스마트 컨트랙트 취약점 검증 기술

오학주

고려대학교 정보대학

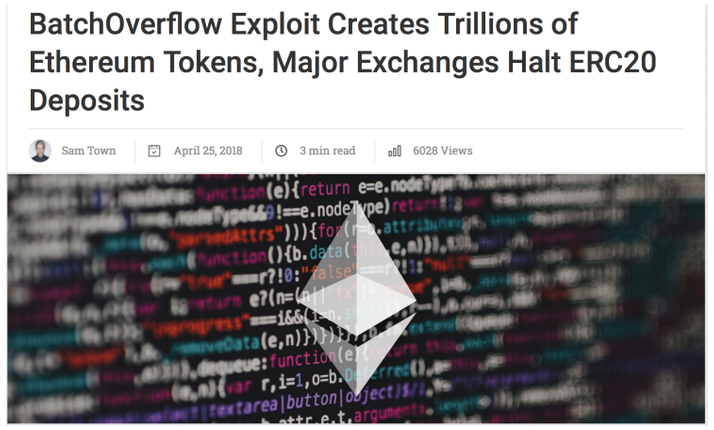
April 25, 2019

스마트 컨트랙트의 안전성 문제

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

The DAO (2016)
750억원

Parity Wallet (2017)
350억원



SmartMesh (2018)
천문학적 금액 인출 시도

정수 오버플로우 취약점 검증기

- Solidity에서는 정수를 유한한 비트로 표현

```
uint public totalSupply;
```

256bit

- 정수 연산시 표현 가능한 범위를 넘어서는 문제가 발생 가능

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

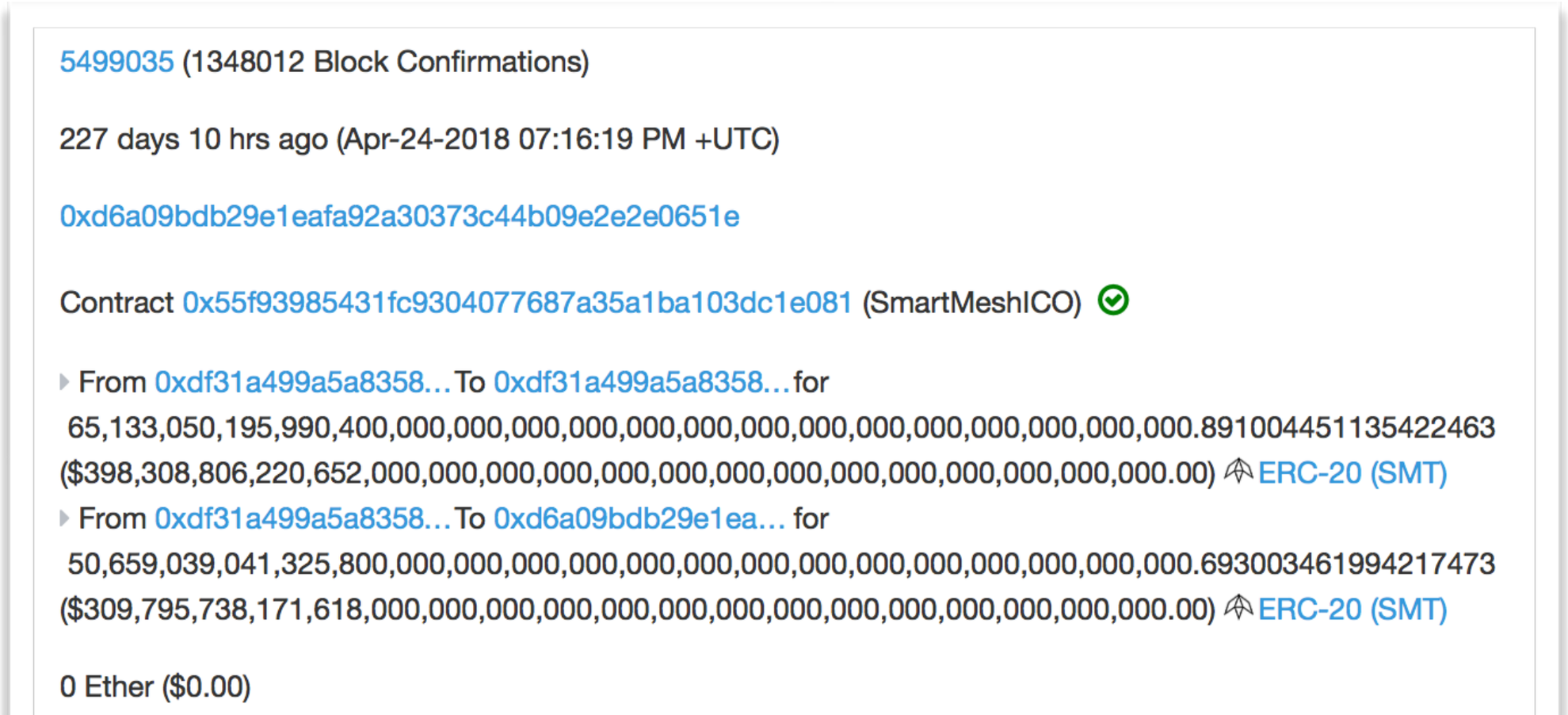
- 오버플로우 유무를 판단하기가 매우 까다로움

- 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

SmartMesh 사례 (2018)

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



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

SmartMesh 사례 (2018)

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

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

CVE-2018-10376

SmartMesh 사례 (2018)

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

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

송금

CVE-2018-10376

SmartMesh 사례 (2018)

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

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

보내는 사람의 잔고가 충분한지 체크

송금

CVE-2018-10376

SmartMesh 사례 (2018)

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

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

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

CVE-2018-10376

SmartMesh 사례 (2018)

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

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

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

오버플로우/언더플로우 발생하지 않음

SmartMesh 사례 (2018)

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

CVE-2018-10376

SmartMesh 사례 (2018)

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

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

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

0!

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

0!

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

8fffff…ff

700…00

CVE-2018-10376

SmartMesh 사례 (2018)

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

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

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

false

false

0!

8fffff…ff

700…00

0!

CVE-2018-10376

기존 스마트 컨트랙트 분석기술의 한계

- 오류 검출기 (e.g., Mythril, Osiris [ACSAC'18], Oyente [CCS'16])

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

Osiris만 검출 가능

CVE-2018-10376

기존 스마트 컨트랙트 분석기술의 한계

- 오류 검출기 (e.g., Mythril, Osiris [ACSAC'18], Oyente [CCS'16])

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

앞의 경우와 비슷한
오류이지만 검출 실패

CVE-2018-14006

기존 스마트 컨트랙트 분석기술의 한계

- 오류 검증기 (SMTChecker, Zeus [NDSS'18])

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

허위 경보 (False alarm)

허위 경보 (False alarm)

