Анализ кода смарт-контракта

сети Ethereum

Исходные данные

Смарт-контракт развернут в сети Ethereum по адресу:

0x41c23Bf53117806cE77Ca65003D435C4D944f519

Код контракта открыт и верифицирован.

```
/**
 * @title Contractus contract
 * Funds distribution:
 * 90% - deposit funds
 * 3%  - support
 * 7% -  marketing
 * Allows you to receive income up to 2 deposit amounts and above if you continue to keep the deposit.
 * You can receive income 200% or more only once. In this case the deposit is closed.
 * Thus, the longer you keep the deposit open and do not withdraw your income, the more your potential gain becomes.
 *Payments are terminated after the completion of 200%. To re-enter the game, you must replenish your deposit.
 * You can receive your income at any time, based on a 2.5% per day calculation to 200%
 *
 * This contract is a game - a lottery, in which prizes - payments on the deposit.
 * The contract is not a pyramid, since all deposits have a finite period of validity of payments.
 * You should not use this contract for investment purposes. Only for the game - lottery.
 * By sending funds to this contract, you should understand that it is possible that the balance
 * of the contract will not be enough to pay all players.
 * Contract developers have not left themselves any functions for the withdrawal of players' funds,
 * but this is just a game - remember this.
 */



pragma solidity ^0.4.24;
contract Contractus {
    mapping (address => uint256) public balances;
    mapping (address => uint256) public timestamp;
    mapping (address => uint256) public receiveFunds;
    uint256 internal totalFunds;

    address support;
    address marketing;

    constructor() public {
        support = msg.sender;
        marketing = 0x53B83d7be0D19b9935363Af1911b7702Cc73805e;
    }

    function showTotal() public view returns (uint256) {
        return totalFunds;
    }

    function showProfit(address _investor) public view returns (uint256) {
        return receiveFunds[_investor];
    }

    function showBalance(address _investor) public view returns (uint256) {
        return balances[_investor];
    }

    /**
```

```
    * The function will show you whether your deposit will remain in the game after the withdrawal of
revenue or close after reaching 200%
    * A value of "true" means that your deposit will be closed after withdrawal of funds
    */
   function isLastWithdraw(address _investor) public view returns(bool) {
       address investor = _investor;
       uint256 profit = calcProfit(investor);
       bool result = !((balances[investor] == 0) || (balances[investor] * 2 > receiveFunds[investor] + profit));
       return result;
   }

   function calcProfit(address _investor) internal view returns (uint256) {
       uint256 profit = balances[_investor]*25/1000*(now-timestamp[_investor])/86400; // a seconds in one
day
       return profit;
   }


   function () external payable {
       require(msg.value > 0,"Zero. Access denied.");
       totalFunds +=msg.value;
       address investor = msg.sender;
       support.transfer(msg.value * 3 / 100);
       marketing.transfer(msg.value * 7 / 100);

       uint256 profit = calcProfit(investor);
       investor.transfer(profit);

       if (isLastWithdraw(investor)){
           /**
            * @title Closing of the deposit
            *
            *  You have received 200% (or more) of your contribution.
            *  Under the terms of the game, your contribution is closed, the statistics are reset.
            *  You can start playing again. We wish you good luck!
            */
           balances[investor] = 0;
           receiveFunds[investor] = 0;

       }
       else {
       receiveFunds[investor] += profit;
       balances[investor] += msg.value;

       }
       timestamp[investor] = now;
   }

}
```

Анализ кода

Дальнейший анализ относится только к данному коду, применительно к адресу сети Ethereum 0x41c23Bf53117806cE77Ca65003D435C4D944f519.

1. Введение

В анализе рассматривается функциональность кода с точки зрения логики и безопасности выполнения функций, безотносительно к финансовой, юридической составляющим.

Контракт содержит 1 внешнюю функцию, 4 публичных функций, доступных для выполнения любым пользователем и одну внутреннюю функцию, служащую для расчета данных. Среди них 4 функции представляют собой «геттеры», не вызывающие изменение состояния данных контракта. Данные функции представлены ниже.

