**Аудит Smart-контрактов DApp**

11.10.2018

**Исходные данные:**

Адрес в сети Интернет: <https://github.com/StayBitDev/RentalContracts>

По этому адресу расположено 8 Smart контрактов: [BaseEscrowLib.sol](https://github.com/StayBitDev/RentalContracts/blob/master/BaseEscrowLib.sol), [DateTime.sol](https://github.com/StayBitDev/RentalContracts/blob/master/DateTime.sol), [FlexibleEscrowLib.sol](https://github.com/StayBitDev/RentalContracts/blob/master/FlexibleEscrowLib.sol), [ModerateEscrowLib.sol](https://github.com/StayBitDev/RentalContracts/blob/master/ModerateEscrowLib.sol), [MyToken.sol](https://github.com/StayBitDev/RentalContracts/blob/master/MyToken.sol), [Ownable.sol](https://github.com/StayBitDev/RentalContracts/blob/master/Ownable.sol), [StayBitContractFactory.sol](https://github.com/StayBitDev/RentalContracts/blob/master/StayBitContractFactory.sol), [StrictEscrowLib.sol](https://github.com/StayBitDev/RentalContracts/blob/master/StrictEscrowLib.sol).

Описание работы: <https://staybit.io/demo/Help.aspx>

Есть работающий DApp в тестовой сети Ropsten и в Главной сети: <https://staybit.io/demo/> и <https://staybit.io/main/>. Разница между ними в том что тестовая версия позволяет симулировать текущую дату и работает с тестовым токеном. Версия в майннете работает с TrueUSD токеном и игнорирует симуляцию даты.

**Краткое описание проекта:**

В области кратковременной посуточной аренды происходит много обмана: фальшивые листинги на бесплатных сайтах предлагают перечислить всю сумму вперед за резервацию жилья, а потом арендаторы приезжают заселяться и оказываются ни с чем. Арендодатели также страдают от жуликов, которые резервируют за день до заселения и платят украденными кредитками или поддельными чеками. Данное DApp позволяет использую эскроу договора на блокчейне Ethereum финансово обезопасить проведения сделок по аренде жилья.

**Цель проведения аудита:**

Проверить утверждение, что арендаторы и арендодатели могут пользоваться этим DApp, не опасаясь за потерю своих финансовых средств.

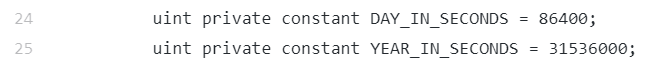
 Арендатор может отменить бесплатно за 30 дней до даты переезда. Если он отменяет менее чем за 30 дней, тогда штраф за отмену составит половину всей арендной платы. Во время операции «Переезд» арендатор может отменить и потребовать искажения владельца, в этом случае плата за кабзеллинг будет разделена между сторонами. Если арендатор аннулируется после даты переезда, он будет платить за свои фактические дни, потраченные плюс штраф за отмену. Оставшаяся рента будет возвращена арендатору. Ранний переезд должен быть подтвержден арендодателем. Если арендодатель подтвердил досрочный выезд на следующий день, то день, подтвержденный арендодателем, будет засчитан как фактический день выезда. Если арендодатель инициировал раннее выезд, то он не будет допущен для получения Гарантийного депозита.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID:** |  | 74 |  | PIN: |  | 6190 |

0x3f3C9E209Ed39CaB936726AbA6b385756092bE42  - [StayBit Token (STBT)](https://ropsten.etherscan.io/token/0x3f3c9e209ed39cab936726aba6b385756092be42)

***Анализ исходного кода***

1. **Контракт DateTime**



(Обратить внимание).В языке Solidity уже есть временные константы. Данный код можно было бы записать более понятно.