VeriSmart

- 목표: 안전하면서 정확한 스마트 컨트랙트 정적 분석기

CVE-2018-10376

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

CVE-2018-14006

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

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

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

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

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

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	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
#1	2018-10299	BEC	299	6	2	0	✓	1	0	✓	1	0	△	0	0	✗	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	0	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	34	0	✓	9	0	✗	4	0	△	0	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	0	0	✗
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✓	2	0	✗	0	0	✗	0	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	✓	10	2	✗	11	4	✗	2	1	✗	0	0	✗
#9	2018-11561	EETHER	146	10	5	0	✓	4	0	✓	2	0	△	0	0	✗	0	0	✗
#10	2018-11687	BTCR	99	20	4	0	✓	2	0	✓	2	0	△	0	0	✗	0	0	✗
#11	2018-12070	SEC	269	40	8	0	✓	5	0	✓	4	0	✗	3	1	✗	0	0	✗
#12	2018-12230	RMC	161	9	5	0	✓	3	0	✓	5	0	✓	0	0	✗	internal error		
#13	2018-13113	ETT	142	9	2	0	N/A	3	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	✓	0	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	✓	0	0	✗	0	0	✗
#18	2018-13131	SpadePreSale	312	3	3	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#19	2018-13132	Spadeico	403	8	6	0	✓	0	0	✗	0	0	✗	0	0	✗	0	0	✗
#20	2018-13144	PDX	103	5	2	0	✓	2	1	✓	2	1	✓	0	0	✗	internal error		
#21	2018-13189	UNLB	335	4	3	0	✓	2	0	✓	3	0	✓	1	0	✗	0	0	✗
#22	2018-13202	MyBO	183	17	11	0	✓	4	0	✗	3	0	✗	0	0	✗	internal error		
#23	2018-13208	MoneyTree	171	17	10	0	✓	4	0	✓	2	0	✗	0	0	✗	0	0	✗
#24	2018-13220	MAVCash	171	15	10	0	✓	4	0	✓	2	0	✗	0	0	✗	internal error		
#25	2018-13221	XT	186	15	10	0	✓	4	0	✓	2	0	✗	2	0	✗	internal error		
#26	2018-13224	VEU	244	16	9	0	✓	5	0	✓	4	0	✗	1	0	✗	internal error		
#27	2018-13227	MCN	172	17	10	0	✓	4	0	✓	2	0	✗	0	0	✗	0	0	✗
#28	2018-13228	CNX	171	17	10	0	✓	3	0	✗	2	0	✗	0	0	✗	0	0	✗
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	0	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	5	0	✓	3	0	✗	1	0	✗	2	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	3	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	✗	internal error		
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	1	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	15	0	✓	7	0	✓	2	0	✓	0	0	✗	internal error		
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	5	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	9	0	✓	0	0	✗	0	0	✗	1	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	0	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	0	0	✗	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	✗	internal error		
#43	2018-13779	YLCToken	180	17	11	0	✓	4	0	✓	6	0	✓	0	0	✗	internal error		
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	0	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	12	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	0	0	△	0	0	✗
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✓	3	0	△	1	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	0	0	△	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✓	2	0	△	1	0	✗	internal error		
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	internal error		
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	0	0	△	internal error		
#52	2018-14006	NGT	249	27	11	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	✗	1	0	✗	internal error		
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	internal error		
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	1	0	✓	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	13	0	✓	internal error		
#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	✗	internal error		
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	3	0	✓	2	0	✗	0	0	✗
Total			12556	973	496	2	✓: 58 △: 0 ✗: 0	234	11	✓: 42 △: 0 ✗: 16	164	13	✓: 23 △: 11 ✗: 24	51	4	✓: 6 △: 3 ✗: 49	0	0	✓:0 △: 0 ✗: 33

