2020 IT 21
Global Conference
Digital New Deal
Technology Essentials
디지털 뉴딜 기술 핵심

Session 5-2

스마트 컨트랙트 안전성 검증 기술

오학주 교수 (고려대학교)

[요약문]

스마트 컨트랙트의 보안 취약점을 자동으로 검출하는 기술인 VeriSmart를 소개한다. 스마트 컨트랙트는 블록체인 기술의 핵심 가운데 하나로, 임의의 계산 가능한 거래를 중개인 없이 가능하게 하는 컴퓨터 프로그램이다. 스마트 컨트랙트는 한번 작성되면 수정이 불가능하고, 주로 금전적 거래를 수행하며, 코드가 공개되어 있다는 점에서 통상적인 프로그램들에 비해 더 높은 수준의 안전성 검증을 필요로 한다. 지금까지 스마트 컨트랙트의 보안 취약점을 검출하거나 검증하는 다양한 기술들이 발표되었지만 모두 성능이 제한적이었다. 취약점 검출을 목표로 하는 기술들은 모든 문제를 잡아내지 못한다는 제한이 있어왔고, 취약점 검증을 목표로 하는 기술들은 취약점이 아닌데 취약점이라고 보고하는 허위 경보 문제가 늘 있어왔다. VeriSmart는 이러한 기존 기술들의 문제들을 해결하여 대상으로 하는 모든 취약점을 검출하면서 허위 경보를 최소화 하였다. 본 발표에서는 VeriSmart의 동작 방식을 설명하고, 기존의 스마트 컨트랙트 취약점 분석 도구들과 비교한다.

[발표자 약력]

2005년 한국과학기술원 전산학 학사
2007년 서울대학교 컴퓨터공학 석사
2012년 서울대학교 컴퓨터공학 박사
2012년~2015 서울대학교 박사후 연구원
2015년~현재 고려대학교 조교수, 부교수

관심분야 : 소프트웨어 보안, 소프트웨어 분석

VeriSmart

스마트 컨트랙트 안전성 검증기

오학주

고려대학교 정보대학 컴퓨터학과

스마트 컨트랙트

블록체인 1.0

코인 거래만 가능

vs.