**uint private constant** DAY \_IN\_SECONDS = 1 days;

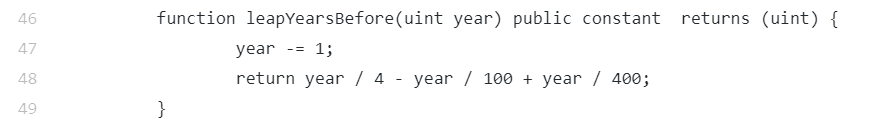
**uint private constant** YEAR\_IN\_SECONDS = 1 years;



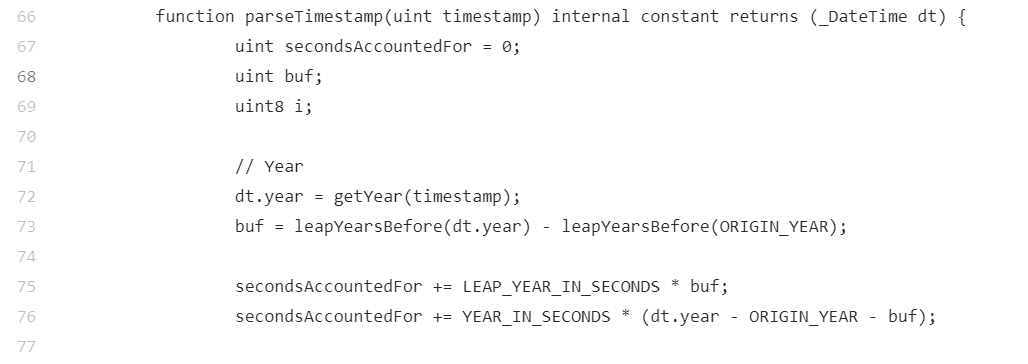
**uint endOfDay** = t2 + 1 days;

(Критично) В контракте не используется безопасная математика.

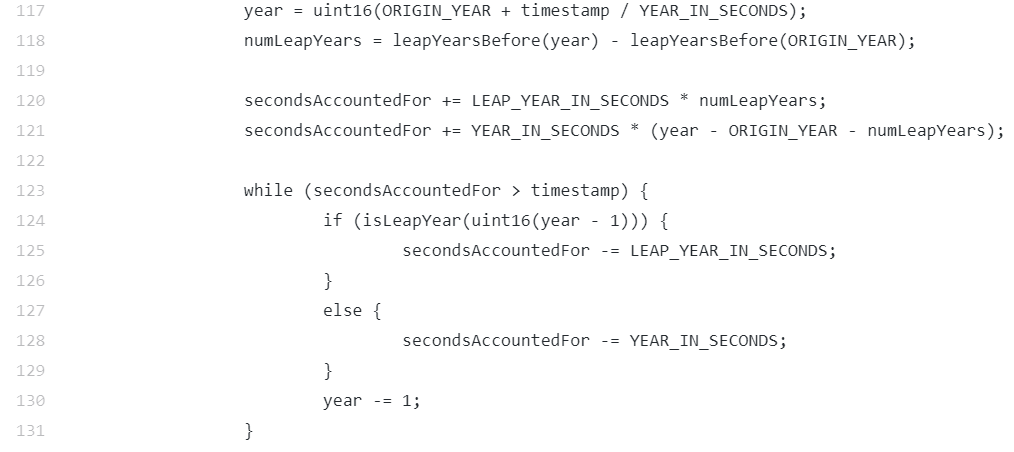
(Критично) Код в следующих строках является не безопасным и его исполнение может привести к непредсказуемым последствиям.



Строки 47 и 48



Строки 73, 76

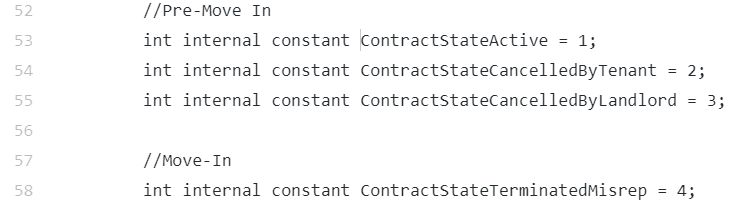


Строки 118, 121, 124, 125, 128, 130

Обезопасить использование данного кода позволяет проведение проверки на допустимость значений, используемых в этих выражениях переменных.

