Agency for
Science, Technology
and Research
SINGAPORE

NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

# 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" finance – banking, exchanges, *etc.*

# Solidity Smart Contracts & DeFi

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

deposit()

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

https://www.defipulse.com/

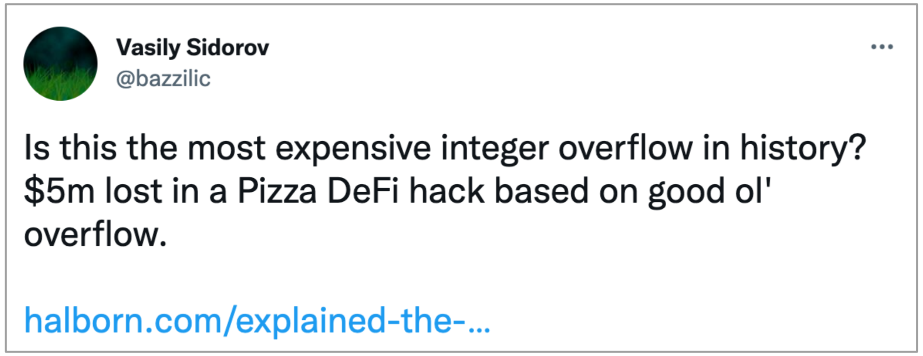
# DeFi Attacks

$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



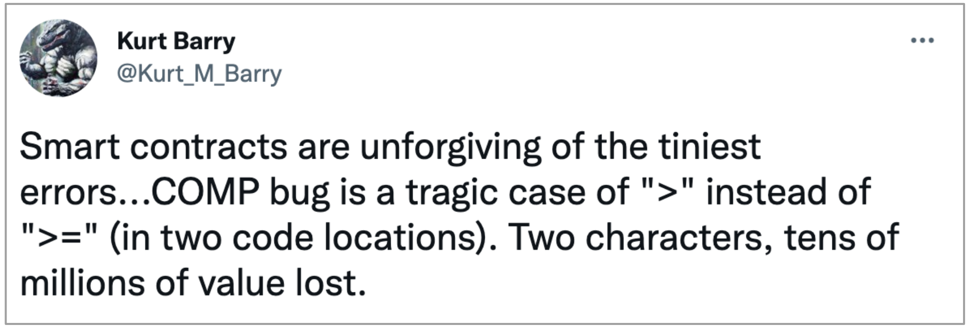
Vasily Sidorov
@bazzilic

Is this the most expensive integer overflow in history? $5m lost in a Pizza DeFi hack based on good ol' overflow.

halborn.com/explained-the-...

## Fidelity Issues

- "**Logical**" bugs
- Especially problematic in DeFi
- **Impossible** to find using patterns

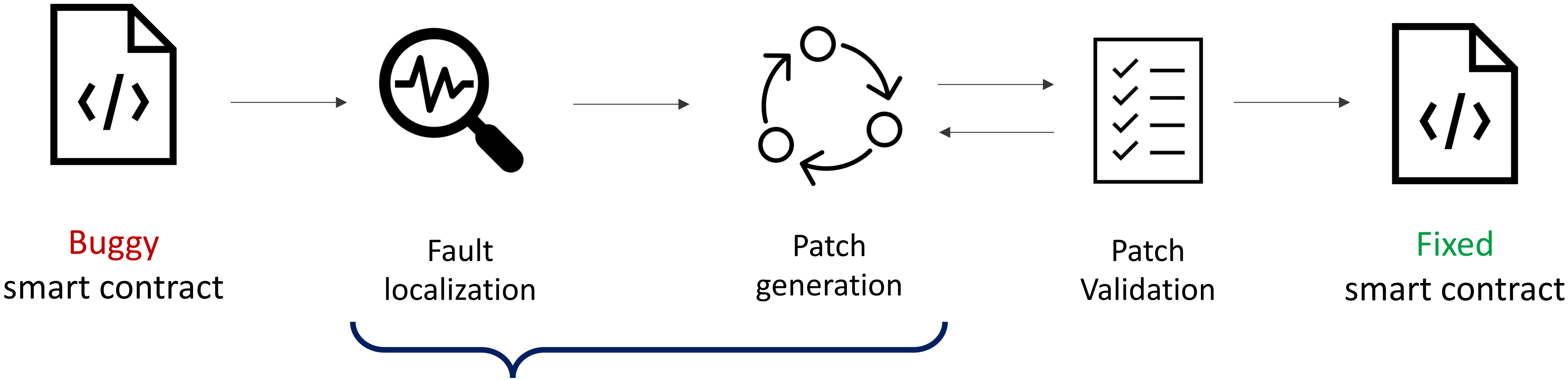


Kurt Barry
@Kurt_M_Barry

Smart contracts are unforgiving of the tiniest errors...COMP bug is a tragic case of ">" instead of ">=" (in two code locations). Two characters, tens of millions of value lost.

