 ($173,496,170,111,934,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) SmartMesh (SMT)

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

SmartMesh 사례 (2018)

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



```
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;
}
```

SmartMesh 사례 (2018)

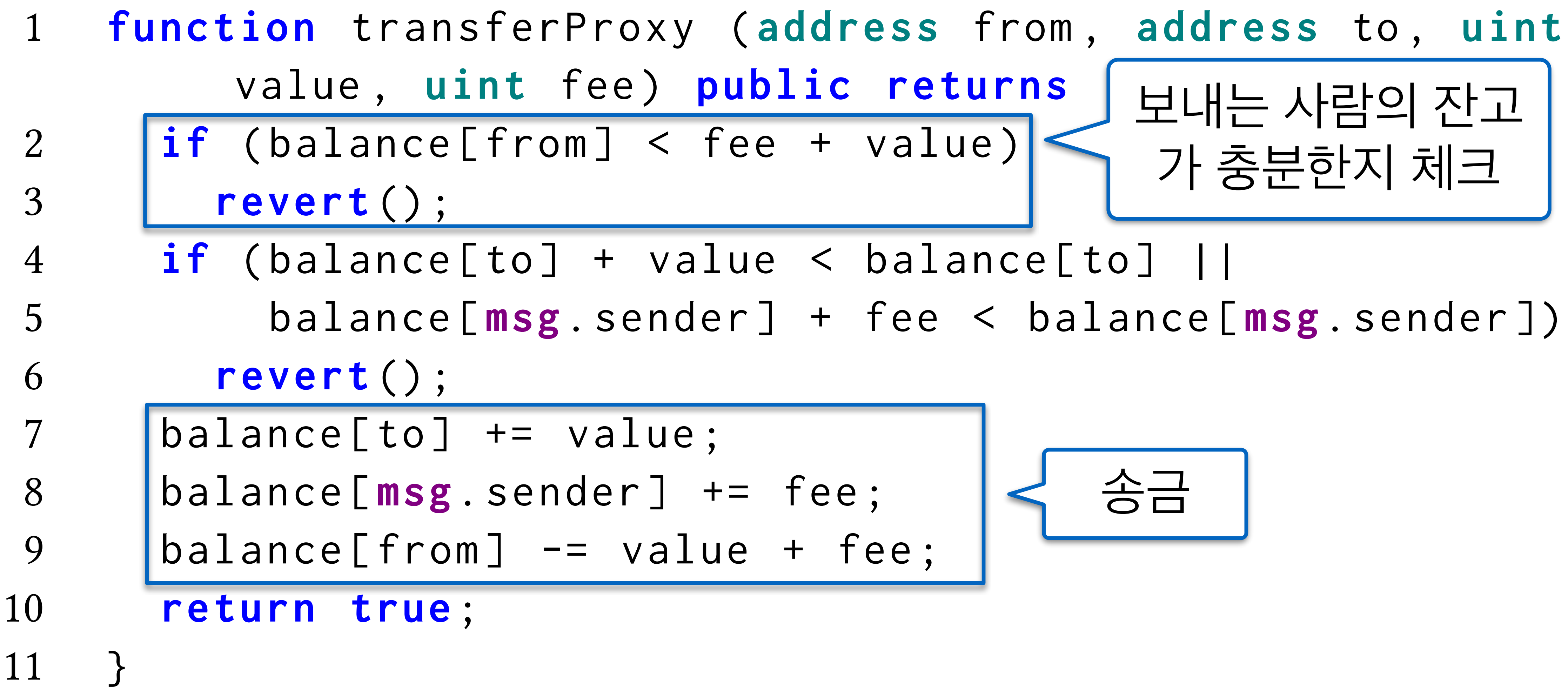
- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
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;
}
```

송금

SmartMesh 사례 (2018)

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



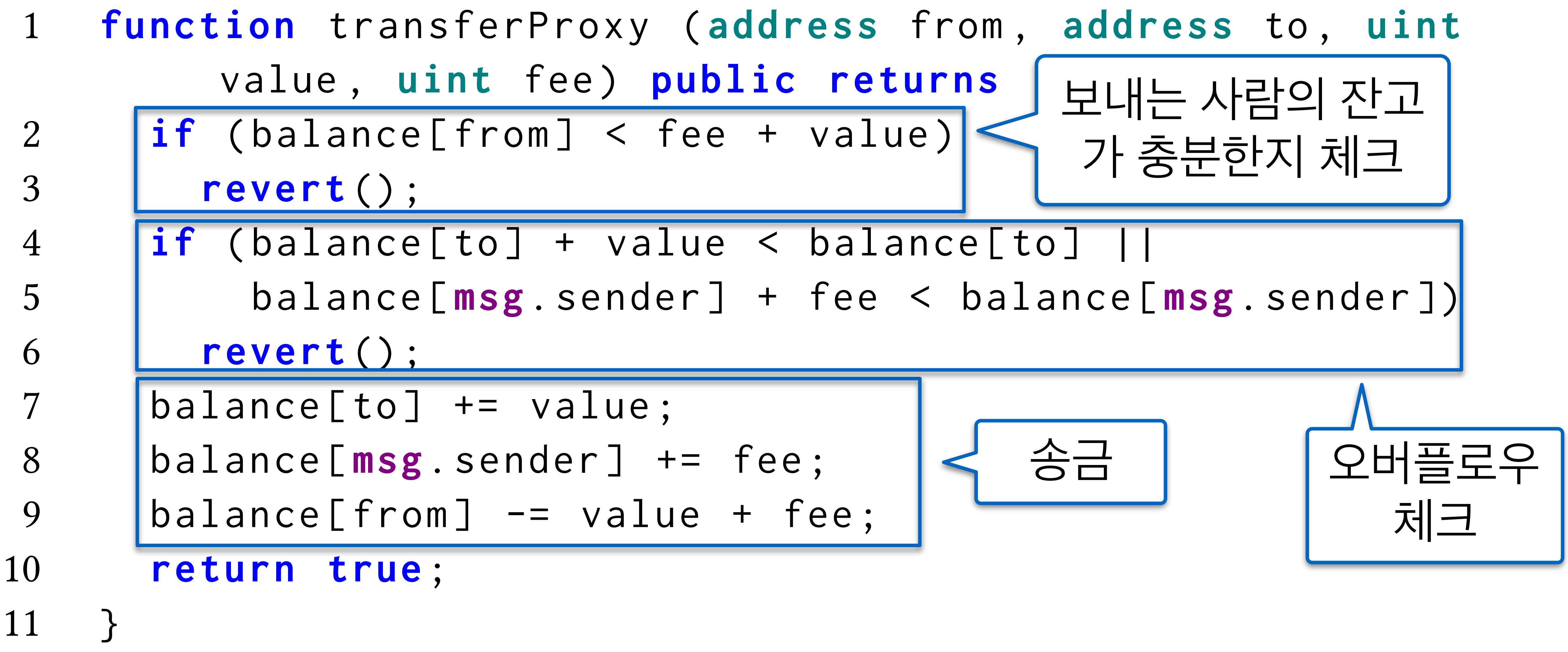
```
function transferProxy (address from, address to, uint
     value, uint fee) public returns
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender]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

보내는 사람의 잔고가 충분한지 체크

송금

SmartMesh 사례 (2018)

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



```
function transferProxy (address from, address to, uint
      value, uint fee) public returns
  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;
}
```

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

SmartMesh 사례 (2018)

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

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns
  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;
}
```

보내는 사람의 잔고가 충분한지 체크

오버플로우 체크

송금

(실질적) 오버플로우/언더플로우 발생하지 않음

SmartMesh 사례 (2018)

```
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;
}
```

SmartMesh 사례 (2018)

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

```
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;
}
```

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
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;
}
```

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
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;
}
```

0!

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (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;
}
```

false

0!

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    (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;
}
```

false

0!

false

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
       (balance[from] < fee + value)
      revert();
       (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;
}
```

false

0!

false

8fffff⋯ff

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    (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;
}
```

false

0!

false

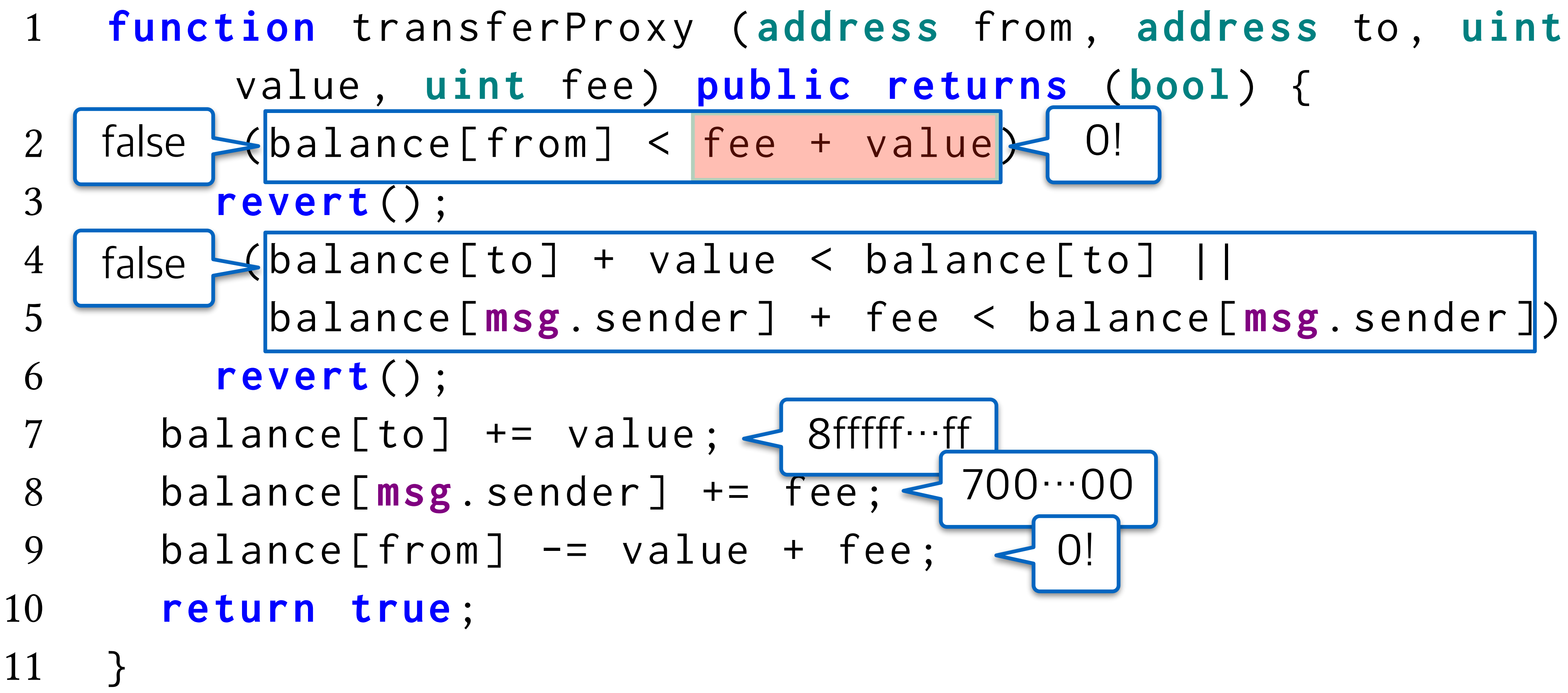
8fffff⋯ff

700⋯00

SmartMesh 사례 (2018)

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

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```



