스마트 컨트랙트 취약점 자동 검증

오학주

http://prl.korea.ac.kr

고려대학교 정보대학

Ethereum Blockchain

- 비트코인에서는 제한적인 종류의 거래만 가능
  - “X가 Y에게 Z 비트코인을 전송”
- 더욱 다양한 형태의 거래가 가능하도록 일반화
  - 코인, 금융거래, 부동산, 공증, 경매 등
- 핵심 아이디어: 스마트 컨트랙트
  - 일반적인 계약을 작성할 수 있는 프로그래밍 언어를 제공
  - 이더리움 가상 머신이 계약을 실행하고 신뢰성을 보장

스마트 컨트랙트의 구조

```
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 컨트랙트의 구조

```
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 컨트랙트의 구조

```
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 컨트랙트의 구조

```
contract Netkoin {
  mapping (address => uint) public balance;   사용자의 계좌 정보
  uint public totalSupply;                                          데이터

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

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

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

스마트 컨트랙트의 구조

```
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 안전성 문제

- 스마트 컨트랙트는 매우 엄밀한 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

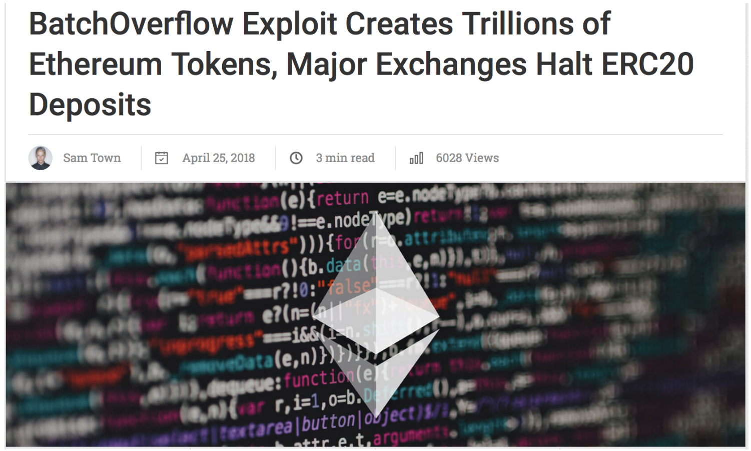
The DAO (2016)
750억원

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



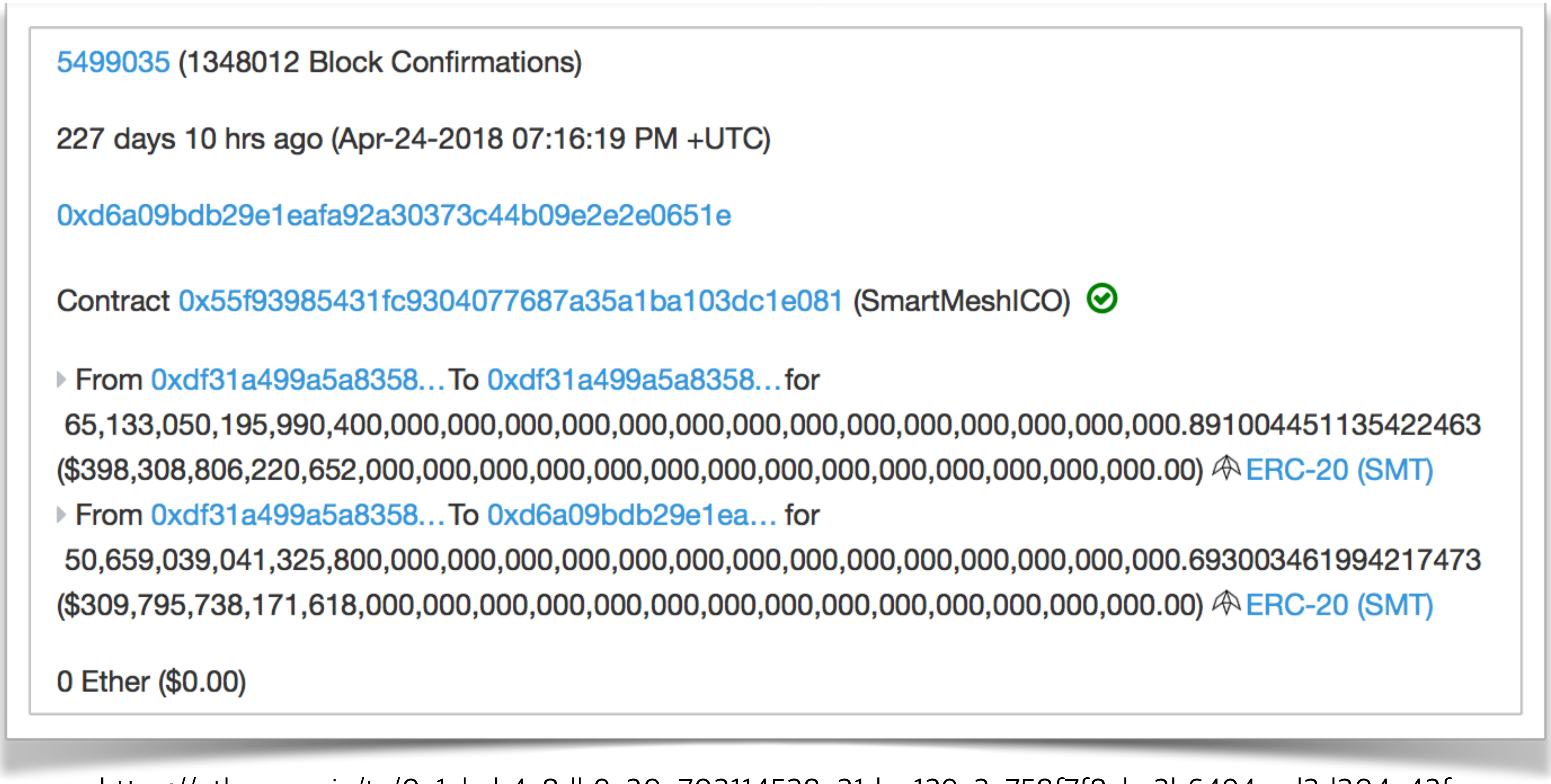
BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

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



5499035 (1348012 Block Confirmations)

227 days 10 hrs ago (Apr-24-2018 07:16:19 PM +UTC)

0xd6a09bdb29e1eafa92a30373c44b09e2e2e0651e

Contract 0x55f93985431fc9304077687a35a1ba103dc1e081 (SmartMeshICO) ✓

From 0xdf31a499a5a8358... To 0xdf31a499a5a8358... for 65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463 ($398,308,806,220,652,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) ERC-20 (SMT)

From 0xdf31a499a5a8358... To 0xd6a09bdb29e1ea... for 50,659,039,041,325,800,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.693003461994217473 ($309,795,738,171,618,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) ERC-20 (SMT)

0 Ether ($0.00)

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;
}
```

CVE-2018-10376

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

CVE-2018-10376

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 잔고
가 충분한지 체크

송금

CVE-2018-10376

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

CVE-2018-10376

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

오버플로우/언더플로우 발생하지 않음

6

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;
}
```

CVE-2018-10376

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;
}
```

CVE-2018-10376

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;
}
```

CVE-2018-10376

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;
}
```

O!

CVE-2018-10376

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!

CVE-2018-10376

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

CVE-2018-10376

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!

