Cyberscope

Audit Report

Wuacoin

June 2023

Network	BSC
Address	0x5C7cBEEDE0DFeeEd9373C76F309335aD52213EDF
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MU	Modifiers Usage	Unresolved
	RVR	Redundant Varibale Reassignment	Unresolved
	MEE	Missing Events Emission	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved

Table of Contents

Review

Contract Name	CoinCoin
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	200 runs
Explorer	https://bscscan.com/address/0x5c7cbeede0dfeeed9373c76f309335ad52213edf
Address	0x5c7cbeede0dfeeed9373c76f309335ad52213edf
Network	BSC
Symbol	WUAO
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	01 Jun 2023

Source Files

Filename	SHA256
CoinCoin.sol	e6bdd1f302cf5609792c8e5f7b272e5e3c441e557c8900881bc606d8f03de99b

Findings Breakdown

Critical	2
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	CoinCoin.sol#L75
Status	Unresolved

Description

The contract does not allow users to transfer tokens on behalf of another user, excluding the manager and the whitelisted addresses. Additionally, even if the manager sets a user as whitelisted, the user would still not be able to transfer on behalf of another, because the `approve` function is only accessible by the contract manager.

```
function transferFrom(address from, address to, uint tokens) override public returns
(bool success) {
    require(proxi[msg.sender],"Only actived proxi");
    require(allowed[from][msg.sender]>=tokens,"Spender without balance");
    allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
    balances[from] = safeSub(balances[from], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    emit Transfer(from, to, tokens);
    return true;
}
```

Recommendation

The team should carefully manage the private keys of the manager's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	CoinCoin.sol#L89
Status	Unresolved

Description

The contract, by default, stops addresses from transactions that require allowance. The contract manager has the authority to whitelist addresses by calling the `setStatusProxi` function.

```
function setStatusProxi(address _proxi, bool _status) public {
    require(msg.sender==manager,"Set status only Manager");
    proxi[_proxi]=_status;
}
```

Recommendation

The team should carefully manage the private keys of the manager's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

MU - Modifiers Usage

Criticality	Minor / Informative
Location	CoinCoin.sol#L69,90
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
require (manager == msg.sender,"Only manager");
require(msg.sender==manager,"Set status only Manager");
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

RVR - Redundant Varibale Reassignment

Criticality	Minor / Informative
Location	CoinCoin.sol#L49
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.

The contract assigns the same value to the variable `balances[msg.sender]` more than once in the constructor. As are result, the second assignment is redundant.

```
balances[msg.sender] = _totalSupply;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	CoinCoin.sol#L89
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function setStatusProxi(address _proxi, bool _status) public {
    require(msg.sender==manager,"Set status only Manager");
    proxi[_proxi]=_status;
}
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	CoinCoin.sol#L91
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.

The contract modifies the state of certain variables without checking if their current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
proxi[_proxi]=_status;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	CoinCoin.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	CoinCoin.sol#L46,50
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_totalSupply
manager
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	CoinCoin.sol#L41,89
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint public _totalSupply
address _proxi
bool _status
```