```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
      (balance[from] < fee + value)
    revert();
      (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;
}
```

false

0!

false

8fffff…ff

700…00

0!

목표: 정수 오버플로우 취약점 검증

- Solidity에서는 정수를 256비트로 표현

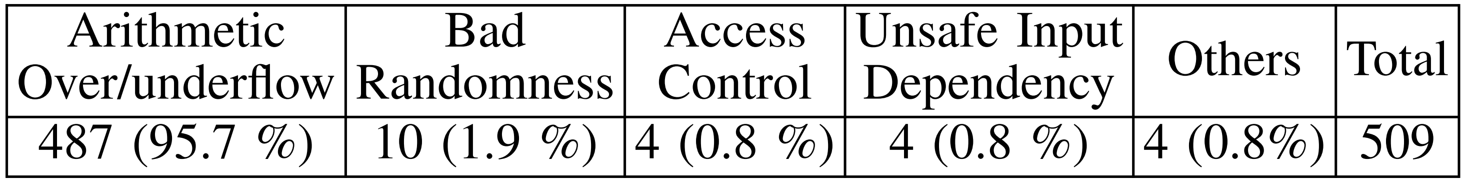
```
uint public totalSupply;
```

- 정수 연산시 표현 가능한 범위를 넘어서는지 여부를 검증

```
totalSupply += value;    balance[msg.sender] -= value;
```

- 사람이 오버플로우 유무를 판단하기는 매우 까다로움

- CVE 등록된 취약점 대부분이 정수 오버플로우에서 비롯



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

(2019.05)

스마트 컨트랙트 자동 분석 기술

- 오류 검출기 (bug-detector)

Oyente

Mythril

manticore

Osiris

- 오류 검증기 (verifier)

Using Solidity's SMTChecker

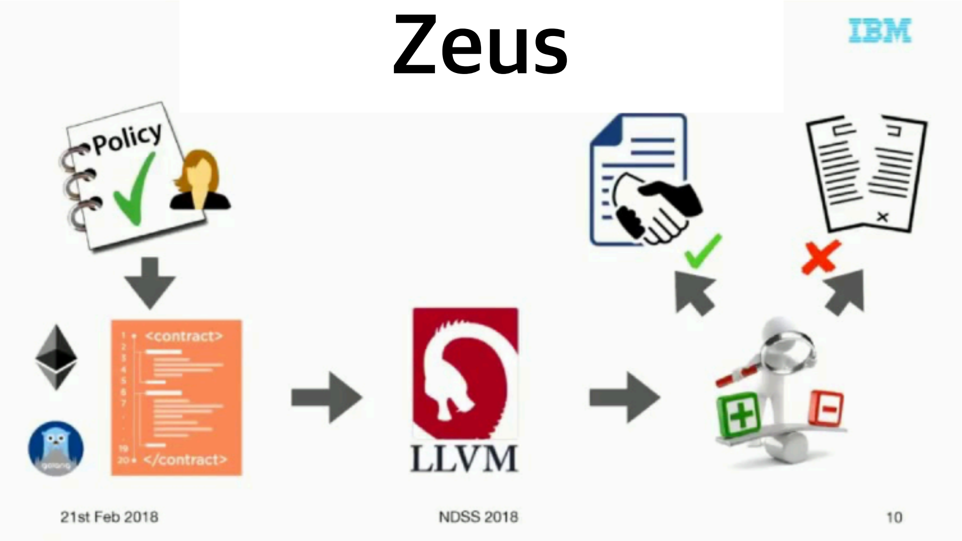
Leonardo Alt

Ethereum Foundation

leonardoalt

leo@ethereum.org

leonardoalt



Zeus

IBM

21st Feb 2018

NDSS 2018

10

현재 자동 분석 기술의 한계 (1)

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

```
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;
}
```

Osiris만 검출 가능

CVE-2018-10376

현재 자동 분석 기술의 한계 (1)

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

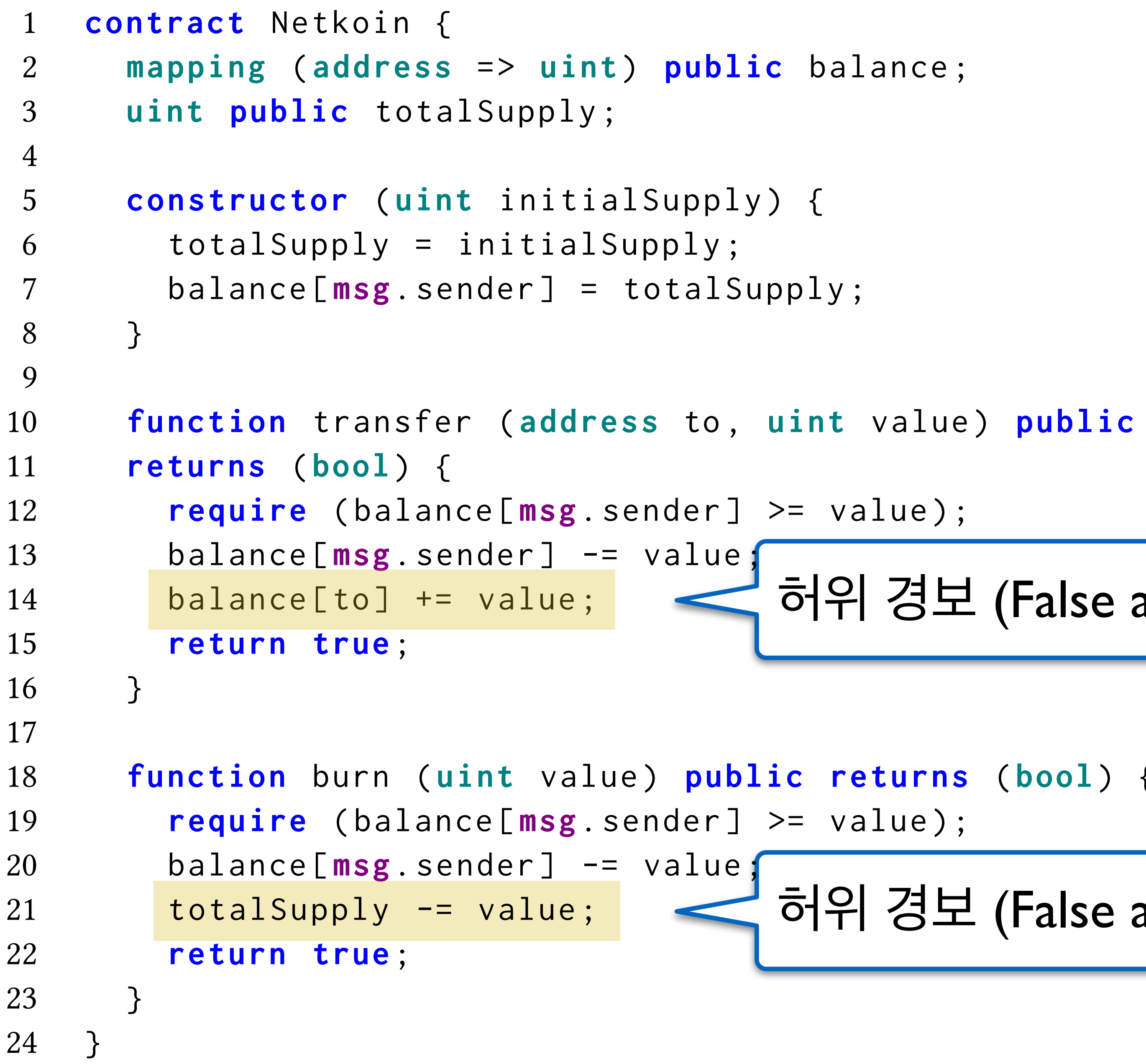
```
function multipleTransfer(address[] to, uint value) {
 require(value * to.length > 0);
 require(balances[msg.sender] >= value * to.length);
 balances[msg.sender] -= value * to.length;
 for (uint i = 0; i < to.length; ++i) {
   balances[to[i]] += value;
 }
 return true;
}
```

앞의 경우와 비슷한 오류
이지만 검출 모두 실패

CVE-2018-14006

현재 자동 분석 기술의 한계 (2)

- 오류 검증기(SMTChecker, Zeus): 허위경보 존재