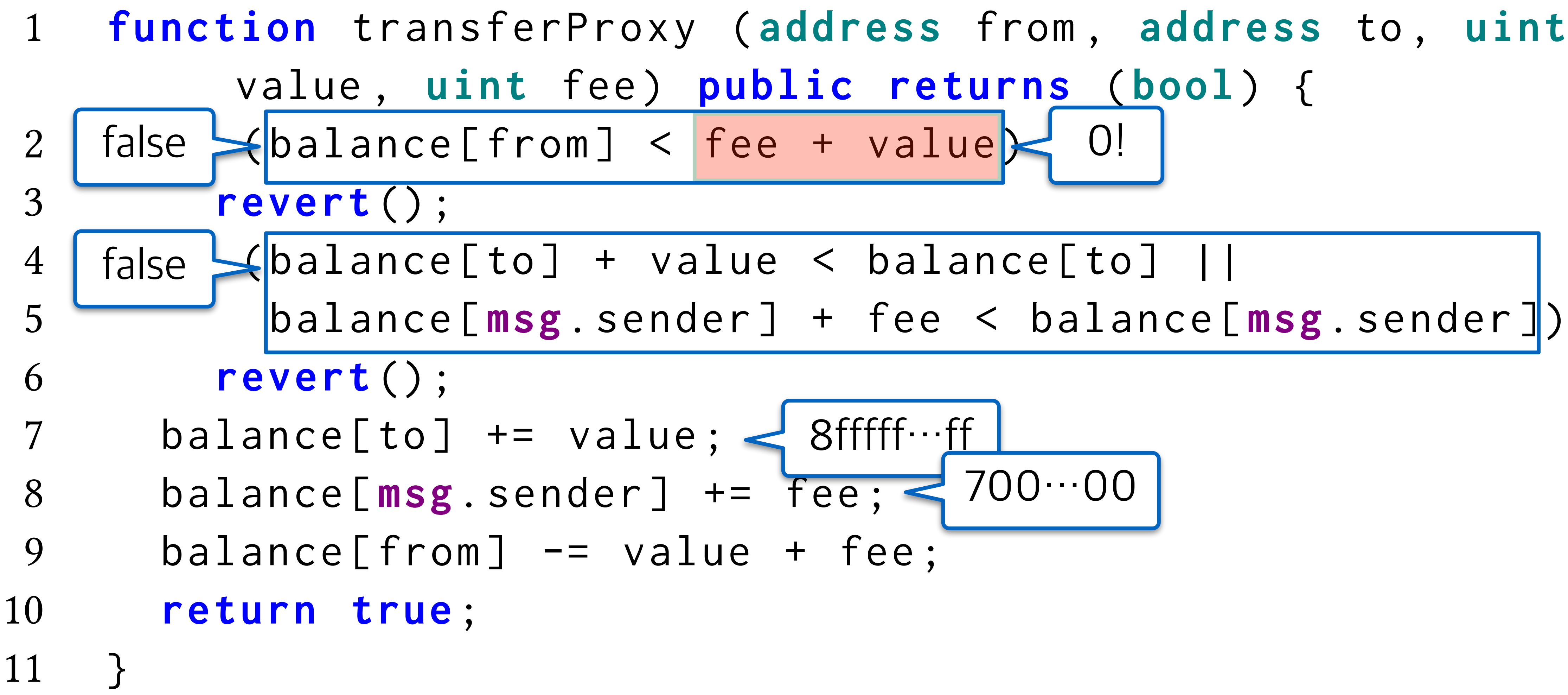
false

8fffff⋯ff

CVE-2018-10376

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

CVE-2018-10376

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;
}
```

false

0!

false

8fffff…ff

700…00

0!

CVE-2018-10376

목표: 정수 오버플로우 취약점 검증

- Solidity에서는 정수를 유한한 비트로 표현