1. **Контракт (BaseEscrowLib)**

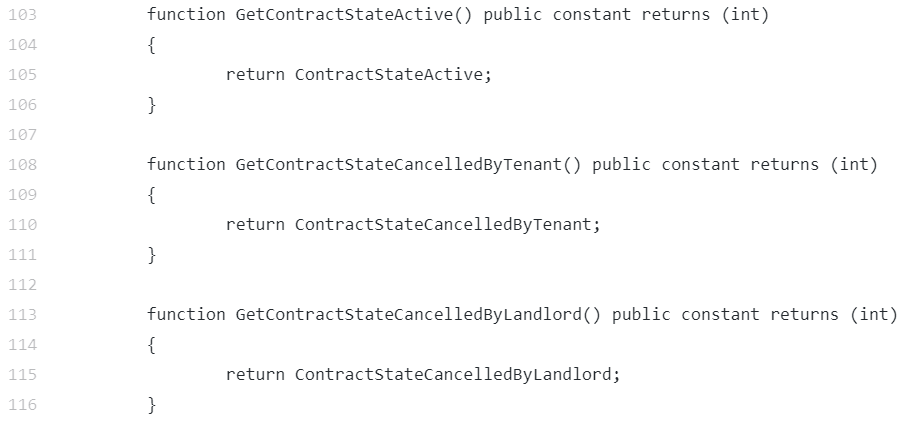
(Обратить внимание) Использование типов данных enum позволит сделать код более компактным.



enum ContractState {ACTIVE, CANCELLED\_BY\_TENANT, CANCELLED\_BY\_LANDLORD, TERMINATED\_MISREP }

ContractState stateData;

Обращение к функциям:



Таким образом вместо описания вызова функциий GetContractState..() в 9 местах достаточно описать геттер только один раз.

Можно будет заменить в библиотеке BaseEscrowLib этот код на следующий код:

function GetContractState(uint \_value) public constant returns (uint) {

return ContractState(\_value);

}

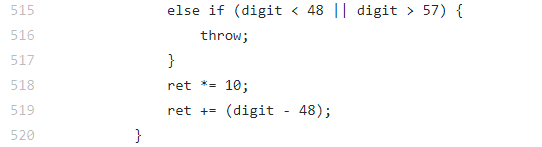
Тогда обращение из других библиотек будет осуществляться вместо такого вызова:

BaseEscrowLib.GetContractStateActive()

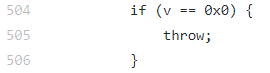
так:

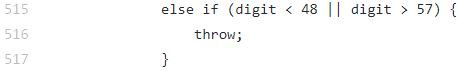
BaseEscrowLib.GetContractState(ContractState.ACTIVE)

Аналогично можно поступить и для 3 функций типа GetContractStage…() и для



Код в строке 519 является безопасным, т.к. в строке 515 проводится проверка на допустимость значения переменной.





(Обратить внимание) Использование throw считается устаревшим в строках 505 и 516

В версии Solidity 0.4.10 были введены функции assert(), require()и revert().

В настоящее время более предпочтительным считается использование assert() и require().

Причем так сложилось, что вызов revert происходит в более «легких» случаях (например при вызове if/else), а вызов assert() в более «тяжелых» случаях (например: превышение предельного значения для переменной, условие, которое не должно возникнуть и т.д).

В данном случае будет предпочтительнее использовать revert().

Также использование throw приведет к расходу всего отправленного газа, а использование revert() позволяет вернуть не использованный газ вызывающей стороне.

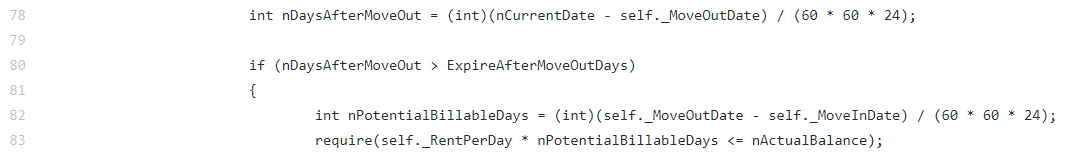
1. **Контракт (FlexibleEscrowLib)**



(Обратить внимание) Код в строке 40 можно заменить на

int nDaysBeforeMoveIn = (int)(self.\_MoveInDate - nCurrentDate) / (1 days);

(Критично) Выполнение кода в строке 40 без проверки значений переменных может привести к непредсказуемым последствиям