```
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;
  }
}
```

허위 경보 (False alarm)

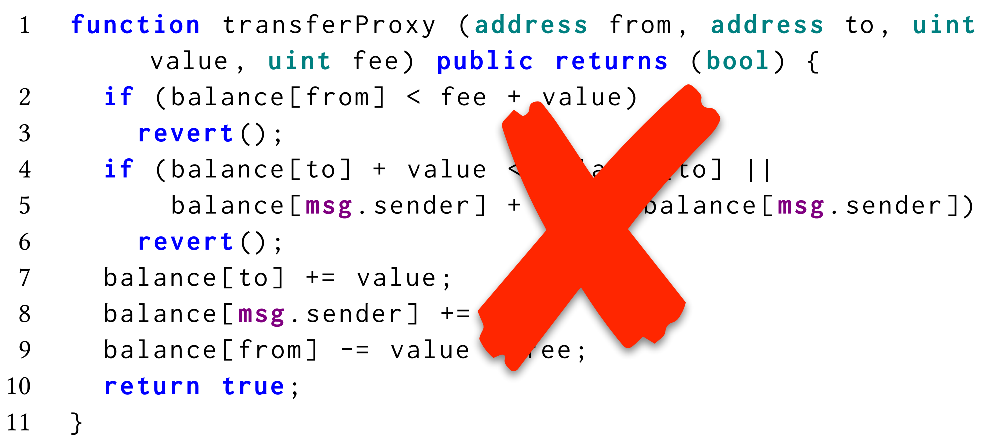
허위 경보 (False alarm)

41st IEEE Symposium on Security and Privacy

VeriSmart

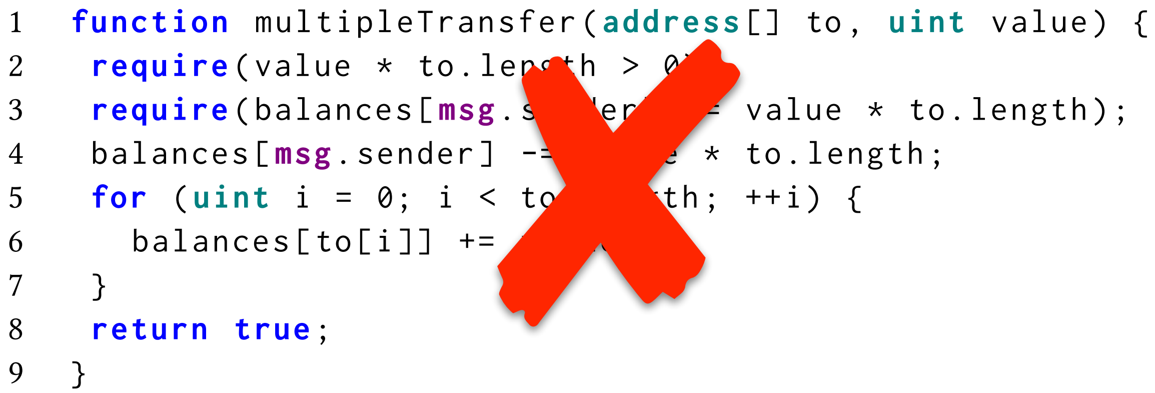
- 안전하면서 정확한 스마트 컨트랙트 취약점 자동 분석기

CVE-2018-10376



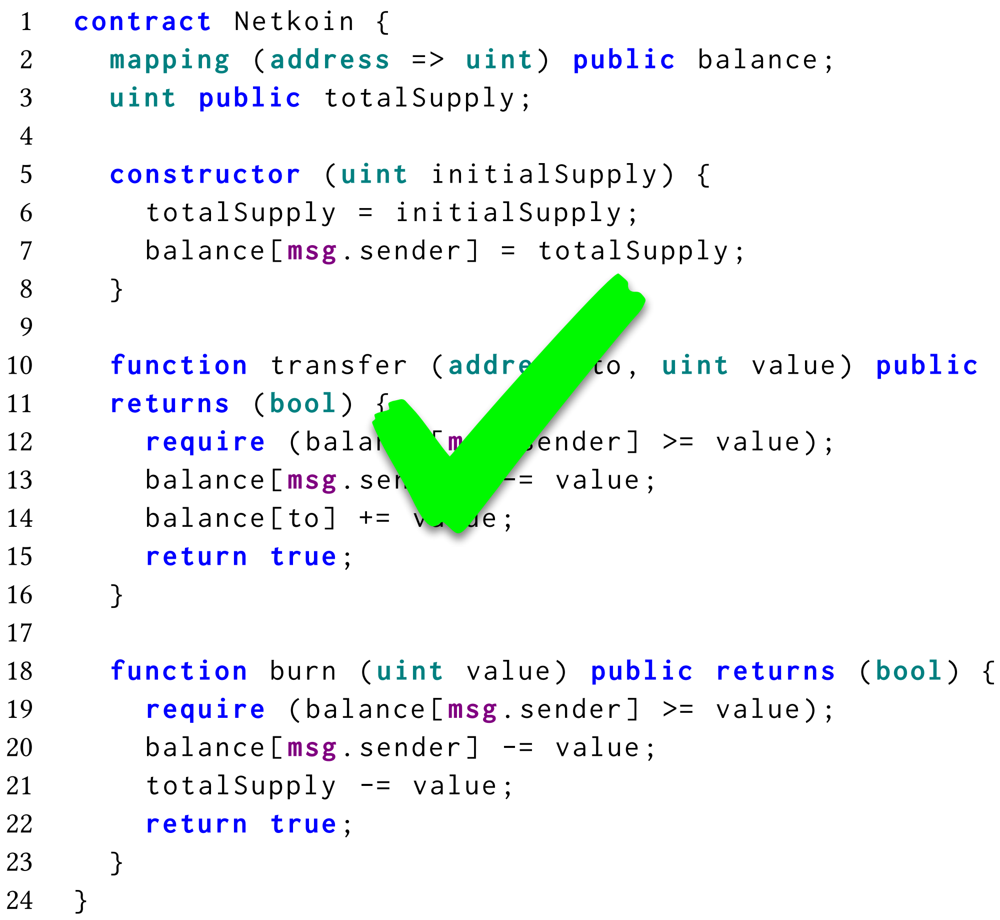
```
function transferProxy (address from, address to, uint
    value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender]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

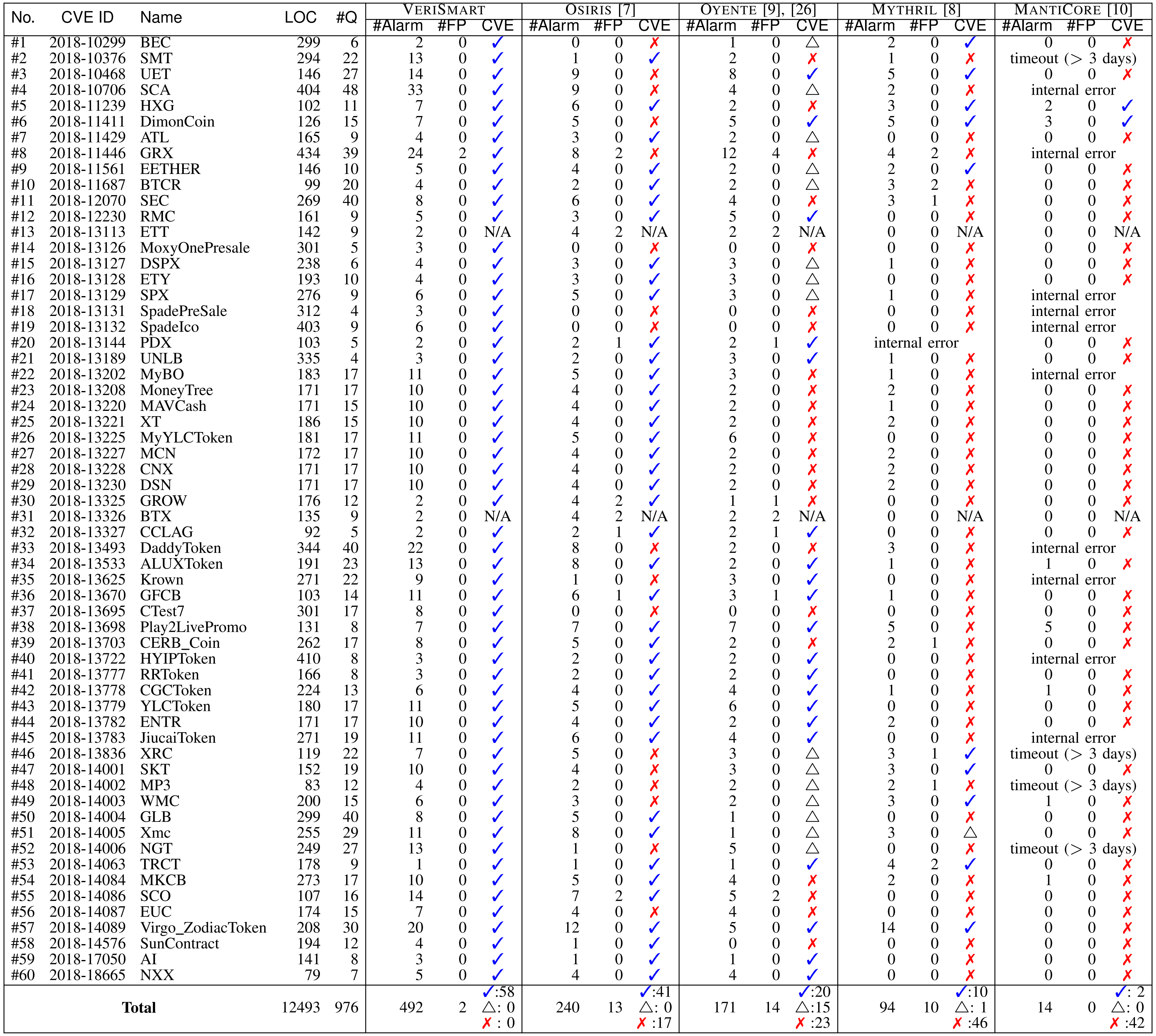
모든 오류를 검출



```
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;
  }
}
```

