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

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

 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 |

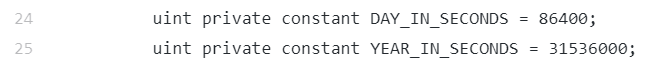
0x3f3C9E209Ed39CaB936726AbA6b385756092bE42  - [StayBit Token (STBT)](https://ropsten.etherscan.io/token/0x3f3c9e209ed39cab936726aba6b385756092be42)

Замечания:

1. Не актуальная версия Solidity
2. Наименования переменных и констант не по codestyle:

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

1. Не используется безопасная математика
2. (Контракт DateTime)



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

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

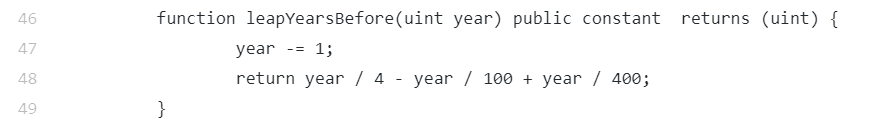
**uint private constant** YEAR\_IN\_SECONDS = 1 year;



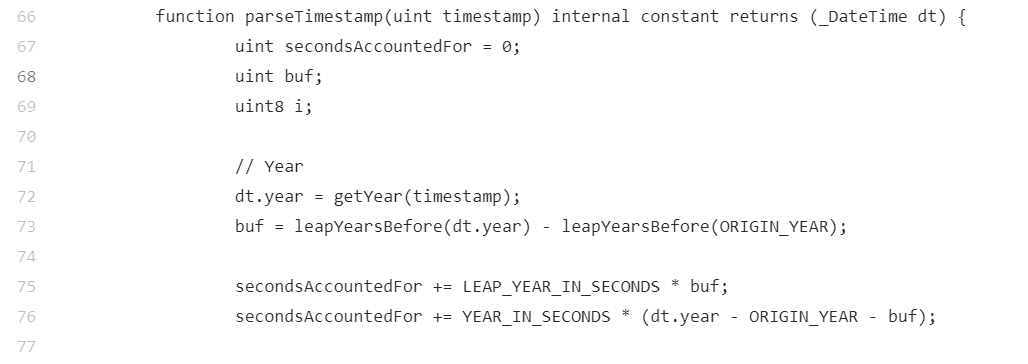
**uint endOfDay** = t2 + 1 day;

Не используется безопасная математика.

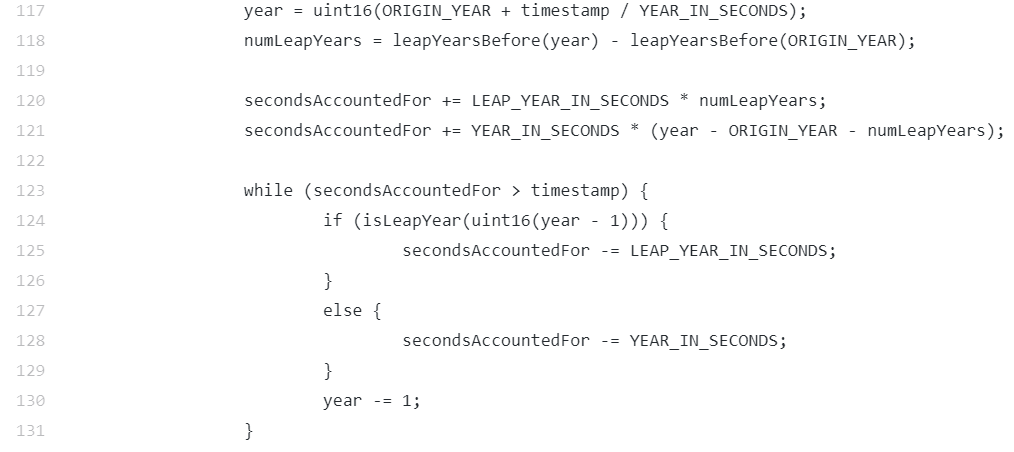
Код в следующих строках является не безопасным и его исполнение может привести к непредсказуемым последствиям:



Строки 47 и 48



Строки 73, 76



Строки 118, 121, 124, 125, 128, 130

1. Контракт (BaseEscrowLib) зачем делать константы и функции геттеры, если можно было сделать enum с областью видимости public

**int** nPotentialBillableDays = (**int**)(self.\_MoveOutDate - self.\_MoveInDate) / (60 \* 60 \* 24);

Можно заменить на day

Использование throw считается устаревшим

Строка 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().

