SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models

Sunbeom So
GIST

Hakjoo Oh
Korea University

5 Dec 2023
ESEC/FSE 2023 @ San Francisco, USA

Smart Contract

- Digital contract written in programming languages.

```
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  }
```

Solidity Contract

Smart Contract

- Digital contract written in programming languages.

```
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  }
```

State (global) variables

Constructor

Function

Function

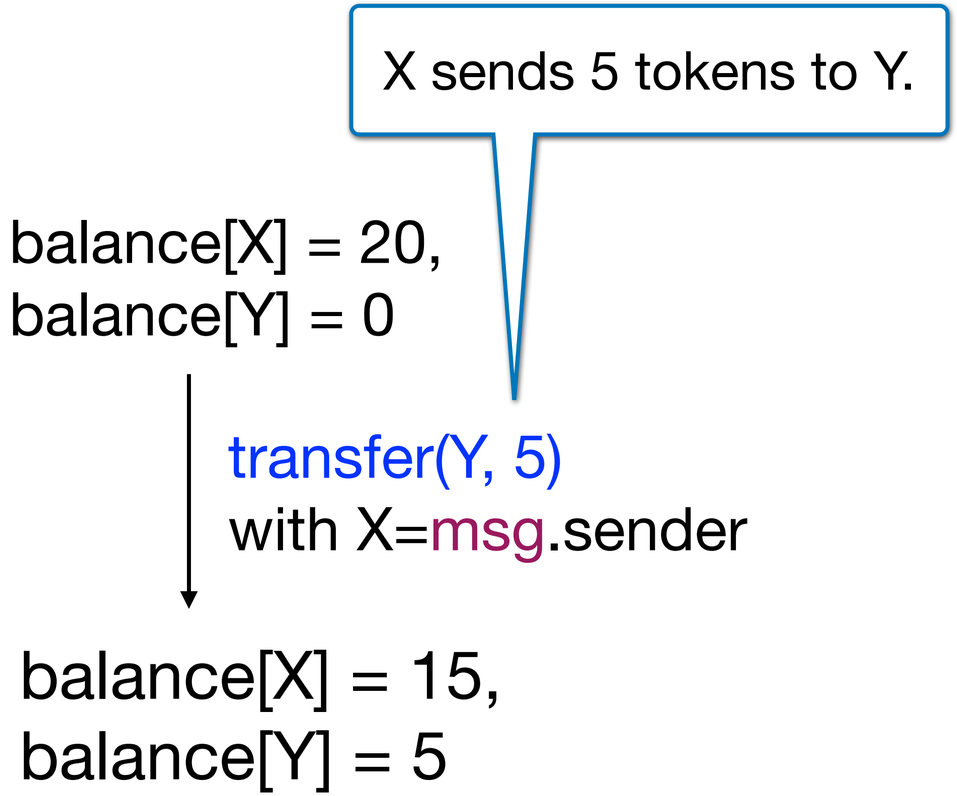
Solidity Contract

Smart Contract

- Transaction execution = Function invocation

```
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  }
```

Solidity Contract



Importance of Safe Smart Contracts

- **Immutable** once deployed on blockchains.
- **Huge financial damage** once exploited.

(2016)

(2018)

(2021)

(2023)

Goal: Ensuring safety before deployment

SmartFix's Goal:
Fixing Vulnerable Contracts Automatically and Safely

CVE-2018-11411

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

SmartFix's Goal:
Fixing Vulnerable Contracts Automatically and Safely

CVE-2018-11411

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

Our Goal

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

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]: Rely only on inserting runtime checks

※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

To pass line 8:
A: `balance[to] + value >= balance[to]`

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]: Rely only on inserting runtime checks

※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

To pass line 8:
A: `balance[to] + value >= balance[to]`

B: `balance[to] + value <= balance[to]`

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]: Rely only on inserting runtime checks

※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

value == 0 in if-branch (by A,B)

To pass line 8:
A: balance[to] + value >= balance[to]

B: balance[to] + value <= balance[to]

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]:
Rely only on inserting runtime checks

※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

value!=0 after line 2

value == 0 in if-branch (by A,B)

To pass line 8:
A: balance[to] + value >= balance[to]

B: balance[to] + value <= balance[to]

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]:
Rely only on inserting runtime checks

※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Limitations of Existing Techniques: Simple Patch Only

Rely on a single repair template for each bug type

CVE-2018-11411

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

value!=0 after line 2

Incorrect Patch (Deadcode)

value == 0 in if-branch (by A,B)

