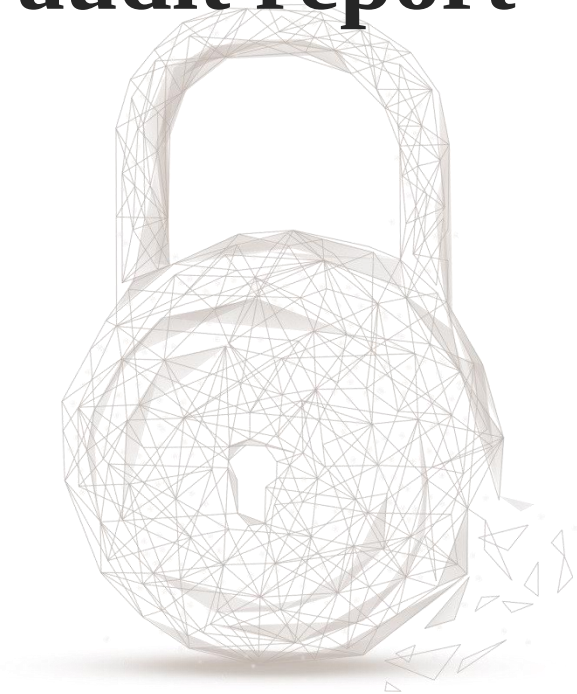
BEOSIN
Blockchain Security

Smart contract security audit report



Audit Number: 202104161645

Report Query Name: ETToken

Smart Contract Link:

https://github.com/ethst20210317/ETHST-Project/blob/main/ETToken.sol

Smart Contract Commit:

c8b18a62d757fcfe7b83aee735ec342fdd9505dd

Start Date: 2021.04.13

Completion Date：2021.04.16

Overall Result：Pass

Audit Team: Beosin (Chengdu LianAn) Technology Co. Ltd.

Audit Categories and Results:

No.	Categories	Subitems	Results
1	Coding Conventions	Compiler Version Security	Pass
		Deprecated Items	Pass
		Redundant Code	Pass
		SafeMath Features	Pass
		require/assert Usage	Pass
		Gas Consumption	Pass
		Visibility Specifiers	Pass
		Fallback Usage	Pass
2	General Vulnerability	Integer Overflow/Underflow	Pass
		Reentrancy	Pass
		Pseudo-random Number Generator (PRNG)	Pass
		Transaction-Ordering Dependence	Pass
		DoS (Denial of Service)	Pass
		Access Control of Owner	Pass
		Low-level Function (call/delegatecall)	Pass

		Security	
		Returned Value Security	Pass
		tx.origin Usage	Pass
		Replay Attack	Pass
		Overriding Variables	Pass
3	Business Security	Business Logics	Pass
		Business Implementations	Pass

Disclaimer: This report is made in response to the project code. No description, expression or wording in this report shall be construed as an endorsement, affirmation or confirmation of the project. This audit is only applied to the type of auditing specified in this report and the scope of given in the results table. Other unknown security vulnerabilities are beyond auditing responsibility. Beosin (Chengdu LianAn) Technology only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Beosin (Chengdu LianAn) Technology lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The security audit analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Beosin (Chengdu LianAn) Technology before the issuance of this report, and the contract provider warrants that there are no missing, tampered, deleted; if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Beosin (Chengdu LianAn) Technology assumes no responsibility for the resulting loss or adverse effects. The audit report issued by Beosin (Chengdu LianAn) Technology is based on the documents and materials provided by the contract provider, and relies on the technology currently possessed by Beosin (Chengdu LianAn). Due to the technical limitations of any organization, this report conducted by Beosin (Chengdu LianAn) still has the possibility that the entire risk cannot be completely detected. Beosin (Chengdu LianAn) disclaims any liability for the resulting losses.

The final interpretation of this statement belongs to Beosin (Chengdu LianAn).

Audit Results Explained:

Beosin (Chengdu LianAn) Technology has used several methods including Formal Verification, Static Analysis, Typical Case Testing and Manual Review to audit three major aspects of smart contract ETToken, including Coding Standards, Security, and Business Logic. **The ETToken contract passed all audit items. The overall result is Pass. The smart contract is able to function properly.**

Audit Contents:

1. Coding Conventions

Check the code style that does not conform to Solidity code style.