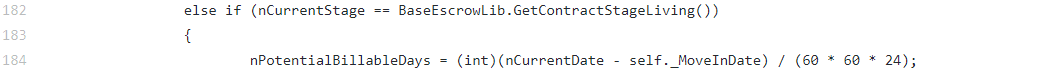
(Обратить внимание) Код в строке 78 можно заменить на

int nDaysAfterMoveOut = (int)(nCurrentDate - self.\_MoveOutDate) / (1 days);

(Обратить внимание) Код в строке 82 можно заменить на

int nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

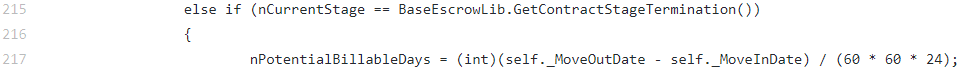
(Критично) Выполнение кода в строках 78 и 82 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 184 можно заменить на

nPotentialBillableDays = (int)(nCurrentDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 184 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 217 можно заменить на

nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 217 без проверки значений переменных может привести к непредсказуемым последствиям

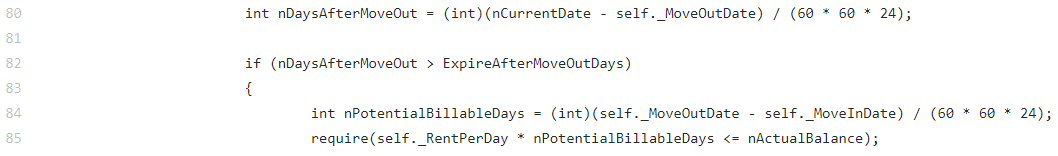
1. **Контракт (ModerateEscrowLib)**



(Обратить внимание) Код в строке 39 можно заменить на

int nDaysBeforeMoveIn = (int)(self.\_MoveInDate - nCurrentDate) / (1 days);

(Критично) Выполнение кода в строке 39 без проверки значений переменных может привести к непредсказуемым последствиям



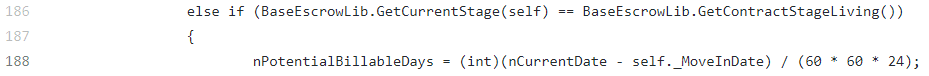
(Обратить внимание) Код в строке 80 можно заменить на

int nDaysAfterMoveOut = (int)(nCurrentDate - self.\_MoveOutDate) / (1 days);

(Обратить внимание) Код в строке 84 можно заменить на

int nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строках 80 и 84 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 188 можно заменить на

nPotentialBillableDays = (int)(nCurrentDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 188 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 224 можно заменить на

nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 224 без проверки значений переменных может привести к непредсказуемым последствиям

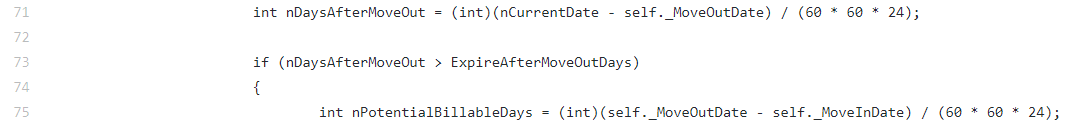
1. **Контракт (StrictEscrowLib)**



(Обратить внимание) Код строке 40 можно заменить на

int nDaysBeforeMoveIn = (int)(self.\_MoveInDate - nCurrentDate) / (1 days);

(Критично) Выполнение кода в строке 40 без проверки значений переменных может привести к непредсказуемым последствиям



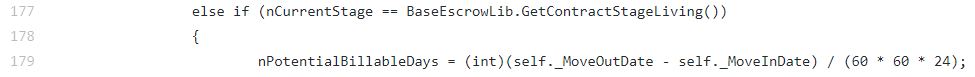
(Обратить внимание) Код в строке 71 можно заменить на

int nDaysAfterMoveOut = (int)(nCurrentDate - self.\_MoveOutDate) / (1 days);

(Обратить внимание) Код в строке 75 можно заменить на

int nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

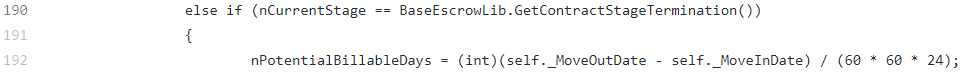
(Критично) Выполнение кода в строках 71 и 75 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 179 можно заменить на

nPotentialBillableDays = (int)(nCurrentDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 179 без проверки значений переменных может привести к непредсказуемым последствиям



