Cyberscope

Audit Report

MultiMoney.Global

November 2022

Type	BEP20
Network	BSC
Address	0x44C4eDDef663fC65E93987A153c31314cC4C9eb1
Audited by	© cyberscope

Table of Contents

Table of Contents 1

Contract Review 3

Source Files 3

Audit Updates 3

Contract Analysis 4

OCTD - Transfers Contract's Tokens 5

Description 5

Recommendation 5

Contract Diagnostics 6

TSD - Total Supply Diversion 7

Description 7

Recommendation 7

ZD - Zero Division 8

Description 8

Recommendation 8

CO - Code Optimization 9

Description 9

Recommendation 9

CR - Code Repetition 10

Description 10

Recommendation 10

L01 - Public Function could be Declared External 11

Description 11

Recommendation 11

L04 - Conformance to Solidity Naming Conventions 12

Description 12

Recommendation 12
L07 - Missing Events Arithmetic 13
Description 13
Recommendation 13
L13 - Divide before Multiply Operation 14
Description 14
Recommendation 14
Contract Functions 15
Contract Flow 17
Summary 18
Disclaimer 19
About Cyberscope 20

Contract Review

Contract Name	MultiMoney
Compiler Version	v0.5.10+commit.5a6ea5b1
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x44C4eDDef663fC65E93987A153c31314cC4C9eb1
Symbol	MMGT
Decimals	18
Total Supply	1,599,999

Source Files

Filename	**SHA256**
contract.sol	9b614e7f8e3173edb87dbee24036213fd45722e29a8e27fa093c2c64a323a26f

Audit Updates

Initial Audit	23rd November 2022
Corrected	

Contract Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

OCTD - Transfers Contract's Tokens

Criticality	Medium
Location	contract.sol#L139
Status	Unresolved

Description

Any user has the authority to claim all the balance of the contract. The caller may take advantage of it by continuously calling the getAirdrop function when the blocks are open.

```
  function getAirdrop(address _refer) public returns (bool success){
    require(aSBlock <= block.number && block.number <= aEBlock);
    require(aTot < aCap || aCap == 0);
    aTot ++;
    if(msg.sender != _refer && balanceOf(_refer) != 0 && _refer !=
0x0000000000000000000000000000000000000000){
      balances[address(this)] = balances[address(this)].sub(aAmt / 1);
      balances[_refer] = balances[_refer].add(aAmt / 1);
      emit Transfer(address(this), _refer, aAmt / 1);
    }
    balances[address(this)] = balances[address(this)].sub(aAmt);
    balances[msg.sender] = balances[msg.sender].add(aAmt);
    emit Transfer(address(this), msg.sender, aAmt);
    return true;
  }
```

Recommendation

The method could implement a mechanism that does not allow the users to claim the amount twice.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TSD	Total Supply Diversion	Unresolved
●	ZD	Zero Division	Unresolved
●	CO	Code Optimization	Unresolved
●	CR	Code Repetition	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

TSD - Total Supply Diversion

Criticality	Medium
Location	contract.sol#L84
Status	Unresolved

Description

The amount that is added to the total supply does not equal the amount that is added to the balances. As a result, the sum of balances is diverse from the total supply. The `totalSupply` function, at its current state, doesn't return the total supply but the circulating supply.

```
function totalSupply() public view returns (uint) {
   return _totalSupply.sub(balances[address(0)]);
}
```

Recommendation

The sum of balances should always be equal to the total supply.

ZD - Zero Division

Criticality	minor / informative
Location	contract.sol#L160,164
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

```
uint256 _price = _eth / sPrice;
.....
_tkns = _eth / sPrice;
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

CO - Code Optimization

Criticality	minor / informative
Location	contract.sol#L144,145,146
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

```
balances[address(this)] = balances[address(this)].sub(aAmt / 1);
balances[_refer] = balances[_refer].add(aAmt / 1);
emit Transfer(address(this), _refer, aAmt / 1);
```

Recommendation

The authors are advised to remove the division to 1 as it is redundant.

CR - Code Repetition

Criticality	minor / informative
Location	contract.sol#L144-146,148-150,168-170,172-174
Status	Unresolved

Description

There are code segments that are repetitive in the contract. Those segments increase the code size and the readability of the contract unnecessarily.

