Автоматическая сертификация смарт-контрактов на предмет надёжности их бизнес-логики

Евгений Шишкин
Научный отдел ИнфоТекс
20 марта 2019

Смарт-контракт

```
contract Send {
    uint amount = 0;
    uint deadline = 0;
    address payable recepient;
    function set(address payable _recp, uint _deadline) external payable {
        require (msg.value > 0 && _deadline > now && deadline == 0);
        amount = msg.value;
        recepient = _recp;
    }
    function withdraw() external {
        require (msg.sender == recepient && now >= deadline);
        recepient.transfer(amount);
        deadline = 0;
    }
}
```

TheDAO

- Децентрализованный инвестиционный фонд
- Размер смарт-контракта: 1238 строк (Solidity)
- Максимальный баланс смарт-контракта:

  ~ 150 000 000 $
- Количество пользователей: ~15 000
- TheDAO Whitepaper: 10 страниц описания

TheDAO Whitepaper

10. Acknowledgements

I want to thank Stephan Tual and Simon Jentzsch for fruitful discussions and corrections, as well as Gavin Wood and Christian Reitwiessner for a review of the contracts and the development of Solidity, the programing language used to write the contracts.

Special thanks goes to Yoichi Hirai and Lefteris Karapetsas for reviewing the smart contracts and making significant improvements.

I also want to thank Griff Green for reviewing and editing the paper.

Last but not least I want to thank our community which has given feedback, corrections and encouragement.

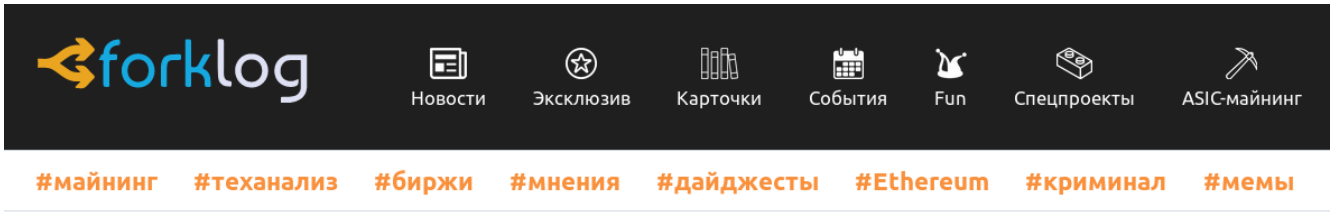
Специалист по EVM

Соавтор языка Solidity

Специалист формальным методам

Соавтор языка Solidity

Взлом TheDAO



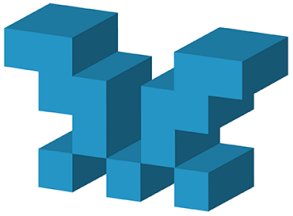
The DAO атакована: украдено $50 миллионов

НОВОСТИ 17.06.2016

Сегодня стало известно об атаке на первый децентрализованный венчурный фонд The DAO. Из фонда начали массово выводиться токены Ehereum, на текущий момент украдено порядка $50 миллионов.

Чему учит эта история?

Как сделать смарт-контракты надёжными?



SmartDec

Два вида ошибок

Операционные ошибки

- Переполнение арифметики
- Деление на 0
- Некорректная работа с динамическими структурами данных
- Reentrancy (TheDAO et al.)
- Недостижимость участка кода
- Оператор send с недостаточным количеством средств на балансе

Логические ошибки

- Алгоритм не во всех случаях достигает результата, ожидаемого пользователями
- Разработчик не предусмотрел все возможные цепочки исполнения, из-за этого часть непредвиденных цепочек приводит к нежелательным последствиям (например, deadlock)

Операционные ошибки

```
function deposit() public payable {
    require(msg.value >= DAO_token_price);
    address sender = msg.sender;
    uint tokens = msg.value / DAO_token_price;
    uint remainder = msg.value - tokens * DAO_token_price;
    balance[sender] += tokens;
    DAO_tokens_emitted += tokens;
    if (remainder > 0)
        sender.transfer(remainder);
    emit Deposited(sender, tokens);
}
```

Логические ошибки

```
function set(address payable _recp, uint _deadline) external payable {
    require (msg.value > 0 && _deadline > now && deadline == 0);
    amount = msg.value;
    recepient = _recp;
}
function withdraw() external {
    require (msg.sender == recepient && now >= deadline);
```

