Cyberscope

Audit Report

Kai ECoin

February 2023

Network	BSC
Address	0x0f10c9e30818cB77d0992D734203F3294027d4A6
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 3
Audit Updates 3
Source Files 3
Introduction 4
Roles 4
Diagnostics 5
DIO - Depositor Information Overwrite 6
Description 6
Recommendation 6
PTAI - Potential Transfer Amount Inconsistency 7
Description 7
Recommendation 7
TCDT - Transfers Contract's Deposited Tokens 9
Description 9
Recommendation 9
MC - Missing Check 10
Description 10
Recommendation 10
RSML - Redundant SafeMath Library 11
Description 11
Recommendation 11
L04 - Conformance to Solidity Naming Conventions 12
Description 12
Recommendation 12
L07 - Missing Events Arithmetic 14
Description 14
Recommendation 14
L08 - Tautology or Contradiction 15
Description 15
Recommendation 15
L19 - Stable Compiler Version 16

Description 16
Recommendation 16
L20 - Succeeded Transfer Check 17
Description 17
Recommendation 17
Functions Analysis 18
Inheritance Graph 21
Flow Graph 22
Summary 23
Disclaimer 24
About Cyberscope 25

Review

Contract Name	Crowdsale
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x0f10c9e30818cb77d0992d734203f3294027d4a6
Address	0x0f10c9e30818cb77d0992d734203f3294027d4a6
Network	BSC

Audit Updates

Initial Audit	08 Feb 2023

Source Files

Filename	SHA256
Crowdsale.sol	6085f310d7b417863d866c376e943539f30c42ff0c12639496a706935b5381e5

Introduction

This smart contract implements a token deposit functionality for two ERC-20 tokens. The deposit process can only occur within a specified deadline.

Roles

The contract Roles consist of the `owner` role. The owner has the authority to configure contract parameters and withdraw tokens from the contract.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DIO	Depositor Information Overwrite	Unresolved
●	PTAI	Potential Transfer Amount Inconsistency	Unresolved
●	TCDT	Transfers Contract's Deposited Tokens	Unresolved
●	MC	Missing Check	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

DIO - Depositor Information Overwrite

Criticality	Minor / Informative
Location	Crowdsale.sol#L794
Status	Unresolved

Description

The depositor is stored in an informational struct. If the same user deposits tokens more than once, the struct is overwritten and only the latest deposit is tracked.

```
function DepositTokens(IERC20 token, uint256 amount) external   {
    ...
    Deposit memory deposite =  Deposit(token, amount, current_time);
    _deposites[msg.sender] = deposite;
    emit TokensDeposit(msg.sender, token, amount, current_time);
}
```

Recommendation

Since a single user can deposit tokens more than once, then the contract should use a collection in order to store the deposits.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	Crowdsale.sol#L799
Status	Unresolved

Description

The `safeTransfer()` and `safeTransferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
token.safeTransferFrom(msg.sender,address(this), amount);
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts.

Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

```
Actual Transferred Amount = Balance After Transfer - Balance Before Transfer
```

TCDT - Transfers Contract's Deposited Tokens

Criticality	Minor / Informative
Location	Crowdsale.sol#L805
Status	Unresolved

Description

The contract owner has the authority to claim all deposited balance of the contract. The owner may take advantage of it by calling the `WithdrawTokens` function.

```
function WithdrawTokens(IERC20 token, uint256 amount) external
onlyOwner {
    token.transfer(msg.sender, amount);
}
```

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.

MC - Missing Check

Criticality	Minor / Informative
Location	Crowdsale.sol#L809,814
Status	Unresolved

Description

The contract is processing arguments that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

```
function setCrowdSaleStep(uint256 deadline, uint256 rate) external
onlyOwner{
    _deadline = deadline;
    _rate = rate;
}

function setStableCoin(IERC20 usdt, IERC20 busd) external onlyOwner{
    _busd = busd;
    _usdt = usdt;
}
```

Recommendation

The team is advised to properly check the arguments according to the required specifications.