기존 취약점 검출기와 성능 비교

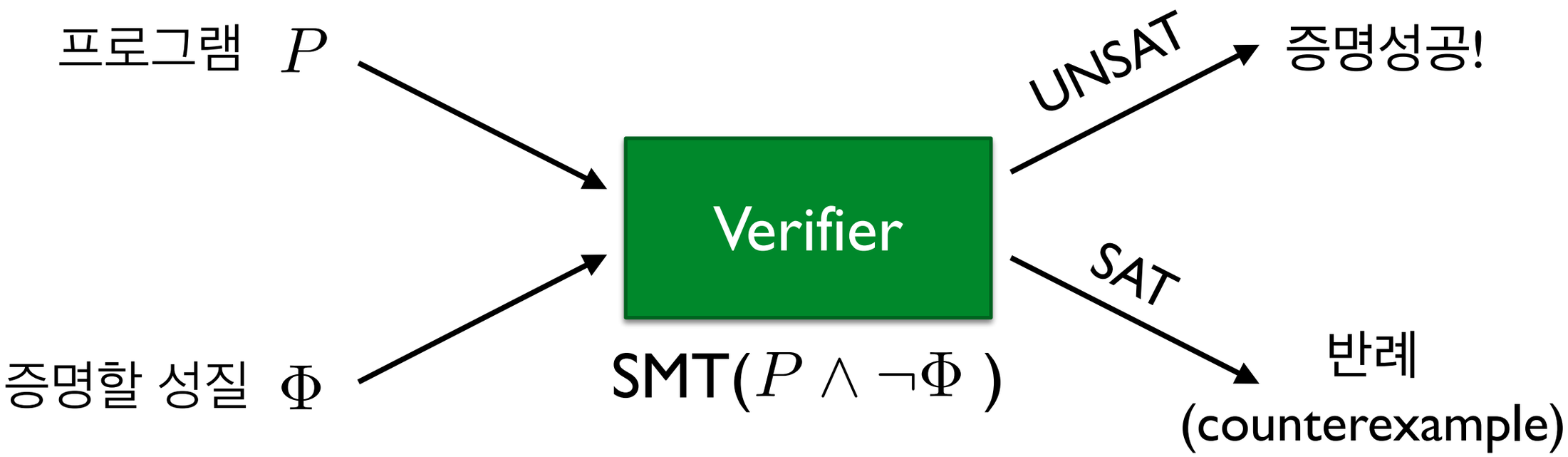
No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	ManticoRE [2] #Alarm	ManticoRE [2] #FP	ManticoRE [2] CVE
#1	2018-10299	BEC	299	6	2	0	✓	1	0	✓	1	0	△	0	0	✗	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	0	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	34	0	✓	9	0	✗	4	0	△	0	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	0	0	✗
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✓	2	0	✗	0	0	✗	0	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	✓	10	2	✗	11	4	✗	2	1	✗	0	0	✗
#9	2018-11561	EETHER	146	10	5	0	✓	4	0	✓	2	0	△	0	0	✗	0	0	✗
#10	2018-11687	BTCR	99	20	4	0	✓	2	0	✓	2	0	△	0	0	✗	0	0	✗
#11	2018-12070	SEC	269	40	8	0	✓	5	0	✓	4	0	✗	3	1	✗	0	0	✗
#12	2018-12230	RMC	161	9	5	0	✓	3	0	✓	5	0	✓	0	0	✗	internal error		
#13	2018-13113	ETT	142	9	2	0	N/A	3	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	✓	0	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	✓	0	0	✗	0	0	✗
#29	2018-13230	DSN	171	17	10	0	✓	4	0	✓	2	0	✗	0	0	✗	0	0	✗
#30	2018-13325	GROW	176	12	5	0	✓	3	0	✗	1	0	✗	2	0	✗	0	0	✗
#31	2018-13326	BTX	135	9	2	0	N/A	3	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	✗	internal error		
#33	2018-13493	DaddyToken	344	40	22	0	✓	8	0	✗	2	0	✗	1	0	✗	internal error		
#34	2018-13533	ALUXToken	191	23	15	0	✓	7	0	✓	2	0	✓	0	0	✗	internal error		
#35	2018-13625	Krown	271	22	9	0	✓	1	0	✗	3	0	✓	0	0	✗	internal error		
#36	2018-13670	GFCB	103	14	11	0	✓	5	1	✓	3	1	✓	1	0	✗	0	0	✗