(Обратить внимание) Код в строке 192 можно заменить на

nPotentialBillableDays = (int)(self.\_MoveOutDate - self.\_MoveInDate) / (1 days);

(Критично) Выполнение кода в строке 192 без проверки значений переменных может привести к непредсказуемым последствиям

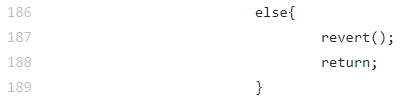
1. **Контракт (StayBitContractFactory)**



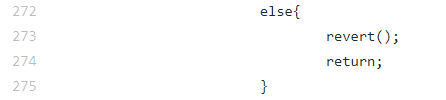
Код в строке 50 является безопасным, т.к. в строке 49 производится проверка на допустимость значений переменных.



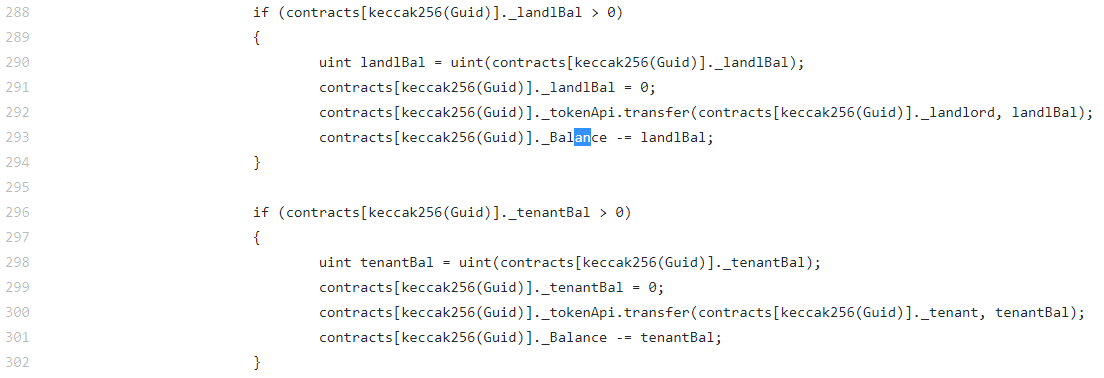
(Критично) Выполнение кода в строке 129 без проверки значений переменных может привести к непредсказуемым последствиям





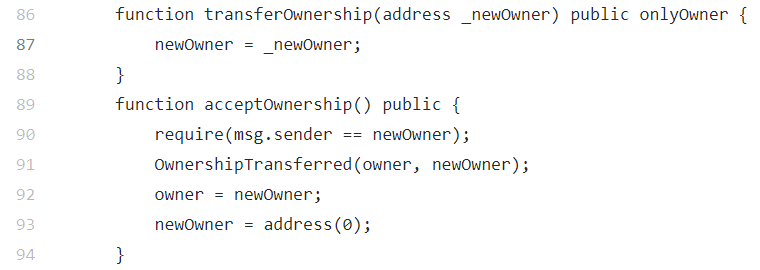


(Обратить внимание) Вызов оператора return в строках 188, 219, 274 никогда не выполнится, т.к. всегда выполнение кода в строках 187, 218, 273 приведет к выходу из функции и завершению транзакции.



(Критично) Выполнение кода в строках 293 и 301 без проверки значений переменных может привести к непредсказуемым последствиям

1. **Контракт (MyToken)**



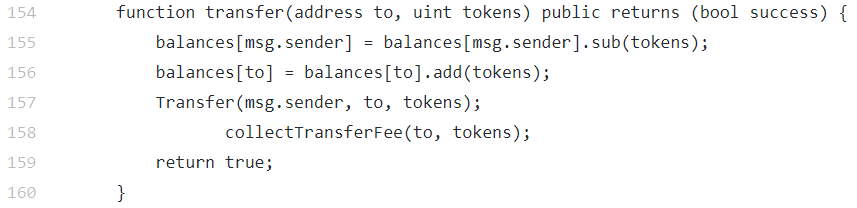
(Критично) Перед выполнением присвоения нового значения адреса, нет проверки значения этого адреса на 0. В случае введения неверного значения возможно возникновения ситуации потери контроля над управлением контрактом.

