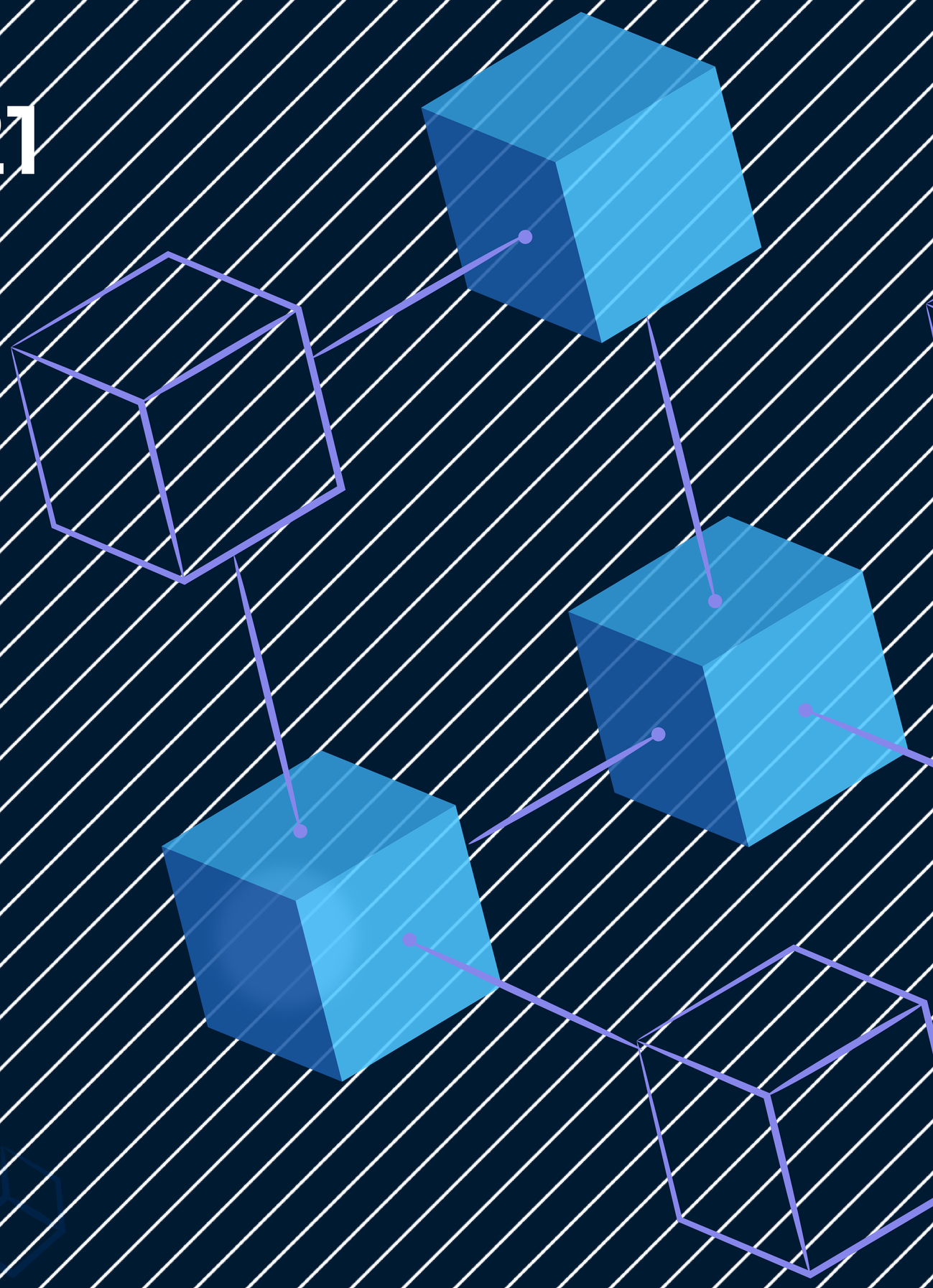
QuillAudits

# Audit Report
# December, 2021



For

BollyCoin

# Contents

## Scope of the Audit

The scope of this audit was to analyze and document the Bollycoin smart contract codebase for quality, security, and correctness.

## Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

# Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

**Structural Analysis**
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

**Static Analysis**
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

**Code Review / Manual Analysis**
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

**Gas Consumption**
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

**Tools and Platforms used for Audit**
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

# Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

| Risk-level | Description |
| --- | --- |
| High | A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment. |
| Medium | The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed. |
| Low | Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future. |
| Informational | These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact. |

## Number of issues per severity

| Type | High | Medium | Low | Informational |
| --- | --- | --- | --- | --- |
| Open | 0 | 0 | 0 | 0 |
| Acknowledged | 0 | 1 | 0 | 0 |
| Closed | 1 | 0 | 2 | 4 |

# Introduction

During the period of **December 18, 2021 to December 23, 2021** - QuillAudits Team performed a security audit for **Bollycoin** smart contracts.

The code for the audit was taken from following the official link:

| V | Date | Commit ID | Files |
|---|---|---|---|
| 1 | 18 Dec 2021 | https://rinkeby.etherscan.io/address/0x913131F748441D3a3BcF3A03EDb01CA7a1790B1f#code | BollyStake.sol |
| 2 | 23 Dec 2021 | https://rinkeby.etherscan.io/address/0x1B9Debe284b61019E64B34989706409 3181F6B1E#code | BollyStake.sol |

# Issues Found

## A. Contract – BollyStake

### High severity issues

1. No access control on removeStakeholder & addStakeholder function

| Line | Code |
| --- | --- |
| 809-817 | function removeStakeholder(address _stakeholder)<br>public<br>{<br>(bool _isStakeholder, uint256 s) = isStakeholder(_stakeholder);<br>if(_isStakeholder){<br>stakeholders[s] = stakeholders[stakeholders.length - 1];<br>stakeholders.pop();<br>}<br>} |
| 798-803 | function addStakeholder(address _stakeholder)<br>public<br>{<br>(bool _isStakeholder, ) = isStakeholder(_stakeholder);<br>if(!_isStakeholder) stakeholders.push(_stakeholder);<br>} |

**Description**

There is no access control on the removeStakeholder & addStakeholder function. So users can remove other users from the stakeholders array. This means at the time of reward distribution any user can front-run and remove others from the stakeholders array and claim 100% of the rewards.

Someone can also use these functions to fill the stakeholder array to a length such that transactions adding/removing more stakeholder will exceed the block gas limit. Such a scenario is very rare, but still it is possible to do so.

**Remediation**
Use access control for removeStakeholder & addStakeholder function.
Or change these functions' visibility to internal/private.

**Status:** Fixed
In version 2, the functions were made private.

## Medium severity issues

### 2. Reward distribution is controlled by the owner

| Line | Code |
| --- | --- |
| 895-911 | function distributeRewards(uint amount) public {<br>uint256 allowance = usdt.allowance(msg.sender, address(this));<br>require(amount > 0, "Nothing to distribute");<br>require(msg.sender == owner, "Caller is not authorised");<br>require(allowance >= amount, "Check the USDT allowance");<br>usdt.transferFrom(owner,address(this),amount);<br>for (uint256 s = 0; s < stakeholders.length; s += 1){<br>address stakeholder = stakeholders[s];<br>uint256 stakeof = stakeOf(stakeholder);<br>uint256 totalstakes = total_eligible_Stakes();<br>if(users[stakeholder].expire > block.timestamp) {<br>uint256 reward = (stakeof.mul(amount)).div(totalstakes);<br>// Transfer the usdt<br>usdt.transfer(stakeholder, reward);<br>}<br>}<br>} |

**Description**
The reward distribution is controlled by the owner. This means that the owner might not give out the rewards according to the agreement, or the amount might not be correct.

**Remediation**
We advise the client to handle the owner account carefully. We also recommend the client to consider the following solutions:
- Timelock with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

**Status:** Acknowledged by the Auditee

Comment by Client: Reward distribution is controlled by the owner - currently kept as it is. Later ownership can be transferred to governance/DAO contract.

## Low severity issues

### 3. Missing zero address validation

| Line | Code |
| --- | --- |
| 759-764 | constructor(IERC20usdt _usdt,IERC20 _BOLLY, address _owner) {<br>require(address(_BOLLY) != address(0), "_BOLLY is a zero address");<br>BOLLY = _BOLLY;<br>owner = _owner;<br>usdt = _usdt;<br>} |

**Description**
The parameters in the constructor should be checked for zero address. Otherwise there may be issues like the owner will never get back the ownership of the contract and will not be able to perform the owner restricted actions.