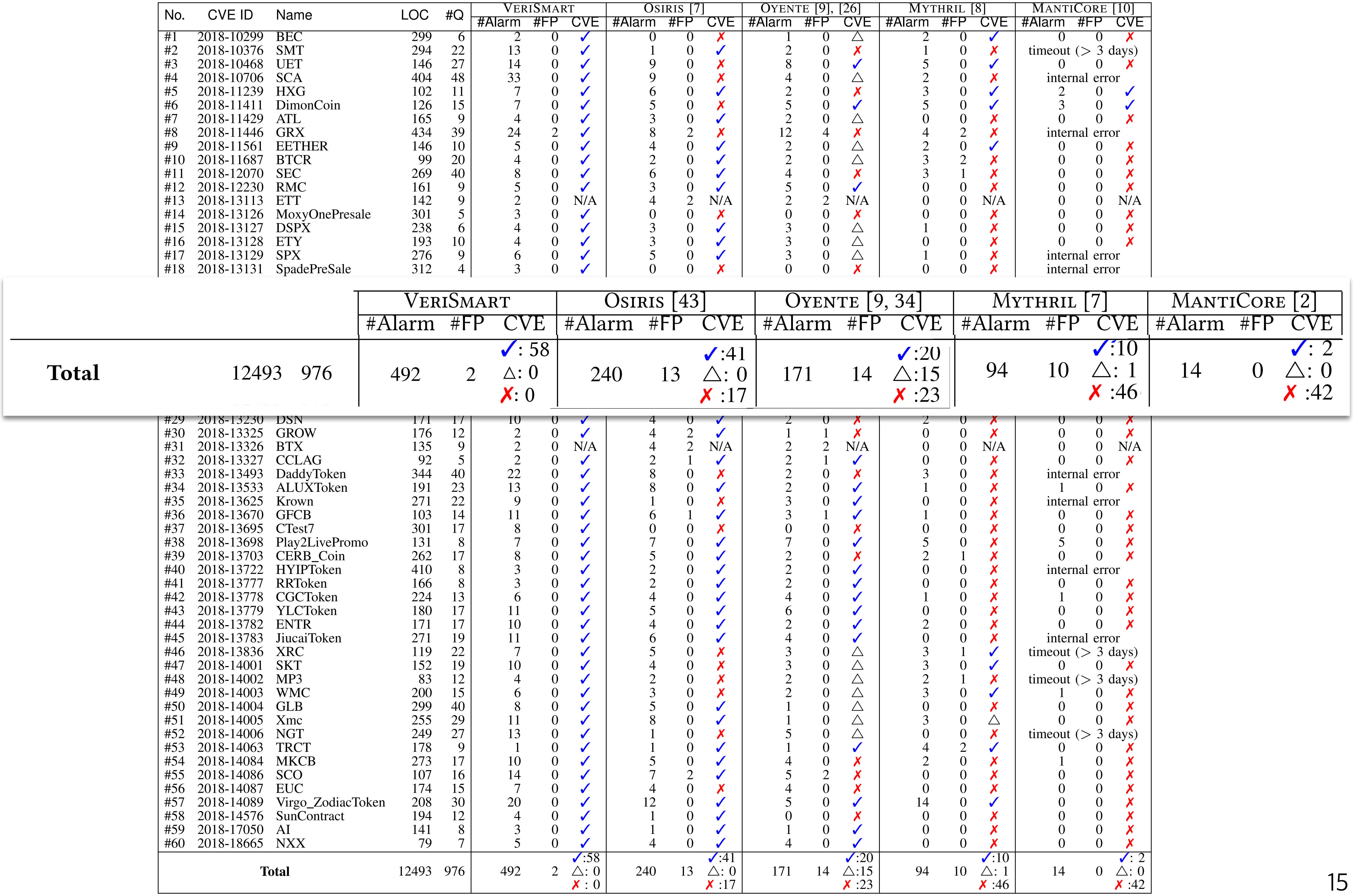
허위 경보 최소화

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

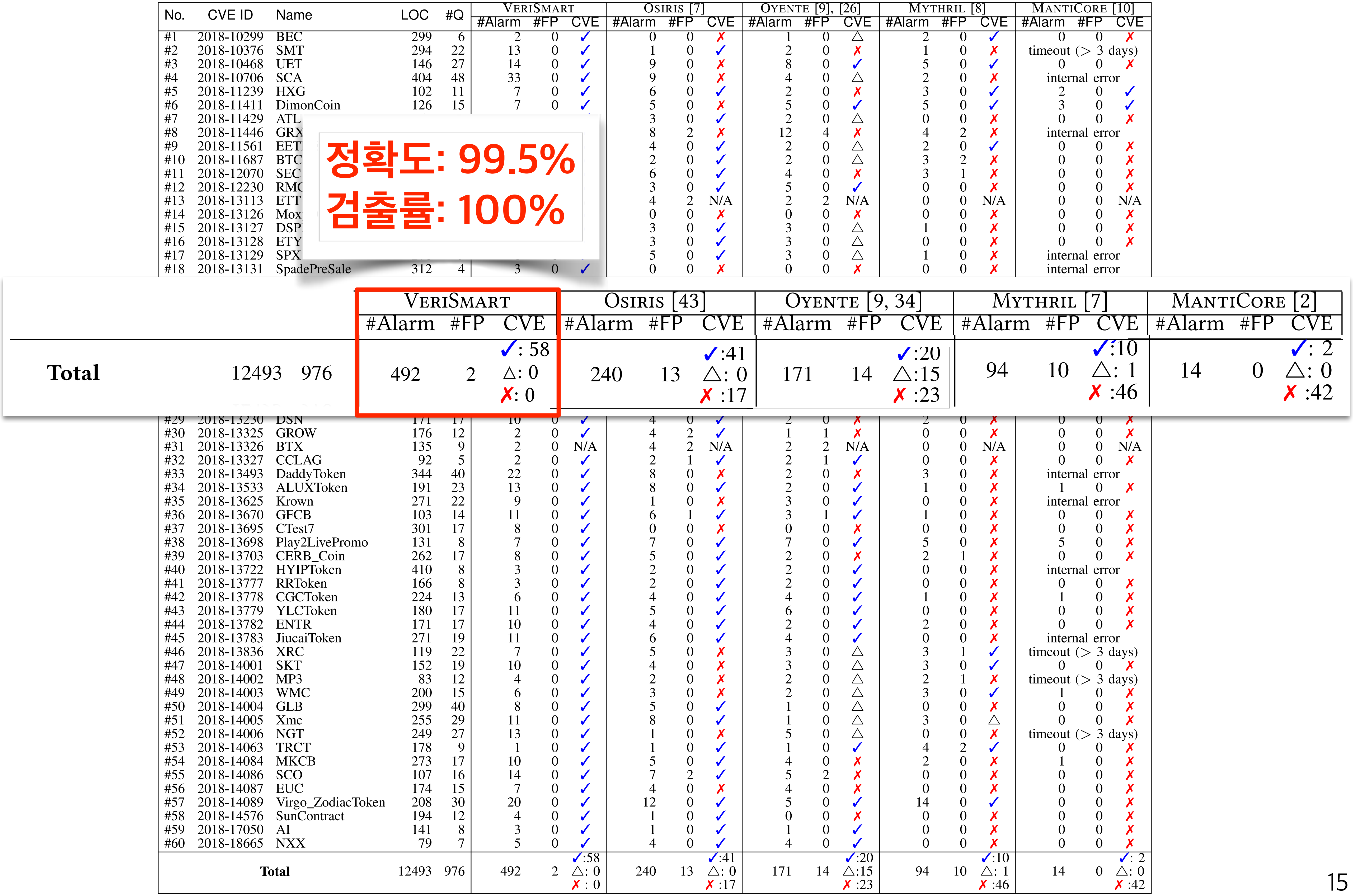
기존 취약점 검출기와 성능 비교



No.	CVE ID	Name	LOC	#Q	VeriSmart			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	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			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 검출기와 성능 비교