1.1 Compiler Version Security

- Description: Check whether the code implementation of current contract contains the exposed solidity compiler bug.
- Result: Pass

1.2 Deprecated Items

- Description: Check whether the current contract has the deprecated items.
- Result: Pass

1.3 Redundant Code

- Description: Check whether the contract code has redundant codes.
- Result: Pass

1.4 SafeMath Features

- Description: Check whether the SafeMath has been used. Or prevents the integer overflow/underflow in mathematical operation.
- Result: Pass

1.5 require/assert Usage

- Description: Check the use reasonability of 'require' and 'assert' in the contract.
- Result: Pass

1.6 Gas Consumption

- Description: Check whether the gas consumption exceeds the block gas limitation.
- Result: Pass

1.7 Visibility Specifiers

- Description: Check whether the visibility conforms to design requirement.
- Result: Pass

1.8 Fallback Usage

- Description: Check whether the Fallback function has been used correctly in the current contract.
- Result: Pass

2. General Vulnerability

Check whether the general vulnerabilities exist in the contract.

2.1 Integer Overflow/Underflow

- Description: Check whether there is an integer overflow/underflow in the contract and the calculation result is abnormal.
- Result: Pass

2.2 Reentrancy

- Description: An issue when code can call back into your contract and change state, such as withdrawing HT.
- Result: Pass

2.3 Pseudo-random Number Generator (PRNG)

- Description: Whether the results of random numbers can be predicted.
- Result: Pass

2.4 Transaction-Ordering Dependence

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: Pass

2.5 DoS (Denial of Service)

- Description: Whether exist DoS attack in the contract which is vulnerable because of unexpected reason.
- Result: Pass

2.6 Access Control of Owner

- Description: Whether the owner has excessive permissions, such as malicious issue, modifying the balance of others.
- Result: Pass

2.7 Low-level Function (call/delegatecall) Security

- Description: Check whether the usage of low-level functions like call/delegatecall have vulnerabilities.
- Result: Pass

2.8 Returned Value Security

- Description: Check whether the function checks the return value and responds to it accordingly.
- Result: Pass

2.9 tx.origin Usage

- Description: Check the use secure risk of 'tx.origin' in the contract.
- Result: Pass

2.10 Replay Attack

- Description: Check whether the implementation possibility of Replay Attack exists in the contract.
- Result: Pass

2.11 Overriding Variables

- Description: Check whether the variables have been overridden and lead to wrong code execution.

- Result: Pass

3. Business Security

Check whether the business is secure.

3.1 Business analysis of Contract Token

Token name	ET TOKEN
Token symbol	ET
decimals	18
totalSupply	Initially is 0 (mintable, max 100 million)
Token type	HRC-20

Table 1 Basic Token Information of ET TOKEN

3.2 Other function

(1) updatePoolAddress

- Description: This function is used to modify the addresses of 5 pools. The poolId parameter is used to specify the pool id, and the pool parameter is used to specify the new address.

```
78      function updatePoolAddress(uint256 poolId, address pool)
79          external
80          virtual
81          requireImpl
82      {
83          require(poolProportion[poolId] > 0, "Pool does not exist");
84          pollAddress[poolId] = pool;
85          addressPoll[pool] = poolId;
86          emit UpdatePoolAddress(poolId, pool);
87      }
```

Figure 1 Source code of updatePoolAddress

- Result: Pass

(2) getPoolLastBlock

- Description: Returns the block number of the last minting of the specified pool.

```
89      function getPoolLastBlock(uint256 poolId)
90          internal
91          view
92          virtual
93          returns (uint256)
94      {
95          uint256 lastBlock = poolLastBlock[poolId];
96
97          if (lastBlock == 0) {
98              lastBlock = MintStartBlock;
99          }
100         return lastBlock;
101     }
```

Figure 2 Source code of getPoolLastBlock

- Result: Pass

(3) getPoolLastStratBlock

- Description: Returns the *_lastStartBlock* of the last minting of the specified pool.

```
103     function getPoolLastStratBlock(uint256 poolId)
104         internal
105         view
106         virtual
107         returns (uint256)
108     {
109         uint256 lastBlock = poolLastStartBlock[poolId];
110
111         if (lastBlock == 0) {
112             lastBlock = MintStartBlock;
113         }
114         return lastBlock;
115     }
```