- The variable `_deadline` should be greater than the current timestamp.
- The variable `_rate` should be greater than zero.
- The variables `_busd` and `_usdt` shouldn't be set to zero address.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Crowdsale.sol#L10
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 on 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 unnecessarily the gas consumption.

```
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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Crowdsale.sol#L646,754,755,769,770,790,801
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.

```
function DOMAIN_SEPARATOR() external view returns (bytes32);
IERC20 public _usdt
IERC20 public _busd
uint256 public _deadline
uint256 public _rate

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	Crowdsale.sol#L806
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_deadline = deadline
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L08 - Tautology or Contradiction

Criticality	Minor / Informative
Location	Crowdsale.sol#L792
Status	Unresolved

Description

A tautology is a logical statement that is always true, regardless of the values of its variables. A contradiction is a logical statement that is always false, regardless of the values of its variables.

Using tautologies or contradictions can lead to unintended behavior and can make the code harder to understand and maintain. It is generally considered good practice to avoid tautologies and contradictions in the code.

```
require(amount >= 0, "check the amount")
```

Recommendation

The team is advised to carefully consider the logical conditions is using in the code and ensure that it is well-defined and make sense in the context of the smart contract.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Crowdsale.sol#L2
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Crowdsale.sol#L802
Status	Unresolved

