Cyberscope

Audit Report

LopeCoin

March 2024

Network	BSC
Address	0xa732FBB2EbB4C89452deFC01a4fb8916effB239d
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	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CR	Code Repetition	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	MTE	Misleading Transfer Event	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RVD	Redundant Variable Declaration	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

Table of Contents

Analysis 1
Diagnostics 2
Table of Contents 3
Review 4
Audit Updates 4
Source Files 4
Findings Breakdown 5
ST - Stops Transactions 6
Description 6
Recommendation 7
ELFM - Exceeds Fees Limit 8
Description 8
Recommendation 9
CR - Code Repetition 10
Description 10
Recommendation 11
IDI - Immutable Declaration Improvement 12
Description 12
Recommendation 12
MTE - Misleading Transfer Event 13
Description 13
Recommendation 13
MEE - Missing Events Emission 14
Description 14
Recommendation 14
RVD - Redundant Variable Declaration 15
Description 15
Recommendation 15
L04 - Conformance to Solidity Naming Conventions 16
Description 16
Recommendation 16
Functions Analysis 17
Inheritance Graph 20
Flow Graph 21
Summary 22
Disclaimer 23
About Cyberscope 24

Review

Contract Name	LopeCoin
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://bscscan.com/address/0xa732fbb2ebb4c89452defc01a4fb8916effb239d
Address	0xa732fbb2ebb4c89452defc01a4fb8916effb239d
Network	BSC
Symbol	LOPE
Decimals	18
Total Supply	400,000,000,000,000
Badge Eligibility	Must Fix Criticals

Audit Updates

Initial Audit	17 Mar 2024

Source Files

Filename	SHA256
LopeCoin.sol	abfff7a4b1e34baf40c74b0f832a25d6017bf73305985484b634fe7710cc8dd0

Findings Breakdown

Critical	2
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	LopeCoin.sol#L299
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by setting the sum of charity wallet percentages greater than 100.

```
uint256 totalDonationAmount = 0;
for (uint i = 0; i < charityWallets.length; i++) {
    uint256 donationAmount = amount * charityWallets[i].percentage / 100;
    totalDonationAmount += donationAmount;
    super._transfer(sender, charityWallets[i].wallet, donationAmount);
}

uint256 transferAmount = amount - totalDonationAmount;
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	LopeCoin.sol#L199
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `addMultipleCharityWallets` function with high percentage values.

```
function addMultipleCharityWallets(address[] memory wallets, uint256[]
memory percentages) public onlyOwner {
    require(wallets.length == percentages.length, "Wallets and percentages
array must be of the same length");

    for (uint i = 0; i < wallets.length; i++) {
        require(wallets[i] != address(0), "Invalid charity wallet
address");
        require(percentages[i] > 0 && percentages[i] <= 100, "Invalid
percentage");
        charityWallets.push(CharityWallet(wallets[i], percentages[i]));
    }
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

CR - Code Repetition

Criticality	Minor / Informative
Location	LopeCoin.sol#L268,305
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
if (sender == _owner || sender == liquidityPoolAddress) {
    bool tokensAreLocked = false;


    for (uint i = 0; i < _locks[sender].length; i++) {

        if (_locks[sender][i].amount > 0 && _locks[sender][i].unlockTime >
block.timestamp) {
            tokensAreLocked = true;
            break;
        }
    }

    require(!tokensAreLocked, "Tokens are locked");

    uint256 lockedAmount = 0;
    for (uint i = 0; i < _locks[sender].length; i++) {
        if (_locks[sender][i].unlockTime > block.timestamp) {
            lockedAmount += _locks[sender][i].amount;
        }
    }

    require(balanceOf(sender) - lockedAmount >= amount, "Amount exceeds
available balance");
}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	LopeCoin.sol#L176
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.

```
_owner
```

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.

