Final Security Audit of Narfex NarfexToken contract by profiterole.group

Conclusion

This audit was made by Profiterole Group https://profiterole.group/
Auditor: Vladimir Smelov.
Date: 2023-04-15

Based on the evaluation, the Client's smart contracts are considered secure.

Our team conducted a code functionality analysis, manual examination, test cases and automated tests using static analyzers. All issues identified during the automated analysis were manually inspected, and significant vulnerabilities are highlighted in the Audit Summary section.

The Client has acknowledged and fixed or accepted all the issues.

Scope

Pre Audit Scope

repo: https://github.com/narfex/fiat-factory
commit: `b8799a8ca2eebcbd91b300a18bedaf299a389b43`

- contracts/NarfexToken.sol (md5 `3cbdc268e5b9279af5187e3e0a79a9f8`)

Final Audit Scope

repo: https://github.com/narfex/fiat-factory
commit: `181324400942b9d36a5f4705b62c14cf76b2b825`

- contracts/NarfexToken.sol (md5 `211f02b6053de2c17e04ef5ca6c618dd`)

Final Deployed Contract

NarfexToken on Ethereum - https://etherscan.io/address/0xCc17e34794B6c160a0F61B58CF30AA6a2a268625#code

Methodology

1. Blind audit. Understand the structure of the code without reading any docs.
2. Ask questions to developers.
3. Run static analyzers.
4. Find problems with:

- backdoors;
- bugs;
- math;
- potential leaking of funds;
- potential locking of the contract;
- validate arguments and events;
- others.

Contract desrciption

This Solidity smart contract is for NarfexToken, an ERC20 token with added features, built on OpenZeppelin's ERC20 and Ownable contracts. The primary feature is the burn/farming fee system for token transfers. When users transfer tokens, a small fee (0.3% by default) is either burned or sent to a farming contract, depending on the total token supply. This mechanism helps maintain a balance between burning and farming rewards. The contract owner can change the farming contract address and the fee percentage at any time.

Additionally, the contract supports migration from an old token to the new NarfexToken. Users can migrate their balance using the "migrate" function, which checks the user's balance in the old token contract, transfers it to the new token contract, and mints the equivalent amount of new tokens for the user. This ensures a smooth transition for users holding the old token.

Result

LOW-1.

At

- NarfexToken.sol:9 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L9

```
contract NarfexToken is ERC20, Ownable {
```

In general, use "//" not "///" for non-NatSpec comments

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-2.

At

- NarfexToken.sol:11 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L11
- NarfexToken.sol:13 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L13

```
    uint16 constant PERCENT_PRECISION = 10**4; /// 10000 = 100%
```

```
    uint16 private _burnPercentage = 30; /// 0.3% of burn/farming fee
```

it is more efficient is to use uint256

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-3.

At

- NarfexToken.sol:13 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L13

```
    uint16 private _burnPercentage = 30; /// 0.3% of burn/farming fee
```

it is more efficient is to use uint256

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-4.

At

- NarfexToken.sol:24 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L24

```
    event Migrate(address account, uint256 amount);
```

place the events declaration on the top of the contract

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-5.

At

- NarfexToken.sol:24 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L24

```
    event Migrate(address account, uint256 amount);
```

use “address indexed account”

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-6.

At

- NarfexToken.sol:31 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#L31

```
        uint256 balance = IERC20(oldToken).balanceOf(msg.sender);
```

be consistent with using _msgSender or msg.sender, actually you dont need _msgSender at all

Status.

ACKNOWLEDGED - not a security/functional issue

LOW-7.

At

- NarfexToken.sol:34 - https://github.com/narfex/fiat-factory/blob/b8799a8ca2eebcbd91b300a18bedaf299a389b43/contracts/NarfexToken.sol#

L34

```
            IERC20(oldToken).transferFrom(msg.sender, address(this), balance);
```

use safeTransferFrom

Status.

ACKNOWLEDGED - not a security/functional issue, since the previous contract never returns “false” from transferFrom.