블록체인 2.0

임의의 거래가 가능

Key: 스마트 컨트랙트

2

스마트 컨트랙트 생김새

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

사용자의 계좌 정보

송금

잔고가 충분하면
거래를 실행

데이터

생성자

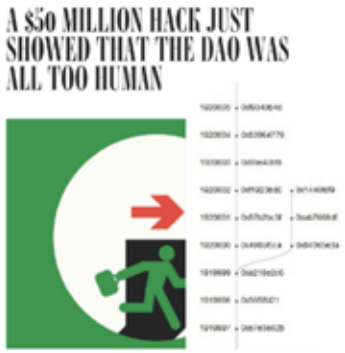
함수

함수

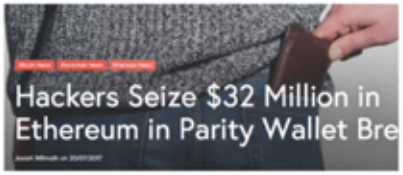
3

스마트 컨트랙트의 위험성

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



The DAO (2016)
750억원



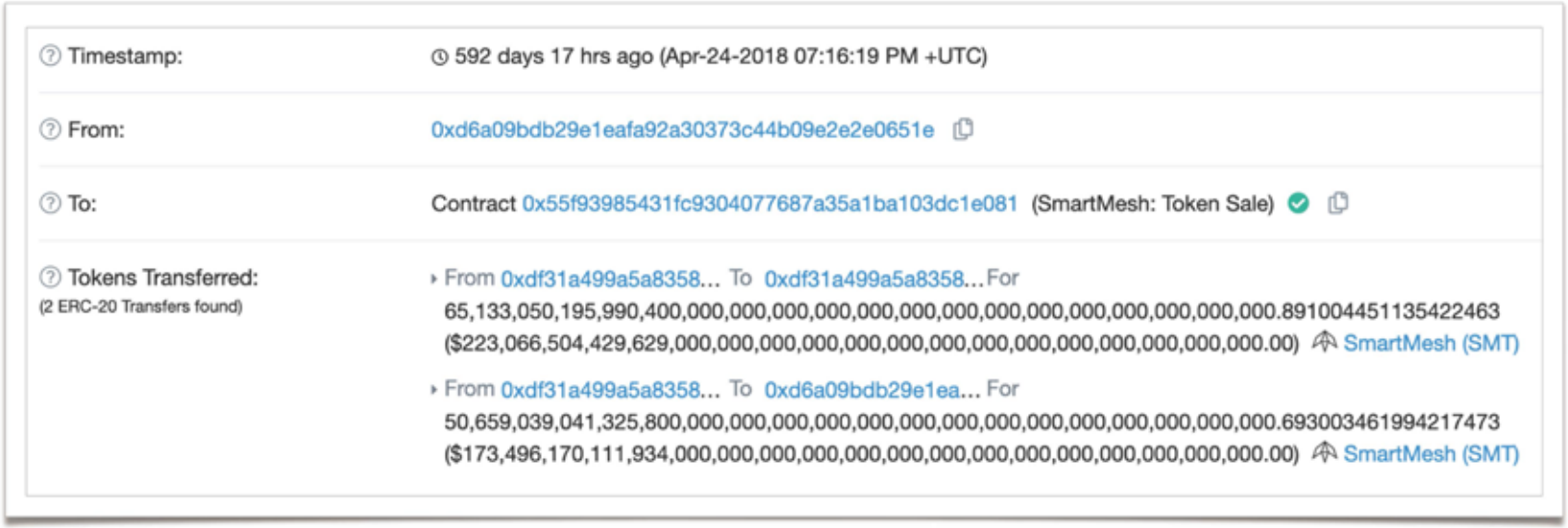
Parity Wallet (2017)
350억원

SmartMesh (2018)
천문학적 금액 인출 시도

4

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
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

오버플로우 체크

송금

(실질적) 오버플로우/언더플로우 발생하지 않음

6

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!

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 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

(2019.05)

8

스마트 컨트랙트 자동 분석 기술

- 오류 검출기 (bug-detector)

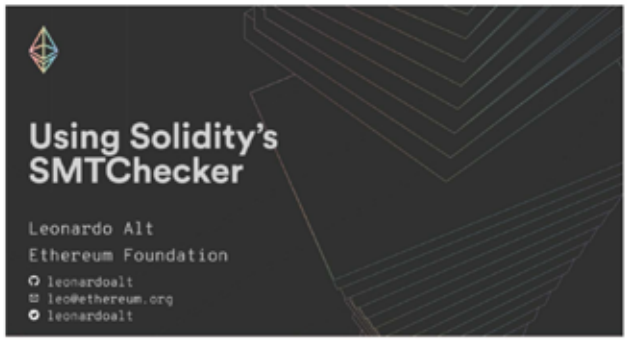
Oyente

Mythril

manticore

Osiris

- 오류 검증기 (verifier)



9

현재 자동 분석 기술의 한계 (1)

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

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

Osiris만 검출 가능

CVE-2018-10376

10

현재 자동 분석 기술의 한계 (1)

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

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

앞의 경우와 비슷한 오류
이지만 검출 모두 실패

CVE-2018-14006

11

현재 자동 분석 기술의 한계 (2)

- 오류 검증기(SMTChecker, Zeus): 허위경보 존재

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

허위 경보 (False alarm)

허위 경보 (False alarm)

12

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 <  [to] ||
      balance[msg.sender] +  balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] +=
  balance[from] -= value  ee;
  return true;
}
```

CVE-2018-14006

```
function multipleTransfer(address[] to, uint value) {
 require(value * to.leng  h > 0
 require(balances[msg.s  er  value * to.length);
 balances[msg.sender] -  * to.length;
 for (uint i = 0; i < to  h; ++i) {
   balances[to[i]] +=
 }
 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 (addre  o, uint value) public
  returns (bool) {
    require (bala  ender] >= value);
    balance[msg.se  = value;
    balance[to] +=  e;
    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

13

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9], [26] #Alarm	Oyente [9], [26] #FP	Oyente [9], [26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL						3	0	✓	2	0	✓						
#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		

정확도: 99.5%
검출률: 100%

정확도: < 94.6%
검출률: < 70.7%

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

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9], [26] #Alarm	Oyente [9], [26] #FP	Oyente [9], [26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#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	BucaiToken	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