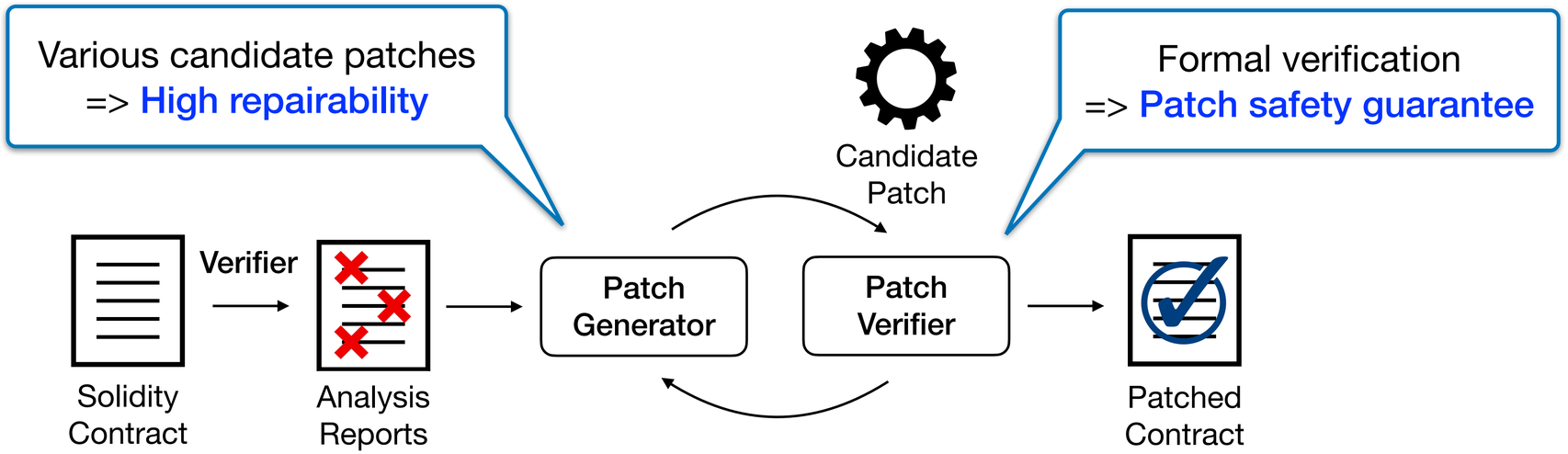
To pass line 8:
A: balance[to] + value >= balance[to]

B: balance[to] + value <= balance[to]

sGuard [IEEE S&P'21], SmartShield [SANER'20], Elysium [RAID'22]:
Rely only on inserting runtime checks

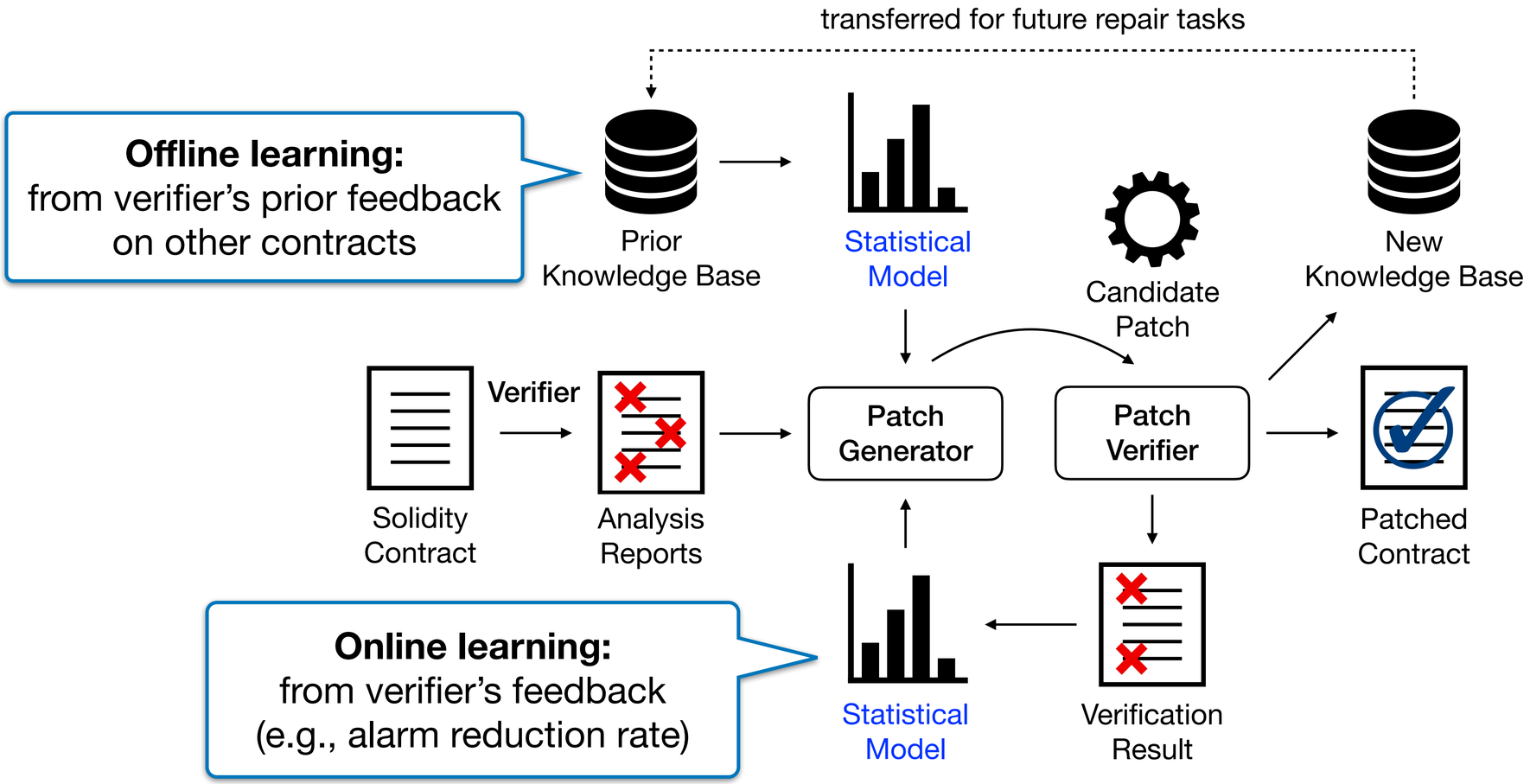
※ **safeAdd/safeSub:** raise exceptions if over/underflows occur at runtime

