ASTROBIATECH
BLOCKCHAIN SECURITY

MADE IN INDIA
BLOCKCHAIN SECURITY

SECURITY ASSESSMENT REPORT

PREPARED FOR

MonieBot Staking

@astrobiatech

Jan
2024

astrobiatech.in

TABLE OF CONTENTS

SCOPE OF AUDIT	1
TECHNIQUES AND METHODS	2
ISSUE CATEGORIES	3
INTRODUCTION	4
OVERVIEW	5
MANUAL ANALYSIS FINDINGS	6
AUTOMATED ANALYSIS	8
SUMMARY	11
DISCLAIMER	12

SCOPE OF AUDIT

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

CHECKED VULNERABILITIES

We have scanned the smart contract for commonly known and morespecific 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 & 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 thesmart contracts.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to testsecurity of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis.Contracts were completely manually analyzed, their logic was checkedand compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

ISSUE CATEGORIES

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

➢ HIGH SEVERITY ISSUES

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 to be fixed before movingto a live environment.

➢ MEDIUM SEVERITY ISSUES

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 SEVERITY ISSUES

Low level severity issues can cause minor impact and or are just warningsthat can remain unfixed for now. It would be better to fix these issues atsome point in the future.

➢ INFORMATIONAL

These are severity four issues which indicate an improvement request, ageneral question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

ISSUES TABLE

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
OPEN	2	-	-	-
ACKNOWLWDGENT	-	-	-	-
CLOSED	-	-	-	-

INTRODUCTION

On 10/01/2024 – Astrobiatech Blockchain Security Team performed security audit for MonieBot Staking smart contract.

CONTRACT NAME	StakingContract
CONTRACT ADDRESS	0x54e9acfe4060893d65659630b659ebf1f3aa523e
BLOCKCHAIN	Ethereum

OVERVIEW

CONTRACT ADDRESS
0x54e9acfe4060893d65659630b659ebf1f3aa523e

CONTRACT NAME
Ethereum

CONTRACT CREATOR
0x154Ea28ea914C84C9d009D2e6BdC66Aa9423e261

OWNER ADDRESS
0x154Ea28ea914C84C9d009D2e6BdC66Aa9423e261

SOURCE CODE
Contract Source Code Verified at Etherscan

OTHER SETTINGS
default evmVersion, MIT license

COMPILER VERSION
v0.8.19+commit.7dd6d404

OPTIMIZATION ENABLED
Yes with 200 runs

Code is truncated to fit the constraints of this document.
https://etherscan.io/address/0x54e9acfe4060893d65659630b659ebf1f3aa523e#code

MANUAL ANALYSIS FINDINGS

HIGH

1. Lock Period variable and function are defined.

Description:-

In the given contract , the lock period variable and function are defined which allows owner to adjust it to any days which give rise to the centralization and potential scam .

Recommendation:-

Remove the Lock Period function and its variable if have to keep zero otherwise define a limit to extend .

2. APR Limit not set

Description:-

In the Given Contract , the limit to minimum APR are not set which allows contract owner to change it to 0 which may rise the risk of potential scam .

Recommendation:-

Set the minimum APR Limit .

.