```
uint public totalSupply;
```

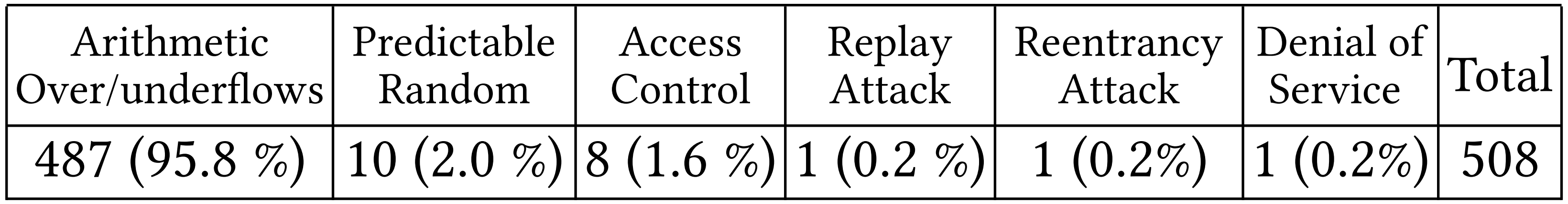
256bit

- 정수 연산시 표현 가능한 범위를 넘어서지 여부를 검증

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

- 오버플로우 유무를 판단하기가 매우 까다로움

- CVE 등록된 취약점 대부분이 정수 오버플로우에서 비롯



Arithmetic Over/underflows	Predictable Random	Access Control	Replay Attack	Reentrancy Attack	Denial of Service	Total
487 (95.8 %)	10 (2.0 %)	8 (1.6 %)	1 (0.2 %)	1 (0.2%)	1 (0.2%)	508

스마트 컨트랙트 자동 분석 기술

- 오류 검출기 (bug-detector)

Oyente Mythril manticore Osiris

- 오류 검증기 (verifier)

Using Solidity's SMTChecker

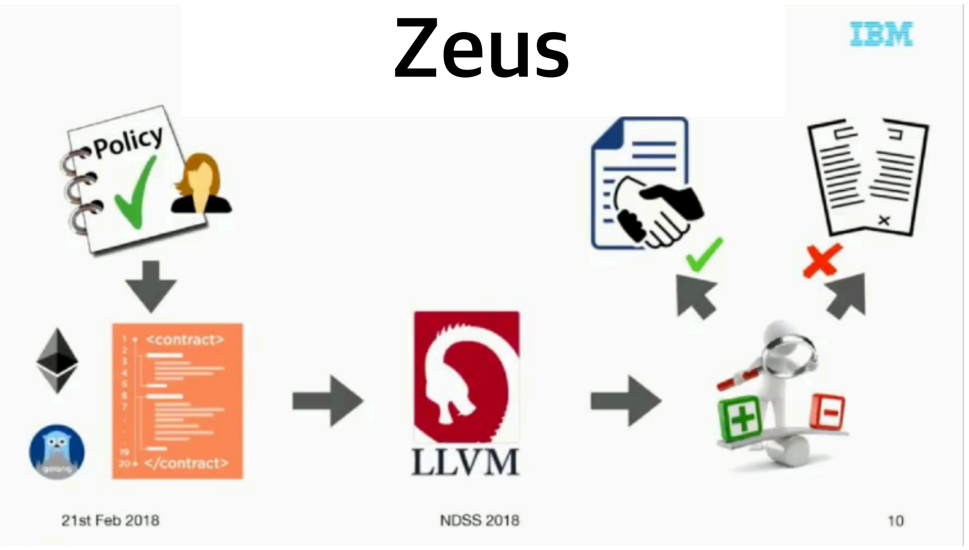
Leonardo Alt

Ethereum Foundation

leonardoalt

leo@ethereum.org

leonardoalt

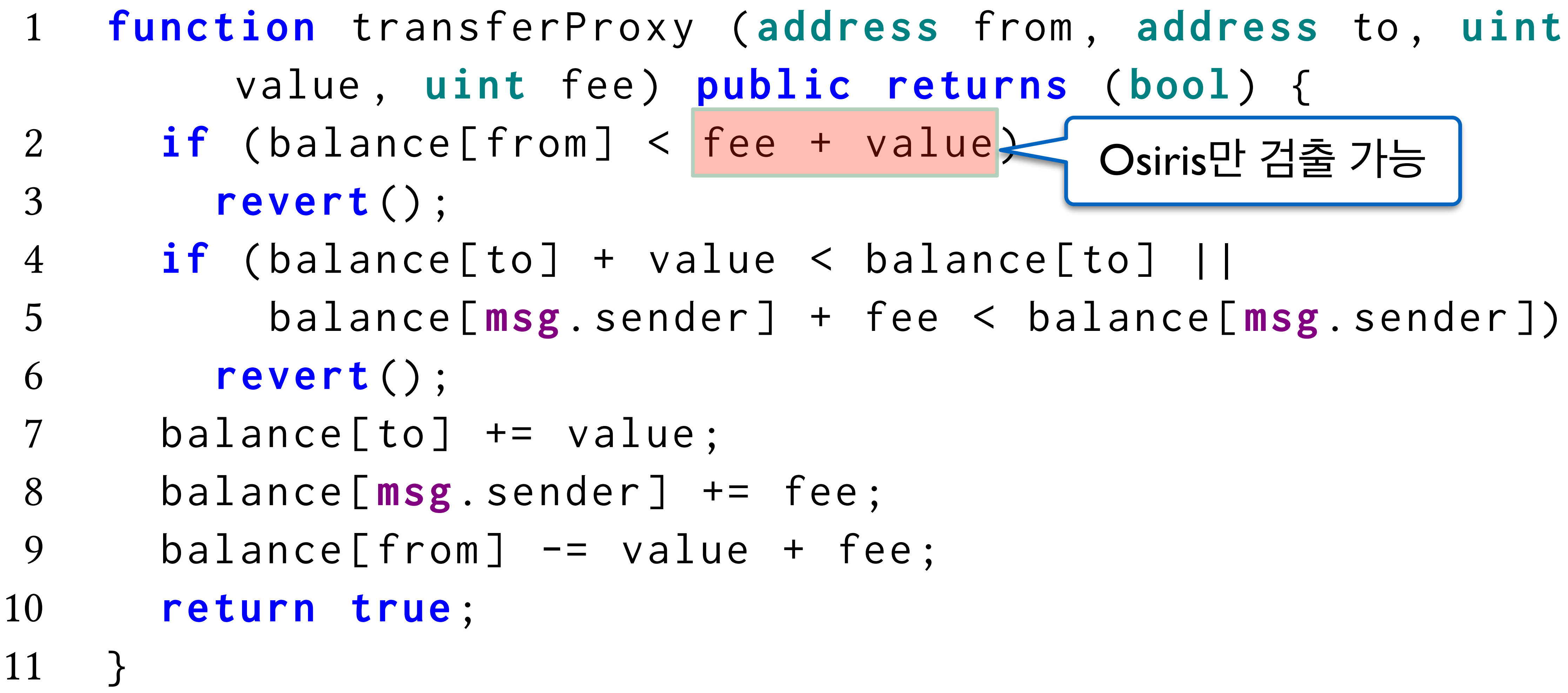


Zeus

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
오류이지만 검출 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)

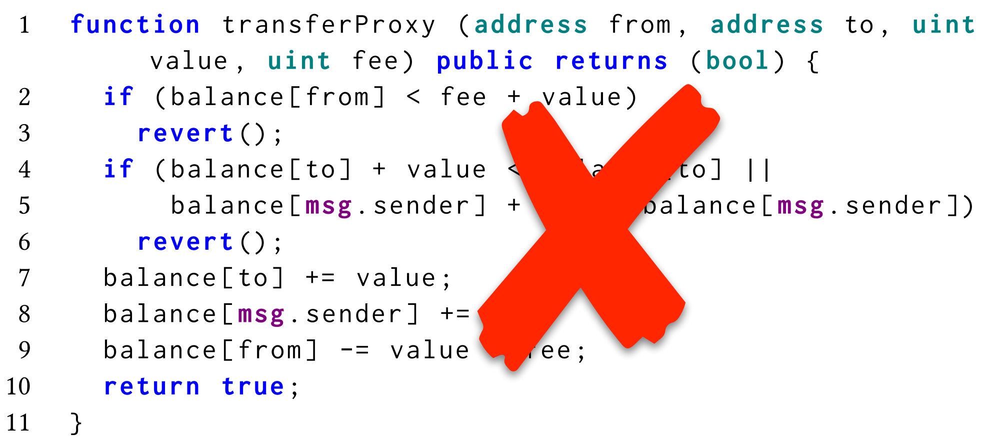
허위 경보 (False alarm)

Oakland'20

VeriSmart

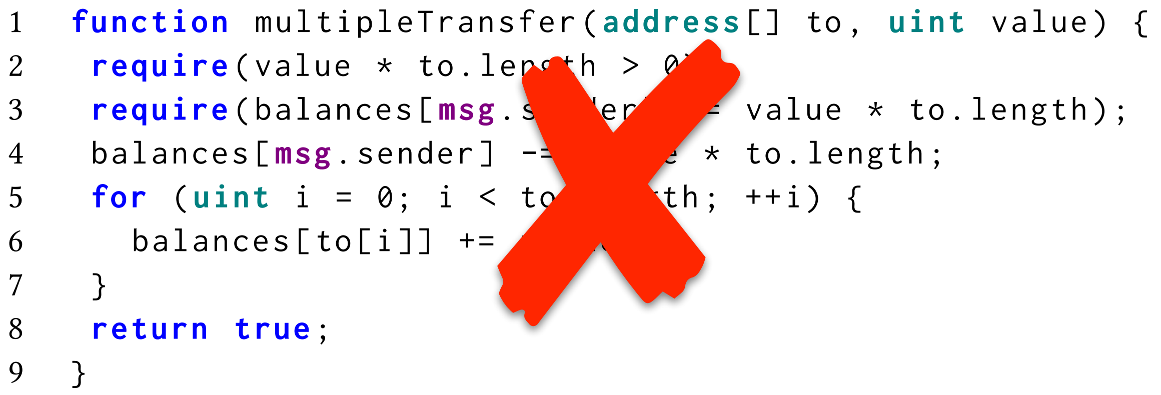
- 안전하면서 정확한 스마트 컨트랙트 취약점 자동 분석기

CVE-2018-10376

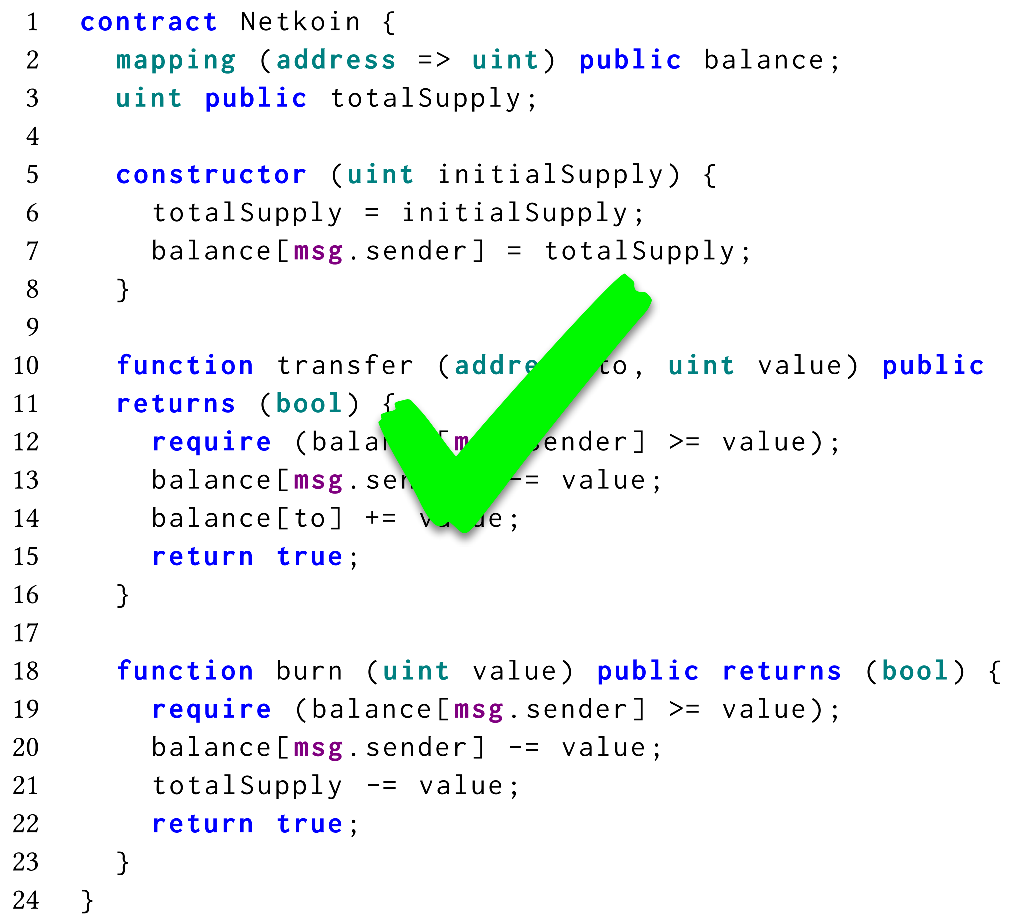