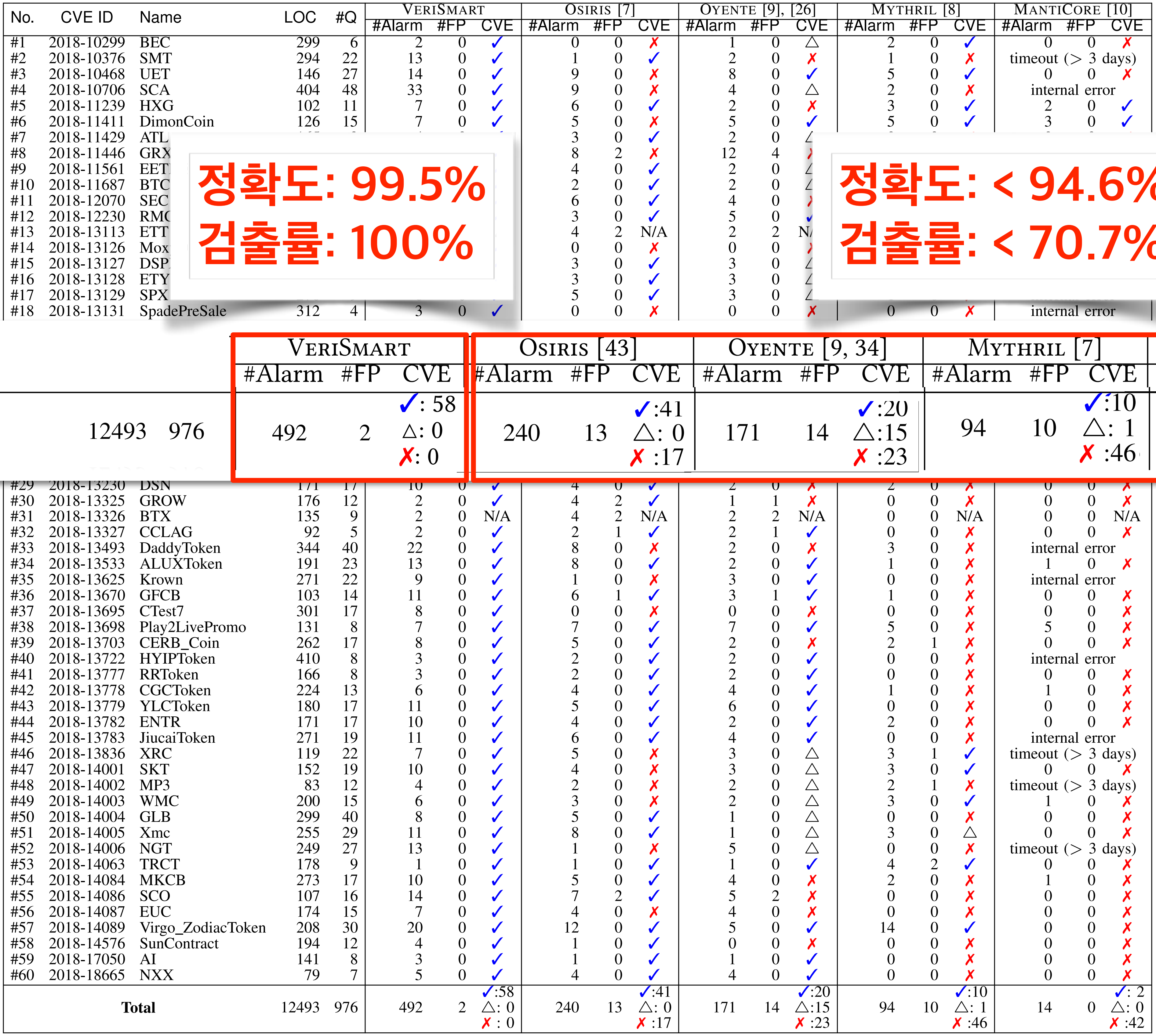
정확도: 99.5%

검출률: 100%

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	△	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	RM						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

			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 검출기와 성능 비교



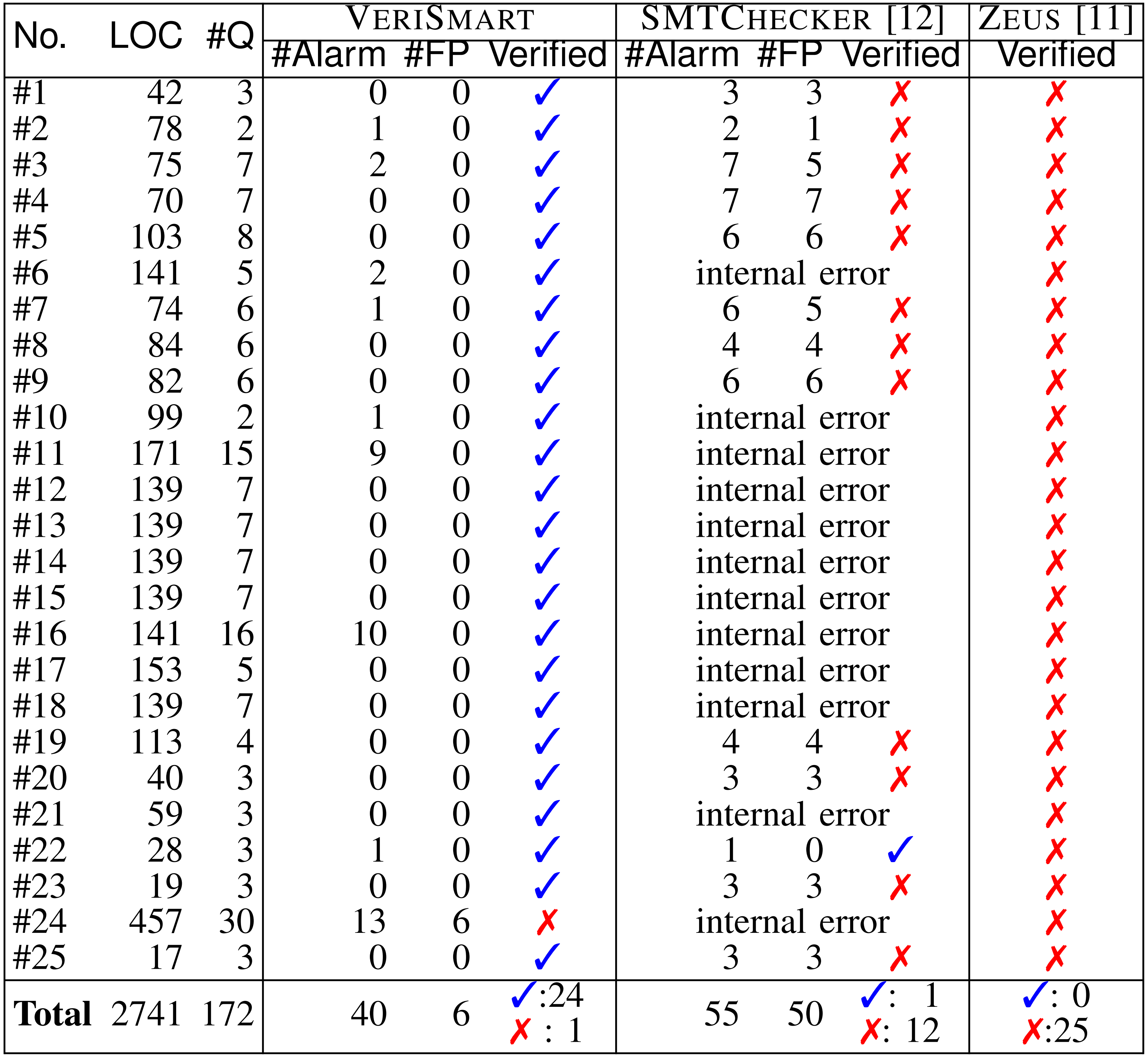
No.	CVE ID	Name	LOC	#Q	VERISMART			OSIRIS [7]			OYENTE [9], [26]			MYTHRIL [8]			MANTICORE [10]		
					#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE	#Alarm	#FP	CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#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						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]			MANTICORE [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

```
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;
  }
}
```

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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;
  }
}
```

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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;
  }
}
```

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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;
  }
}
```

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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;
  }
}
```

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

```
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;
  }
}
```

totalSupply = ∑balance

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)

$\text{totalSupply} = \sum\text{balance}$ … transaction invariant

$\geq \text{balance[msg.sender]}$ … def. of $\sum\text{balance}$

$\geq \text{value}$ … 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

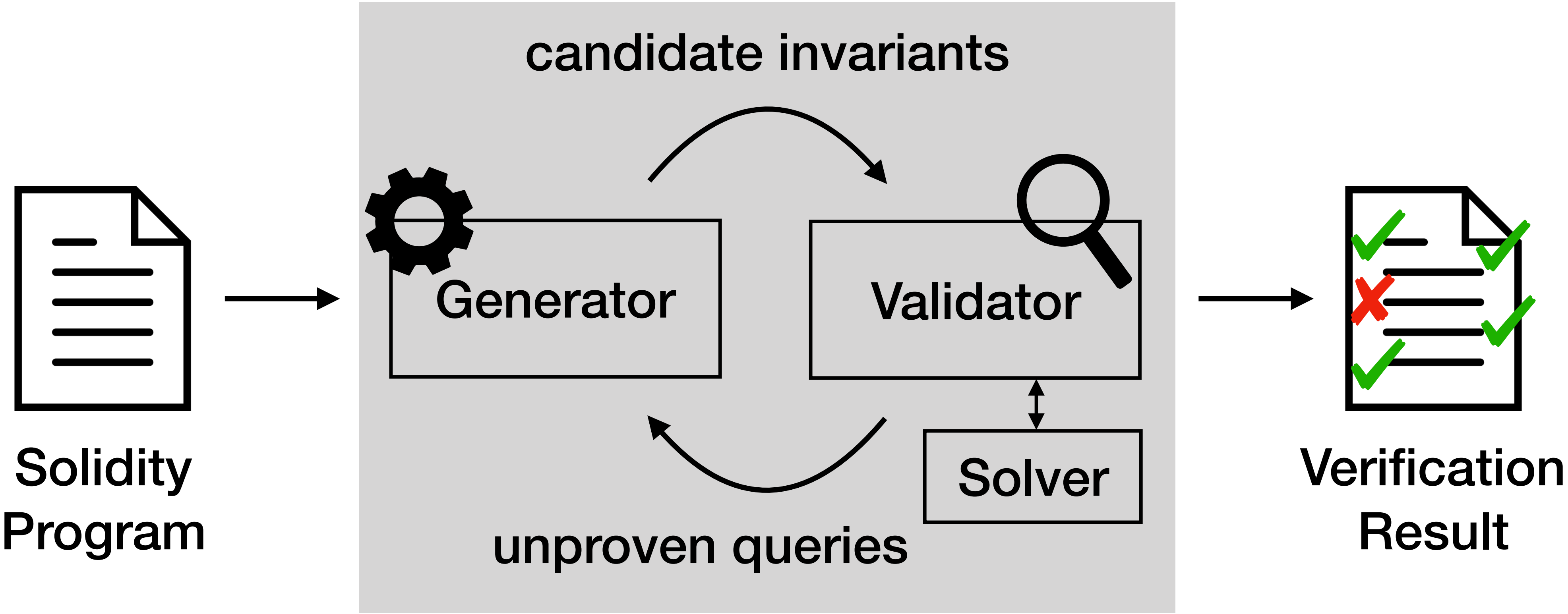
≥ balance[msg.sender] … def. of ∑balance

≥ value … assumption (require)