Recommendation

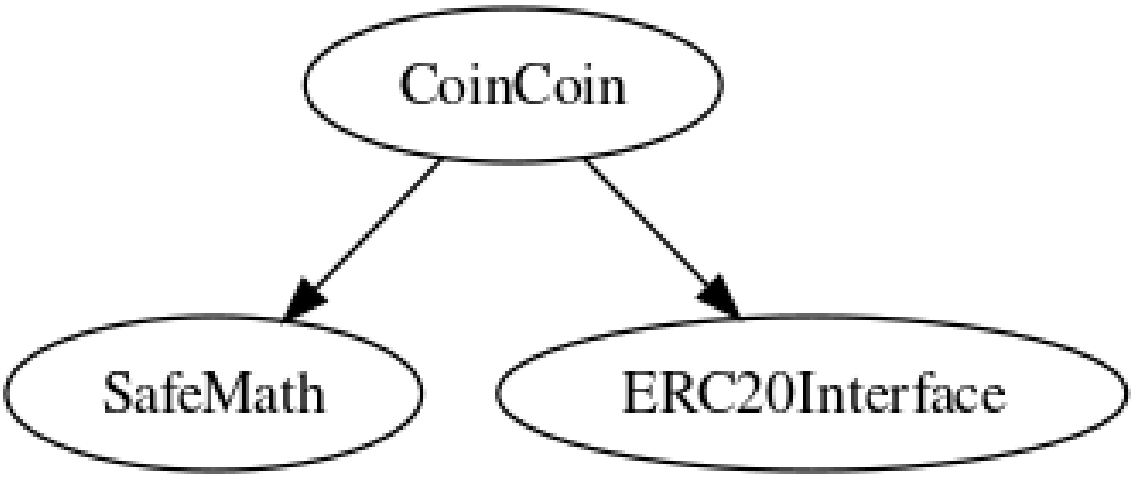
By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

Functions Analysis

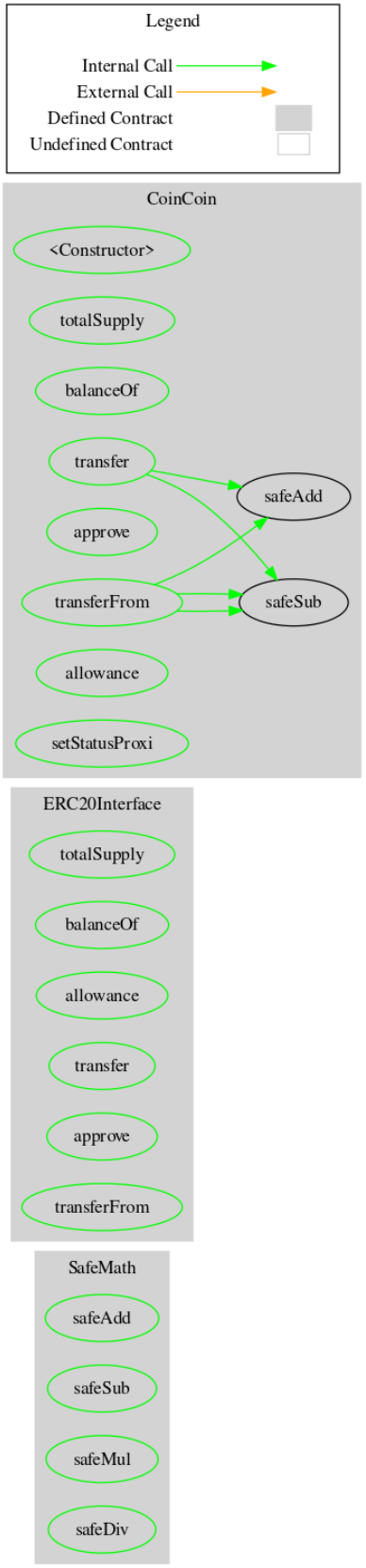
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Implementation			
	safeAdd	Public		-
	safeSub	Public		-
	safeMul	Public		-
	safeDiv	Public		-
ERC20Interface	Implementation			
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
CoinCoin	Implementation	ERC20Interface, SafeMath		
		Public	✓	-
	totalSupply	Public		-
	balanceOf	Public		-

	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	allowance	Public		-
	setStatusProxi	Public	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
CoinCoin
<Constructor>
totalSupply
balanceOf
transfer
safeAdd
approve
transferFrom
safeSub
allowance
setStatusProxi
ERC20Interface
totalSupply
balanceOf
allowance
transfer
approve
transferFrom
SafeMath
safeAdd
safeSub
safeMul
safeDiv

Summary

Wuacoin contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the manager like stop transactions and massively blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's 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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 represents an extensive assessment 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

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