



Smart contracts security assessment

Final report

[Tariff: Top](#)

Brick By Brick

September 2023



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	4
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	6
7. Conclusion	10
8. Disclaimer	11

Introduction

The report has been prepared for **Brick By Brick**.

The audited contract is a Ponzi-like staking with rewards in the same token with fixed ROI without explicitly defined source of such rewards.

Up to 2.5% of deposit goes to the marketing address, 0.5% - to the dev address, 3% - to the admin address, and 61% - to the bbb address. Claiming and compounding requires 3.5% fee to the claim address and 0.5% to dev address to be taken. All these external wallets are out of scope and can be either EOA or contracts.

The SHA-1 hashes of audited files are:

brick (2).sol 863ded9a8c48ab7ea8ad82d0bfc9f5cba20ddd9

Update. The updated code was deployed to the following address in the BNB Smart Chain:

StakingContract [0xCCACc32C4429683c4035C6A497D04ec78dDe0961](https://bscscan.com/address/0xCCACc32C4429683c4035C6A497D04ec78dDe0961).

Name	Brick By Brick
Audit date	2023-09-14 - 2023-09-15
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
StakingContract	0xCCACc32C4429683c4035C6A497D04ec78dDe0961

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed

<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

Classification of issue severity

High severity

High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity

Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity

Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Owner can change token address (StakingContract)

Status: Open

Modification of the staking token address will break the economic model of the contract as users will receive different tokens compared to the ones they've staked.

```
function setCoin(address _coin) external {
    require(
        msg.sender == adminWallet,
        "Only admin can whitelist addresses"
    );
    IERC20(coin).transfer(msg.sender, IERC20(coin).balanceOf(address(this)));
    coin = _coin;
}
```

Recommendation: Remove the `setCoin` function or include restrictions, e.g., no active users.

2. Source of rewards (StakingContract)

Status: Open

The staking offers fixed ROI percent but there's no source of rewards to harden this reward rate. The only possible source is the unclear nature of bbb wallet, which receives 61% of all deposits. If there will be not enough rewards, the claim and compound functions become unusable as token transfer will be reverted.

Recommendation: Fixed ROI should be removed, or source of rewards should be made public.

3. Logic error in claim function (StakingContract)

Status: Fixed

The `claimedAmount[msg.sender]` is increased twice: once in `if-else` section and then in the end. This results in constant revert of `claim` function once

```
uint256 remainingPayment = (totalStaked[msg.sender] * 3) - claimedAmount[msg.sender]
```

starts underflowing.

```
function claim() public {
    uint256 claimableAmount = calculateClaimableAmount(msg.sender);
    require(claimableAmount > 0, "No claimable amount");
    uint256 claimedAmountAfterTax = 0;
    uint256 claimDevFee = 0;
    uint256 claimWalletFee = 0;
    if (
        claimedAmount[msg.sender] + claimableAmount <=
        totalStaked[msg.sender] * 3
    ) {
        ...
        claimedAmount[msg.sender] += claimableAmount;
    } else {
        ...
        claimedAmount[msg.sender] += remainingPayment;
    }
    ...
    lastClaimTimestamp[msg.sender] = block.timestamp;
}
```

Recommendation: Remove the line `claimedAmount[msg.sender] += claimableAmount` in the end of `claim` function.

Medium severity issues

No issues were found

Low severity issues

1. Division before multiplication (StakingContract)

Status: Open

A loss of precision may occur in `calculateClaimableAmount` function:

```
function calculateClaimableAmount(address _user)
    public
    view
    returns (uint256)
{
    uint256 stakedAmount = totalStaked[_user];
    uint256 diff = block.timestamp - lastClaimTimestamp[msg.sender];
    uint256 claimableAmount = diff *
(calculateClaimAmountInSeconds(calculateROIPercentage(stakedAmount) / 1e18,
stakedAmount));
    return claimableAmount;
}

function calculateClaimAmountInSeconds(uint256 _percentage, uint256 _amountToken)
    internal
    pure
    returns (uint256)
{
    return (((_amountToken / 30 / 24 / 60 / 60) * (_percentage)) / 100);
}
```

2. Immutable variables (StakingContract)

Status: Fixed

Constructor-initializing variables can be declared as immutable: `marketingWallet`, `bbbWallet`, `adminWallet`, `claimWallet`, `devFeeWallet`, `decimal`.

3. Decimals possible mismatch (StakingContract)

Status: Partially fixed

The decimal variable is not derived from the actual `IERC20(coin).decimals()` value. Moreover, the coin address can be updated, but `decimal` can't.

Recommendation: Remove or modify `setCoin` function. Initialize decimal variable with on-chain data.

Conclusion

Brick By Brick StakingContract contract was audited. 3 high, 3 low severity issues were found. 1 high, 1 low severity issues have been fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.



 Guard