```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + [illegible] balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender]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

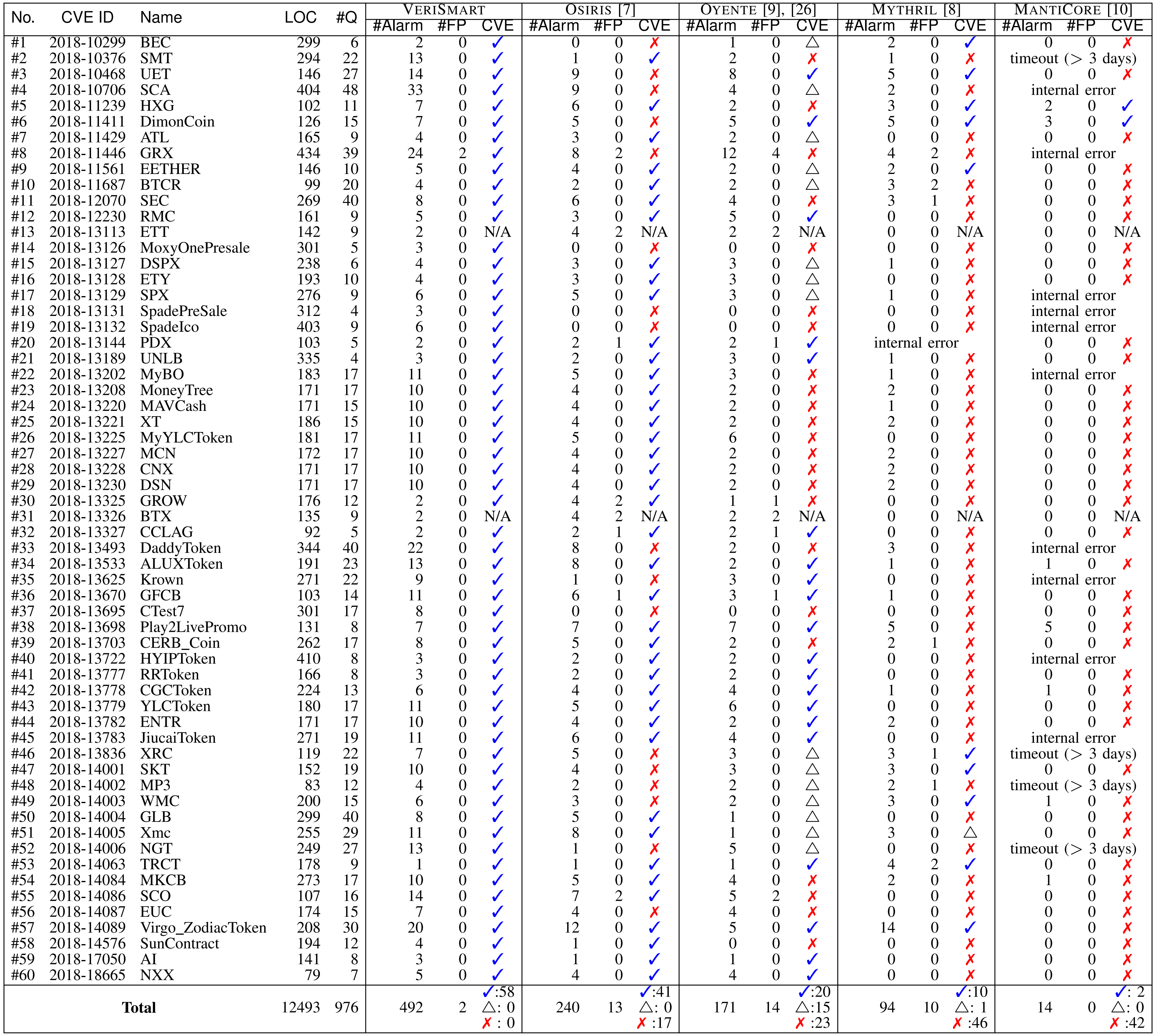


```
contract Netkoin {
  mapping (address => uint) public balance;
  uint public totalSupply;

  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;
  }
}
```

