BEOSIN

Blockchain Security

Smart contract security audit report

Audit Number：202011252112

Report Query Name： Stkr

Smart Project Name：

Stkr

Smart Contract Address：

None

Smart Contract Address Link：

None

Start Date：2020.11.10

Completion Date：2020.11.25

Overall Result：Pass (Developing)

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

Note: Audit results and suggestions in code comments

Disclaimer: 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 project Stkr, including Coding Standards, Security, and Business Logic. **Stkr contract passed all audit items. The overall result is Pass (Note: some related functions are developing, some parts of functions are not fully implemented, the current finished function logic is pass).** Please find below the basic information of the smart contract:

Business Audit:

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 ETH.
- 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 weather 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 AETH_R1 Contract Audit

3.1.1 Basic token information of AETH

The contract implements a basic ERC20 token, and its basic information is as follows:

Token name	Input when deploy
Token symbol	Input when deploy
decimals	Input when deploy
totalSupply	Initial supply is 0 (Mintable without cap; Burnable)
Token type	TRC20

Table 2 – Basic Token Information

3.1.2 AETH Token Functions

- Description: This contract token implements the basic functions of ERC20 standard tokens, and token holders can call corresponding functions for token transfer, approve and other operations.
- Related functions: *name, symbol, decimals, balanceOf, transfer, transferFrom, allowance, approve*
- Safety Suggestion: Beware that changing an allowance with this method brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering. It is recommended that users reset the allowance to zero, and then set a new allowance.
- Result: **Pass**

3.1.3 AETH Token burning

- Description: Users who hold the tokens of this contract can call the *burn* function to destroy their specific number of tokens.
- Related functions: *burn*
- Safety Suggestion: None

- Result: **Pass**

3.1.4 AETH Token minting

- Description: This contract implements the *mint* function to issue tokens to a specified address. The function limits that the caller should be the _globalPoolContract address. And the minting amount is according to the set minting ratio. The contract owner can update(decrease) the minting ratio. Mistake operation can cause the ratio is decreased to 0, then the mint function will be invalid.
- Related functions: *mint*, *updateRatio*
- Safety Suggestion: Cautiously using function updateRatio is recommended.
- Result: **Pass**

3.1.5 AETH Token management

- Description: This contract inherits the Ownable module, the related functions are implemented there. The owner of the contract (the default is the contract deployer) can call the *transferOwnership* function to transfer the management permission of the contract to a specified non-zero address.
- Related functions: *transferOwnership*
- Safety Suggestion: None
- Result: **Pass**

3.2 GlobalPool_R17 Contract Audit

3.2.1 Stake ETH

- Description: This contract implements the internal function *_stake* for stake users to stake ETH to this contract. Users call the *stake* function to do this operation. This function requires that the minimum staking amount should be 0.5 ETH(500 finney), related staking information is updated in this internal function. Note: this function is limited by the modifier notExitRecently, and because of the slash related functions included in this modifier are developing now, there could be some questions in the future. Currently the function logic is pass.
- Related functions: *stake*, *_stake*
- Safety Suggestion: None
- Result: **Pass**

3.2.2 Top up ETH

- Description: This contract implements the *topUpETH* function for a provider to deposit and stake ETH to this contract for related operations. This function requires that the minimum staking amount should be 2 ETH. Then the internal function *_stake* is called to stake corresponding amount of ETH. Then the corresponding staking rewards is calculated and added into specified staker address. Note: this function is limited by the modifier notExitRecently, and because of the slash related functions included

in this modifier are developing now, there could be some questions in the future. Currently the function logic is pass.

- Related functions: *topUpETH*, *_stake*
- Safety Suggestion: None.
- Result: **Pass**

3.2.3 Unstake ETH

- Description: This contract implements the *unstake* function for providers to withdraw the staked ETH to the caller before their pending stakes have been cleared(sent to Beacon chain). This function requires that the caller should have provider balance. Note: this function is limited by the modifier notExitRecently, and because of the slash related functions included in this modifier are developing now, there could be some questions in the future. Currently the function logic is pass.
  - Related functions: *stake*, *_stake*
  - Safety Suggestion: None
  - Result: **Pass**

3.2.4 Claim AETH reward

- Description: This contract implements the functions *claim* and *claimFor* for stake users to claim AETH rewards for himself or a specified staker. The value of _rewards[staker] is not updated. Note: these functions are limited by the modifier notExitRecently, and because of the slash related functions included in this modifier are developing now, there could be some questions in the future. Currently the function logic is pass.
  - Related functions: *claim*, *claimFor*, *_claim*
  - Safety Suggestion: Updating related value is recommended.
  - Fixed Result: Fixed. The project party said the not updated reward value is their original design intent, the total reward of a specified staker should be known. The redundant line is deleted.
  - Result: **Pass**

3.2.5 Push To Beacon Chain (ETH2.0)

- Description: This contract implements the function *pushToBeacon* for the operator or contract owner to deposit 32 ETH to the DepositContract address. The corresponding stake rewards(AETH) of each staker is calculated in this function.
  - Related functions: *pushToBeacon*
  - Safety Suggestion: None
  - Result: **Pass**

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com