- showTotal();
- showProfit(address _investor);
- showBalance(address _investor);
- isLastWithdraw(address _investor);

Функции служат для получения пользователями текущей информации о состоянии данных контракта и не могут тем или иным способом влиять на состояние контракта.

Внутренняя функция предназначена для расчета процентов:

- calcProfit(address _investor).

Данная функция выполняет расчет процентов, исходя из логики — баланс пользователя умножается на 25/1000 (или 2,5%) и на разницу между текущим временем и временем последнего движения по балансу пользователя, выраженному в секундах и равному дням, прошедшим с момента последнего движения по балансу.

Внешняя функция контракта является «fallback payable» функцией, Таким образом, данная функция будет выполнена всякий раз, когда контракт получает транзакцию перевода средств.

- function () external payable.

Логика работы функции.

Стр 69	require(msg.value > 0,"Zero. Access denied.");	Проверка на сумму транзакции. Транзакции с нулевой суммой не допускаются.
Стр 70	totalFunds +=msg.value;	Внутренняя переменная totalFunds отслеживает общую сумму собранных средств.
Стр 71	address investor = msg.sender;	В переменную investor записывается адрес отправителя

		транзакции
Стр 72-73	support.transfer(msg.value * 3 / 100); marketing.transfer(msg.value * 7 / 100);	На административные адреса переводятся 3 и 7% суммы транзакции, соответственно
Стр 75	uint256 profit = calcProfit(investor);	Вычисляется начисленный процент по балансу отправителя
Стр 76	investor.transfer(profit);	Отправителю передается начисленный процент по его балансу на контракте.
Стр 78-95	(isLastWithdraw(investor)){ ...	Проверка состояния баланса отправителя. При условии, что баланс отправителя равен или менее суммы начисленных и выплаченных процентов, баланс отправителя и учет полученных процентов устанавливаются в ноль. Иначе, к балансу и выплаченным процентам добавляются текущие значения. Затем обновляется время последнего движения по балансу.

2. Использование безопасной логики

Контракт не использует библиотек безопасных математических операций. В контракте имеются 7 случаев использования небезопасной математики:

- строка 58;
- строка 63;
- строка 70;
- строка 72;
- строка 73;
- строка 91;
- строка 92.

Все указанные случаи **теоретически** могут приводить к переполнению переменных. Однако, с учетом логики работы контракта, на практике невозможно представить ситуации, при которых переполнение переменных станет возможным. В контексте данного смарт-контракта указанные случаи можно считать безопасными.

3. Использование шаблона Checks-Effects-Interaction

В контракте используется безопасная функция transfer, которая предотвращает атаки Re-Entrancy.

4. Использование оператора «now»

В контракте используется оператор «now». Однако манипулирование временной меткой в пределах нескольких секунд не может оказать существенного влияния на состояние данных контракта. Таким образом, можно сделать вывод, что в контексте данного контракта, использование оператора «now» можно считать безопасным.

5. Функции контракта с привилегированными правами

Контракт не имеет в своем коде функций с привилегированными правами.

Рекомендации

Для исключения даже теоретической возможности переполнения в будущих реализациях предусмотреть использование библиотеки безопасных математических операций.

Выводы

Контракт скомпилирован с использованием современной (на сентябрь 2018 года) версии компилятора: v0.4.24+commit.e67f0147.

Все функции контракта соответствуют заявленному функционалу.

Функций, выполняющих злонамеренные или деструктивные воздействия на данные контракта не обнаружено.

Собственник контракта не имеет привилегированных прав или функций.

Административные адреса, на которые выполняются комиссионные переводы, являются жестко прописанными неизменяемыми адресами. Анализ данных адресов говорит о том, что это адреса кошельков, которые не являются контрактами и, таким образом, не могут блокировать операции по выводу средств с основного контракта.

Аудит контракта провел:

Челбухов А.А.

телеграм: https://t.me/alex_sysadm

Дата документа:

06.10.2018