#37	2018-13695	CTest7	301	17	9	0	✓	0	0	✗	0	0	✗	1	0	✗	0	0	✗
#38	2018-13698	Play2LivePromo	131	8	7	0	✓	7	0	✓	7	0	✓	5	0	✗	0	0	✗
#39	2018-13703	CERB_Coin	262	17	8	0	✓	5	0	✓	2	0	✗	0	0	✗	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	✗	internal error		
#43	2018-13779	YLCToken	180	17	11	0	✓	4	0	✓	6	0	✓	0	0	✗	internal error		
#44	2018-13782	ENTR	171	17	10	0	✓	4	0	✓	2	0	✓	0	0	✗	0	0	✗
#45	2018-13783	JiucaiToken	271	19	12	0	✓	6	0	✓	4	0	✓	0	0	✗	internal error		
#46	2018-13836	XRC	119	22	7	0	✓	5	0	✗	3	0	△	0	0	△	0	0	✗
#47	2018-14001	SKT	152	19	10	0	✓	4	0	✓	3	0	△	1	0	✓	0	0	✗
#48	2018-14002	MP3	83	12	4	0	✓	2	0	✗	2	0	△	0	0	△	timeout (> 3 days)		
#49	2018-14003	WMC	200	15	6	0	✓	3	0	✓	2	0	△	1	0	✗	internal error		
#50	2018-14004	GLB	299	40	8	0	✓	5	0	✓	1	0	△	0	0	✗	internal error		
#51	2018-14005	Xmc	255	29	11	0	✓	8	0	✓	1	0	△	0	0	△	internal error		
#52	2018-14006	NGT	249	27	11	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	✗	1	0	✗	internal error		
#55	2018-14086	SCO	107	16	14	0	✓	7	2	✓	5	2	✗	0	0	✗	internal error		
#56	2018-14087	EUC	174	15	7	0	✓	4	0	✗	4	0	✗	1	0	✓	0	0	✗
#57	2018-14089	Virgo_ZodiacToken	208	30	20	0	✓	12	0	✓	5	0	✓	13	0	✓	internal error		
#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	✗	internal error		
#60	2018-18665	NXX	79	7	5	0	✓	4	0	✓	3	0	✓	2	0	✗	0	0	✗
Total			12556	973	496	2	✓: 58 △: 0 ✗: 0	234	11	✓: 42 △: 0 ✗: 16	164	13	✓: 23 △: 11 ✗: 24	51	4	✓: 6 △: 3 ✗: 49	0	0	✓:0 △: 0 ✗: 33

		VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	ManticoRE [2] #Alarm	ManticoRE [2] #FP	ManticoRE [2] CVE
Total	12556 973	496	2	✓: 58 △: 0 ✗: 0	234	11	✓: 42 △: 0 ✗: 16	164	13	✓: 23 △: 11 ✗: 24	51	4	✓: 6 △: 3 ✗: 49	0	0	✓:0 △: 0 ✗: 33

기존 취약점 검증기와 성능 비교

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

Software Verification 기술 활용



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

예제

- 증명할 성질을 assert 문으로 표현

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

예제

- 프로그램과 증명할 성질의 반대(negation)를 논리식으로 표현

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

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

예제

- 프로그램과 증명할 성질의 반대(negation)를 논리식으로 표현

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

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

SAT/SMT solver: unsatisfiable!

예제

- 증명이 불가능한 경우에는 반례를 생성

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

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

예제

- 증명이 불가능한 경우에는 반례를 생성

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

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

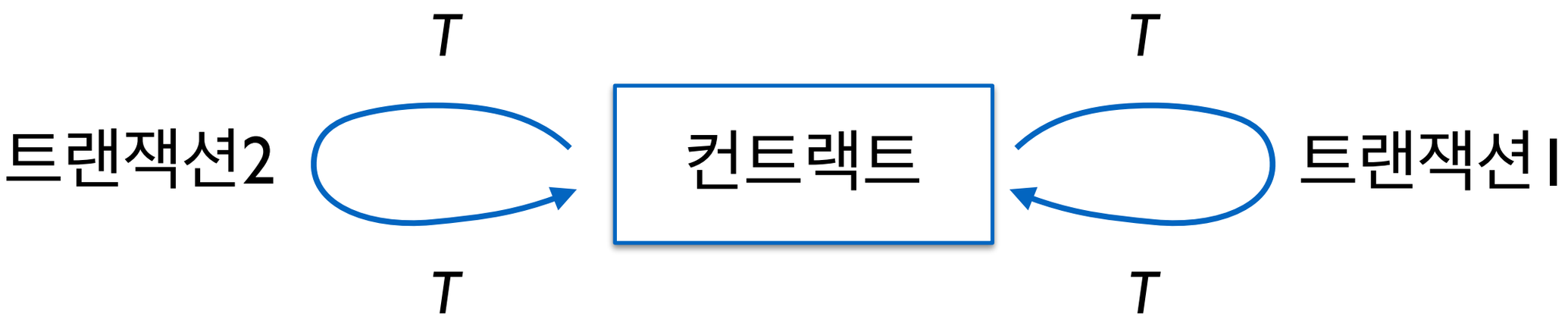
SAT/SMT solver:
satisfiable when a=1 and b=0

정수 오버플로우 안전성 명세

- `a + b` ⇨ `assert (a + b >= a)`
- `a * b` ⇨ `assert ((a * b) / a == b)`
- `a - b` ⇨ `assert (a >= b)`

정확한 검증의 핵심

- 트랜잭션의 불변 성질 (Transaction invariant)을 유추하고 이를 검증에 활용하는 것이 필요
- 트랜잭션 불변 성질의 조건:
  - 생성자 실행후 성립
  - 각 트랜잭션의 실행전/후에 변함없이 성립



