Securing Smart Contracts with Behavioral Contracts

Guannan Wei[1] Danning Xie[1]◆ Wuqi Zhang[2] Yongwei Yuan[1] Zhuo Zhang[1]◆

[1] Purdue University [2] HKUST ◆ Poster presenter ◆ Corresponding author

Motivation: Contracts for Contracts

- Ensuring the security and reliability of Decentralized Finance (DeFi) programs is essential.
- Current DeFi languages (e.g., Solidity) do not support effectively specifying and enforcing rich behavioral properties in a modular way.
- Our idea: extend Solidity with Eiffel-style *behavioral contracts*.

Behavioral contracts should compliment smart contracts

ConSol

A behavioral contract system for Solidity that provides practical specification and runtime monitoring system.

Usage:
1. Attach specifications for function definitions
2. Attach specifications for addresses when used as arguments or returned from functions.

- ***Non-intrusive*** : preserves static/dynamic program behaviors.
- ***Expressive*** : programmers can liberally write and enforce any computable properties.
- ***Gradual*** : programmers only write necessary specifications.
- ***Efficient*** : incurring minimal monitoring overhead (gas consumption) at runtime.

Monitoring Address Calls

Addresses
- Unsigned integers (160-bits)
- First-class citizens
- Unique identifiers for contracts or accounts
- May induce latent computational behaviors

Challenge: tracking and enforcing address behaviors

```
deposit(token, amount) where {
  IERC20(token).transferTo(addr, amt) returns (success)
  requires amt > 10
  ensures success
}
function deposit(address token, uint amount) {
  address token2 = id(token);
  actualDeposit(token2, amount - 10);
}
function actualDeposit(address token2, uint amount) {
  // call happens here
  IERC20(token2).transferTo(..., amount);
}
```

Motivating Example

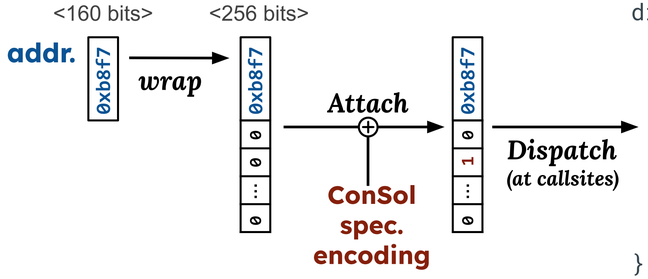
```
function getPrice(address chainklink) returns
(uint256) {
  (_, uint256 ethPrice, _, uint256 updatedAt, _) =
    IChainlinkAggregator(chainlink).latestRoundData();
  require(updatedAt > block.timestamp - 1 days);
  require(ethPrice > 0);

  uint256 price = ORACLE.getRate() * ethPrice;
  require(price * 95 / 100 < ORACLE.getLatestPrice()
    && price * 105 / 100 > ORACLE.getLatestPrice());

  return price;
}
```

Source code of readonly reentrancy vulnerability, and its fix with `require`, ensuring the price fluctuation within 5%.

```
getPrice(chainlink) returns (price)
  ensures price * 95 / 100 < ORACLE.getLatestPrice() &&
          price * 105 / 100 > ORACLE.getLatestPrice()
  where {
    IChainlinkAggregator(chainlink).latestRoundData()
    returns (_, ethPrice, _, updatedAt, _)
    ensures updatedAt > block.timestamp - 1 days && ethPrice > 0
  }
function getPrice(address chainlink) returns (uint256){
  (_, uint256 ethPrice, _, _, _)  =
    IChainlinkAggregator(chainlink).latestRoundData();
  return ORACLE.getRate() * ethPrice;
}
```

The fix with ConSol specifications, ***decoupling*** the specification and business logic, enhancing ***readability*** and ***maintainability.***

Translation




```
dispatch_IERC20_transferTo(uint256 addr, uint256 amount){
  uint96 specId = getSpecId(addr);
  if ((specId & 0000001) != 0) { ... }
  if ((specId & 0000010) != 0) {
    dispatch_transferTo_pre1(addr, amount);
  }
  ...
  bool success = IERC20(token2).transferTo(_unwrap(addr), amount);
  // similarly for dispatching post-condition check
}
```

Evaluation: 10 Real-World Attacks

- Effectiveness: ConSol can express defenses of these attacks meanwhile exhibiting better readability.
- Efficiency: ConSol only introduces on average 0.14% ($0.05 worth) more gas consumption.

Project	Attack Type	Date	Loss ($)	#Tx	Gas Inc. ($)		Gas Inc. (%)		LoC Reduced (%)
					Asset.	ConSol	Asset.	ConSol	
Umbrella	Integer Over/Underflow	03-20-22	700K	3	0.012	0.021	0.151	0.249	33.33
EFLeverVault	Business Logic Flaw	10-14-22	1M	13	0.043	0.05	0.094	0.109	25.00
N00d	Reentrancy	10-26-22	29K	46	0.022	0.027	1.656	2.099	11.11
Dexible	Arbitrary External Call	02-17-23	1.5M	54	0.048	0.126	0.088	0.23	11.76
SushiSwap	Unchecked User Input	04-09-23	3.3M	3	0.479	0.511	4.82	5.147	54.55
SwaposV2	Erroneous Accounting	04-16-23	468K	1	0.019	0.025	0.188	0.244	25.00
Unknown	Missing Slippage Check	05-31-23	111K	10	0.018	0.434	0.01	0.269	30.00
Sturdy	Readonly Reentrancy	06-12-23	800K	19	1.138	1.139	1.221	1.223	57.14
LEVUSDC	Access Control	06-15-23	105K	7	0.051	0.054	0.143	0.152	33.33
Bao	Inflation Manipulate	07-04-23	46K	15	0.004	0.007	0.036	0.069	83.33
Average			805K	17.1	0.183	0.239	0.841	0.979	36.46