OCEANICO.IO

THE ABYSS

SMART CONTRACTS EXPERTISE — RIGHT WAY TO SUCCESS
THE ABYSS 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.
- Correctness of the code.
- Check 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

3. Testing according to the requirements

- Control testing of a smart contract for compliance with specified customer requirements.
- Running tests of the properties of the smart contract in test net.

Terms of Reference for the creation of a smart contract

DAICO Smart Contract

TOKEN SALE DETAILS:

- Token symbol: ABYSS
- Decimals: 18

TOKEN SALE (DAICO):

- Sale date: APR 16 - MAY 16, 2018
- Rate: 1 ABYSS = 0.08 USD

(We accept ETH and BNB. We will set a course for ETH and BNB in a few days before the start of Token Sale, depending on the market rates)

Soft cap: 6.000.000 USD (If this goal is not met, all funds will be returned)

Max total token supply: 905.625.000 ABYSS

(Maximum token Circulation Supply during the first 6 months will be 426M. All unsold tokens will be burnt by Smart Contract)

INTERNATIONAL

- Hard cap: 18.000.000 USD + 300.000 BNB
- ETH

Minimum contribution is 0.2 ETH

First 24HRS Maximum contribution is 20 ETH

Days 2+ no ETH contribution limit

UNITED STATES

- BNB

BNB Tokens are acceptable from the international (non U.S.) participants only

Minimum contribution is 1000 BNB tokens

- BONUSES

FIRST 48HRS +25%

DAYS 3–7 +15%

DAYS 8–14 +10%

DAYS 15–21 +5%

- TOKENS DISTRIBUTION

Bounty 1%

Company & Advisors 21%

Reserve 18%

Foundation 20%

Crowdsale 40%

Sources

- https://www.theabyss.com/
- https://medium.com/theabyss/a-guide-to-the-abyss-daico-smart-contract-b66a04bddaa

List of audited files

Github:

https://github.com/theabyssportal/DAICO-Smart-Contract

The following 23 .sol files were audited:

- ICrowdsaleFund.sol
- ICrowdsaleReservationFund.sol
- IPollManagedFund.sol
- SafeMath.sol
- MultiOwnable.sol
- Ownable.sol
- BasePoll.sol
- ERC20Token.sol
- IERC20Token.sol
- ITokenEventListener.sol
- LockedTokens.sol
- ManagedToken.sol
- TransferLimitedToken.sol
- AbyssToken.sol
- Crowdsale.sol
- DateTime.sol
- Fund.sol
- ISimpleCrowdsale.sol
- Pausable.sol
- PollManagedFund.sol
- RefundPoll.sol
- ReservationFund.sol
- TapPoll.sol

Review of smart contract #1

The Abyss DAICO smart contract review #1

https://github.com/theabyssportal/DAICO-Smart-Contract

Important

1. Possible bug

1.1. BaseVoting (54): require(_startTime > now && _endTime > startTime);

1.1.1. _endTime > startTime always true, because startTime is state variable and it zero by default.

Code quality

2. Overview

2.1. Code quality is very good. Code style follow to the solidity recommendations. Almost all public methods have comments.

3. Critical

3.1. No critical issues found;

4. Gas usage

4.1. LockedTokens.Tokens: field lockEndTime might be stored as uint64. It saves about 1k gas;

4.2. bonusWindow#EndTime might be constants, it saves constructor gas; and the same thing with any TokenWallets;

4.3. Mappings telegramMembers and telegramMemberHadPayment might be merged to one mapping with uint (e.g. 1 means registered, 2 means had payed) value instead of bool.

5. General

5.1. External specification uses when it is not required: it does better performance when memory holded object passed as parameters;

5.2. Using modifiers for single invocation looks excessively, and it make code more unclear;

5.3. It's better to avoid using global properties (like msg.sender, msg.value) inside internal method, and pass values as arguments;

5.4. There is a duplicate logic implementation in processBNBContribution and processContribution which might be shared;

6. Notice

6.1. There is no possibility to transfer tokens to the locked account;

6.2. When constant defined as public, compiler generates accessor method for it;

6.3. If any hard cap will almost reached user must enter right value (calculated by themselves);

6.4. TokenPrice might be set only once (per crowdsale contract) - there is no chance to fix it; also, token price might be 0;

6.5. BNB contributions is not included to the soft cap checking.

Review of smart contract #2

https://github.com/theabyssportal/DAICO-Smart-Contract/tree/93524737dbcb5ec68060c04466847656ed6dff0b

Important

1. Overview

1.1. Code quality is very good. Code style follows the solidity recommendations. Almost all public method has comments.

2. Critical

2.1. No critical issue found

3. Possible bug

3.1. No critical issue found

4. Gas usage

4.1. LockedTokens.Tokens: field lockEndTime might be stored as uint64. It saves about 1k gas.

5. Common

5.1. The external specification uses when it is not required: it does better performance when memory holded object passed as parameters

5.2. Using modifiers for single invocation looks excessively, and it make the code more unclear.

5.3. It's better to avoid using global properties (like msg.sender, msg.value) inside an internal method, and pass values as arguments.

5.4. Using assert() in ManagedToken(23,28). It's better to use require() instead. If transaction failed with assert() - it just reverts and burns all remaining gas. If transaction failing with require - it will undo all changes, refund all remaining gas, and also can return value.

5.5. There is a duplicate logic implementation in processBNBContribution and processContribution which might be shared. 2Also duplicate logic in isValidContribution and isValidBNBContribution.

6. Notice

6.1. If any hard cap will almost reached user must enter a right value (calculated by themselves). From guide: "If the hard cap is exceeded, the excess is returned to respective contributor.". But, if value exceeds hard cap, the transaction will be reverted.

7. Testing

https://github.com/kirillsurkov/AbyssToken

7.1. Testing the contract was successful. No critical issue found.

Results of contract audit

https://github.com/theabyssportal/DAICO-Smart-Contract

The information in this report is a list of recommendations what needs to be done to ensure the quality and security of the smart contract.

The OCEANICO experts conducted the verification of the smart contract. Based on the results, the customer's developers were given recommendations for optimizing the smart contract code.

This smart contract complies with the specifications specified in the terms of reference.

During the audit of the contract, critical errors and possible vulnerabilities were not identified.

For all questions regarding the audit and testing of the smart contract, we recommend contacting zoia@oceanico.io