Cyberscope

Audit Report

Baltic Miners Staking

May 2022

Type	BEP20
Network	BSC
Address	0x10008DB953Db6bD170A22bA135883a3cCB1bD470
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	BMIStaking
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0x10008DB953Db6bD170A22bA135883a3cCB1bD470

Source Files

Filename	**SHA256**
contract.sol	b137615bc3d28fb39da651da676175ac0ec8738db952eb616681fc1aa2545411

Audit Updates

Initial Audit	21st June 2022
Corrected	

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	STC	Succeeded Transfer Check
	CO	Code Optimization
	DSRC	DSRC
	BA	Blacklist Addresses
	TAM	Token Address Mutation
	FO	Fees Overflow
	OCTD	Owner Contract Tokens Drain
	L01	Public Function could be Declared External
	L04	Conformance to Solidity Naming Conventions
	L09	Dead Code Elimination

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L1273,1280,1297,1308

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(BMIToken).transfer(owner, fee);
//
IERC20(BMIToken).transfer(msg.sender, outAmount);
```

Recommendation

The contract should check if the result of the transfer methods is successful.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L1221

Description

The contract declares “storage” variables that are used in a “view” method.

```
 function pendingReward(uint256 _pid, address account) public view returns
(uint256) {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfos[_pid][account];
    if (user.lastChangeTime == 0){
        return 0;
    }
    uint256 endTime = (block.timestamp>user.lockEndTime)? user.lockEndTime :
block.timestamp;
    uint256 deltaTime = (endTime > user.lastChangeTime)?
endTime.sub(user.lastChangeTime) : 0;

    uint256 rewardVal = deltaTime.mul(user.stakedAmount).mul(pool.apr).div(365
days);
    return rewardVal.div(100).add(user.rewardDebt);
  }
```

Recommendation

Rewrite some code segments so the runtime will be more performant.

DSRC - Data Structure Range Check

Criticality	minor
Location	contract.sol

Description

The contract could check as a precondition if the pid index is inside the pool range. This could be implemented in all the methods that accepts the pid as argument.

```
  function stakeToken(uint256 _pid, uint256 _amount, address _ref) external
nonReentrant {
    require(!blacklist[msg.sender], "blacklist");
    require(_amount > 0, "invalid deposit amount");
    ...
  }
```

Recommendation

The contract should properly check the variables according to the required specifications.

BA - Blacklist Addresses

Criticality	critical
Location	contract.sol#L1235,1260,1288

Description

The contract owner has the authority to blacklist addresses that have staked tokens. The owner may take advantage of it by calling the ‘setBlackList’ method. As a result the investors will not be able to either claim their rewards or withdraw their deposits.

```
  function claim(uint256 _pid) external nonReentrant {
    require(!blacklist[msg.sender], "blacklist");
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

TAM - Token Address Mutation

Criticality	minor
Location	contract.sol#L1204

Description

The contract owner has the authority to change the address of the staked token. The owner may take advantage of it by calling the ‘resetToken’. As a result the users will not be able to claim their rewards or withdraw their deposits since the balance of the new token address will be different from the initial.

```
function resetToken(address _token) public onlyOwner{
    BMIToken = _token;
    totalStakedAmount = 0;
}
```

Recommendation

The contract could prevent the staked token address mutation after the first initialization.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

FO - Fees Overflow

Criticality	critical
Location	contract.sol

Description

The properties earlyWithdrawFee, earlyClaimFee and refFee are used to calculate percentages values. The contract owner has the authority to set any value to these variables. If the owner sets a value higher than 100, the contract will overflow and revert the transactions.

```
  function setReferralFees(uint256 _fee) public onlyOwner{
      refFee = _fee;
  }

  function setEarlyClaimFee(uint256 _fee) external onlyOwner {
      earlyClaimFee = _fee;
  }

  function setEarlyWithdrawFee(uint256 _fee) external onlyOwner {
      earlyWithdrawFee = _fee;
  }
```

```
uint256 fee = outAmount.mul(earlyWithdrawFee).div(100);
outAmount = outAmount.sub(fee);
//
fee = outAmount.mul(earlyClaimFee).div(100);
outAmount = outAmount.sub(fee);
//
fee = outAmount.mul(refFee).div(100);
...
outAmount = outAmount.sub(fee);
```

Recommendation

The contract should embody a check for not allowing values greater than 100.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user

from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

OCTD - Owner Contract Tokens Drain

Criticality	medium
Location	contract.sol#L1349

Description

The contract owner has the authority to claim the balance that has accumulated to the staking contract from the depositors. The owner may take advantage of it by:

- Call the ‘resetToken’ method with an address different from the token.
- Call the ‘withDrawUnknownToken’ method with the token address.

```
function withDrawUnknownToken(address _token, address _address) external
onlyOwner{
    require(_token != BMIToken, "staking token can't be withdrawn");
    uint256 balance = IERC20(_token).balanceOf(address(this));
    IERC20(_token).transfer(_address, balance);
}
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L91,99,793,801,818,825,832,844,867,889,912,932,1177,1181,1185,1189,1193,1204

Description

Public functions that are never called by the contract should be declared external to save gas.

```
resetToken
set
setReferralFees
getReceivedReferalRewards
getPendingReferalRewards
getReferalCounts
decreaseAllowance
increaseAllowance
transferFrom
...
```

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L71,1189,1193,1204,1209,1221,1234,1259,1287,1315,1322,1329,1333,1337,1341,1345,1349,1140

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
BMIToken
_address
_token
_bool
_fee
_pid
_amount
_ref
_claimLockPeriod
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L486,496,515,529,575,585,548,558,437,461,602,1011,988

Description

Functions that are not used in the contract, and make the code's size bigger.

```
_mint
_burn
verifyCallResult
sendValue
isContract
functionStaticCall
functionDelegateCall
functionCallWithValue
functionCall
...
```

Recommendation

Remove unused functions.

Contract Functions

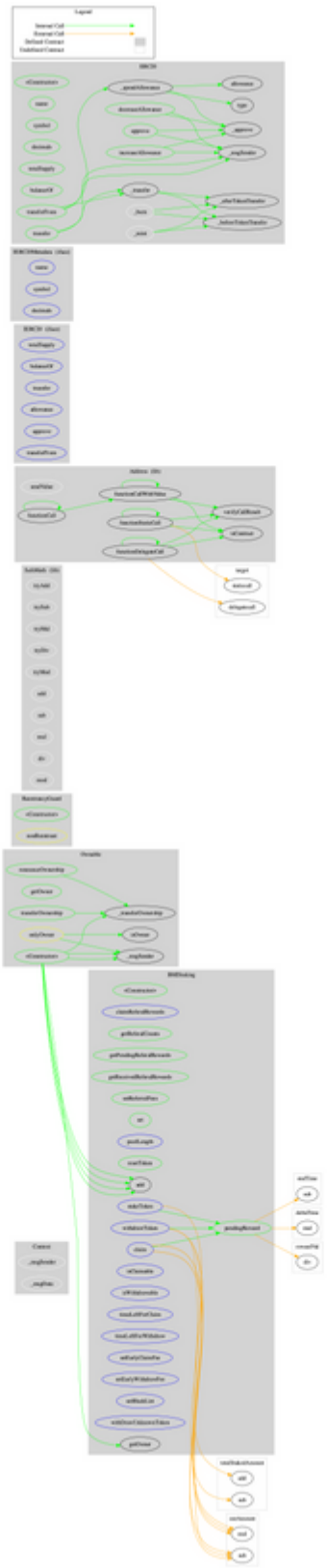
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	getOwner	Public		-
	isOwner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		

	mod	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-

	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
BMIStaking	Implementation	Ownable, ReentrancyGuard		
	<Constructor>	Public	✓	-
	claimReferalRewards	External	✓	-
	getReferalCounts	Public		-
	getPendingReferalRewards	Public		-
	getReceivedReferalRewards	Public		-
	setReferralFees	Public	✓	onlyOwner
	set	Public	✓	onlyOwner
	poolLength	External		-
	resetToken	Public	✓	onlyOwner
	add	Public	✓	onlyOwner
	pendingReward	Public		-
	stakeToken	External	✓	nonReentrant
	withdrawToken	External	✓	nonReentrant
	claim	External	✓	nonReentrant
	isClaimable	External		-
	isWithdrawable	External		-
	timeLeftForClaim	External		-

	timeLeftForWithdraw	External		-
	setEarlyClaimFee	External	✓	onlyOwner
	setEarlyWithdrawFee	External	✓	onlyOwner
	setBlackList	External	✓	onlyOwner
	withDrawUnknownToken	External	✓	onlyOwner

Contract Flow



Summary

Baltic Miners Staking implements a staking contract where users can deposit money and claim rewards proportional to the staking period. This audit mentions potential vulnerabilities, security concerns, business logic suggestions and potential optimizations.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io