(Критично) Функцию в строке 89 может вызвать любой пользователь и произвести изменение владельца контракта. В случае некорректного значения переменной newOwner, например, сразу после установки контракта в сеть, вызов функции любым пользователем приведет к потере возможности управлением контрактом.



(Обратить внимание) В строке 108, либо надо назвать переменную totalSupply и тогда убрать вызов функции totalSupply() из строки 136, либо сделать переменную \_totalSupply внутренней:

uint private \_totalSupply;

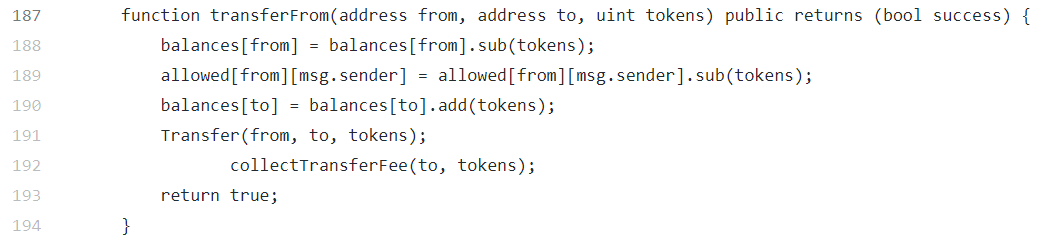


(Критично) В стоках 155 и 156 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

Необходимо осуществлять проверку на допустимость значений.

require(tokens <= \_balances[msg.sender]);

require(to != address(0));



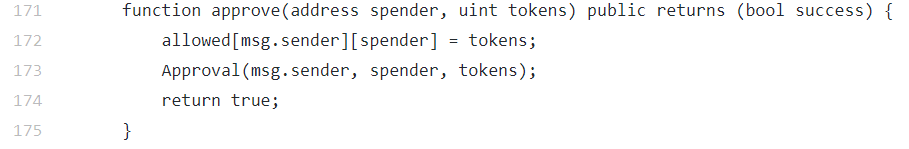
(Критично) В строках 188, 189 и 190 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

Необходимо осуществлять проверку на допустимость значений.

require(tokens <= \_balances[from]);

require(to != address(0));

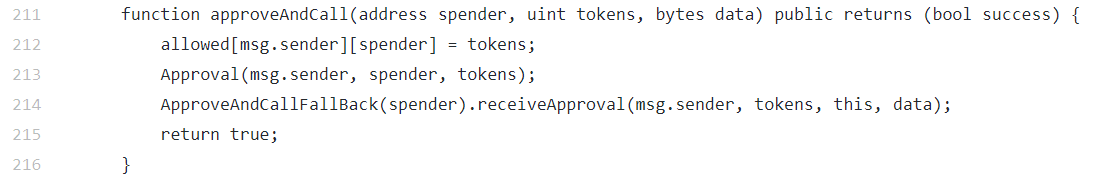
require(tokens <= \_allowed[from][msg.sender]);



(Критично) Перед выполнением кода в строке 172 не сделана проверка на корректность введенного значения адреса.

Необходимо осуществлять проверку на допустимость значения:

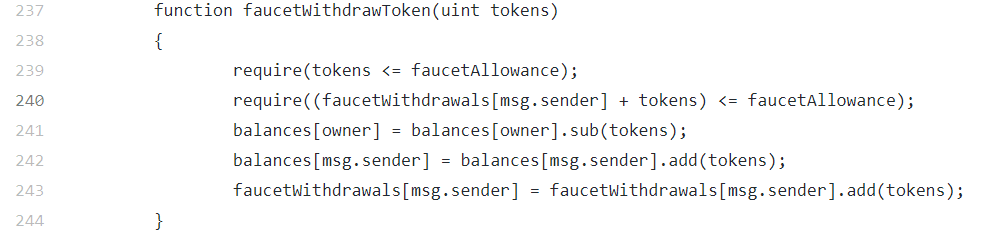
require(spender != address(0));



(Критично) Перед выполнением кода в строке 212 не сделана проверка на корректность введенного значения адреса.