모든 오류를 검출

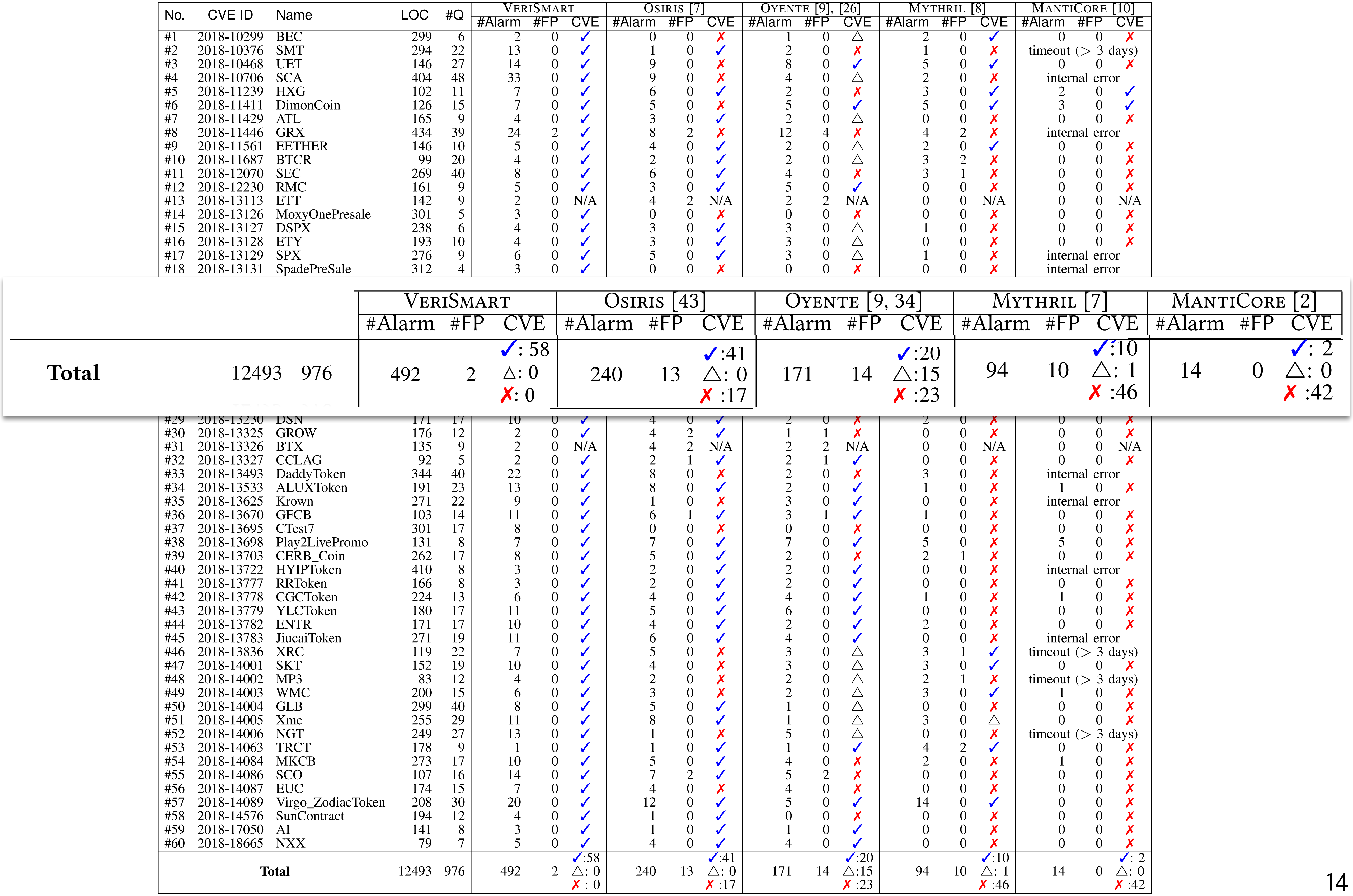
허위 경보 최소화

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

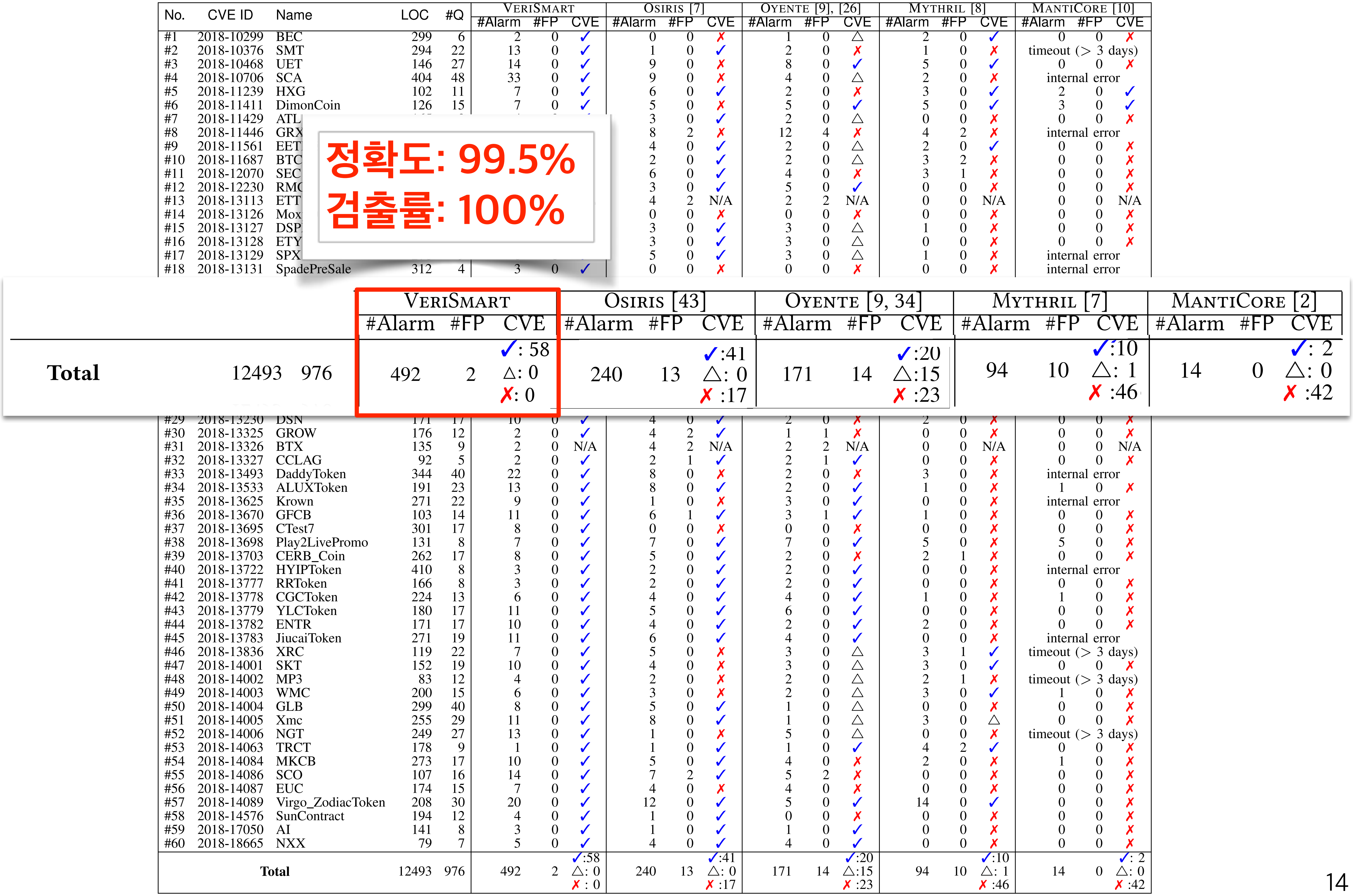
기존 취약점 검출기와 성능 비교



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



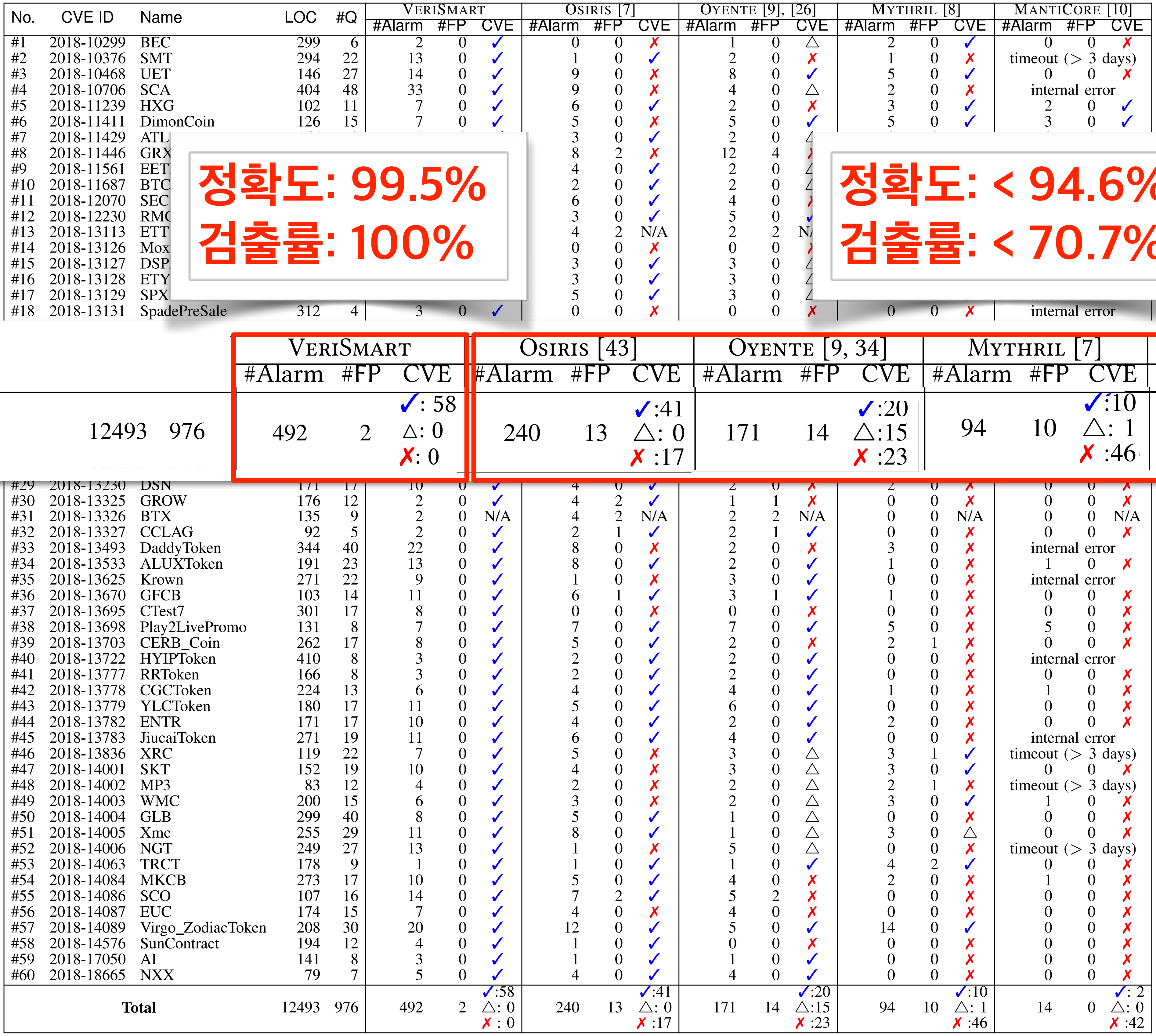
정확도: 99.5%

검출률: 100%

		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