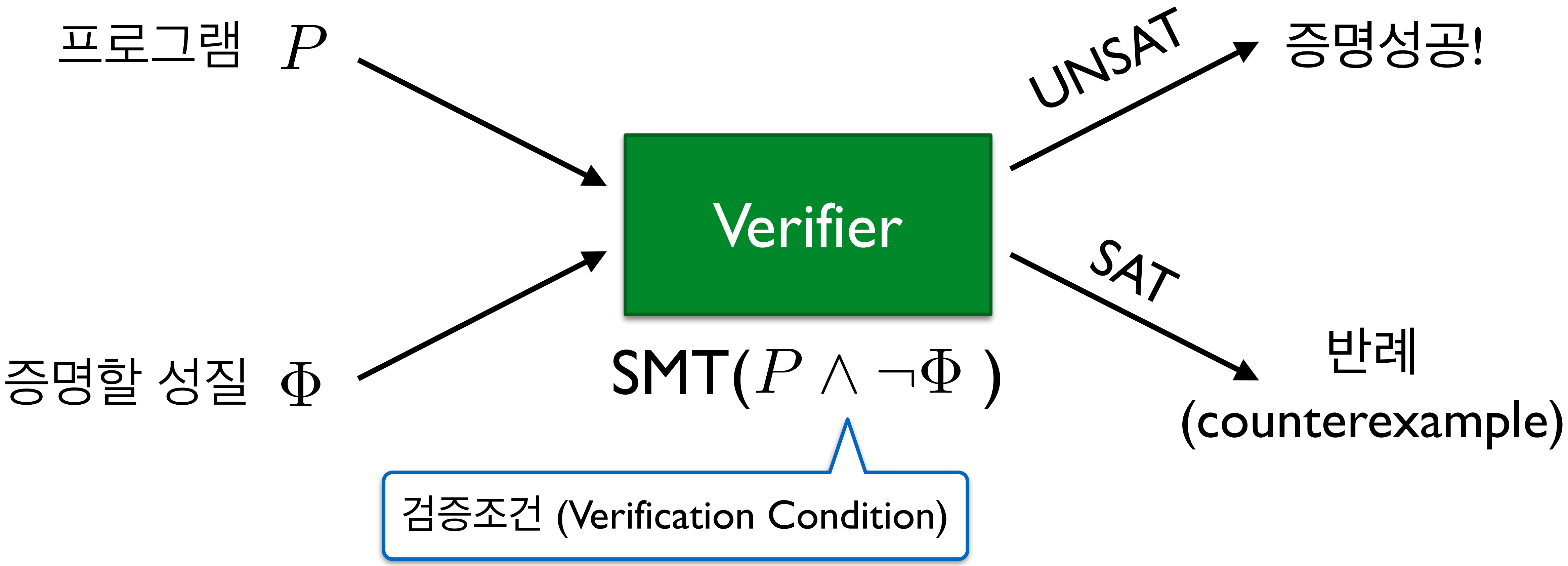
기존 취약점 검출기 / 검증기들은 이러한 추론을 못하고 FN / FP 발생

VeriSmart 검증 알고리즘

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



기반 기술: Software Verification



- 프로그램과 증명할 성질을 일차 논리식(first-order logic)으로 표현
- 논리식의 satisfiability 여부를 SMT solver로 판별

예제

```
int f(bool a) {
  x = 0; y = 0;
  if (a) {
    x = 1;
  }
  if (a) {
    y = 1;
  }
  assert (x == y)
}
```

예제

```
int f(bool a) {
   x = 0; y = 0;
   if (a) {
      x = 1;
   }
   if (a) {
      y = 1;
   }
   assert (x == y)
}
```

① ⇨

Verification Condition:

```
((a ∧ x) ∨ (¬a ∧ ¬x)) ∧
((a ∧ y) ∨ (¬a ∧ ¬y)) ∧
¬(x == y)
```

예제

```
int f(bool a) {
  x = 0; y = 0;
  if (a) {
    x = 1;
  }
  if (a) {
    y = 1;
  }
  assert (x == y)
}
```

①

Verification Condition:

$$((a \land x) \lor (\neg a \land \neg x)) \land$$
$$((a \land y) \lor (\neg a \land \neg y)) \land$$
$$\neg(x == y)$$

②

SMT solver: unsatisfiable!

예제

```
int f(a, b) {
  x = 0; y = 0;
  if (a) {
    x = 1;
  }
  if (b) {
    y = 1;
  }
  assert (x == y)
}
```

① →

Verification Condition:

$$((a \land x) \lor (\neg a \land \neg x)) \land$$
$$((b \land y) \lor (\neg b \land \neg y)) \land$$
$$\neg(x == y)$$

예제

```
int f(a, b) {
  x = 0; y = 0;
  if (a) {
    x = 1;
  }
  if (b) {
    y = 1;
  }
  assert (x == y)
}
```

①

Verification Condition:

$$((a \land x) \lor (\neg a \land \neg x)) \land$$
$$((b \land y) \lor (\neg b \land \neg y)) \land$$
$$\neg(x == y)$$

②

SMT solver:

satisfiable when $a=1$ and $b=0$

반복문 불변 성질

```
i = 0;
j = 0;
while
(i < 10) {
  i++;
  j++;
}
assert (i-j==0)
```

반복문 불변 성질

```
i = 0;
j = 0;
while @(i==j)
(i < 10) {
  i++;
  j++;
}
assert (i-j==0)
```

반복문 불변 성질

```
i = 0;
j = 0;
while @(i==j)
(i < 10) {
  i++;
  j++;
}
assert (i-j==0)
```

```
((i=0∧j=0)→(i=j))
 ∧ ((i=j)→(i−j))
```

반복문 불변 성질

```
i = 0;
j = 0;
while @(i==j)
(i < 10) {
  i++;
  j++;
}
assert (i-j==0)
```

증명에 실패하는 불변 성질은 무용지물
(i >= 0,j >= 0,i == j,true,⋯)

$$((i=0 \land j=0) \rightarrow (i=j))$$
$$\land\ ((i=j) \rightarrow (i-j))$$

소프트웨어 자동 검증의 어려움

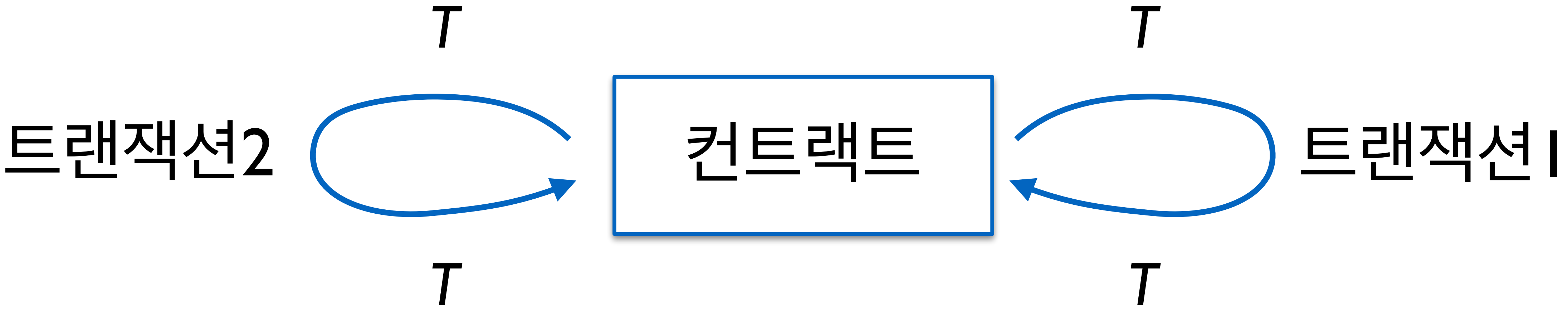
```
@pre : $\top$
@post : sorted$(rv, 0, |rv| - 1)$
bool BubbleSort (int $a$[]) {
  int[] $a := a_0$
  @$L_1$ $\left[\begin{array}{l} -1 \leq i < |a| \\ \land\ \text{partitioned}(a, 0, i, i+1, |a|-1) \\ \land\ \text{sorted}(a, i, |a|-1) \end{array}\right]$
  for (int $i := |a| - 1$; $i > 0$; $i := i - 1$) {
    @$L_2$ $\left[\begin{array}{l} 1 \leq i < |a|\ \land\ 0 \leq j \leq i \\ \land\ \text{partitioned}(a, 0, i, i+1, |a|-1) \\ \land\ \text{partitioned}(a, 0, j-1, j, j) \\ \land\ \text{sorted}(a, i, |a|-1) \end{array}\right]$
    for (int $j := 0$; $j < i$; $j := j + 1$) {
      if ($a[j] > a[j+1]$) {
        int $t := a[j]$;
        int $a[j] := a[j+1]$;
        int $a[j+1] := t$;
      }
    }
  }
  return $a$;
}
```

스마트 컨트랙트의 경우

- 상대적으로 코드가 단순하여 불변식 자동 추론이 가능

```
for(i = 0; i < x ; i++)
```

- 단, 반복문이 트랜잭션 호출로 주로 만들어지므로 트랜잭션 불변 성질 (Transaction invariant) 유추가 중요



탐색 기반 프로그램 합성을 이용

- 스마트 컨트랙트 불변식을 위한 도메인 특화 언어 설계
    - 단순한 형태의 연산식 $x = y, x \geq y, x = n, x \geq n, \ldots$
    - 스마트 컨트랙트에서 자주 사용되는 데이터 특성 반영 (e.g. the sum of balance is equal to totalSupply)
    - quantifier-free, conjunctive formulas
- 모든 가능한 불변식을 크기순으로 탐색하면서 검증 시도

In the paper https://arxiv.org/pdf/1908.11227.pdf

VeriSmart: A Highly Precise Safety Verifier for Ethereum Smart Contracts

Sunbeom So, Myungho Lee, Jisu Park, Heejo Lee, Hakjoo Oh*
Department of Computer Science and Engineering
Korea University

***Abstract*—We present VeriSmart, a highly precise verifier for ensuring arithmetic safety of Ethereum smart contracts. Writing safe smart contracts without unintended behavior is critically important because smart contracts are immutable and even a single flaw can cause huge financial damage. In particular, ensuring that arithmetic operations are safe is one of the most important and common security concerns of Ethereum smart contracts nowadays. In response, several safety analyzers have been proposed over the past few years, but state-of-the-art is still unsatisfactory; no existing tools achieve high precision and recall at the same time, inherently limited to producing annoying false alarms or missing critical bugs. By contrast, VeriSmart aims for an uncompromising analyzer that performs exhaustive verification without compromising precision or scalability, thereby greatly reducing the burden of manually checking undiscovered or incorrectly-reported issues. To achieve this goal, we present a new domain-specific algorithm for verifying smart contracts, which is able to automatically discover and leverage transaction invariants that are essential for precisely analyzing smart contracts. Evaluation with real-world smart contracts shows that VeriSmart can detect all arithmetic bugs with a negligible number of false alarms, far outperforming existing analyzers.**

I. Introduction

Safe smart contracts are indispensable for trustworthy blockchain ecosystems. Blockchain is widely recognized as one of the most disruptive technologies and smart contracts lie at the heart of this revolution (e.g., [1], [2]). Smart contracts are computer programs that run on blockchains in order to automatically fulfill agreed obligations between untrusted parties without intermediaries. Unfortunately, despite their potential, smart contracts are more likely to be vulnerable than traditional programs because of their unique characteristics such as openness and immutability [3]. As a result, unsafe smart contracts are prevalent and are increasingly becoming a serious threat to the success of the blockchain technology. For example, recent infamous attacks on the Ethereum blockchain such as the DAO [4] and the Parity Wallet [5] attacks were caused by unsafe smart contracts.