Необходимо осуществлять проверку на допустимость значения:

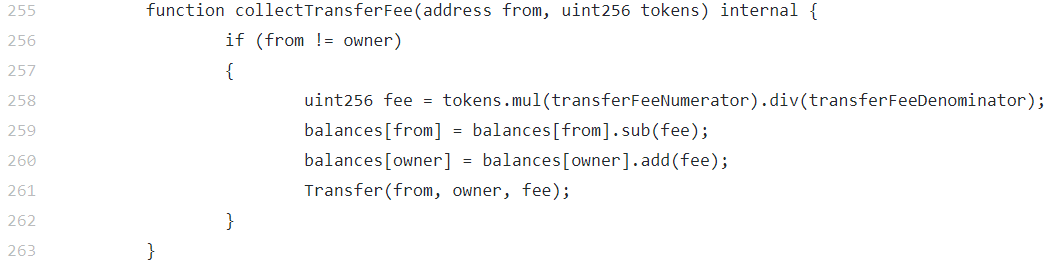
require(spender != address(0));



(Критично) В стоках 241, 242,243 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

Необходимо осуществлять проверку на допустимость значений.

require(tokens <= \_balances[owner]);

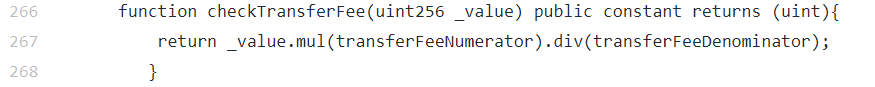


(Критично) В стоках 258, 259, 260 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

Необходимо осуществлять проверку на допустимость значений.

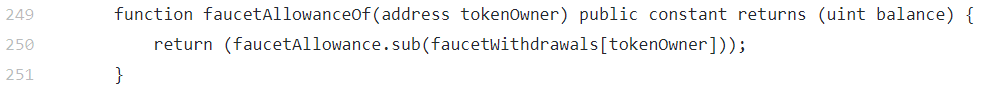
require(transferFeeDonominator > 0);

require(fee <= \_balances[from]);



(Критично) В строке 267 необходимо произвести проверку значения переменной на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

require(transferFeeDonominator > 0);



(Критично) Перед выполнением кода в строке 250 необходимо произвести проверку значения переменной на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям.

require(faucetAllowance >= faucetWithdrawals[tokenOwner]);

1. **Контракт (Ownable)**

К этому контракту замечаний нет. Контракт написан безопасно.

***Экспериментальная часть***

Для проведения тестирования Smart контрактов с помощью фреймворка Truffle был создан файл ..\migrations\2\_deploy\_contracts.js

с таким содержимым:

|  |
| --- |
| **const** BaseEscrowLib = artifacts.require(**'./BaseEscrowLib.sol'**) **const** DateTime = artifacts.require(**'./DateTime.sol'**) **const** ModerateEscrowLib = artifacts.require(**'./ModerateEscrowLib.sol'**) **const** Ownable = artifacts.require(**'./Ownable.sol'**) **const** FlexibleEscrowLib = artifacts.require(**'./FlexibleEscrowLib.sol'**) **const** StrictEscrowLib = artifacts.require(**'./StrictEscrowLib.sol'**) **const** StayBitContractFactory = artifacts.require(**'./StayBitContractFactory.sol'**)  module.**exports** = (deployer) => {  deployer.deploy(DateTime);  deployer.link(DateTime, BaseEscrowLib);  deployer.deploy(BaseEscrowLib);   deployer.link(DateTime, FlexibleEscrowLib);  deployer.link(BaseEscrowLib, FlexibleEscrowLib);  deployer.deploy(FlexibleEscrowLib);   deployer.link(DateTime, ModerateEscrowLib);  deployer.link(BaseEscrowLib, ModerateEscrowLib);  deployer.deploy(ModerateEscrowLib);   deployer.deploy(Ownable);   deployer.link(DateTime, StrictEscrowLib);  deployer.link(BaseEscrowLib, StrictEscrowLib);  deployer.deploy(StrictEscrowLib);   deployer.link(BaseEscrowLib, StayBitContractFactory);  deployer.link(FlexibleEscrowLib, StayBitContractFactory);  deployer.link(ModerateEscrowLib, StayBitContractFactory);  deployer.link(StrictEscrowLib, StayBitContractFactory);  deployer.link(Ownable, StayBitContractFactory);   deployer.deploy(StayBitContractFactory); }; |