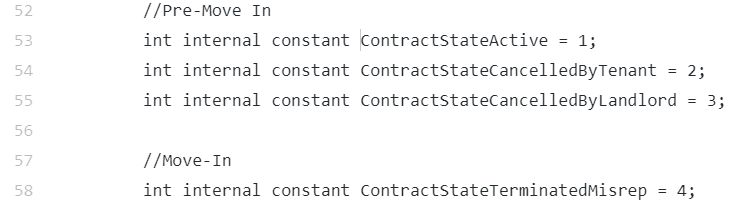
Причем так сложилось, что вызов revert происходит в более «легких» случаях (например при вызове if/else), а вызов assert() в более «тяжелых» случаях (например: превышение предельного значения для переменной, условие, которое не должно возникнуть).

В данном случае будет предпочтительнее использовать revert().

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

1. Контракт (BaseEscrowLib)

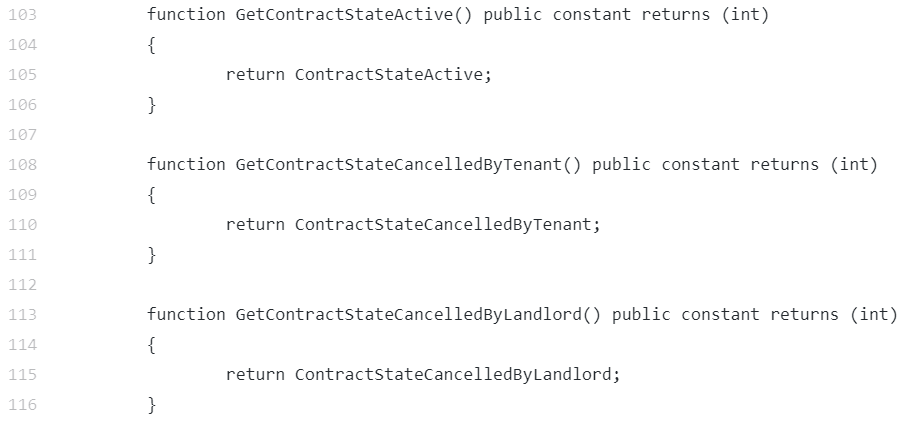
Использование типов данных enum позволит сделать код более компактным.



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);

}

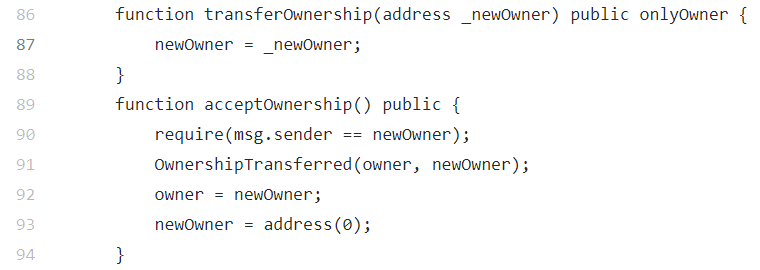
Тогда обращение из других библиотек будет осуществляться вместо такого вызова:

BaseEscrowLib.GetContractStateActive()

так:

BaseEscrowLib.GetContractState(ContractState.ACTIVE)

1. Контракт (MyToken) неправильная работа функций: transfer()



Перед выполнением присвоения нового значения адреса, нет проверки значения этого адреса на 0.

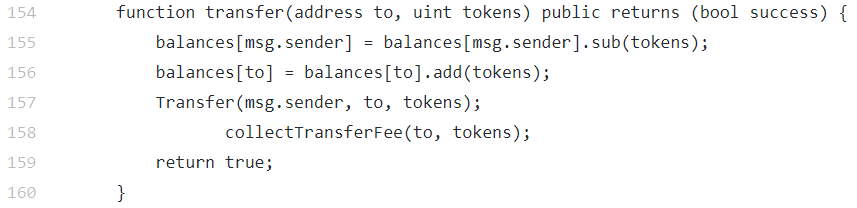
В случае введения неверного значения возможно возникновения ситуации потери контроля над управлением контрактом.