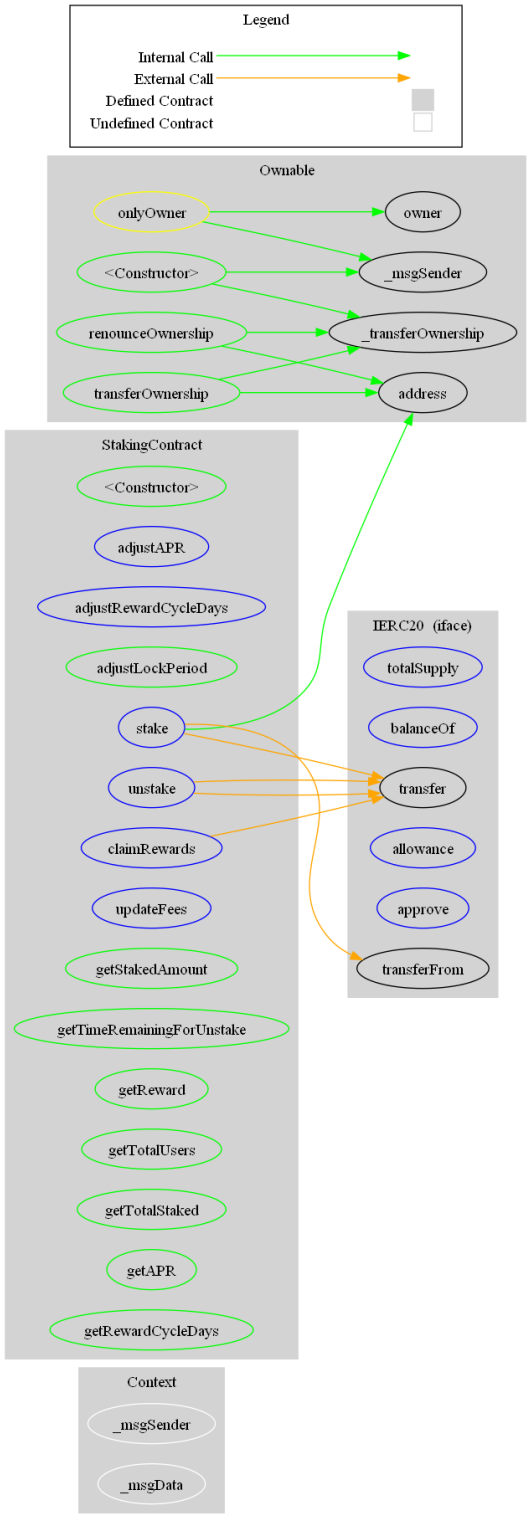
AUTOMATED ANALYSIS

```
INFO:Detectors:
StakingContract.stake(uint256) (token.sol#249-270) ignores return value by
token.transferFrom(msg.sender,address(this),stakedAmountAfterFee) (token.sol#262)
StakingContract.stake(uint256) (token.sol#249-270) ignores return value by token.transfer(feeRecipient,stakingFee) (token.sol#266)
StakingContract.unstake(uint256) (token.sol#272-289) ignores return value by token.transfer(msg.sender,unstakedAmountAfterFee)
(token.sol#283)
StakingContract.unstake(uint256) (token.sol#272-289) ignores return value by token.transfer(feeRecipient,unstakingFee) (token.sol#287)
StakingContract.claimRewards() (token.sol#291-300) ignores return value by token.transfer(msg.sender,reward +
users[msg.sender].accumulutedreward) (token.sol#296)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in StakingContract.claimRewards() (token.sol#291-300):
        External calls:
        - token.transfer(msg.sender,reward + users[msg.sender].accumulutedreward) (token.sol#296)
        State variables written after the call(s):
        - users[msg.sender].accumulutedreward = 0 (token.sol#297)
        StakingContract.users (token.sol#216) can be used in cross function reentrancies:
        - StakingContract.claimRewards() (token.sol#291-300)
        - StakingContract.getReward(address) (token.sol#326-329)
        - StakingContract.getStakedAmount(address) (token.sol#311-313)
        - StakingContract.getTimeRemainingForUnstake(address) (token.sol#315-324)
        - StakingContract.stake(uint256) (token.sol#249-270)
        - StakingContract.unstake(uint256) (token.sol#272-289)
        - StakingContract.users (token.sol#216)
        - users[msg.sender].lastClaimTimestamp = block.timestamp (token.sol#298)
        StakingContract.users (token.sol#216) can be used in cross function reentrancies:
        - StakingContract.claimRewards() (token.sol#291-300)
        - StakingContract.getReward(address) (token.sol#326-329)
        - StakingContract.getStakedAmount(address) (token.sol#311-313)
        - StakingContract.getTimeRemainingForUnstake(address) (token.sol#315-324)