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.

```
token.transfer(msg.sender, amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

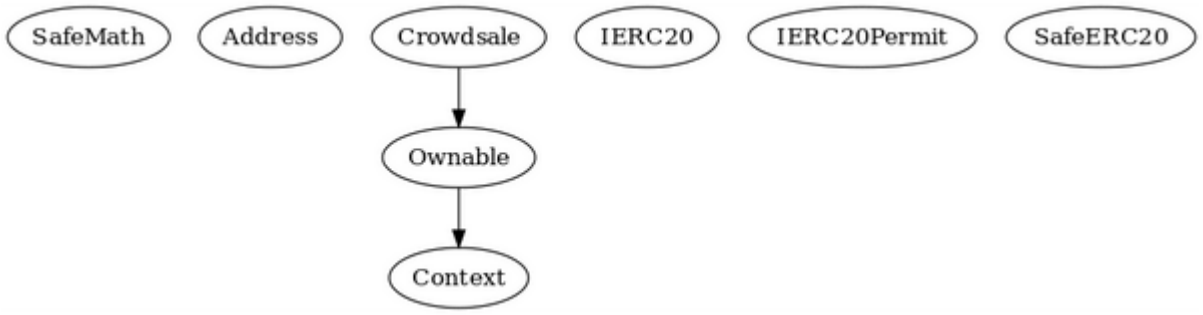
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
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		

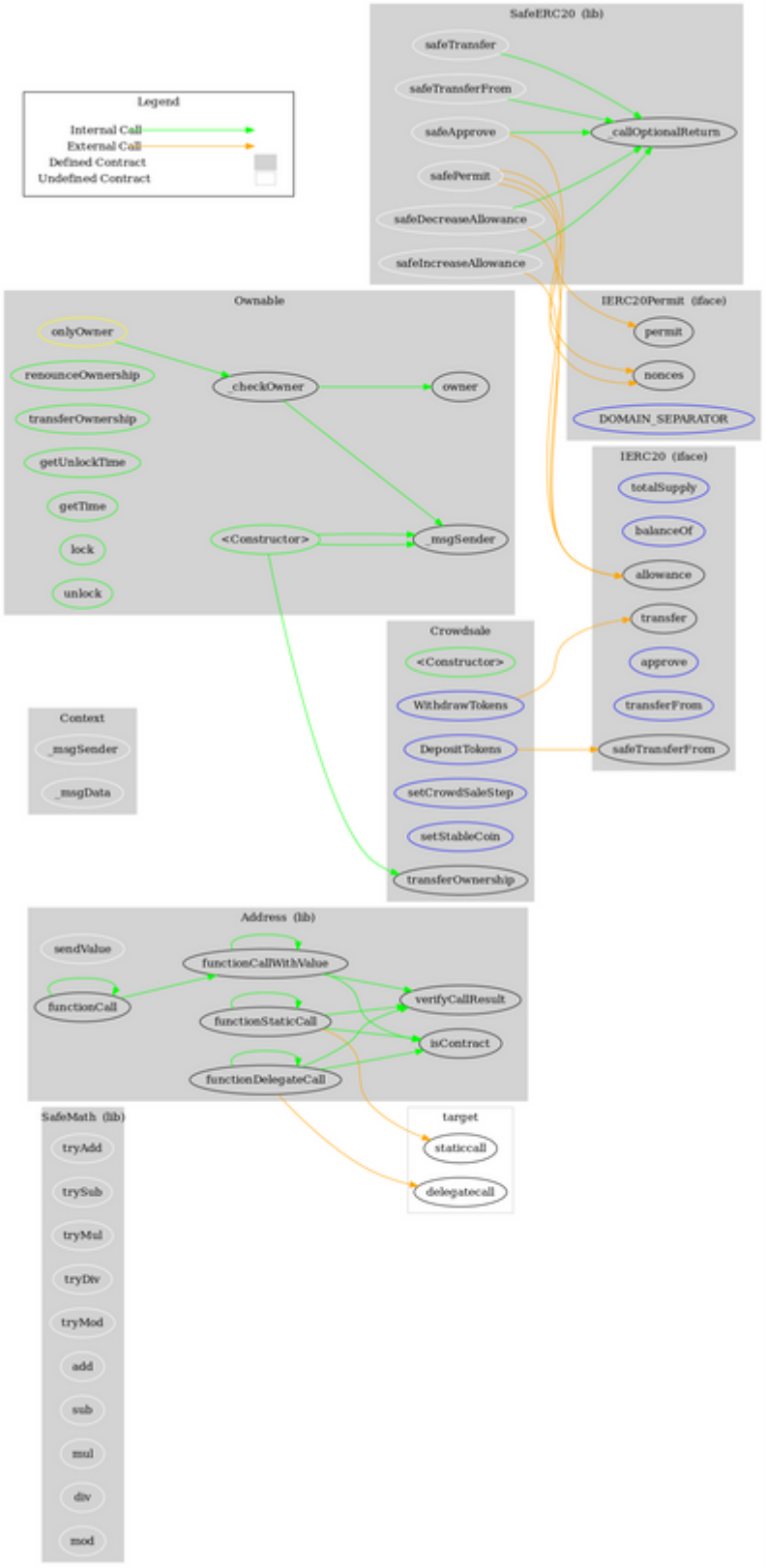
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	getUnlockTime	Public		-
	getTime	Public		-
	lock	Public	✓	onlyOwner
	unlock	Public	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Permit	Interface			
	permit	External	✓	-

	nonces	External		-
	DOMAIN_SEPARATOR	External		-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
Crowdsale	Implementation	Ownable		
		Public	✓	-
	DepositTokens	External	✓	-
	WithdrawTokens	External	✓	onlyOwner
	setCrowdSaleStep	External	✓	onlyOwner
	setStableCoin	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeERC20 (lib)
safeTransfer
safeTransferFrom
safeApprove
safePermit
safeDecreaseAllowance
safeIncreaseAllowance
_callOptionalReturn
Ownable
onlyOwner
renounceOwnership
transferOwnership
getUnlockTime
getTime
lock
unlock
_checkOwner
owner
<Constructor>
_msgSender
IERC20Permit (iface)
permit
nonces
DOMAIN_SEPARATOR
IERC20 (iface)
totalSupply
balanceOf
allowance
transfer
approve
transferFrom
Crowdsale
<Constructor>
WithdrawTokens
DepositTokens
setCrowdSaleStep
setStableCoin
transferOwnership
safeTransferFrom
Context
_msgSender
_msgData
Address (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
verifyCallResult
isContract
target
staticcall
delegatecall
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod

Summary

Kai ECoin contract implements a crowdsale mechanism. This audit investigates security issues, business logic concerns and potential improvements.

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