Анализ кода смарт-контракта

сети Ethereum

06.10.2018

Исходные данные

Смарт-контракт развернут в сети Ethereum по адресу:

0xf2d32cFA422A4A5B7074050651ca380EB0Cf0a8C

Код контракта открыт и верифицирован.

```
/**
 *
 * Easy Investment Contract version 2.0
 * It is a copy of original Easy Investment Contract
 * But here a unique functions is added
 *
 * For the first time you can sell your deposit to another user!!!
 *
 */
pragma solidity ^0.4.24;

contract EasyStockExchange {
    mapping (address => uint256) invested;
    mapping (address => uint256) atBlock;
    mapping (address => uint256) forSale;
    mapping (address => bool) isSale;

    address creator;
    bool paidBonus;
    uint256 success = 1000 ether;

    event Deals(address indexed _seller, address indexed _buyer, uint256 _amount);
    event Profit(address indexed _to, uint256 _amount);

    constructor () public {
        creator = msg.sender;
        paidBonus = false;
    }

    modifier onlyOnce () {
        require (msg.sender == creator,"Access denied.");
        require(paidBonus == false,"onlyOnce.");
        require(address(this).balance > success,"It is too early.");
        _;
        paidBonus = true;
    }

    // this function called every time anyone sends a transaction to this contract
    function () external payable {
        // if sender (aka YOU) is invested more than 0 ether
        if (invested[msg.sender] != 0) {
            // calculate profit amount as such:
            // amount = (amount invested) * 4% * (blocks since last transaction) / 5900
            // 5900 is an average block count per day produced by Ethereum blockchain
            uint256 amount = invested[msg.sender] * 4 / 100 * (block.number - atBlock[msg.sender]) / 5900;

            // send calculated amount of ether directly to sender (aka YOU)
```

```
        address sender = msg.sender;
        sender.transfer(amount);
        emit Profit(sender, amount);
    }

    // record block number and invested amount (msg.value) of this transaction
    atBlock[msg.sender] = block.number;
    invested[msg.sender] += msg.value;
}


/**
 * function add your deposit to the exchange
 * fee from a deals is 10% only if success
 * fee funds is adding to main contract balance
 */
function startSaleDepo (uint256 _salePrice) public {
    require (invested[msg.sender] > 0,"You have not deposit for sale.");
    forSale[msg.sender] = _salePrice;
    isSale[msg.sender] = true;
}

/**
 * function remove your deposit from the exchange
 */
function stopSaleDepo () public {
    require (isSale[msg.sender] == true,"You have not deposit for sale.");
    isSale[msg.sender] = false;
}

/**
 * function buying deposit
 */
function buyDepo (address _depo) public payable {
    require (isSale[_depo] == true,"So sorry, but this deposit is not for sale.");
    isSale[_depo] = false; // lock reentrance

    require (forSale[_depo] == msg.value,"Summ for buying deposit is incorrect.");
    address seller = _depo;


    //keep the accrued interest of sold deposit
    uint256 amount = invested[_depo] * 4 / 100 * (block.number - atBlock[_depo]) / 5900;
    invested[_depo] += amount;


    //keep the accrued interest of buyer deposit
    if (invested[msg.sender] > 0) {
        amount = invested[msg.sender] * 4 / 100 * (block.number - atBlock[msg.sender]) / 5900;
        invested[msg.sender] += amount;
    }

    // change owner deposit
    invested[msg.sender] += invested[_depo];
    atBlock[msg.sender] = block.number;
```

```
        invested[_depo] = 0;
        atBlock[_depo] = block.number;


        isSale[_depo] = false;
        seller.transfer(msg.value * 9 / 10); //10% is fee for deal. This funds is stay at main contract
        emit Deals(_depo, msg.sender, msg.value);
    }

    function showDeposit(address _depo) public view returns(uint256) {
        return invested[_depo];
    }

    function showUnpaidDepositPercent(address _depo) public view returns(uint256) {
        return invested[_depo] * 4 / 100 * (block.number - atBlock[_depo]) / 5900;
    }

    function Success () public onlyOnce {
        // bonus 5% to creator for successful project
        creator.transfer(address(this).balance / 20);

    }
}
```

Анализ кода

Дальнейший анализ относится только к данному коду, применительно к адресу сети Ethereum 0xf2d32cFA422A4A5B7074050651ca380EB0Cf0a8C.

1. Анализ функционала смарт-контракта

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

Контракт содержит одну внешнюю “payable” функцию, 2 публичных функций-геттера, 3 публичных функции-сеттера, одну публичную функцию-сеттер с привилегированными правами и один модификатор.

Публичные функции-геттеры предназначены для получения информации о состоянии данных смарт-контракта и не имеют возможности влиять на состояние данных:

- showDeposit;
- showUnpaidDepositPercent;

В качестве аргумента обе функции принимают адрес.