        - StakingContract.stake(uint256) (token.sol#249-270)
        - StakingContract.unstake(uint256) (token.sol#272-289)
        - StakingContract.users (token.sol#216)
Reentrancy in StakingContract.stake(uint256) (token.sol#249-270):
        External calls:
        - token.transferFrom(msg.sender,address(this),stakedAmountAfterFee) (token.sol#262)
        - token.transfer(feeRecipient,stakingFee) (token.sol#266)
        State variables written after the call(s):
        - users[msg.sender].startTime = block.timestamp (token.sol#267)
        StakingContract.users (token.sol#216) can be used in cross function reentrancies:
        - StakingContract.claimRewards() (token.sol#291-300)
        - StakingContract.getReward(address) (token.sol#326-329)
        - StakingContract.getStakedAmount(address) (token.sol#311-313)
        - StakingContract.getTimeRemainingForUnstake(address) (token.sol#315-324)
        - StakingContract.stake(uint256) (token.sol#249-270)
        - StakingContract.unstake(uint256) (token.sol#272-289)
        - StakingContract.users (token.sol#216)
        - users[msg.sender].stakedAmount += _amount (token.sol#268)
        StakingContract.users (token.sol#216) can be used in cross function reentrancies:
        - StakingContract.claimRewards() (token.sol#291-300)
        - StakingContract.getReward(address) (token.sol#326-329)
        - StakingContract.getStakedAmount(address) (token.sol#311-313)
        - StakingContract.getTimeRemainingForUnstake(address) (token.sol#315-324)
        - StakingContract.stake(uint256) (token.sol#249-270)
        - StakingContract.unstake(uint256) (token.sol#272-289)
        - StakingContract.users (token.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

FUNCTIONAL ANALYSIS

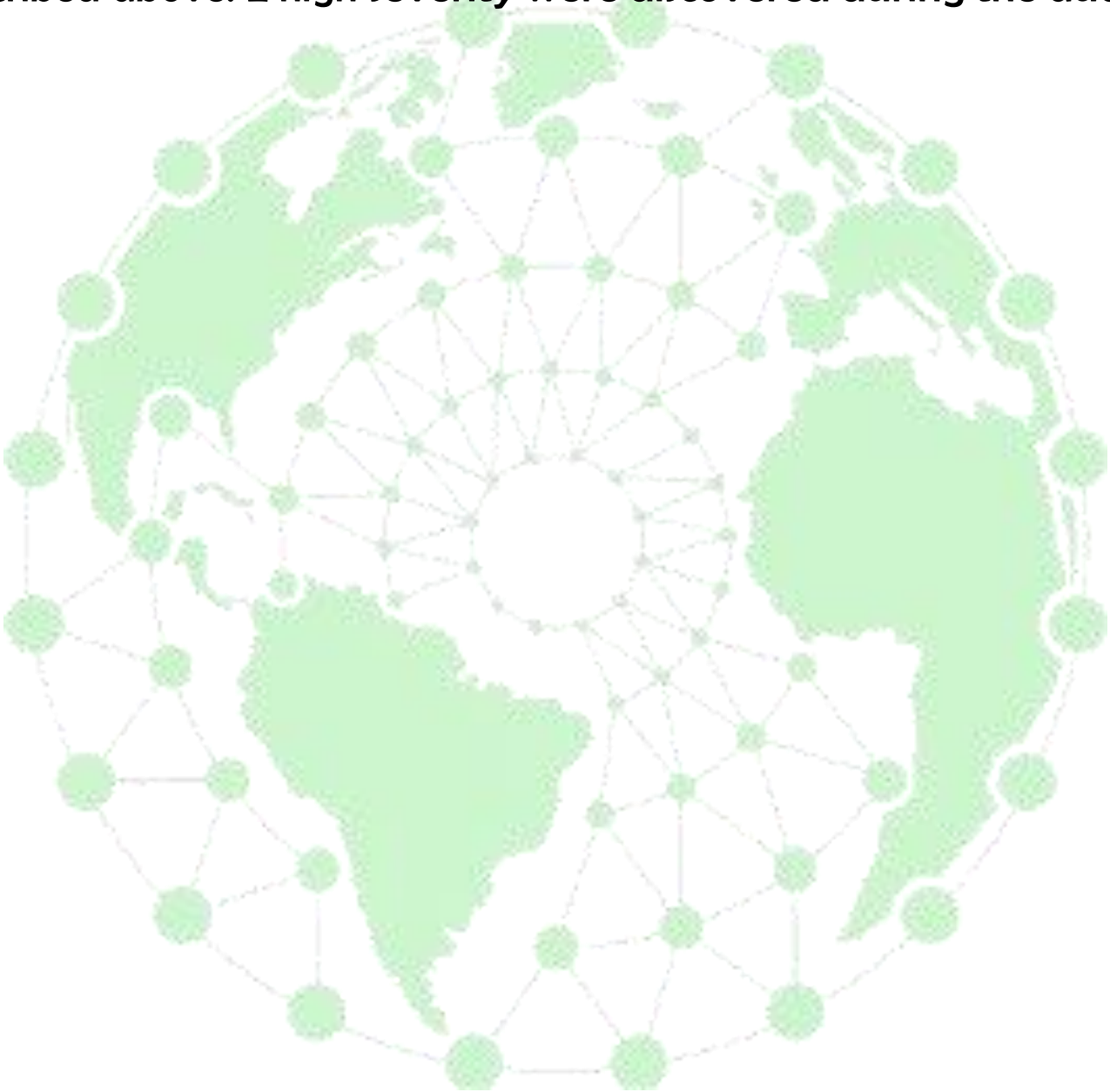
Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public !	🔴	NO !
└	owner	Public !		NO !
└	renounceOwnership	Public !	🔴	onlyOwner
└	transferOwnership	Public !	🔴	onlyOwner
└	_transferOwnership	Internal 🔒	🔴	
IERC20	Interface			
└	totalSupply	External !		NO !
└	balanceOf	External !		NO !
└	transfer	External !	🔴	NO !
└	allowance	External !		NO !