**Remediation**
Use a require statement to check for zero address in the constructor.

**Status:** Fixed
In version 2, all parameters were checked for zero address.

## 4. User’s locked_amount is not decreased when removing stake

| Line | Code |
| --- | --- |
| 887-893 | function remove_stake(uint256 _share) public {<br>require( (users[msg.sender].expire < block.timestamp) && (_share <= users[msg.sender].locked_amount) ,"Please wait 365 days until removing stake");<br>_burn(msg.sender, _share);<br>stakes[msg.sender] = stakes[msg.sender].sub(_share);<br>if(stakes[msg.sender] == 0) removeStakeholder(msg.sender);<br>BOLLY.transfer(msg.sender, _share);<br>} |

**Description**

The user’s locked_amount should be decreased when removing the stake. But in the remove_stake() function, the locked_amount value of the user is not changed.

It does not have any serious security issues but should be fixed as leaves the following require statement(one line #888) useless :

**_share <= users[msg.sender].locked_amount**

**Remediation**

Subtract the share amount from the user's locked_amount.

**Status:** Fixed

In version 2,_share amount is subtracted from the locked_amount.

## Informational issues

### 5. Multiple pragma statements

| Line | Code |
| --- | --- |
| 10 | pragma solidity ^0.8.0; |
| 240 | pragma solidity ^0.8.0; |
| 267 | pragma solidity ^0.8.0; |
| 352 | pragma solidity ^0.8.0; |
| 382 | pragma solidity ^0.8.0; |
| 739 | pragma solidity ^0.8.7; |

**Description**
There are multiple pragma statements in the code. Keeping only one pragma statement helps in maintaining readability of the code.

**Remediation**
Keep a single pragma statement.

**Status:** Fixed
In version 2, only one pragma statement is present.

### 6. Missing Events for Significant Transactions

**Description**
The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the function:
- set_owner
- addStakeholder
- removeStakeholder
- enter_stake
- relock_stake
- remove_stake

**Remediation**
We recommend emitting an event for the above-mentioned transactions.

**Status:** Fixed
In version 2, events were emitted for the above-mentioned functions.

### 7. No need to check for allowance

| Line | Code |
| --- | --- |
| 864-877 | function enter_stake(uint256 _amount) public {<br>// Gets the amount of BOLLY locked in the contract<br>uint256 allowance = BOLLY.allowance(msg.sender, address(this));<br>require(_amount >= 100000000000000000000,"minimum 10000 BOLLY needs to be staked");<br>require(allowance >= _amount, "Check the Bolly allowance");<br>if(stakes[msg.sender] == 0) addStakeholder(msg.sender);<br>stakes[msg.sender] = stakes[msg.sender].add(_amount);<br>// Lock the BOLLY in the contract<br>locked storage userInfo = users[msg.sender];<br>userInfo.expire = block.timestamp + _TIMELOCK;<br>userInfo.locked_amount = userInfo.locked_amount + _amount;<br>BOLLY.transferFrom(msg.sender, address(this), _amount);<br>_mint(msg.sender, _amount);<br>} |

**Description**
There is no need to check for the allowance separately as the transferFrom function call will handle that. Remove this check can help with some gas savings as well as low code size.

**Remediation**
Remove the allowance check.

**Status:** Fixed
In version 2, allowance checks were removed.

8. **Public function that could be declared external**

**Description**
The following public functions that are never called by the contract should be declared external to save gas:
- set_owner()
- totalStakes()
- enter_stake()
- relock_stake()
- remove_stake()
- distributeRewards()

**Remediation**
Use the external attribute for functions never called from the contract.

**Status:** Fixed
In version 2, the functions above were declared as external.

## Functional Test Cases

- Should be able to transfer/transferFrom **PASS**
- Can enter the staking with the specified amount **PASS**
- Can remove the staked amount after 365 days have passed **PASS**
- Cannot remove the stake before the expiry **PASS**
- Can relock the stake **PASS**
- Owner should be able to distribute the rewards **PASS**