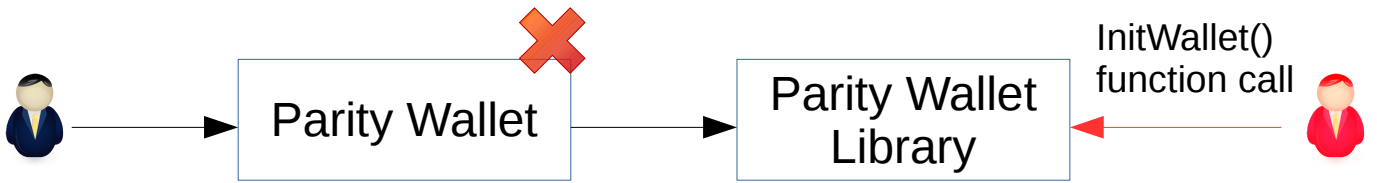
```
function getWinners() public view finishedGame
    returns(address[] memory players, uint[] memory prizes)
{
    uint8 totalNumWinners;
    for (uint8 i = 0; i < totalNumWinners; i++) {
        if ( i > winnerIndex ) {
            index = ( ( players.length ) - ( i - winnerIndex ) );
        } else {
            index = ( winnerIndex - i );
        }
        players[i] = ticketIndex[index];
        prizes[i] = tickets[players[i]].prize;
    }
    return (players, prizes);
}
```

Parity Wallet Library

ЛОГИЧЕСКАЯ ОШИБКА

Ethereum Parity Hack May Impact ETH 500,000 or $146 Million

November 7, 2017 @ 9:56 am By JD Alois

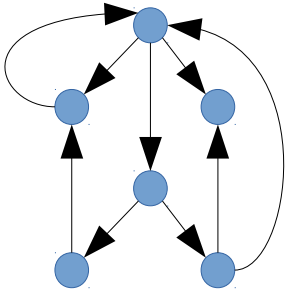


Промежуточные итоги

Формальная верификация

$$check(Code,\ Spec) = True \,|\, \{False, Trace\}$$

$$Impl \Rightarrow Spec$$



$$M \models \varphi$$

Шишкин Е.С. «*О построении среды для конструирования гарантированно надёжных смарт-контрактов*», РусКрипто 2018

Верификация модели программы

$$check(Code,\ Spec) = True \,|\, \{False, Trace\}$$

$$Code \approx Model$$

$$check(Model,\ Spec) = True \,|\, \{False, Trace'\}$$

Сертификация

Эффективность алгоритма верификации

Автоматическое достоверное построение модели из кода

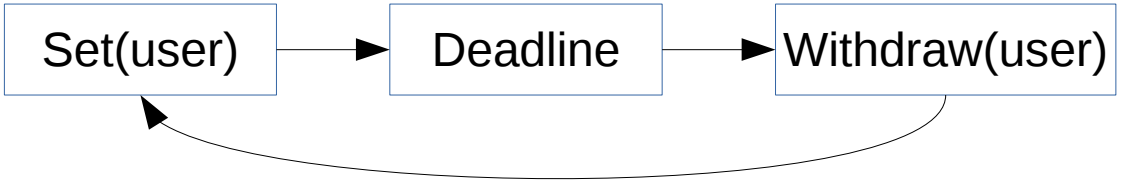
На каком языке задавать спецификацию?

$$\begin{cases} check(Model,\ Spec) = \{True, Cert\} \mid \{False, Trace\} \\ validate(Code,\ Spec,\ Cert) = True \mid False \end{cases}$$

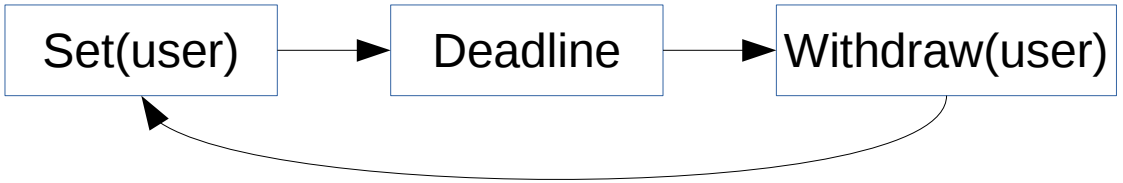
Как избавить конечного пользователя от необходимости проводить проверку корректности ***повторно***?

Спецификация

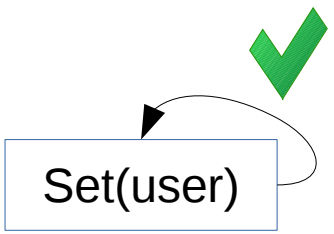
```
1  pragma solidity >=0.4.0 <0.6.0;
2  interface Send {
3      function set(address recepient, uint deadline) external;
4      function withdraw() external;
5      event Set(address recepient);
6      event Deadline();
7      event Withdraw(address recepient);
8  }
```