14

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

15

VeriSmart 핵심 차별점

- 트랜잭션 불변 성질 (Transaction invariant) 자동 추론

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

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

totalSupply = ∑balance

16

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

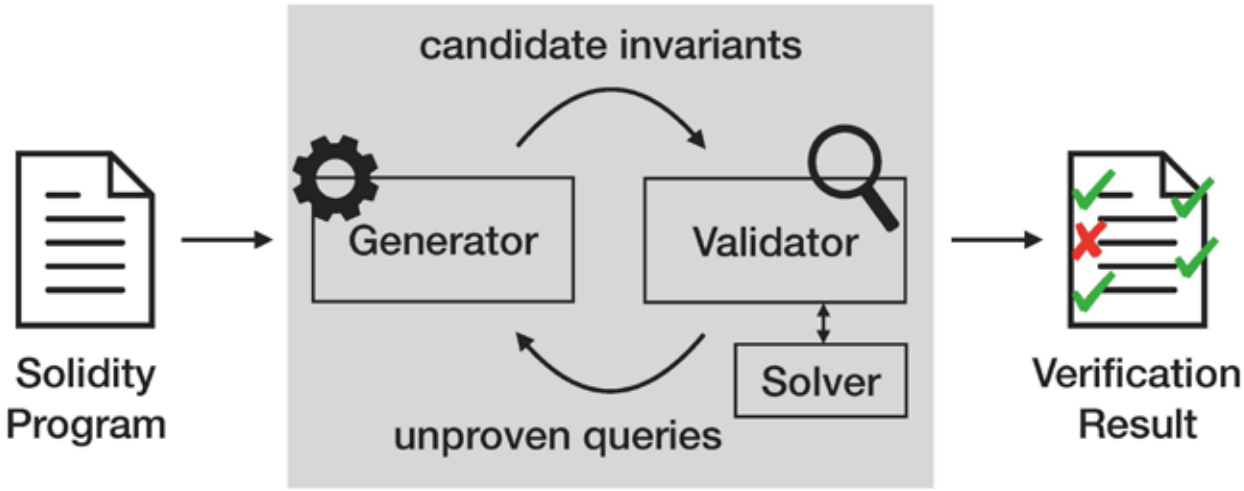
≥ value … assumption (require)

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

17

VeriSmart 검증 알고리즘

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



18

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

19

기존 취약점 검출기와 성능 비교

No.	CVE ID	Name	LOC	#Q	VeriSmart #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [7] #Alarm	Osiris [7] #FP	Osiris [7] CVE	Oyente [9], [26] #Alarm	Oyente [9], [26] #FP	Oyente [9], [26] CVE	Mythril [8] #Alarm	Mythril [8] #FP	Mythril [8] CVE	MantiCore [10] #Alarm	MantiCore [10] #FP	MantiCore [10] CVE
#1	2018-10299	BEC	299	6	2	0	✓	0	0	✗	1	0	△	2	0	✓	0	0	✗
#2	2018-10376	SMT	294	22	13	0	✓	1	0	✓	2	0	✗	1	0	✗	timeout (> 3 days)		
#3	2018-10468	UET	146	27	14	0	✓	9	0	✗	8	0	✓	5	0	✓	0	0	✗
#4	2018-10706	SCA	404	48	33	0	✓	9	0	✗	4	0	△	2	0	✗	internal error		
#5	2018-11239	HXG	102	11	7	0	✓	6	0	✓	2	0	✗	3	0	✓	2	0	✓
#6	2018-11411	DimonCoin	126	15	7	0	✓	5	0	✗	5	0	✓	5	0	✓	3	0	✓
#7	2018-11429	ATL						3	0	✓	2	0	✓						
#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	2018-13250	USN	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	BucaiToken	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 #Alarm	VeriSmart #FP	VeriSmart CVE	Osiris [43] #Alarm	Osiris [43] #FP	Osiris [43] CVE	Oyente [9, 34] #Alarm	Oyente [9, 34] #FP	Oyente [9, 34] CVE	Mythril [7] #Alarm	Mythril [7] #FP	Mythril [7] CVE	MantiCore [2] #Alarm	MantiCore [2] #FP	MantiCore [2] CVE
Total	12493	976	492	2	✓: 58 △: 0 ✗: 0	240	13	✓:41 △: 0 ✗ :17	171	14	✓:20 △:15 ✗ :23	94	10	✓:10 △: 1 ✗ :46	14	0	✓: 2 △: 0 ✗ :42

기존 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 (...,...)
```

21

VeriSmart 한계

- 복잡한 불변 성질은 유추하지 못하고 허위 경보 발생

```
1  function unlockReward(address addr, uint value) {
2    require(totalLocked[addr] > value);
3    require(locked[addr][msg.sender] >= value);
4    if(value == 0) value = locked[addr][msg.sender];
5    totalLocked[addr] -= value;  // false positive
6    locked[addr][msg.sender] -= value;
7  }
```

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

22

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.,

```
1  contract BTX {
2    mapping (address => uint) public balance;
3    uint public totalSupply;
4
5    constructor () {
6      totalSupply = 10000;
7      balance[msg.sender] = 10000;
8    }
9
10   function transfer (address to, uint value) {
11     require (balance[msg.sender] >= value);
12     balance[msg.sender] -= value;
13     balance[to] += value; // Safe
14   }
15
16   function transferFrom (address from, address to, uint
         value) {
17     require (balance[from] >= value);
18     balance[to] += value; // Safe
19     balance[from] -= value;
20   }
21 }
```

23

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

24

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

25

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