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*

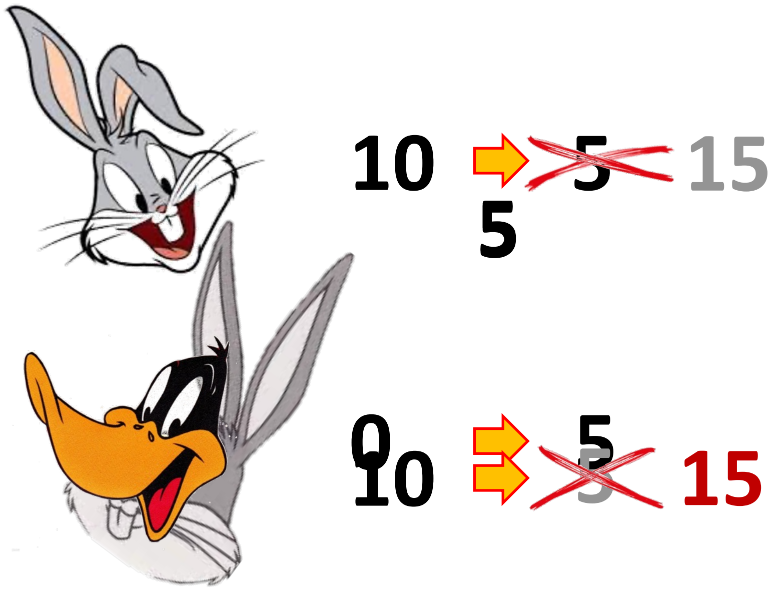
# iToken Duplication Issue ($8M loss)

```
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»*

Correct Behavior

Incorrect Behavior



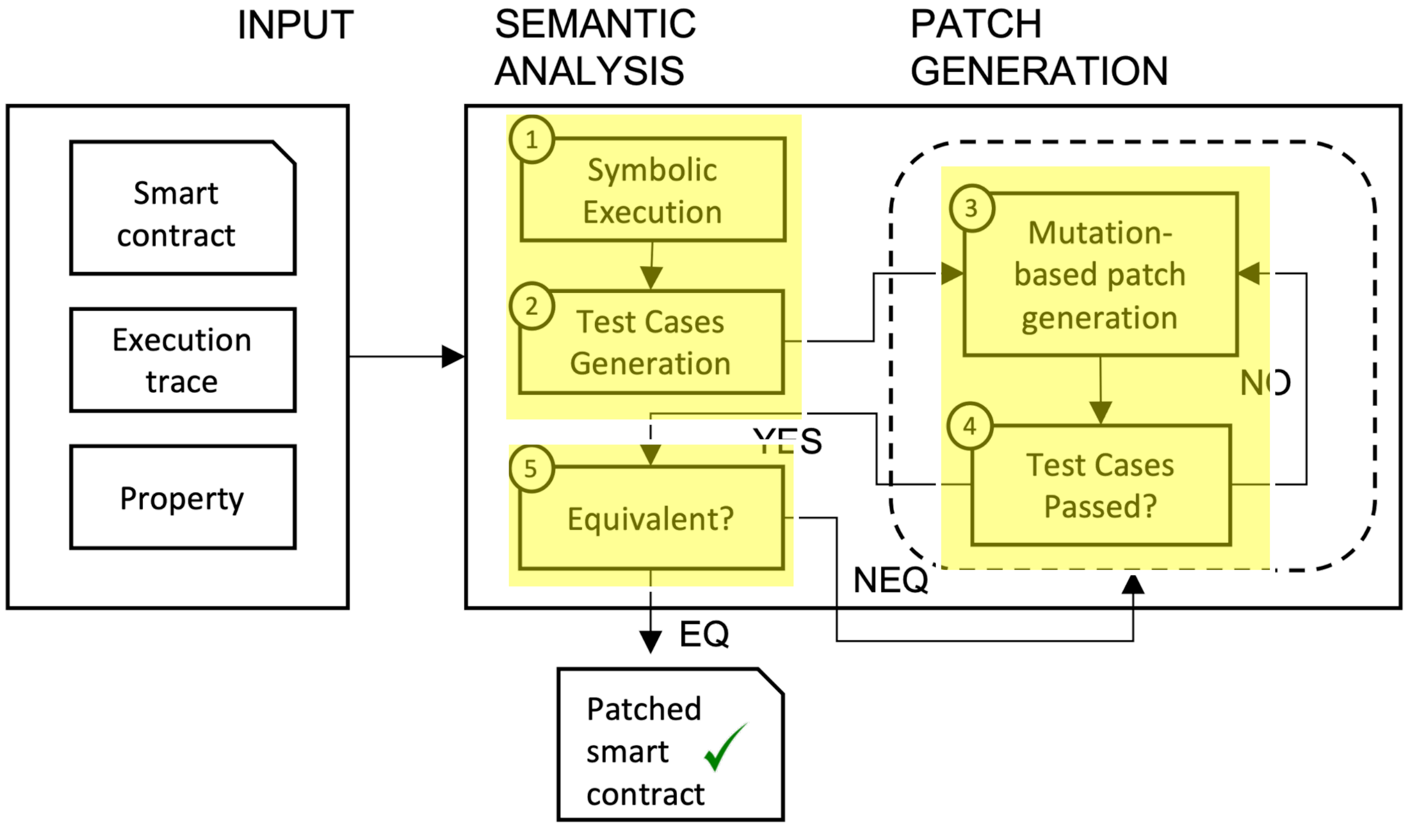
https://fullycrypto.com/bzx-suffers-token-duplication-incident

# 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

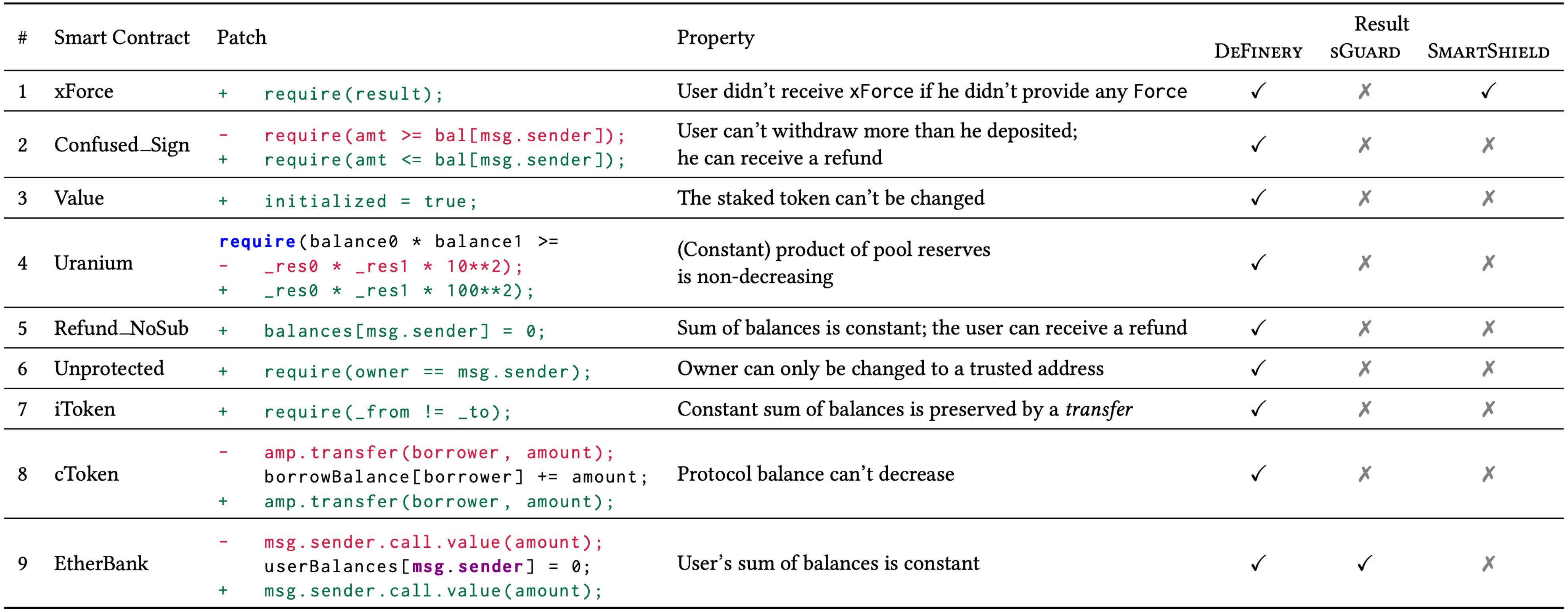
Generates patches using AST-based mutations, evaluates the patches using test cases



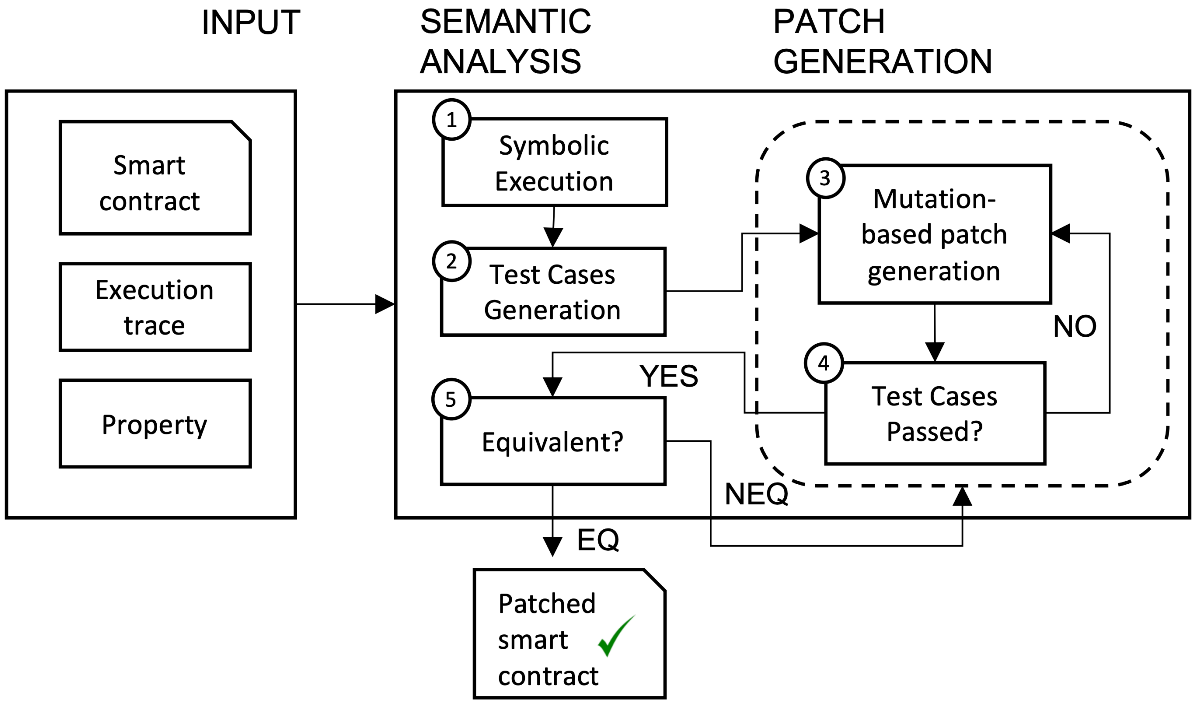
Checks conditional equivalence between original and patched smart contracts based on symbolic summaries

# 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



| # | Smart Contract | Patch | Property | Result<br>DEFINERY | <br>SGUARD | <br>SMARTSHIELD |
