Аудит смарт контрактов TST

Статус - завершен.

Код контрактов проверен на предмет программных закладок и критических ошибок способных привести к потере денег инвесторами, а также на предмет логики функционала и оптимизации стоимости транзакций.

Результат Аудита

Проблемы критического уровня:

Ошибки способные привести к потере средств, а также проблемы нарушающие обязательный функционал контрактов.

Выявлено: 0.

Проблемы среднего уровня.

Ошибки логики и функционала контрактов, не приводящие к потере средств.

Выявлено: 1.

Проблемы оптимизации.

Возможности снизить стоимость транзакций и уменьшить количество строк кода.

Выявлено: 5.

Примечания и рекомендации.

Советы и рекомендации, а также ошибки, не влияющие на функционал смарт контракта.

Выявлено: 5.

Смарт контракты

Библиотека SafeMath:

Данная библиотека используется для проведения безопасных вычислений (позволяет исключить overfrow и underflow в смарт-контрактах). Как правило, включает в себя функции прибавления, вычитания, умножения деления, извлечения модуля (**add**, **sub**, **mul**, **div**, **mod**).

- Примечание: в данной версии библиотеки функция извлечения модуля отсутствует за ненадобностью.

- Оптимизация: можно добиться небольшой уменьшения стоимости транзакций если преобразовать требование во второй строке функции **mul** (строка 32) на две строки: условие и требование. Итоговый код будет выглядеть следующим образом:

```
function mul(uint a, uint b) internal pure returns (uint c) {
    if (a == 0) {
        return 0;
    }
    c = a * b;
    require(c / a == b);
}
```

Контракт Owned:

Функционал контракта **Owned** реализует ограничение доступа к функциям контракта посредством модификатора **onlyOwner**. Модификатор **onlyOwner** ограничивает вызов функции ролью **owner**. Роль **owner** устанавливается при деплое контракта в значение адреса, с которого происходит деплой. Установка нового значения **owner** осуществляется посредством вызова функции **transferOwnership** через параметр address **newOwner**.

- Примечание: Конструктор контракта можно обозначить словом **constructor**(), без повторения имени контракта.

Функция transferOwnership:

Доступна только владельцу контракта

Принимает один параметр: адрес нового владельца

Устанавливает нового владельца.

- Проблема: В функции **transferOwnership** нет проверки на наличие адреса, следовательно из-за ошибки владельца функция может позволить навсегда потерять право владения контрактом. Дополнительная строчка должна выглядеть так:

```
require(_newOwner != address(0));
```

- Примечание: Также ввиду очевидной важности данной функции ее можно разбить на две из соображений безопасности:

```
function transferOwnership(address _newOwner) public onlyOwner {
    _transferOwnership(_newOwner);
}
function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    owner = _newOwner;
}
```

- Оптимизация: модификатор **onlyOwner** абсолютно не используются в контрактах TST. Никаких полномочий, функций для владельца контракта не предусмотрено. Единственный случай использования переменной **owner** в конструкторе можно исключить. Следовательно, без внесения таковых полномочий, контракт Owned не имеет смысла.

Контракт TST

Смарт контракт **TST** – токен стандарта ERC20.

Контракт наследует функционал контрактов **STokenInterface** и **Owned**.

Переменные:

running (по умолчанию устанавливается как true).

- Оптимизация: данная переменная не используется в контракте, следовательно ее можно убрать.

symbol – сокращение названия. (по умолчанию устанавливается как TST).

name – название токена. (по умолчанию устанавливается как TEST).

decimals – количество знаком после запятой (по умолчанию устанавливается как 18).

_totalSupply – количество первоначально созданных токенов (по умолчанию устанаваливается как 1 миллиард).

balances – переменная типа mapping, хранящая балансы держателей токена.

allowed - переменная типа mapping, хранящая значения доступные для вывода с баланса определенного участника другим пользователем.

Ивенты:

Transfer – перевод средств.

Approval – разрешение на вывод средств.

Основной фукционал:

Конструктор контракта (**function TST**) выполняется единоразово при создании контракта и не принимает параметров. Все переменные устанавливаются в значения указанные выше. Весь выпуск токенов отправляется на баланс владельцу контракта.

- Примечание: Конструктор контракта можно обозначить словом **constructor**(), без повторения имени контракта.

- Оптимизация: Выпуск токенов отправляется на переменную **owner**, которую можно заменить на **msg.sender** и полностью избавиться от использования контракта **Owned**.

- Примечание: Для вызова всех ивентов (строки 103, 131, 146, 161) можно использовать маркер **emit**.

Функция totalSupply:

Общедоступна для вызова.

Не принимает параметров.

Не изменяет данные в блокчейне , следовательно не требует отправки транзакции.

Возвращает значение первоначальной эмиссии **_totalSupply** с вычтенным балансом нулевого адреса.

- Оптимизация: По умолчанию вычитание баланса нулевого адреса не требуется.

Функция balanceOf:

Общедоступна для вызова.

Принимает один параметр: адрес держателя токена.

Не изменяет данные в блокчейне , следовательно не требует отправки транзакции.

Возвращает значение баланса заданного пользователя.

Функция transfer:

Общедоступна для вызова.

Изменяет данные в блокчейне, следовательно, требует отправки транзакции.

Принимает два параметра:

1. Адрес назначения
2. Количество токенов

Выполняет следующее:

1. Проверяет баланс отправителя на наличие нужной суммы.
2. Проверяет наличие адреса назначения.
3. Снимает сумму с адреса отправителя.
4. Начисляет сумму на адрес назначения.
5. Вызывает ивент **Transfer**.
6. Возвращает true (маркер выполнения функции).

Функция approve:

Общедоступна для вызова.

Изменяет данные в блокчейне, следовательно, требует отправки транзакции.

Принимает два параметра:

1. Адрес пользователя которому дается разрашение пользоваться токенами с адреса отправителя.
2. Количество токенов.

Выполняет следующее:

1. Сохраняет значение токенов доступных к использованию конечному пользователю в mapping allowed.
2. Вызывает ивент **Approval**.
3. Возвращает **true** (маркер выполнения функции).

Функция transferFrom:

Общедоступна для вызова.

Изменяет данные в блокчейне, следовательно, требует отправки транзакции.

Принимает три параметра:

1. Адрес, с которого необходимо списать токены.
2. Адрес назначения.
3. Количество токенов.

Выполняет следующее:

1. Проверяет баланс отправителя на наличие нужной суммы.
2. Проверяет разрешение на вывод средств.
3. Проверяет наличие адреса назначения.
4. Снимает сумму с адреса отправителя.
5. Вычитает снятую сумму из mapping **allowed**.
6. Начисляет сумму на адрес назначения.
7. Вызывает ивент **Transfer**.
8. Возвращает true (маркер выполнения функции).

Функция allowance:

Общедоступна для вызова.

Не изменяет данные в блокчейне, следовательно, не требует отправки транзакции.

Принимает два параметра:

1. Адрес, с которого дается разрешение.
2. Адрес, которому дается разрешение.
3. Возвращает значение разрешенных к выводу токенов.

Предупреждение об ответственности

Этот аудит касается только исходных кодов смарт контрактов и не должен рассматриваться как одобрение платформы, команды или компании.

Авторы

Аудит провёла команда **EthereumWorks.** По вопросам проведения аудитов и разработки смарт контрактов обращайтесь: Telegram - **@gafagilm**.