In this paper, we present VeriSmart, a fully automated safety analyzer for verifying Ethereum smart contracts with a particular focus on arithmetic safety. We focus on detecting arithmetic bugs such as integer over/underflows and division-by-zeros because smart contracts typically involve lots of arithmetic operations and they are major sources of security vulnerabilities nowadays. For example, arithmetic over/underflows account for 95.7% (487/509) of CVEs assigned to Ethereum smart contracts, as shown in Table I. Even worse, arithmetic bugs, once exploited, are likely to cause significant but unexpected financial damage (e.g., the integer overflow in the SmartMesh contract [6] explained in Section II). Our goal is to detect all arithmetic bugs before deploying smart contracts on the blockchain.

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

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

Unlike existing techniques, VeriSmart aims to be a truly practical tool by performing automatic, scalable, exhaustive, yet highly precise verification of smart contracts. Recent years have seen an increased interest in automated tools for analyzing arithmetic safety of smart contracts [7], [8], [9], [10], [11], [12]. However, existing tools are still unsatisfactory. A major weakness of bug-finding approaches (e.g., [7], [9], [8], [10]) is that they are likely to miss fatal bugs (i.e., resulting in false negatives), because they do not consider all the possible behaviors of the program. On the other hand, verification approaches (e.g., [11], [12]) are exhaustive and therefore miss no vulnerabilities, but they typically do so at the expense of precision (i.e., resulting in false positives). In practice, both false negatives and positives burden developers with error-prone and time-consuming process for manually verifying a number of undiscovered issues or incorrectly reported alarms. VeriSmart aims to overcome these shortcomings of existing approaches by being exhaustive yet precise.

To achieve this goal, we present a new verification algorithm for smart contracts. The key feature of the algorithm, which departs significantly from the existing analyzers for smart contracts [7], [8], [9], [10], [11], [12], is to automatically discover domain-specific invariants of smart contracts during the verification process. In particular, our algorithm automates the discovery of *transaction invariants*, which are distinctive properties of smart contracts that hold under arbitrary interleaving of transactions and enable to analyze smart contracts exhaustively without exploring all program paths separately. A technical challenge is to efficiently discover precise invariants

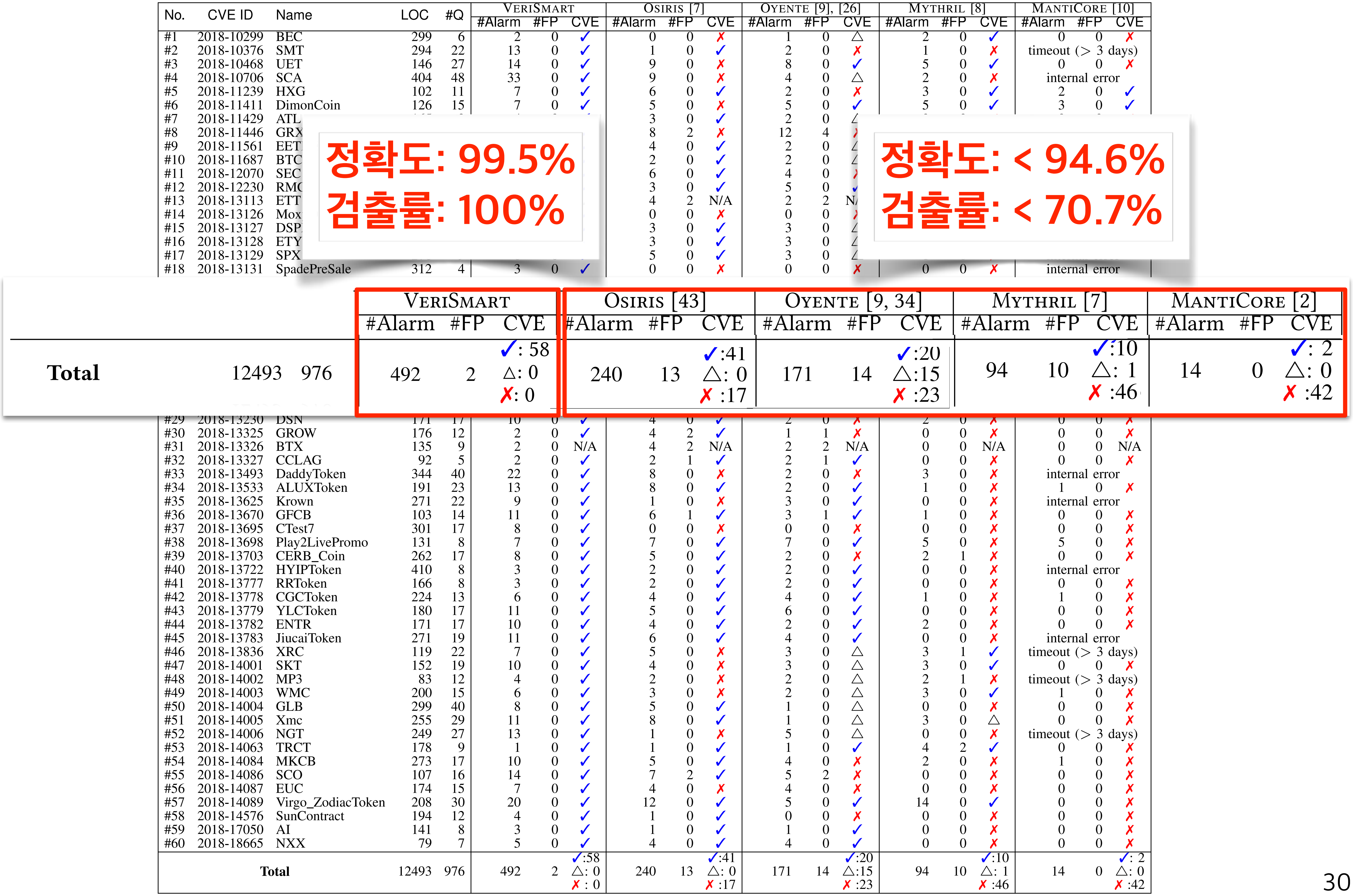
*Corresponding author: Hakjoo Oh, hakjoo_oh@korea.ac.kr

- VC 생성 & 불변식 유추
- VC 효율적으로 풀기
- 구현 이슈
- …

실험

- 벤치마크 (https://github.com/kupl/VeriSmart-benchmarks)
    - CVE 취약점이 있는 60개 컨트랙트
    - Zeus (NDSS'18) 공개 데이터 25개
- 비교 대상 분석기
    - 오류 검출기: Oyente, Mythril, Manticore, Osiris,
    - 오류 검증기: Zeus, SMTChecker

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/A						
#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 오류 검출기들의 한계

- 총 37개의 허위 경보중 18개는 불변 성질 유추에 실패해서, 19개는 조건식을 정교하게 추적 못해서 발생

```
function transfer(address _to, uint _value) {
  if (msg.sender.balance < min)
    sell((min - msg.sender.balance) / sellPrice);
}
```

- 컨트랙트간 함수 호출로 발생하는 취약점 탐지에 주로 실패

```
function mint (address holder, uint value) {
  require (total+ value <= TOKEN_LIMIT); // CVE bug
  balances[holder] += value;            // CVE bug
  total += value;                       // CVE bug
}
...
token.mint (...,...)
```

VeriSmart 한계

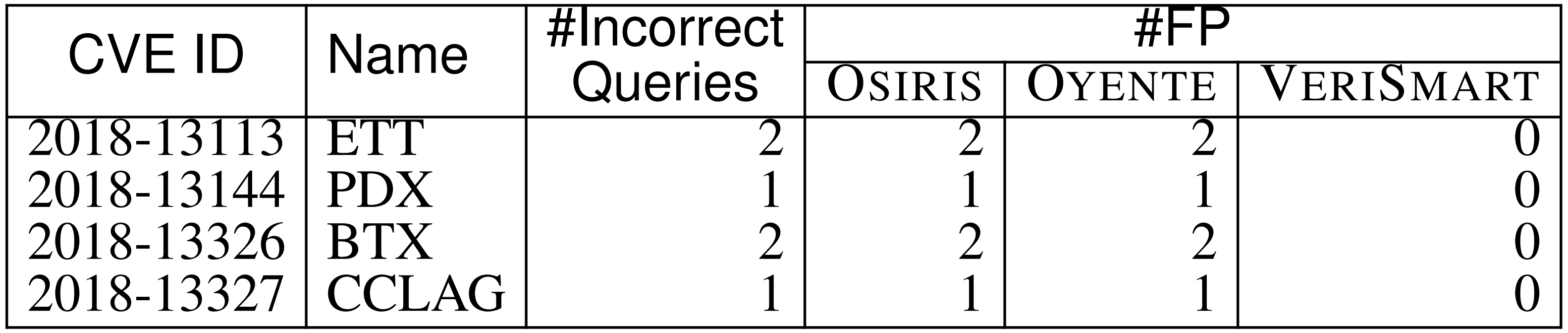
- 복잡한 불변 성질은 유추하지 못하고 허위 경보 발생

```
function unlockReward(address addr, uint value) {
  require(totalLocked[addr] > value);
  require(locked[addr][msg.sender] >= value);
  if(value == 0) value = locked[addr][msg.sender];
  totalLocked[addr] -= value;   // false positive
  locked[addr][msg.sender] -= value;
}
```