|---|---|---|---|---|---|---|
| 1 | xForce | `+ require(result);` | User didn't receive `xForce` if he didn't provide any `Force` | ✓ | ✗ | ✓ |
| 2 | Confused_Sign | `- require(amt >= bal[msg.sender]);`<br>`+ require(amt <= bal[msg.sender]);` | User can't withdraw more than he deposited; he can receive a refund | ✓ | ✗ | ✗ |
| 3 | Value | `+ initialized = true;` | The staked token can't be changed | ✓ | ✗ | ✗ |
| 4 | Uranium | `require(balance0 * balance1 >=`<br>`- _res0 * _res1 * 10**2);`<br>`+ _res0 * _res1 * 100**2);` | (Constant) product of pool reserves is non-decreasing | ✓ | ✗ | ✗ |
| 5 | Refund_NoSub | `+ balances[msg.sender] = 0;` | Sum of balances is constant; the user can receive a refund | ✓ | ✗ | ✗ |
| 6 | Unprotected | `+ require(owner == msg.sender);` | Owner can only be changed to a trusted address | ✓ | ✗ | ✗ |
| 7 | iToken | `+ require(_from != _to);` | Constant sum of balances is preserved by a *transfer* | ✓ | ✗ | ✗ |
| 8 | cToken | `- amp.transfer(borrower, amount);`<br>`borrowBalance[borrower] += amount;`<br>`+ amp.transfer(borrower, amount);` | Protocol balance can't decrease | ✓ | ✗ | ✗ |
| 9 | EtherBank | `- msg.sender.call.value(amount);`<br>`userBalances[msg.sender] = 0;`<br>`+ msg.sender.call.value(amount);` | User's sum of balances is constant | ✓ | ✓ | ✗ |



# Thanks!

✉ yi_li@ntu.edu.sg

🐦 @liyistc

# Main contributions

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