Property-Based Automated Repair of DeFi Protocols

Palina Tolmach *, Yi Li *, Shang-Wei Lin *
* Nanyang Technological University
+ Institute of High Performance Computing, A*STAR

ASE 2022
Oct 12, 2022
Definitions

Blockchain:
Append-only distributed database of global state

Smart Contract:
Program that can write into blockchain

Primary application:
“Decentralized” fiinance – banking, exchanges, etc.
Solidity Smart Contracts & DeFi

• **Computer programs** running on blockchain
• ** Govern billions of dollars** that can be stolen

```solidity
contract Token {
    mapping (address => uint256) public balances;
    string public name;
    constructor(string memory _tokenName) public payable {
        name = _tokenName;
        deposit();
    }
    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }
    ...
}
```

https://www.defipulse.com/
DeFi Attacks

$1.3bn lost in 2021
$1.6bn lost in 2022H1

Common Issues
• Bad practices, common mistakes
  • Integer overflows
  • SC-specific security issues
• Detectable by static analysis

Fidelity Issues
• “Logical” bugs
• Especially problematic in DeFi
• Impossible to find using patterns

Automated repair of logical issues in DeFi smart contracts
(Typical) Smart Contract Repair

Buggy smart contract → Fault localization → Patch generation → Patch Validation → Fixed smart contract

Pattern-based vulnerability detection and patch generation limited to a set of predefined vulnerabilities
iToken Duplication Issue ($8M loss)

```solidity
contract iToken ... {

    function transfer(address _from, address _to, uint256 _value) public returns (bool res) {
        require(_from != _to);
        uint256 _balancesFrom = balances[_from];
        uint256 _balancesTo = balances[_to];
        require(_balancesFrom >= _value);
        uint256 _balancesFromNew = _balancesFrom - _value;
        balances[_from] = _balancesFromNew;
        uint256 _balancesToNew = _balancesTo + _value;
        balances[_to] = _balancesToNew;
    }
}
```

«the sum of sender and recipient’s balances before and after transfer doesn’t change»

https://fullycrypto.com/bzx-suffers-token-duplication-incident
DeFinery

• Automated property-based repair for smart contracts
• Combining search-based patch generation with semantic inference

Symbolically executes the trace, generates valid and invalid test cases
Generates patches using AST-based mutations, evaluates the patches using test cases
Checks conditional equivalence between original and patched smart contracts based on symbolic summaries
### Evaluation

- **Dataset**: 9 smart contracts (5 exploited DeFi protocols, 4 from SmartBugs dataset)
- **Average time**: 53 seconds
- **Fixes**: missing pre-/postconditions and variable updates, common security issues

<table>
<thead>
<tr>
<th>#</th>
<th>Smart Contract</th>
<th>Patch</th>
<th>Property</th>
<th>DeFinery</th>
<th>Result</th>
<th>SmartShield</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xForce</td>
<td>+</td>
<td>require(result); User didn’t receive xForce if he didn’t provide any Force</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>Confused_Sign</td>
<td>-</td>
<td>require(amt &gt;= bal[msg.sender]); User can’t withdraw more than he deposited; he can receive a refund</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
<td>require(amt &lt;= bal[msg.sender]);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Value</td>
<td>+</td>
<td>initialized = true; The staked token can’t be changed</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>4</td>
<td>Uranium</td>
<td></td>
<td>require(balance0 * balance1 &gt;= -_res0 * _res1 * 10**2); (Constant) product of pool reserves is non-decreasing</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ _res0 * _res1 * 10**2);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Refund_NoSub</td>
<td>+</td>
<td>balances[msg.sender] = 0; Sum of balances is constant; the user can receive a refund</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>6</td>
<td>Unprotected</td>
<td>+</td>
<td>require(owner == msg.sender); Owner can only be changed to a trusted address</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>7</td>
<td>iToken</td>
<td>+</td>
<td>require(_from != _to); Constant sum of balances is preserved by a transfer</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>8</td>
<td>cToken</td>
<td>-</td>
<td>amp.transfer(borrower, amount); Protocol balance can’t decrease</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>borrowBalance[borrower] := amount;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
<td>amp.transfer(borrower, amount);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>EtherBank</td>
<td>-</td>
<td>msg.sender.call.value(amount); User’s sum of balances is constant</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
<td>userBalances[msg.sender] = 0;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>msg.sender.call.value();</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Main contributions

- Automated property-based repair for smart contracts
- Combination of semantic analysis and search-based repair
- Public repository: https://github.com/polinatolmach/DeFinery

Thanks!

yi_li@ntu.edu.sg
@liyistc