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	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

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/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

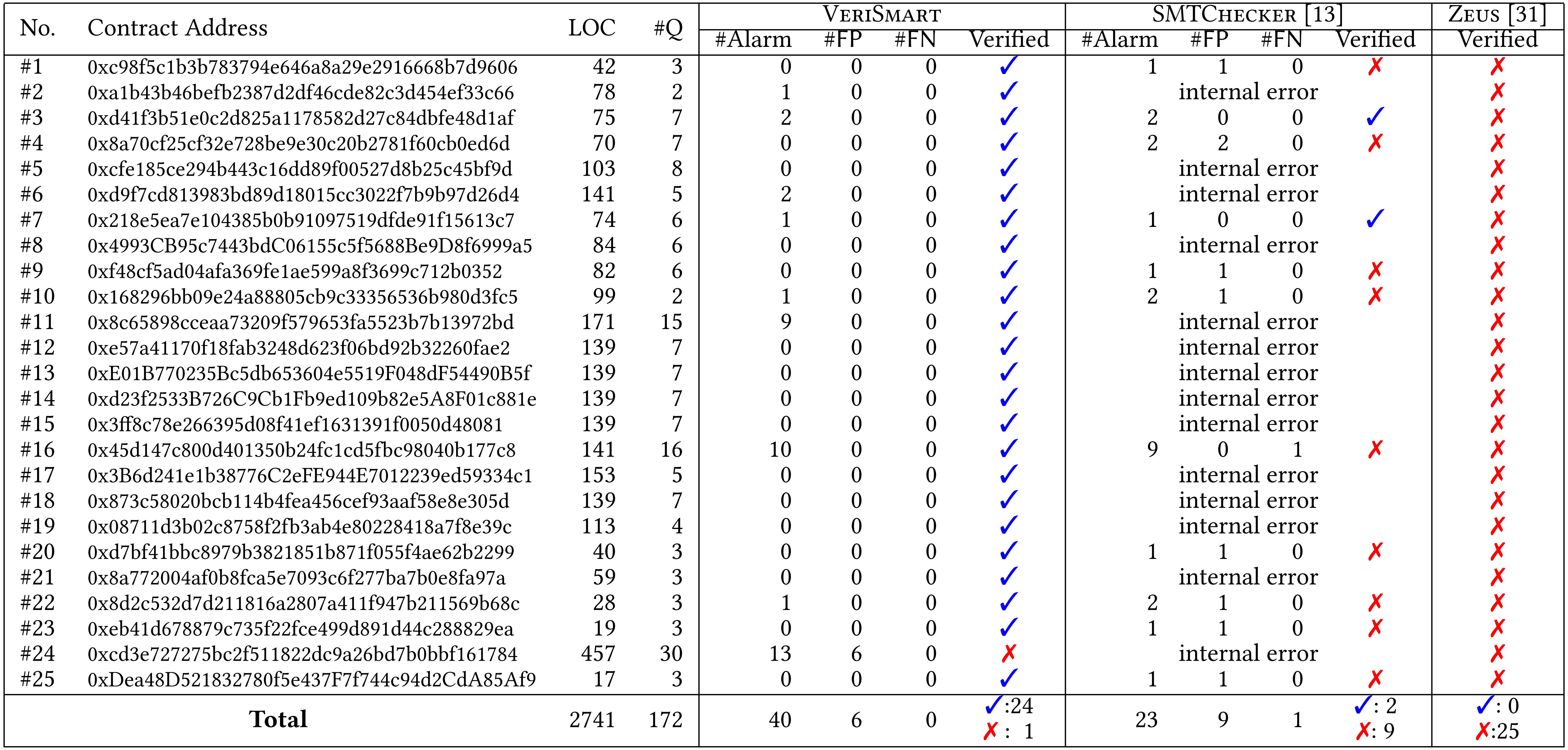
정확도: 99.5%
검출률: 100%

정확도: < 94.6%
검출률: < 70.7%

			VeriSmart			Osiris [43]			Oyente [9, 34]			Mythril [7]			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