Basic Approach: Generate-and-Verify



Challenge: Repair efficiency
(large search space + patch verification cost)

SmartFix Approach:
Speeding up Generate-and-Verify using Statistical Models



Key Idea: prioritize likely candidates using learned models

Quantifying Verifier's Feedback

- The most important part for learning useful statistical models.
- **Main metric**: alarm reduction rate
  - (Example 1) $\#\text{Alarm}_{\text{org}} = 5$, $\#\text{Alarm}_{\text{pat}} = 3 \rightarrow 0.4 \left(= \frac{5-3}{5} \right)$
  - (Example 2) $\#\text{Alarm}_{\text{org}} = 5$, $\#\text{Alarm}_{\text{pat}} = 7 \rightarrow$ negative score
- The complete definition of the quantifying function can be found in our paper.

Evaluation: Setup

- **Comparison Target**: sGuard [IEEE S&P'21]
  - State-of-the-art fixing tool for Solidity smart contracts.
- **Benchmark**: collected 361 contracts from multiple sources (5 types of known security bugs)
  - **Integer over/underflow:** 200 CVE-reported contracts
  - **Ether-Leak, Suicidal:** 104 from SmartTest [USENIX Sec'21]
  - **Reentrancy:** 28 from (SODA [NDSS'20],SmartBugs [ICSE'20]) + 17 by bug-injection + 2 from the wild
  - **Dangerous tx.origin:** 1 from SmartBugs [ICSE'20] + 9 by bug-injection

Evaluation: Fix Rate (vs. sGuard [IEEE S&P'21])

Bug Type	#Bug	SmartFix					sGuard [IEEE S&P '21]				
		#BugRun	#Generated	#Correct	**Success Rate**	**Accuracy**	#BugRun	#Generated	#Correct	**Success Rate**	**Accuracy**
IO	229	228	218	218	95.6%	100.0%	170	103	103	60.6%	100.0%
RE	52	51	46	46	90.2%	100.0%	33	33	29	87.9%	87.9%
TX	12	12	12	12	100.0%	100.0%	2	2	2	100.0%	100.0%
EL	137	134	83	76	56.7%	91.6%	n/a	n/a	n/a	n/a	n/a
SU	53	51	40	34	66.7%	85.0%	n/a	n/a	n/a	n/a	n/a
IO+RE+TX	293	291	276	276	**94.8%**	**100.0%**	205	138	134	**65.4%**	**97.1%**
Total	483	476	399	386	**81.1%**	**96.7%**	-	-	-	-	-

Fix Success Rate:
94.8% (Ours) vs. 65.4% (sGuard)