# Automated Tests

## Slither

```
Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884) ignores return value by BOLLY.transferFrom(msg.sender,address(this),
_amount) (contracts\BollyStake.sol#881)
Bollystake.remove_stake(uint256) (contracts\BollyStake.sol#895-903) ignores return value by BOLLY.transfer(msg.sender,_share) (contract
s\BollyStake.sol#901)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

IERC20usdt (contracts\BollyStake.sol#744-748) has incorrect ERC20 function interface:IERC20usdt.transfer(address,uint256) (contracts\Bo
llyStake.sol#745)
IERC20usdt (contracts\BollyStake.sol#744-748) has incorrect ERC20 function interface:IERC20usdt.transferFrom(address,address,uint256) (
contracts\BollyStake.sol#746)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

Bollystake.enter_stake(uint256).allowance (contracts\BollyStake.sol#871) shadows:
        - ERC20.allowance(address,address) (contracts\BollyStake.sol#498-500) (function)
        - IERC20.allowance(address,address) (contracts\BollyStake.sol#299) (function)
Bollystake.distributeRewards(uint256).allowance (contracts\BollyStake.sol#906) shadows:
        - ERC20.allowance(address,address) (contracts\BollyStake.sol#498-500) (function)
        - IERC20.allowance(address,address) (contracts\BollyStake.sol#299) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Bollystake.constructor(IERC20usdt,IERC20,address)._owner (contracts\BollyStake.sol#759) lacks a zero-check on :
                - owner = _owner (contracts\BollyStake.sol#763)
Bollystake.set_owner(address)._owner (contracts\BollyStake.sol#774) lacks a zero-check on :
                - owner = _owner (contracts\BollyStake.sol#776)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Bollystake.distributeRewards(uint256) (contracts\BollyStake.sol#905-922) has external calls inside a loop: usdt.transfer(stakeholder,re
ward) (contracts\BollyStake.sol#919)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884):
        External calls:
        - BOLLY.transferFrom(msg.sender,address(this),_amount) (contracts\BollyStake.sol#881)
        State variables written after the call(s):
        - _mint(msg.sender,_amount) (contracts\BollyStake.sol#882)
                - _balances[account] += amount (contracts\BollyStake.sol#635)
        - _mint(msg.sender,_amount) (contracts\BollyStake.sol#882)
                - _totalSupply += amount (contracts\BollyStake.sol#634)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884):
        External calls:
        - BOLLY.transferFrom(msg.sender,address(this),_amount) (contracts\BollyStake.sol#881)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (contracts\BollyStake.sol#636)
                - _mint(msg.sender,_amount) (contracts\BollyStake.sol#882)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Bollystake.total_eligible_Stakes() (contracts\BollyStake.sol#854-866) uses timestamp for comparisons
        Dangerous comparisons:
        - users[stakeholders[s]].expire > block.timestamp (contracts\BollyStake.sol#861)
```

```
Bollystake.remove_stake(uint256) (contracts\BollyStake.sol#895-903) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((users[msg.sender].expire < block.timestamp) && (_share <= users[msg.sender].locked_amount),Please wait
365 days until removing stake) (contracts\BollyStake.sol#896)
Bollystake.distributeRewards(uint256) (contracts\BollyStake.sol#905-922) uses timestamp for comparisons
        Dangerous comparisons:
        - users[stakeholder].expire > block.timestamp (contracts\BollyStake.sol#915)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Bollystake.relock_stake() (contracts\BollyStake.sol#885-891) compares to a boolean constant:
        -require(bool,string)(_isStakeholder == true,only current stakeholders can relock stake) (contracts\BollyStake.sol#887)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.7']
        - ^0.8.0 (contracts\BollyStake.sol#10)
        - ^0.8.0 (contracts\BollyStake.sol#240)
        - ^0.8.0 (contracts\BollyStake.sol#267)
        - ^0.8.0 (contracts\BollyStake.sol#352)
        - ^0.8.0 (contracts\BollyStake.sol#382)
        - ^0.8.7 (contracts\BollyStake.sol#739)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (contracts\BollyStake.sol#257-259) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts\BollyStake.sol#197-206) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts\BollyStake.sol#157-159) is never used and should be removed


SafeMath.mod(uint256,uint256,string) (contracts\BollyStake.sol#223-232) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts\BollyStake.sol#174-183) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts\BollyStake.sol#28-34) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts\BollyStake.sol#70-75) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts\BollyStake.sol#82-87) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts\BollyStake.sol#53-63) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts\BollyStake.sol#41-46) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts\BollyStake.sol#10) allows old versions