Публичные функции-сеттеры предназначены для изменения состояния данных смарт-контракта:

- startSaleDepo;
- stopSaleDepo;
- buyDepo.

Функция **startSaleDepo** служит для установки логического маркера в «true» на маппинг isSale, а также для установки стоимости сделки на маппинг forSale. Функция принимает явный параметр — сумма сделки и неявный параметр — адрес отправителя.

Функция **stopSaleDepo** служит для установки логического маркера в «false» на маппинг isSale.

Таким образом, с помощью данных функций можно пометить баланс отправителя функций меткой «for Sale» либо снять эту метку.

Функция **buyDepo** служит для перевода средств отправителя функции получателю — собственнику баланса, указанного в качестве аргумента функции. В качестве параметра функция принимает адрес. Данная функция является “**payable**”, т. е. предусматривает **передачу эфиров** при выполнении. Сумма эфира должна точно соответствовать сумме продаваемого баланса в маппинге forSale. Проверка на соответствие выполняется в строке 85:

- require (forSale[_depo] == msg.value,"Summ for buying deposit is incorrect.");

Также здесь идет проверка на наличие булевского маркера в маппинге isSale. Иными словами, функция проверяет, действительно ли был выставлен указанный баланс на продажу и соответствует ли сумма транзакции сумме, указанной продавцом. Если данные условия выполняются, происходит расчет и причисление % на баланс продавца (строки 90-91), затем идет расчет и причисление % на баланс покупателя, если он отличен от нуля (строки 95-98).

После этого к балансу отправителя добавляется баланс продавца.

Затем баланс продавца устанавливается в ноль.

Также обновляются временные метки, которые в данном смарт-контракте привязаны к номеру текущего блока.

С баланса продавца снимается метка «for Sale», затем продавцу отправляется транзакция с суммой 90% от суммы сделки (строка 110) и в лог событий записывается событие «Сделка» (строка 111).

!!! Внимание!!! При анализе работы данной функции выявлена недоработка, которая с учетом вероятной ошибки пользователя может привести к необратимой потере баланса!

В функции нет проверки, что адрес отправителя **не является** аргументом функции. Иными словами, пользователь, выполняющий данную функцию, который по ошибке установит в качестве аргумента свой собственный адрес, потеряет весь свой баланс на контракте, так как в этом случае в строке 105 произойдет сброс данных о балансе отправителя.

Внешняя «payable» функция предназначена для ввода и вывода средств. Функция делится на 2 блока, в зависимости от условия — существует ли отправитель в маппинге invested или нет.

Если запись не найдена, происходит добавление отправителя в маппинг с указанием суммы транзакции и номера блока.

Если запись найдена, функция определяет размер %, исходя из логики 4% в день.[1]

1 В смарт-контракте не используется оператор «now». Вместо него для определения времени выполняется подсчет количества блоков, прошедших с момента последнего изменения баланса. За усредненное количество блоков в сутки принято число 5900, что эквивалентно появлению

Далее отправителю переводятся средства, с использованием функции transfer (строка 49) и в лог событий записывается соответствующее событие.

2. Использование безопасной логики

В контракте не используется библиотека безопасных математических операций.

Выявлено 8 случаев использования небезопасной математики:

- строка 45;
- строка 55;
- строка 90;
- строка 91;
- строка 97;
- строка 101;
- строка 110;
- строка 119.

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. Функции контракта с привилегированными правами

Контракт имеет функцию с привилегированными правами.

- Success () public onlyOnce

Данная функция предполагает перевод средств создателю контракта в размере 5% баланса контракта. С учетом анализа логики модификатора onlyOnce нужно иметь ввиду, что данную функцию может выполнить только создатель контракта, только при условии, что баланс контракта превысил величину 1000 эфиров. Кроме того, в модификаторе используется логический триггер:

нового блока в сети Ethereum каждые 14,6 сек.

- bool paidBonus

который позволяет выполнить данную функцию только один раз и предотвращает повторное выполнение.

Выводы

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

Все функции контракта в целом соответствуют заявленному функционалу.

Функций, выполняющих злонамеренные воздействия на данные контракта не обнаружено.

Обнаружена функция, выполняющая деструктивное воздействие на данные контракта!!! - в функции **buyDepo** выявлена **опасная недоработка**, которая в случае ненадлежащего использования, ошибочно обнуляет данные о балансе пользователя — отправителя функции. Для предотвращения указанной ситуации необходимо убедиться, что отправитель случайно не поставил свой собственный адрес в качестве аргумента функции.

Собственник контракта имеет функцию с **привилегированными правами**, позволяющую один и только один раз получить 5% баланса контракта при условии, что баланс контракта превысит 1000 эфиров.

К **недостаткам разработки** контракта можно отнести отсутствие средств для выяснения состояния маркера «isSale» и суммы продажи «forSale» искомого баланса.

Аудит контракта провел:

Челбухов А.А.

телеграм: https://t.me/alex_sysadm

Дата документа:

06.10.2018