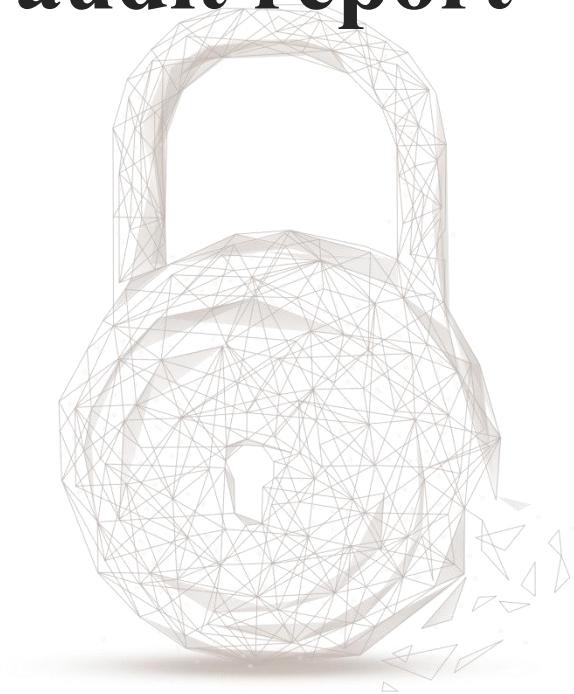
BEOSIN

Blockchain Security

Smart contract security audit report



Audit Number: 202104302110

Report Query Name: BLES_Staking

Smart Contract Link:

Staking: https://github.com/BlindBoxesNFT/blindboxes-contracts/blob/main/contracts/Staking.sol

VoteStaking: https://github.com/BlindBoxesNFT/blindboxes-contracts/blob/main/contracts/VoteStaking.sol

Origin commit hash: 40b34a9603fa9b49b0b22e95fd07106095c39e73

Final commit hash: 3409c6cba1952840a7d895ae5106bdc66ee59a53

Start Date: 2021.04.26

Completion Date: 2021.04.30

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) Security	Pass
		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 contracts Staking and VoteStaking of in project BLES_Staking, including Coding Standards, Security, and Business Logic. **The Staking and VoteStaking contracts 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 contract and change state, such as withdrawing BNB.
- 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 the whether the implement 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

3.1 Business analysis of contract Staking

(1) Add new staking pool

- Description: The contract implements the *add* function to add the staking pool of token, which can only be called by the owner of the contract. When adding a new staking pool, the owner will initialize the information of the pool, including the address of the token, the start block and end block of the reward distribution, and the number of reward tokens in each block. The owner can choose whether to execute the *massUpdatePools* function to update all the stake pools.
- Related functions: *add, massUpdatePools*
- Result: Pass

(2) Modify the parameters of the staking pool

- Description: The contract implements the *set* function to modify the parameters of the staking pool, which can only be called by the owner of the contract. This function can modify the start block and end block of the reward distribution, and the number of reward tokens in each block. The owner can choose whether to execute the *massUpdatePools* function to update all the stake pools.
- Related functions: *set, massUpdatePools*
- Result: Pass

(3) Update specified staking pool

- Description: The contract implements the *updatePool* function to update the specified staking pool information. Calling this function requires that the current block number is greater than the last reward block number, and the deposited token of the specified staking pool is greater than 0. After calculating the staking pool's BLES rewards, *accRewardPerShare* and *lastRewardBlock* of pool will be updated. If the current pool is voting Pool, then the *updatePool* function of VotingStaking contract will be called.
- Related functions: *updatePool, getReward*
- Result: Pass

(4) Deposit token

- Description: The contract implements the *deposit* function for users to deposit tokens to the specified staking pool for rewards. When the users call this function, the current staking pool information will be updated first. If the user has previously staked in the current staking pool, the reward will be calculated and add to user record. Then the *safeTransferFrom* function will be called to transfer the user's corresponding tokens into this contract, so the user needs to authorize this contract in advance. Finally update user-related parameters.
- Related functions: *deposit, updatePool, safeTransferFrom*
- Result: Pass

(5) Withdraw token

- Description: The contract implements the *withdraw* function to enable user to withdraw the staked tokens from the specified staking pool. If the pool is a voting pool, user will not be able to withdraw the voted token. After the voting, these tokens can be withdrawn together with the staked tokens in the VoteStaking contract. The relevant parameters of the user and pool will be updated before sending the tokens to be withdrawn to the user.
- Related functions: *withdraw, getUserStakedAmount, updatePool, safeTransfer*
- Result: Pass

(6) Claim reward

- Description: The contract has 3 claim reward-related functions, *claimNow, claimLater* and *claimLaterReady*. *claimNow* will directly claim the reward, but half of the claimed reward will be destroyed. *claimLater* will establish a lock-up for the corresponding reward, and the user can call *claimLater* to claim for all the rewards after the lock-up ends. The rewards claimed by users consist of three parts. 1. The first part is the staking reward in the staking contract; 2. the second part is the voting staking reward in the VoteStaking contract. This part of the reward will be sent directly to the user without locking and destroying; 3. the third part It is the second part of the same reward. This part of the reward will be sent to the user together with the first part in the staking contract. Depending on the claim method, it may be necessary to lock the position or destroy half of it.
- Related functions: *claimNow, claimLater claimLaterReady, updatePool, claim*
- Result: Pass

(7) Proposal

- Description: The owner can add proposal. Proposal cannot overlap. Proposal has a start and end block limit, and only users within this range can vote. The proposal will also set up a staking pool in the VoteStaking contract, and users will deposit the same number of BLES into the VoteStaking contract each time they vote, and withdraw them after the end of the proposal. Users can vote on multiple options of the proposal at the same time, as long as the user deposits in the voting pool. The maximum number of votes each time cannot exceed the deposit.
- Related functions: *addProposal, voteProposal, getProposalOptionVotes, getProposalUserOptionVotes*
- Result: Pass

(8) Contract parameter settings

- Description: The owner can set the relevant parameters of the contract. **Note that the owner should use these functions with caution. If the setting is wrong, the user will not be able to normally deposit, claim reward, withdraw, vote, etc.**
- Related functions: *setClaimWaitTime, setVoteStaking, setVotingPoolId, setMaximumVotingBlocks*
- Result: Pass

3.2 Business analysis of contract VoteStaking

(1) Set pool parameters

- Description: Each proposal in the staking contract will set the pool parameter in this contract. The pool's reward start and end block number is the start and end block of the proposal.
- Related functions: *set*
- Result: Pass

(2) Voting related

- Description: Users who vote in the Staking contract will add a deposit to this contract, which can generate rewards, and users can claim rewards. After the proposal is finished, users can withdraw the deposit when voting through the Staking contract. Users cannot directly call the *deposit, withdraw,* and *claim* functions of this contract, and must perform related operations through the staking contract.
- Related functions: *set*
- Result: Pass

(3) Change Staking contract address

- Description: The owner can change Staking contract address. **Note that the owner should use this function with caution. If the Staking contract address is wrong, the user will not be able to normally *deposit, claim, withdraw*, etc. It will also affect the function of the Staking contract.**
- Related functions: *changeStakingAddress*
- Result: Pass

4. Conclusion

Beosin(ChengduLianAn) conducted a detailed audit on the design and code implementation of the smart contracts Staking and VoteStaking in project BLES. The problems found by the audit team during the audit process have been notified to the project party and repaired. The overall audit result of the smart contracts Staking and VoteStaking is **Pass**.

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com