Pragma version^0.8.0 (contracts\BollyStake.sol#240) allows old versions
Pragma version^0.8.0 (contracts\BollyStake.sol#267) allows old versions
Pragma version^0.8.0 (contracts\BollyStake.sol#352) allows old versions
Pragma version^0.8.0 (contracts\BollyStake.sol#382) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Struct Bollystake.locked (contracts\BollyStake.sol#769-772) is not in CapWords
Function Bollystake.set_owner(address) (contracts\BollyStake.sol#774-778) is not in mixedCase
Parameter Bollystake.set_owner(address)._owner (contracts\BollyStake.sol#774) is not in mixedCase
Parameter Bollystake.isStakeholder(address)._address (contracts\BollyStake.sol#786) is not in mixedCase
Parameter Bollystake.addStakeholder(address)._stakeholder (contracts\BollyStake.sol#801) is not in mixedCase
Parameter Bollystake.removeStakeholder(address)._stakeholder (contracts\BollyStake.sol#813) is not in mixedCase
Parameter Bollystake.stakeOf(address)._stakeholder (contracts\BollyStake.sol#830) is not in mixedCase
Function Bollystake.total_eligible_Stakes() (contracts\BollyStake.sol#854-866) is not in mixedCase
Function Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884) is not in mixedCase
Parameter Bollystake.enter_stake(uint256)._amount (contracts\BollyStake.sol#869) is not in mixedCase
```

```
Function Bollystake.relock_stake() (contracts\BollyStake.sol#885-891) is not in mixedCase
Function Bollystake.remove_stake(uint256) (contracts\BollyStake.sol#895-903) is not in mixedCase
Parameter Bollystake.remove_stake(uint256)._share (contracts\BollyStake.sol#895) is not in mixedCase
Variable Bollystake.BOLLY (contracts\BollyStake.sol#753) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Bollystake.stakeOf(address)._stakeholder (contracts\BollyStake.sol#830) is too similar to Bollystake.stakeholders (contracts\B
ollyStake.sol#779)
Variable Bollystake.addStakeholder(address)._stakeholder (contracts\BollyStake.sol#801) is too similar to Bollystake.stakeholders (cont
racts\BollyStake.sol#779)
Variable Bollystake.removeStakeholder(address)._stakeholder (contracts\BollyStake.sol#813) is too similar to Bollystake.stakeholders (c
ontracts\BollyStake.sol#779)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884) uses literals with too many digits:
        - require(bool,string)(_amount >= 10000000000000000000,minimum 10000 BOLLY needs to be staked) (contracts\BollyStake.sol#872
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

name() should be declared external:
        - ERC20.name() (contracts\BollyStake.sol#439-441)
symbol() should be declared external:
        - ERC20.symbol() (contracts\BollyStake.sol#447-449)
decimals() should be declared external:
        - ERC20.decimals() (contracts\BollyStake.sol#464-466)
totalSupply() should be declared external:
        - ERC20.totalSupply() (contracts\BollyStake.sol#471-473)


        - ERC20.balanceOf(address) (contracts\BollyStake.sol#478-480)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (contracts\BollyStake.sol#490-493)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (contracts\BollyStake.sol#498-500)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (contracts\BollyStake.sol#509-512)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (contracts\BollyStake.sol#527-541)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (contracts\BollyStake.sol#555-558)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (contracts\BollyStake.sol#574-582)
set_owner(address) should be declared external:
        - Bollystake.set_owner(address) (contracts\BollyStake.sol#774-778)
totalStakes() should be declared external:
        - Bollystake.totalStakes() (contracts\BollyStake.sol#842-852)
enter_stake(uint256) should be declared external:
        - Bollystake.enter_stake(uint256) (contracts\BollyStake.sol#869-884)
relock_stake() should be declared external:
        - Bollystake.relock_stake() (contracts\BollyStake.sol#885-891)
remove_stake(uint256) should be declared external:
        - Bollystake.remove_stake(uint256) (contracts\BollyStake.sol#895-903)
distributeRewards(uint256) should be declared external:
        - Bollystake.distributeRewards(uint256) (contracts\BollyStake.sol#905-922)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

## Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

# Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

Several Issues found during the Audit, All Issues have been resolved, and the Auditee has acknowledged one medium-severity issue.

# Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **Bollycoin** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Bollycoin** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

# Audit Report
# December, 2021

For

BollyCoin

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com