$$\forall x.\texttt{totalLocked[x]} = \sum_{i} \texttt{locked[x][}i\texttt{]}$$

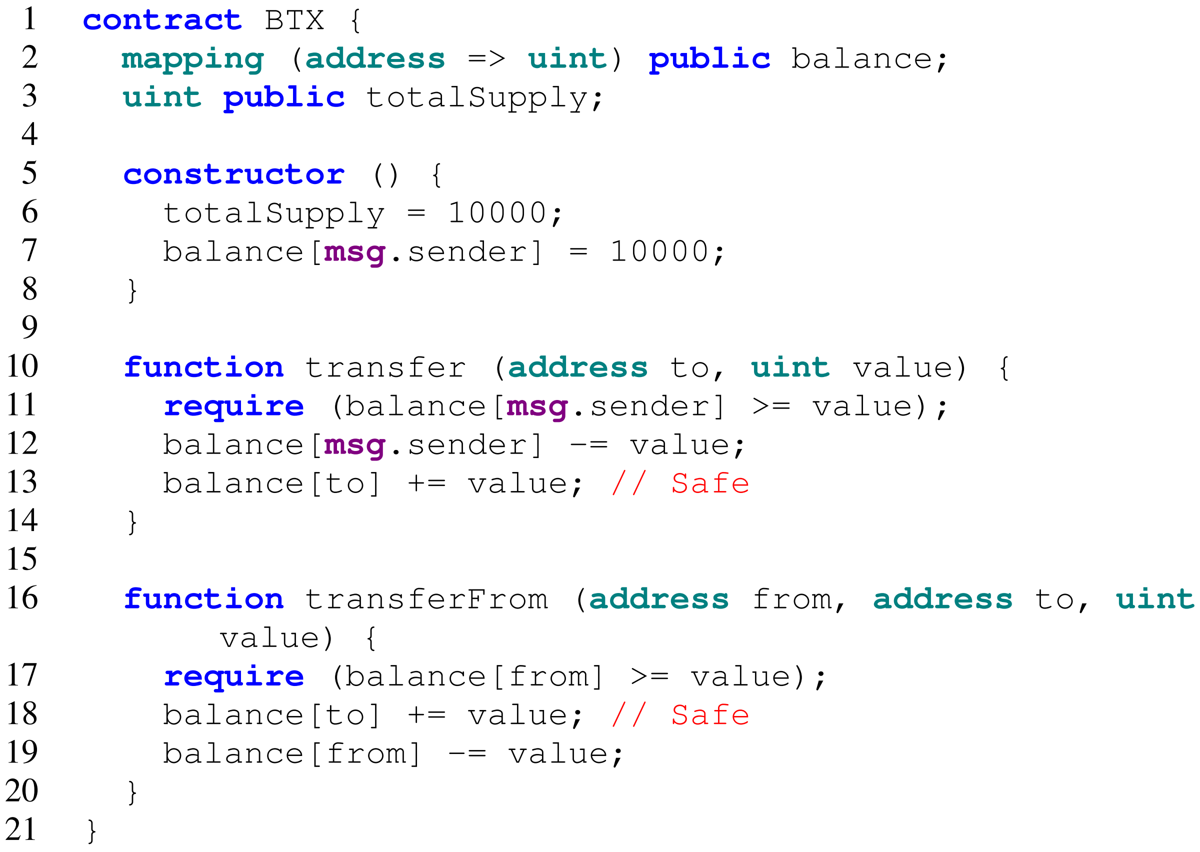
잘못된 CVE 발견

- CVE를 부여받은 일부 취약점이 실제 취약점이 아님을 발견



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

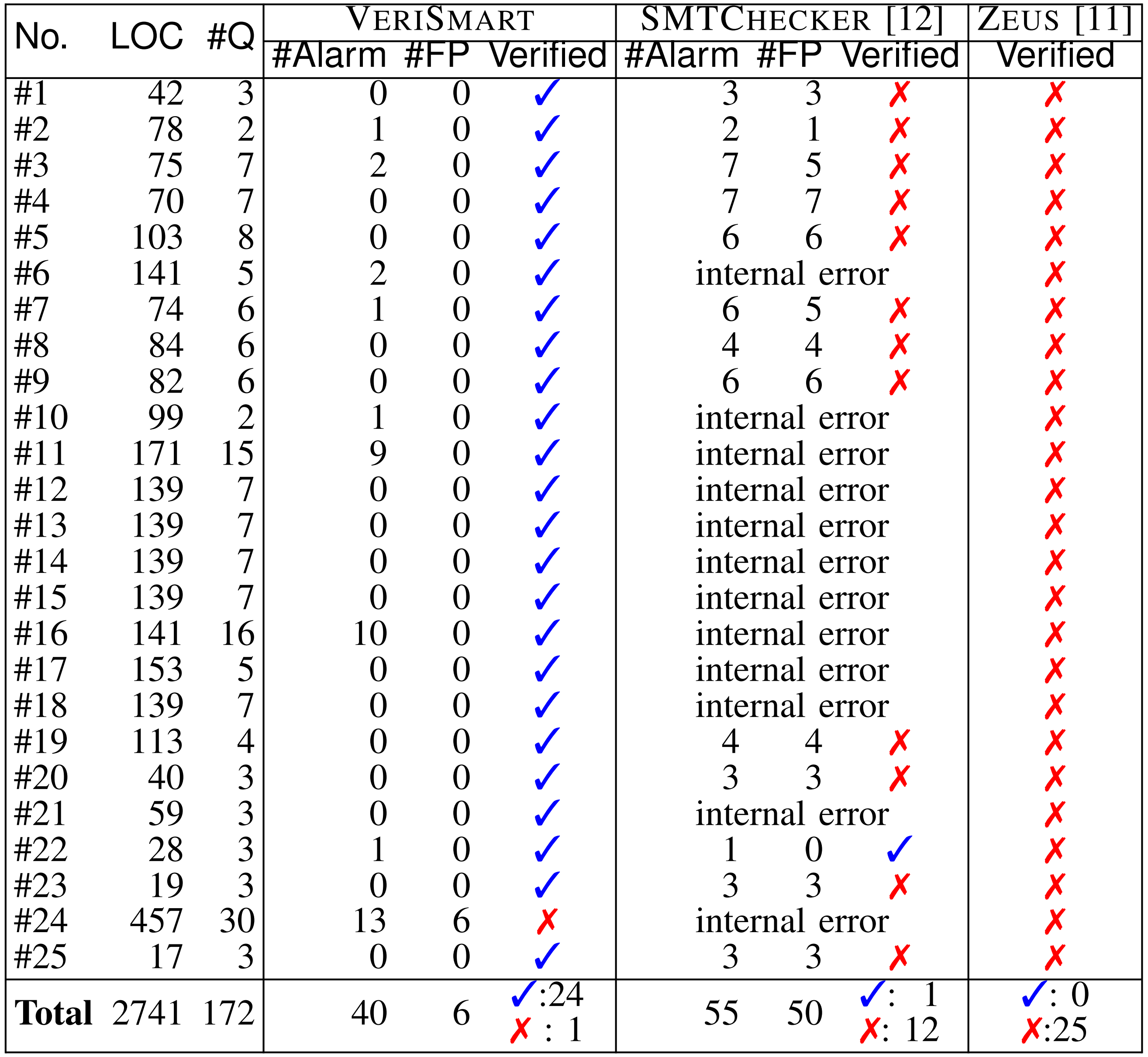
- E.g.,



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

  constructor () {
    totalSupply = 10000;
    balance[msg.sender] = 10000;
  }

  function transfer (address to, uint value) {
    require (balance[msg.sender] >= value);
    balance[msg.sender] -= value;
    balance[to] += value; // Safe
  }

  function transferFrom (address from, address to, uint
      value) {
    require (balance[from] >= value);
    balance[to] += value; // Safe
    balance[from] -= value;
  }
}
```

기존 취약점 검증기와 성능 비교

- 기존 검증기들은 스마트 컨트랙트 주요 성질 검증에 실패
- 트랜잭션 자동 유추 기능을 끄면 VeriSmart도 17개 실패



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

다른 종류의 취약점 검출에 응용

- 일반적으로 임의의 assert로 표현된 성질 검증에 활용 가능

- 액세스 컨트롤 관련 취약점: e.g. CVE 2018-11329

```
function DrugDealer() public { ceoAddr = msg.sender; }
function buyDrugs () public payable {
  ceoAddr.transfer(msg.value); // send Ether to ceoAddr
  drugs[msg.sender] += ...; // buy drugs by paying Ether
}
```

- 액세스 컨트롤 관련 모든 CVE 검출 (CVE-10666, 2018-10705, 2018-11329)

- 60개 중 55개 컨트랙트에 대해서 안전성 검증 성공

마무리

- 스마트 컨트랙트는 보안취약점 검증이 필수
- 현재 스마트 컨트랙트 분석 기술은 성능이 제한적
  - 안전성과 정확성 둘 중 하나를 포기
- **VeriSmart:** 안전하면서 정확한 스마트 컨트랙트 자동 검증기
  - 트랜잭션 불변 성질을 자동 추론하며 검증하는 첫 사례
  - 소프트웨어 검증 기술을 자동으로 유용하게 사용한 사례

Thank you!

- Research areas: programming languages, software engineering, software security
  - program analysis and testing
  - program synthesis and repair
- Publication: top-venues in PL, SE, Security, and AI:
  - PLDI(’12,’14),OOPSLA(’15,’17a,’17b,’18a,’18b,’19),TOPLAS(’14,’16,’17,’18,’19), ICSE(’17,’18,’19,’20), FSE(’18,’19), ASE’18, S&P(’17,’20), IJCAI(’17,’18), etc

http://prl.korea.ac.kr