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. (Критично)



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; (Обратить внимание)

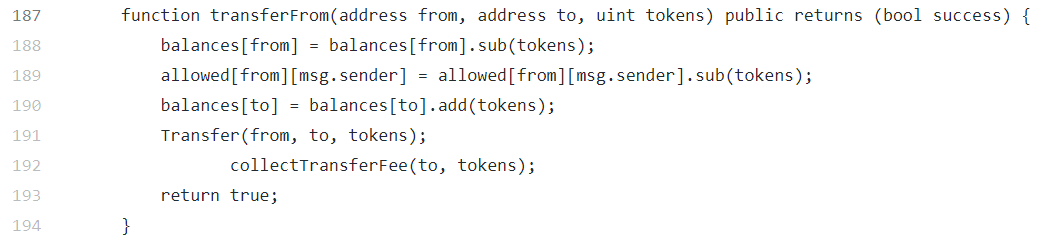


В стоках 155 и 156 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям. (Критично)

Необходимо осуществлять проверку на допустимость значений.

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

require(to != address(0));



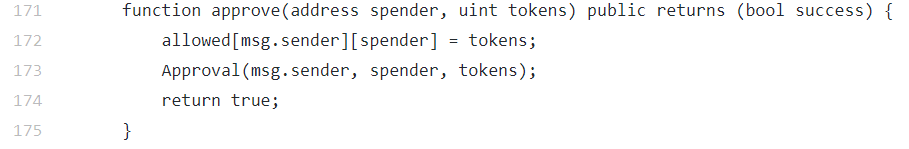
В стоках 188, 189 и 190 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям. (Критично)

Необходимо осуществлять проверку на допустимость значений.

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

require(to != address(0));

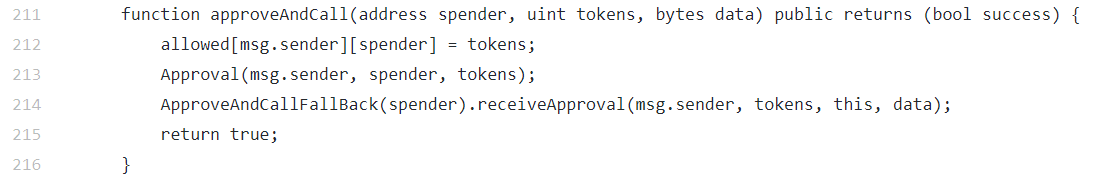
require(tokens <= \_allowed[from][msg.sender]);



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

Необходимо осуществлять проверку на допустимость значения:

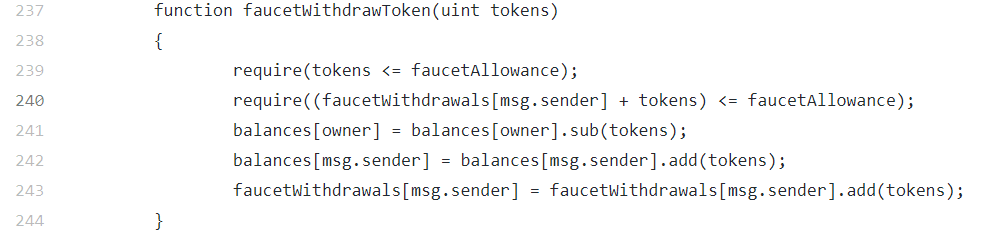
require(spender != address(0));



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

Необходимо осуществлять проверку на допустимость значения:

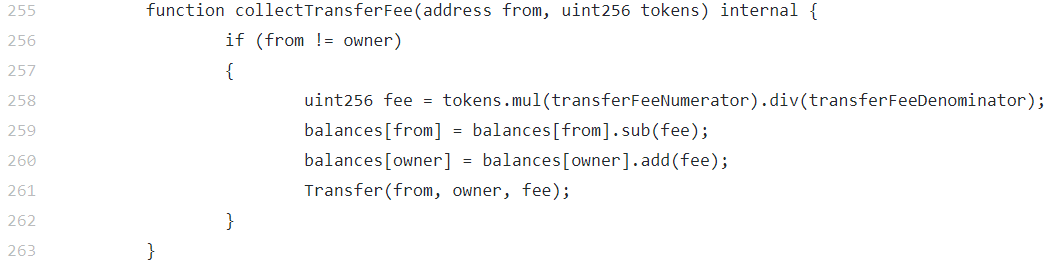
require(spender != address(0));



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. (Критично)

Необходимо осуществлять проверку на допустимость значений.

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

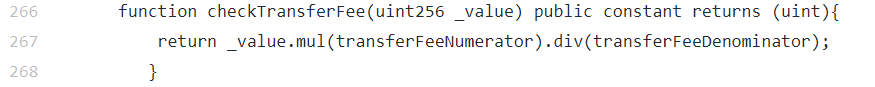


В стоках 258, 259, 260 изменение числа токенов происходит без проверки значения этих токенов на допустимость. Вызов данной функции с не корректными значениями приведет к непредсказуемым последствиям. (Критично)

Необходимо осуществлять проверку на допустимость значений.