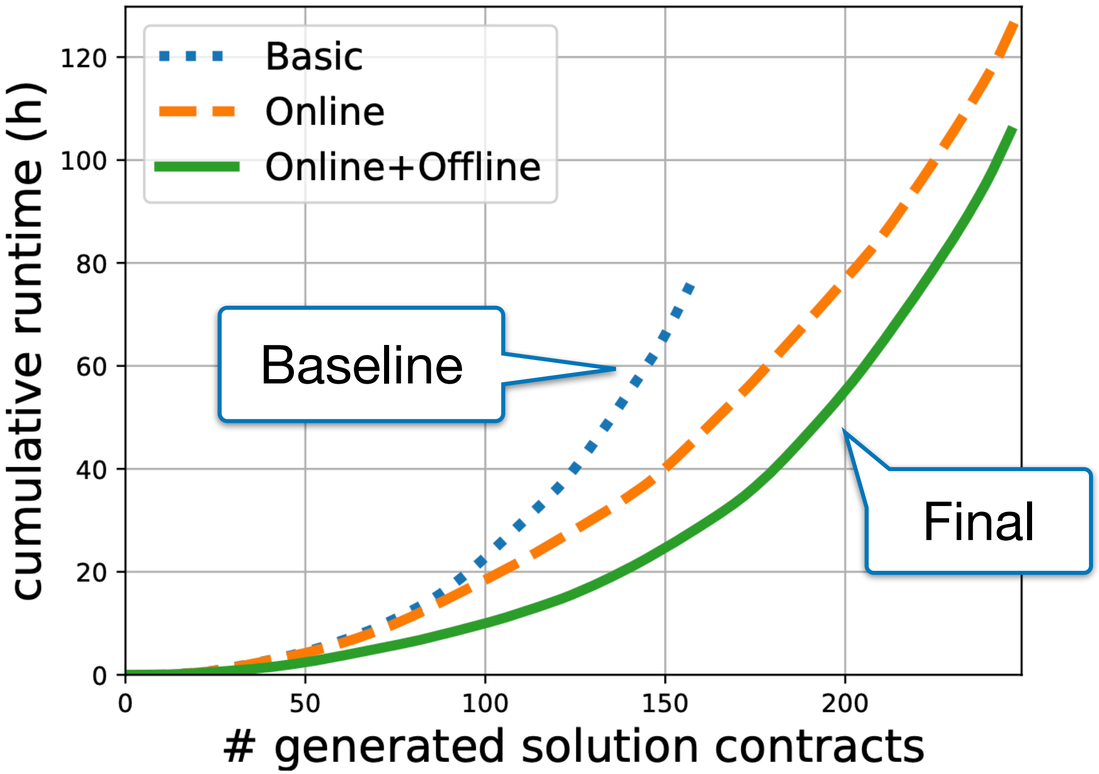
Evaluation: Patch Simplicity (vs. sGuard)

Dataset	Sol	SmartFix		sGuard [42]		$\frac{\text{SmartFix}}{\text{sGuard}}$
		#BC	#NR	#BC	#NR	#BC+#NR
IO Bench	73	213	0	783	0	27.2%
RE Bench	25	27	1	53	61	24.6%
TX Bench	2	3	0	10	0	30.0%
Total	100	243	1	846	61	26.9%

- SmartFix accurately identified where to apply patches.
  - Thanks to the use of verification-based patch validation.

Patch Size Reduction 73.1%

Evaluation: Impact of Using Statistical Models



- #Generated Bug-free contracts
  - Baseline (157) vs. Final (246)

Performance Up 56.7%

$$= \frac{246 - 157}{157}$$

Summary

- **SmartFix's Goal:** high repairability and patch safety
- **Key Idea:** generate-and-verify repair + statistical models
- **Source code & Benchmark:** https://github.com/kupl/SmartFix-Artifact
- In the paper:
  - Details of the learning process (e.g., feature vector representation)
  - Functional regression detection to reject likely incorrect patches
  - Optimizations

Thank you!

Backup Slide

Evaluation: Patch Simplicity (vs. sGuard)

sGuard's Patch [IEEE S&P'21]

```
...
1     function mintToken (address target, uint amount) {
2       require(owner == msg.sender);
        require(balance[target] + amount >= amount);
3       balance[target] += amount;
        require(totalSupply + amount >= totalSupply);
4       totalSupply += amount;
5     }
6
7
8     function burnFrom (address from, uint value)
9     public returns (bool) {
10      require(balance[from] >= value);
11      require(allowed[from][msg.sender] >= value);
        require(balance[from] >= value);
12      balance[from] -= value;
        require(allowed[from][msg.sender] >= value);
13      allowed[from][msg.sender] -= value;
        require(totalSupply >= value);
14      totalSupply -= value;
15      return true;
16    }
...
```

Blindly insert runtime checks to fix IO

SmartFix's Patch (Ours)

```
...
1     function mintToken (address target, uint amount) {
2       require(owner == msg.sender);
3       balance[target] += amount;
        require(totalSupply + amount >= totalSupply);
4       totalSupply += amount;
5     }
6
7
8     function burnFrom (address from, uint value)
9     public returns (bool) {
10      require(balance[from] >= value);
11      require(allowed[from][msg.sender] >= value);
12      balance[from] -= value;
13      allowed[from][msg.sender] -= value;
14      totalSupply -= value;
15      return true;
16    }
...
```

Patch Verifier:
After fixing line 4, All Safe!