Figure 3 Source code of getPoolLastStratBlock

- Result: Pass

(4) calculateMint

- Description: Specify the number of blocks and pools, and calculate the corresponding number of mintable tokens *_reward*.

```
117     function calculateMint(uint256 blockNumber, uint256 poolId)
118         public
119         view
120         virtual
121         returns (
122             uint256 _lastStartBlock,
123             uint256 _releaseAmount,
124             uint256 _reward
125         )
126     {
127         uint256 proportion = poolProportion[poolId];
128         require(proportion > 0, "Pool does not exist");
129         if (blockNumber == 0) {
130             blockNumber = block.number;
131         }
132         uint256 lastBlock = getPoolLastBlock(poolId);
133         if (blockNumber <= lastBlock) {
134             return (_lastStartBlock, _releaseAmount, _reward);
135         }
136 
137         _lastStartBlock = getPoolLastStratBlock(poolId);
138         bool updateBlockCount = blockNumber.sub(_lastStartBlock) > Constant.DAY_BLOCK_COUNT;
139         uint256 rel =
140             updateBlockCount
141                 ? uint256(poolTotalReward[poolId]).sub(
142                     uint256(poolLastReleaseReward[poolId]).mul(proportion).div(
143                         100
144                     )
145                 )
146                 : 0;
147         uint256 releaseAmount = poolLastReleaseReward[poolId];
148         uint256 subBlockCount = blockNumber.sub(_lastStartBlock);
149         uint256 blockCount = blockNumber.sub(lastBlock);
150         uint256 reward = 0;
151         if (updateBlockCount) {
152             uint256 day = subBlockCount.div(Constant.DAY_BLOCK_COUNT);
153             for (uint256 i = 0; i < day; i++) {
154                 uint256 release = 0;
155                 if (TotalMintAmount > releaseAmount) {
156                     release = TotalMintAmount.sub(releaseAmount).div(1000);
157                 }
158                 reward = reward.add(release);
159                 releaseAmount = releaseAmount.add(release);
160                 _lastStartBlock = _lastStartBlock.add(Constant.DAY_BLOCK_COUNT);
161             }
162             blockCount = blockNumber.sub(_lastStartBlock);
163         }
164 
165         uint256 add =
166             TotalMintAmount.sub(releaseAmount).div(1000).mul(blockCount).div(
167                 Constant.DAY_BLOCK_COUNT
168             );
169         // 加上当日区块的释放
170         reward = reward.add(add).mul(proportion).div(100).sub(rel);
171         return (_lastStartBlock, releaseAmount, reward);
172     }
```

Figure 4 Source code of calculateMint

Result: Pass

(5) mint

- Description: Only the designated 5 pool addresses can call the *mint*. After calling, it will be mint tokens to caller, the number of tokens minted is related to the input blockNumber.

```
175     function mint(uint256 blockNumber) external virtual {
176         uint256 pid = addressPoll[msg.sender];
177         require(pid > 0, "Pool does not exist");
178 
179         //允许提前超发
180         //require(blockNumber <= block.number, "Illegal block number");
181 
182         (uint256 _lastStartBlock, uint256 _releaseAmount, uint256 _reward) =
183             calculateMint(blockNumber, pid);
184         if (_reward > 0) {
185             if (_lastStartBlock > 0) {
186                 poolLastStartBlock[pid] = _lastStartBlock;
187                 poolLastReleaseReward[pid] = _releaseAmount;
188             }
189 
190             _mint(msg.sender, _reward);
191             poolLastBlock[pid] = blockNumber;
192             poolTotalReward[pid] = poolTotalReward[pid].add(_reward);
193         }
194     }
```

Figure 5 Source code of mint

- Result: Pass

4. Conclusion

Beosin(ChengduLianAn) conducted a detailed audit on the design and code implementation of the smart contract ETToken. The problems found by the audit team during the audit process have been notified to the project party and reached an agreement on the repair results, the overall audit result of the ETToken is **Pass**.

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com