└	approve	External !	🔴	NO !
└	transferFrom	External !	🔴	NO !
StakingContract	Implementation	Ownable		
└	<Constructor>	Public !	🔴	NO !
└	adjustAPR	External !	🔴	onlyOwner
└	adjustRewardCycleDays	External !	🔴	onlyOwner
└	adjustLockPeriod	Public !	🔴	NO !
└	stake	External !	🔴	NO !
└	unstake	External !	🔴	NO !
└	claimRewards	External !	🔴	NO !
└	updateFees	External !	🔴	onlyOwner
└	getStakedAmount	Public !		NO !
└	getTimeRemainingForUnstake	Public !		NO !
└	getReward	Public !		NO !
└	getTotalUsers	Public !		NO !
└	getTotalStaked	Public !		NO !
└	getAPR	Public !		NO !
└	getRewardCycleDays	Public !		NO !

GRAPH TREE


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
address
StakingContract
<Constructor>
adjustAPR
adjustRewardCycleDays
adjustLockPeriod
stake
unstake
claimRewards
updateFees
getStakedAmount
getTimeRemainingForUnstake
getReward
getTotalUsers
getTotalStaked
getAPR
getRewardCycleDays
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

SUMMARY

In this report, we have considered the security of the MonieBot Staking smart contract. We performed our audit according to the procedure described above. 2 high severity were discovered during the audit.



DISCLAIMER

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constituteinvestment advice. No one shall have any right to rely on the report or its contents, and Astrobiatech Blockchain Security and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Astrobiatech BlockchainSecurity) owe no duty of care towards you or any other person, nor does Astrobiatech Blockchain Security make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Astrobiatech Blockchain Security hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effectin relation to the report. Except and only to the extent that it is prohibited by law, Astrobiatech Blockchain Security hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Astrobiatech Blockchain Security, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic lossor damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Proofed by Astrobiatech

ASTROBIATECH
BLOCKCHAIN SECURITY

https://astrobiatech.in

@astrobiatech