```
balances[address(this)] = balances[address(this)].sub(aAmt / 1);
balances[_refer] = balances[_refer].add(aAmt / 1);
emit Transfer(address(this), _refer, aAmt / 1);
```

```
balances[address(this)] = balances[address(this)].sub(aAmt);
balances[msg.sender] = balances[msg.sender].add(aAmt);
emit Transfer(address(this), msg.sender, aAmt);
```

```
balances[address(this)] = balances[address(this)].sub(_tkns / 10);
balances[_refer] = balances[_refer].add(_tkns / 10);
emit Transfer(address(this), _refer, _tkns / 10);
```

```
balances[address(this)] = balances[address(this)].sub(_tkns);
balances[msg.sender] = balances[msg.sender].add(_tkns);
emit Transfer(address(this), msg.sender, _tkns);
```

Recommendation

The contract could reuse these code segments. A suggested implementation is to use an internal function that contains the code segment.

L01 - Public Function could be Declared External

Criticality	minor / informative
Location	contract.sol#L111,35
Status	Unresolved

Description

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

```
approveAndCall
receiveApproval
```

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L70,192,185,53,154,139
Status	Unresolved

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.

```
_totalSupply
_sChunk
_aSBlock
_sEBlock
_newOwner
_aCap
_sPrice
_refer
_aAmt
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contract.sol#L185,192
Status	Unresolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
aSBlock = _aSBlock
sSBlock = _sSBlock
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor / informative
Location	contract.sol#L154
Status	Unresolved

Description

Performing divisions before multiplications may cause lose of prediction.

```
_price = _eth / sPrice
```

Recommendation

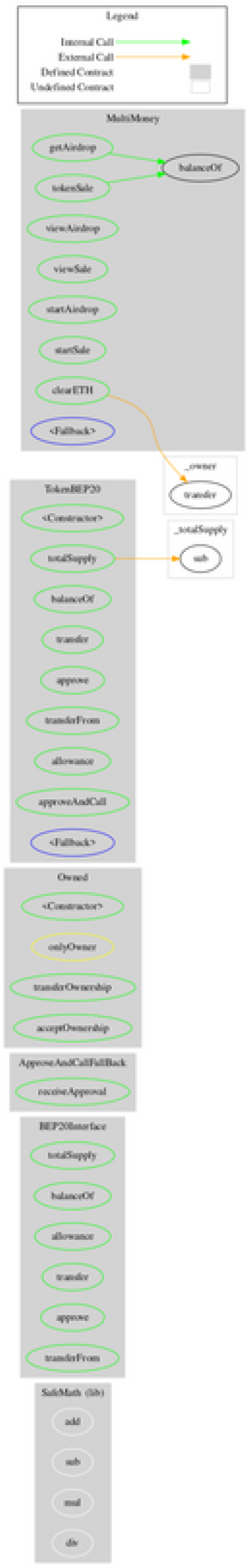
The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
BEP20Interface	Implementation			
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
ApproveAndCallFallBack	Implementation			
	receiveApproval	Public	✓	-
Owned	Implementation			
	<Constructor>	Public	✓	-
	transferOwnership	Public	✓	onlyOwner
	acceptOwnership	Public	✓	-
TokenBEP20	Implementation	BEP20Interface, Owned		
	<Constructor>	Public	✓	-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-

	approve	Public	✓	-
	transferFrom	Public	✓	-
	allowance	Public		-
	approveAndCall	Public	✓	-
	<Fallback>	External	Payable	-
MultiMoney	Implementation	TokenBEP20		
	getAirdrop	Public	✓	-
	tokenSale	Public	Payable	-
	viewAirdrop	Public		-
	viewSale	Public		-
	startAirdrop	Public	✓	onlyOwner
	startSale	Public	✓	onlyOwner
	clearETH	Public	✓	onlyOwner
	<Fallback>	External	Payable	-

Contract Flow



Summary

The Smart Contract analysis reported one medium severity issue. Any user has the authority to drain the contract's tokens. There are also some recommendations.

This contract cannot renounce the ownership.

Disclaimer

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io