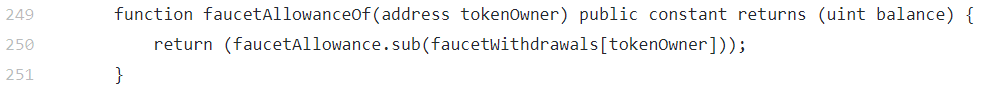
require(transferFeeDonominator > 0);

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



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

require(transferFeeDonominator > 0);



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

require(faucetAllowance >= faucetWithdrawals[tokenOwner]);

1. initialize public

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); }; |

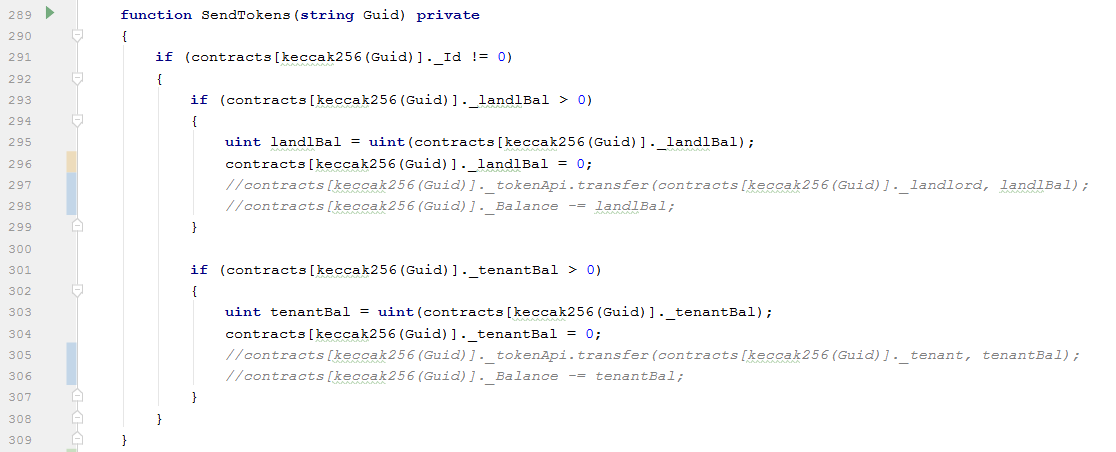
Но осуществить деплой контракта StayBitContractFactory включая все используемые в нем библиотеки не удалось, т.к. на экран была выведена ошибка  


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.

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.

**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. Тогда для использования функционала библиотек: BaseEscrowLib, FlexibleEscrowLib, ModerateEscrowLib, StrictEscrowLib**

Вместо

**import "./FlexibleEscrowLib.sol"**;  
**import "./ModerateEscrowLib.sol"**;  
**import "./StrictEscrowLib.sol"**;

Необходимо использовать следующий код.  
  
Описать интерфейсы.

**interface** IFlexibleEscrowLib {  
 **function** TenantTerminate(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantMoveIn(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantTerminateMisrep(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** LandlordTerminate(BaseEscrowLib.EscrowContractState **storage** self, **uint** SecDeposit) **public**;  
}  
  
**interface** IModerateEscrowLib {  
 **function** TenantTerminate(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantMoveIn(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantTerminateMisrep(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** LandlordTerminate(BaseEscrowLib.EscrowContractState **storage** self, **uint** SecDeposit) **public**;  
}  
  
**interface** IStrictEscrowLib {  
 **function** TenantTerminate(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantMoveIn(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** TenantTerminateMisrep(BaseEscrowLib.EscrowContractState **storage** self) **public**;  
 **function** LandlordTerminate(BaseEscrowLib.EscrowContractState **storage** self, **uint** SecDeposit) **public**;  
}

Провести инициализацию.

**function** SetLibrary(**address** addressFlexible, **address** addressModerate, **address** addressStrict) **public** onlyOwner  
{  
 IFlexibleEscrowLib FlexibleEscrowLib = IFlexibleEscrowLib(addressFlexible);  
 IModerateEscrowLib ModerateEscrowLib = IModerateEscrowLib(addressModerate);  
 IStrictEscrowLib StrictEscrowLib = IStrictEscrowLib(addressStrict);  
}

contracts[keccak256(Guid)].initialize();