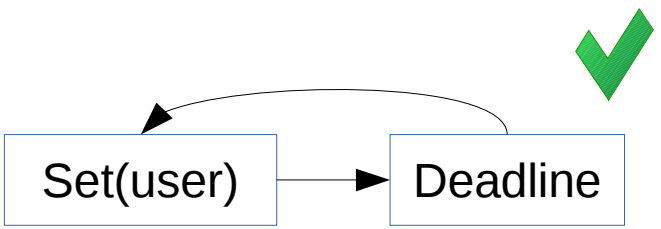


Спецификация на LTL



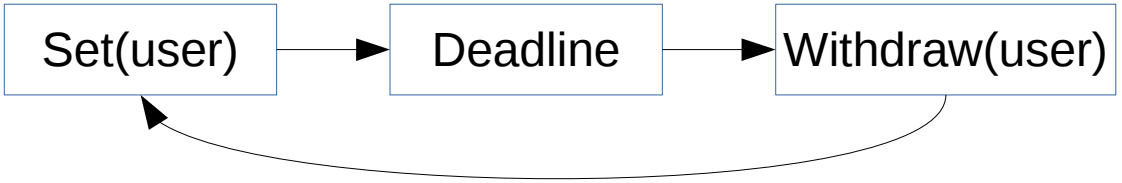
$$\Box(Set \Rightarrow \circ(Deadline \Rightarrow \circ(Withdraw \Rightarrow \circ Set))) \wedge \Box\Diamond Set$$





Поведения ошибочно удовлетворяют спецификации!

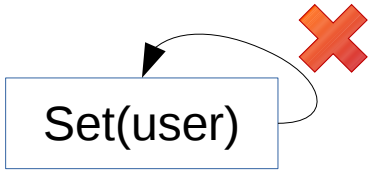
Спецификация через цепочки событий

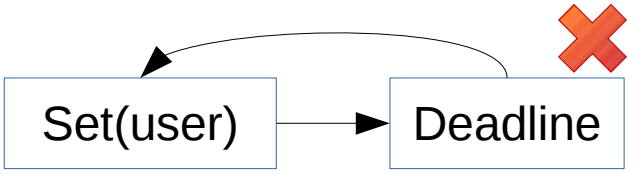


$$(Set,\ Deadline,\ Withdraw)^{\omega}$$

$$(Set\,(\neg Set|Withdraw|Deadline)^{*}\ Deadline$$

$$(\neg Set|Withdraw|Deadline)^{*}\ Withdraw)^{\omega}$$





Слабейшее предусловие Дейкстры

```
1 contract WP {
2  function send_if(uint amount, address payable to) external {
3      if (amount > 10**6 && amount < 10**7) {
4          to.transfer(amount);
5      }
6  }
7 }
```

- Функцию send_if можно вызвать не менее, чем $2^{256} * 2^{256} = 2^{512} \sim \mathbf{10^{154}}$ способами (не учитывая различные варианты остатков балансов вызывающей стороны и баланса адреса to)
- Но эффект из них имеют всего лишь $\mathbf{10^7} * n$, где n = потенциальное число пользователей смарт-контракта
- Техника вычисления предиката, позволяющего сузить пространство параметров одной функции, называется слабейшим предусловием Дейкстры

Сертификат надёжности

$$check(Code,\ Spec) = \{True, Cert\} \mid \{False, Trace\}$$
$$validate(Code,\ Spec,\ Cert) = True \mid False$$

- Автоматическая верификация смарт-контракта может занимать много времени и вычислительного ресурса
- Как избавить конечного пользователя от необходимости перепроверять СК на соответствие спецификации?
- Сертификат Cert является доказательством такой проверки; для заданных Code и Spec, его можно быстро проверить.
- «Подделать» Cert вычислительно сложно.

Эксперименты

- Упрощённая версия TheDAO была подвергнута проверке
- За несколько секунд прототип инструмента нашёл две ошибки в бизнес-логике
- Первая из них могла быть использована для вывода средств из смарт-контракта. Вторая могла привести к замораживанию средств в смарт-контракте.

Для цитирования: Шишкин Е.С. Проверка функциональных свойств смарт-контрактов методом символьной верификации модели. Труды ИСП РАН, том 30, вып. 5, 2018 г., стр. 265-288. DOI: 10.15514/ISPRAS-2018-30(5)-16

http://www.ispras.ru/proceedings/docs/2018/30/5/isp_30_2018_5_265.pdf

Итоги

Автоматическая сертификация смарт-контрактов на предмет надёжности их бизнес-логики

- **Автоматическая** : *с минимальным участием человека-эксперта*
- **Сертификация** : *не только механизм проверки, но и механизм быстрой перепроверки по выданному сертификату*
- **Надёжности** : *соответствие программы смарт-контракта своей спецификации*
- **Бизнес-логики** : *доказывается отсутствие (в том числе) логических ошибок*

Евгений Шишкин
Ведущий исследователь

ОАО «ИнфоТеКС»
127287, Москва, Старый Петровско-Разумовский проезд, 1/23, стр. 1

+7 (495) 737 61 92 (доб.4726)
evgeny.shishkin@infotecs.ru

https://unboxedtype.bitbucket.io