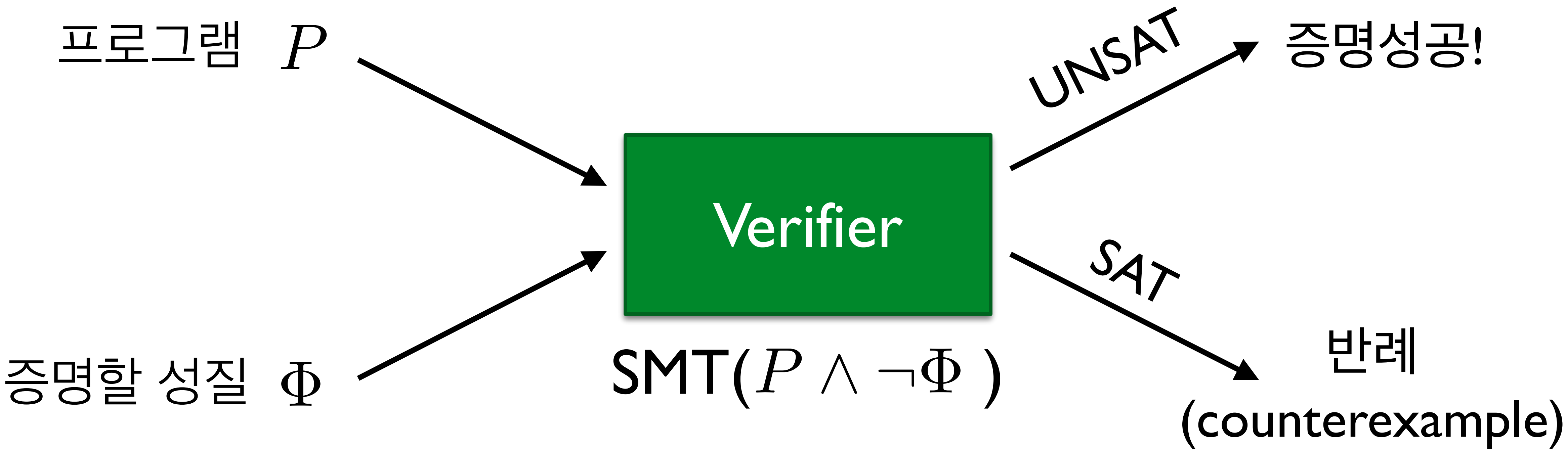
기존 취약점 검증기와 성능 비교

- 기존 검증기들은 스마트 컨트랙트 주요 성질 검증에 실패



No.	Contract Address	LOC	#Q	VeriSmart				SMTChecker [13]				Zeus [31]
				#Alarm	#FP	#FN	Verified	#Alarm	#FP	#FN	Verified	Verified
#1	0xc98f5c1b3b783794e646a8a29e2916668b7d9606	42	3	0	0	0	✓	1	1	0	✗	✗
#2	0xa1b43b46befb2387d2df46cde82c3d454ef33c66	78	2	1	0	0	✓	internal error				✗
#3	0xd41f3b51e0c2d825a1178582d27c84dbfe48d1af	75	7	2	0	0	✓	2	0	0	✓	✗
#4	0x8a70cf25cf32e728be9e30c20b2781f60cb0ed6d	70	7	0	0	0	✓	2	2	0	✗	✗
#5	0xcfe185ce294b443c16dd89f00527d8b25c45bf9d	103	8	0	0	0	✓	internal error				✗
#6	0xd9f7cd813983bd89d18015cc3022f7b9b97d26d4	141	5	2	0	0	✓	internal error				✗
#7	0x218e5ea7e104385b0b91097519dfde91f15613c7	74	6	1	0	0	✓	1	0	0	✓	✗
#8	0x4993CB95c7443bdC06155c5f5688Be9D8f6999a5	84	6	0	0	0	✓	internal error				✗
#9	0xf48cf5ad04afa369fe1ae599a8f3699c712b0352	82	6	0	0	0	✓	1	1	0	✗	✗
#10	0x168296bb09e24a88805cb9c33356536b980d3fc5	99	2	1	0	0	✓	2	1	0	✗	✗
#11	0x8c65898cceaa73209f579653fa5523b7b13972bd	171	15	9	0	0	✓	internal error				✗
#12	0xe57a41170f18fab3248d623f06bd92b32260fae2	139	7	0	0	0	✓	internal error				✗
#13	0xE01B770235Bc5db653604e5519F048dF54490B5f	139	7	0	0	0	✓	internal error				✗
#14	0xd23f2533B726C9Cb1Fb9ed109b82e5A8F01c881e	139	7	0	0	0	✓	internal error				✗
#15	0x3ff8c78e266395d08f41ef1631391f0050d48081	139	7	0	0	0	✓	internal error				✗
#16	0x45d147c800d401350b24fc1cd5fbc98040b177c8	141	16	10	0	0	✓	9	0	1	✗	✗
#17	0x3B6d241e1b38776C2eFE944E7012239ed59334c1	153	5	0	0	0	✓	internal error				✗
#18	0x873c58020bcb114b4fea456cef93aaf58e8e305d	139	7	0	0	0	✓	internal error				✗
#19	0x08711d3b02c8758f2fb3ab4e80228418a7f8e39c	113	4	0	0	0	✓	internal error				✗
#20	0xd7bf41bbc8979b3821851b871f055f4ae62b2299	40	3	0	0	0	✓	1	1	0	✗	✗
#21	0x8a772004af0b8fca5e7093c6f277ba7b0e8fa97a	59	3	0	0	0	✓	internal error				✗
#22	0x8d2c532d7d211816a2807a411f947b211569b68c	28	3	1	0	0	✓	2	1	0	✗	✗
#23	0xeb41d678879c735f22fce499d891d44c288829ea	19	3	0	0	0	✓	1	1	0	✗	✗
#24	0xcd3e727275bc2f511822dc9a26bd7b0bbf161784	457	30	13	6	0	✗	internal error				✗
#25	0xDea48D521832780f5e437F7f744c94d2CdA85Af9	17	3	0	0	0	✓	1	1	0	✗	✗
Total		2741	172	40	6	0	✓:24 ✗ : 1	23	9	1	✓: 2 ✗: 9	✓: 0 ✗:25

VeriSmart 검증 알고리즘

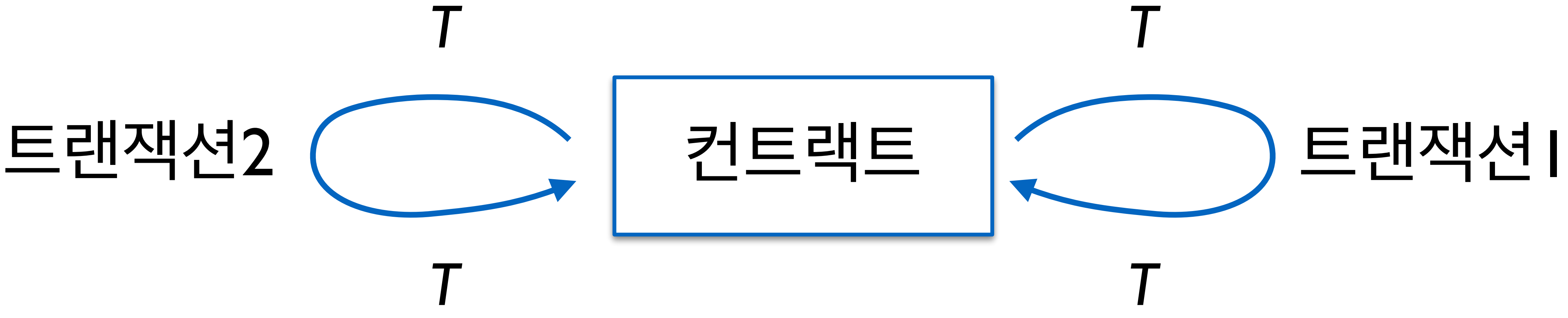


- 프로그램과 증명할 성질을 일차 논리식으로 표현
- 논리식의 satisfiability 여부를 판별

핵심 차별점

- 트랜잭션의 불변 성질 (Transaction invariant)을 자동으로 유추하고 이를 검증에 활용

- 트랜잭션 불변 성질의 조건:
    - 생성자 실행후 성립
    - 각 트랜잭션의 실행전/후에 변함없이 성립



T

트랜잭션2

T

컨트랙트

T

트랜잭션1

T

Netkoin 예제

```
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;
  }
}
```

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = Σbalance

```
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;
  }
}
```

