SMART CONTRACTS EXPERTISE — RIGHT WAY TO SUCCESS
Faceter Smart Contract Audit

If you have any questions concerning smart contract design and audit, feel free to contact zoia@oceanico.io

Content

Description of the complex of procedures for auditing a smart contract

1. Primary architecture review

- Checking the architecture of the contract.
- The correctness of the code.
- Checking for linearity, shortness, and self-documentation.
- Static verification and code analysis for validity and the presence of syntactic errors.

2. Comparison of requirements and implementation

- Checking the code of the smart contract for compliance with the requirements of the customer code logic, writing algorithms, matching the initial constant values.
- Identification of potential vulnerabilities

Faceter Token Contract Review

- Token protocol: ERC20
- Token name: Faceter Token
- Token Symbol: FACE
- Decimals: 18

Faceter Token Lock Contract Review

This smart-contract is used to lock the team tokens for 2 years.

List of audited files

The experts conducted an audit of 2 deployed contract on the list:

- Faceter Token Contract

https://etherscan.io/address/0x4695c7AC68eb86c1079c7d7D53Af2F42DB8a6799#code

- Faceter Token Lock Contract

https://etherscan.io/address/0x7d1efaa19817f519631e1568c725e2da19cea9bc#code

Review of smart contract #1

Faceter Token Contract Review

Faceter Token Lock Contract Review

Important

1. Token Contract contains unnecessary logic. Libraries and links to the contract are used, and the code of contract is hidden on the Etherscan. We recommend using proven OpenZeppelin solutions. This will simplify the architecture of the contract, make it transparent, reliable and will reduce the cost of the deploy contract.

2. Token Lock Contract. The contract contains an incorrect token address.

Code quality

1. This function initializes the contract, it must be done by the constructor.

```
137     function init(EToken2Interface _etoken2, string _symbol, string _name) returns(bool) {
138         if (address(etoken2) != 0x0) {
139             return false;
140         }
141         etoken2 = _etoken2;
142         etoken2Symbol = _bytes32(_symbol);
143         name = _name;
144         symbol = _symbol;
145         return true;
146     }
```

2. We recommend replacing this function with the constructor as follows:

```
137     function FaceterToken(EToken2Interface _etoken2, string _symbol, string _name) {
138         etoken2 = _etoken2;
139         etoken2Symbol = _bytes32(_symbol);
140         name = _name;
141         symbol = _symbol;
142     }
```

Gas optimisation

1. The contract contains an unused function. We recommend to remove it to save gas when deployed.

```
80      function _assemblyCall(address _destination, uint _value, bytes _data) internal returns(bool success) {
81          assembly {
82              success := call(div(mul(gas, 63), 64), _destination, _value, add(_data, 32), mload(_data), 0, 0)
83          }
84      }
```

2. We recommend to replace “if” with ”require(address(etoken2) == 0x0);” to save gas when deployed.

```
138         if (address(etoken2) != 0x0) {
139             return false;
140         }
```

3. We recommend to replace “if” with ”require(msg.sender == address(etoken2)); _; to save gas when deployed.

```
152         if (msg.sender == address(etoken2)) {
153             _;
154         }
```

4. We recommend to replace “if” with ”require(etoken2.isOwner (msg.sender, etoken2Symbol)); _; to save gas when deployed.

```
161         if (etoken2.isOwner(msg.sender, etoken2Symbol)) {
162             _;
163         }
```

5. We recommend to replace “if” with ”require(getVersionFor(_sender) == msg.sender); _; to save gas when deployed.

```
468         if (getVersionFor(_sender) == msg.sender) {
469             _;
470         }
```

6. We recommend to replace “if” with ”require(pendingVersion == 0x0); to save gas when deployed.

```
524         if (pendingVersion != 0x0) {
525             return false;
526         }
```

7. We recommend to replace “if” with ”require(_newVersion != 0x0); to save gas when deployed.

```
528         if (_newVersion == 0x0) {
529             return false;
530         }
```

8. We recommend to replace “if” with ”require(pendingVersion != 0x0); to save gas when deployed.

```
550         if (pendingVersion == 0x0) {
551             return false;
552         }
```

9. We recommend to replace “if” with ”require(pendingVersion != 0x0); to save gas when deployed.

```
567         if (pendingVersion == 0x0) {
568             return false;
569         }
```

10. We recommend to replace “if” with ”require(now >= pendingVersionTimestamp + UPGRADE_FREEZE_TIME); to save gas when deployed.

```
570         if (pendingVersionTimestamp + UPGRADE_FREEZE_TIME > now) {
571             return false;
572         }
```

11. We recommend to replace “if” with ”require(userOptOutVersion [msg.sender] == 0x0); to save gas when deployed.

```
587         if (userOptOutVersion[msg.sender] != 0x0) {
588             return false;
589         }
```

Results of contract audit

The information in this report is a list of recommendations that need to be followed to ensure the quality and safety of the smart contract.

The experts audited the contract. Based on the results, the developers of the smart contract were given recommendations for improving the optimization of the smart contract code.

For all questions regarding the audit and testing of the smart contract, we recommend contacting zoe@targead.com