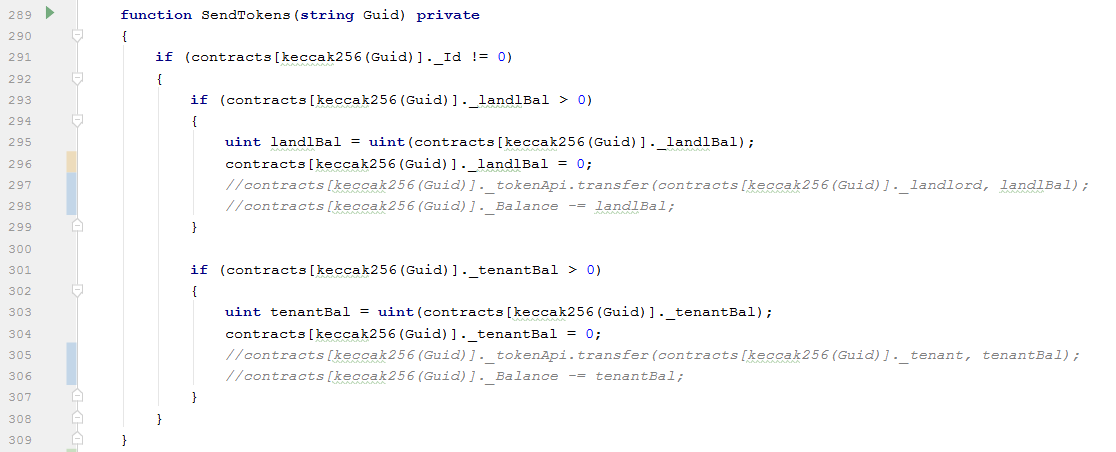
Но осуществить деплой контракта StayBitContractFactory включая все используемые в нем библиотеки не удалось, т.к. на экран была выведена ошибка  


В качестве одной из причин появления этой ошибки является попытка записать в Блокчейн Ethereum контракты, включающие в себя объем логики больший, чем 4 600 000 unit газа.

Была сделана проверка этого предположения. Для этого сначала в контракте были закомментированы вызовы всех функций, кроме конструктора. В результате контракт задеплоился в основную сеть.

Затем был закомментирован код следующий функций:  
TenantTerminate(string Guid), TenantTerminateMisrep(string Guid).  
И частично код функции SendTokens(string Guid)

*На рисунке ниже показано, как это было сделано.*



При таком подходе для установки контракта по прежнему не хватало газа.

Однако, если закомментировать какую-либо одну из строк: 296 или 304, то этого уже было достаточно для того, чтобы хватило газа на установку контракта. Причем не важно какая из этих строк была закомментирована 296 или 303.

Это говорит о правильном первоначальном предположении о том, что для успешной установки контракта его нужно разбить на отдельные модули. Но данный факт потребует внесения изменений в исходный код контрактов.

# Заключение

1. В процессе аудита было найдено много критических замечаний. В основном они связаны с отсутствием использования безопасной математики SafeMath и неверной логикой работы некоторых функций.
2. Ни в одном из представленных к изучению контрактов не используется актуальная версия Solidity – 0.4.25.
3. Наименования переменных и констант не по codestyle:

<https://solidity.readthedocs.io/en/v0.4.25/style-guide.html>

1. В соответствии с принципами объекто-ориентированного программирования – SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation и Dependency inversion) при установке данных контрактов в сеть их необходимо разбить на отдельные самостоятельные сущности. Установка контрактов (не всех), как отдельных сущностей позволит в процессе дальнейшей работы с DApp менять отдельные модули DApp без нарушения работы всего приложения.
2. Перед началом промышленной эксплуатации этого DApp потребуется существенная переработка всего исходного кода.