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = Σbalance

```
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 = Σbalance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = Σbalance

```
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 = Σbalance

totalSupply = Σbalance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = Σbalance

```
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 = Σbalance

totalSupply = Σbalance

totalSupply = Σbalance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = Σbalance

```
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 = Σbalance

totalSupply = Σbalance

totalSupply = Σbalance

오버플로우로 착각하기 쉬움

totalSupply = Σbalance

언더플로우로 착각하기 쉬움

Netkoin 예제

트랜잭션 불변 성질:
totalSupply = ∑balance

```
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

오버플로우로 착각하기 쉬움

totalSupply = ∑balance

totalSupply = ∑balance

언더플로우로 착각하기 쉬움

Netkoin 예제

- 트랜잭션의 불변 성질을 이용한 안전성 증명

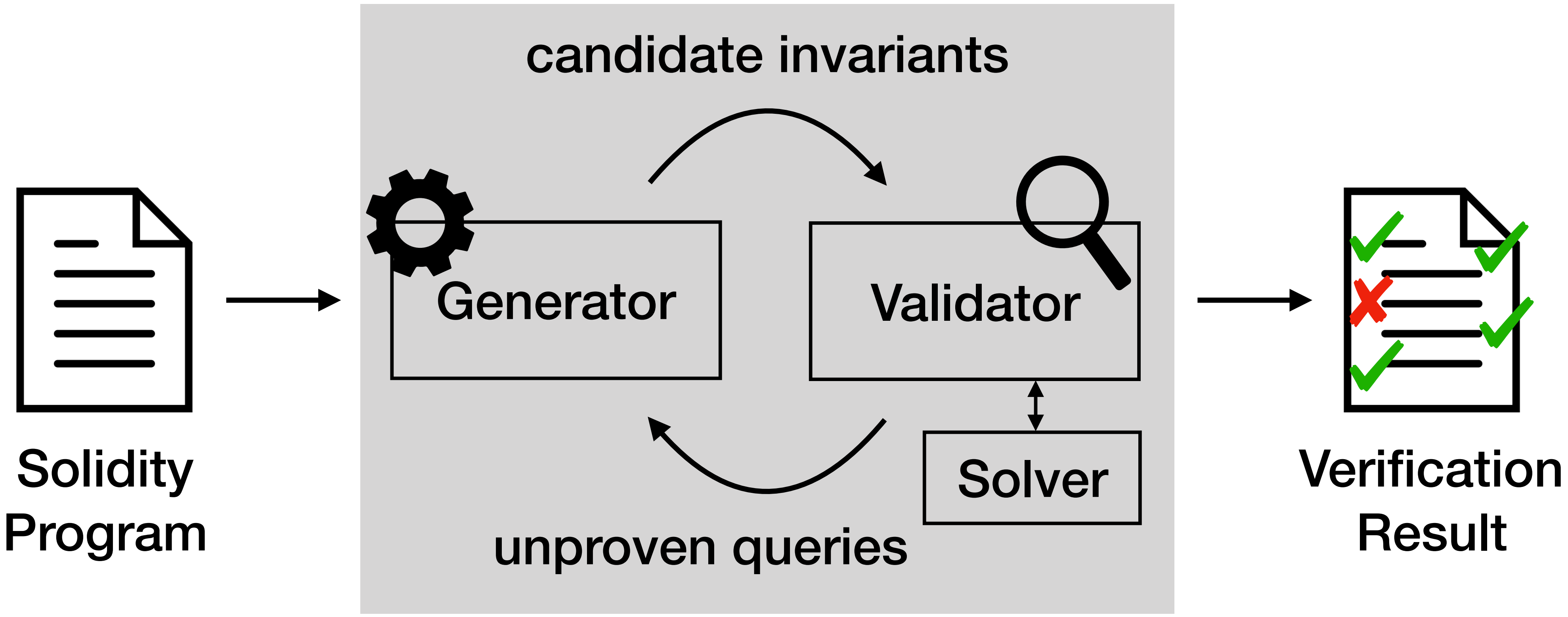
```
require (balance[msg.sender] >= value);
balance[msg.sender] -= value;
totalSupply -= value;
```

assert (totalSupply >= value)

$$\begin{aligned} \text{totalSupply} &= \sum \text{balance} && \text{… transaction invariant} \\ &\geq \text{balance[msg.sender]} && \text{… def. of } \sum \text{balance} \\ &\geq \text{value} && \text{… assumption (require)} \end{aligned}$$

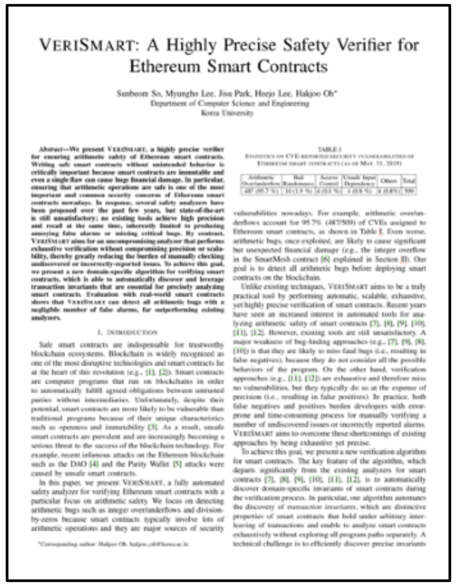
VeriSmart 검증 알고리즘

- 스마트 컨트랙트를 위한 새로운 소프트웨어 검증 기술
- 기존 기술들은 단순한 기호 실행 또는 정적 분석에 의존



Summary

- 스마트 컨트랙트는 보안취약점 자동 분석이 필수
- 현재 스마트 컨트랙트 분석 기술은 성능이 제한적
    - 안전성과 정확성 둘 중 하나를 포기
- VeriSmart: 안전하면서 정확한 스마트 컨트랙트 자동 분석기



VERISMART: A Highly Precise Safety Verifier for Ethereum Smart Contracts

Sunbeom So, Myungho Lee, Jisu Park, Heejo Lee, Hakjoo Oh*

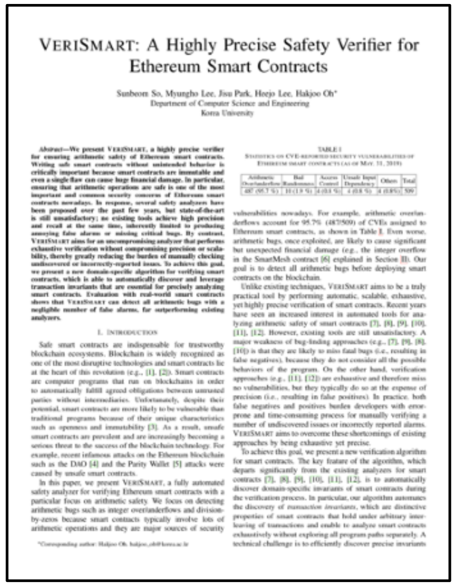
Department of Computer Science and Engineering

Korea University

IEEE S&P 2020 (Oakland) 발표 예정

Summary

- 스마트 컨트랙트는 보안취약점 자동 분석이 필수
- 현재 스마트 컨트랙트 분석 기술은 성능이 제한적
    - 안전성과 정확성 둘 중 하나를 포기
- VeriSmart: 안전하면서 정확한 스마트 컨트랙트 자동 분석기



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

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

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.

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

IEEE S&P 2020 (Oakland) 발표 예정

감사합니다!