Netkoin 예제

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

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

totalSupply = ∑balance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

totalSupply = ∑balance

totalSupply = ∑balance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

오버플로우로 착각하기 쉬움

언더플로우로 착각하기 쉬움

Netkoin 예제

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

트랜잭션 불변 성질:
totalSupply = Σbalance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

오버플로우로 착각하기 쉬움

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)

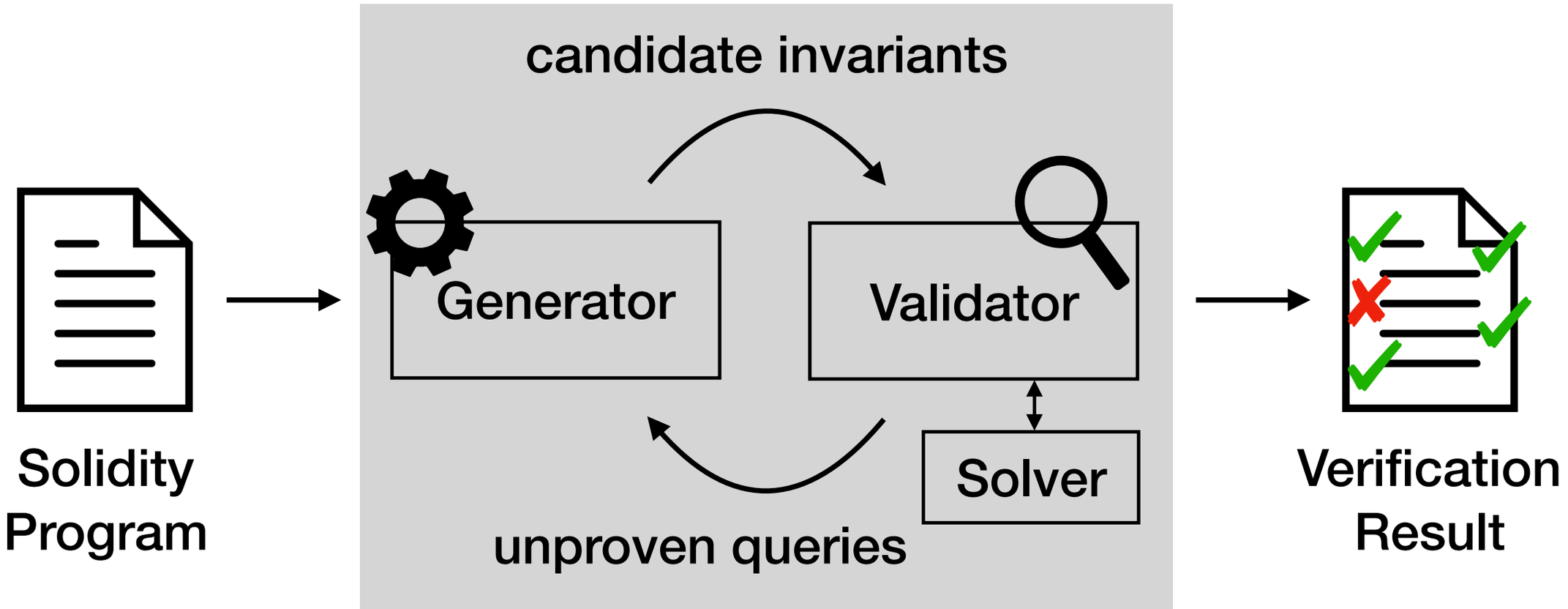
totalSupply = ∑balance … transaction invariant

>= balance[msg.sender] … def. of ∑balance

>= value … assumption (require)

VeriSmart 검증 알고리즘

- 프로그램 증명과 불변 성질 합성을 동시에 진행



Summary

- VeriSmart: 안전하고 정확한 스마트 컨트랙트 안전성 분석기
- Future work
  - 정교한 트랜잭션 불변 성질 추론 $\forall x.\texttt{totalLocked[x]} = \sum_i \texttt{locked[x][i]}$
  - 일반 취약점 검출기로 확장 (Reentrancy, access control, DDoS etc)
  - 취약점 익스플로잇 자동 생성