MTE - Misleading Transfer Event

Criticality	Minor / Informative
Location	LopeCoin.sol#L250
Status	Unresolved

Description

The `lockTokens` function in the contract emits a Transfer event, implying that tokens have been transferred from the sender to the contract. However, no such transfer occurs within the function. This discrepancy between the emitted event and the actual functionality of the function can be misleading to developers and users interacting with the contract.

```
emit Transfer(msg.sender, address(this), amount);
```

Recommendation

To ensure clarity and accuracy in contract events, the team is advised to update the `lockTokens` function to accurately reflect the actions taking place. If tokens are not being transferred to the contract, the team could consider emitting a different event that accurately describes the locking action, such as TokensLocked or similar. By aligning event emissions with actual contract actions, the contract can enhance transparency and facilitate easier understanding for all users.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	LopeCoin.sol#L186,191,196,205,225,233
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.

```
liquidityPoolAddress = _liquidityPoolAddress;
liquidityPoolPercentage = _percentage;
redistributionPercentage = _percentage;
charityWallets.push(CharityWallet(wallets[i], percentages[i]));
charityWallets[index].percentage = newPercentage;
charityWallets.pop();
```

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.

RVD - Redundant Variable Declaration

Criticality	Minor / Informative
Location	LopeCoin.sol#L156,157
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 declares certain variables that are not used in a meaningful way by the contract. As a result, these variables are redundant.

```
uint256 public liquidityPoolPercentage;
uint256 public redistributionPercentage;
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	LopeCoin.sol#L184,189,194
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.

```
address _liquidityPoolAddress
uint256 _percentage
```

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.

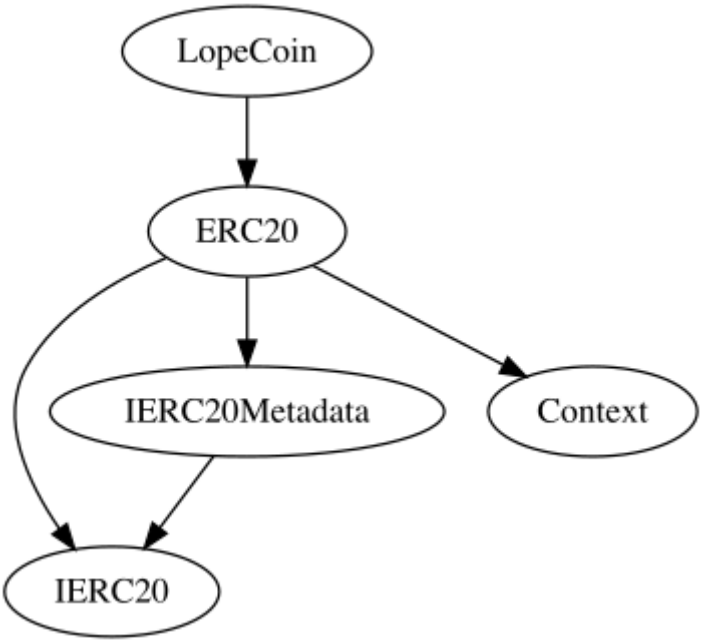
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		

ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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	✓	
LopeCoin	Implementation	ERC20		
		Public	✓	ERC20
	setLiquidityPoolAddress	External	✓	onlyOwner
	setLiquidityPoolPercentage	External	✓	onlyOwner

	setRedistributionPercentage	External	✓	onlyOwner
	addMultipleCharityWallets	Public	✓	onlyOwner
	getCharityWallets	Public		-
	updateCharityWalletPercentage	Public	✓	onlyOwner
	removeCharityWalletByAddress	Public	✓	onlyOwner
	lockTokens	External	✓	-
	getLockedTokens	Public		-
	_transfer	Internal	✓	
	burn	Public	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20
<Constructor>
name
symbol
decimals
totalSupply
transfer
allowance
transferFrom
approve
increaseAllowance
decreaseAllowance
_mint
_transfer
_approve
_msgSender
_burn
balanceOf
LopeCoin
burn
_mint
onlyOwner
setLiquidityPoolAddress
setLiquidityPoolPercentage
setRedistributionPercentage
addMultipleCharityWallets
getCharityWallets
updateCharityWalletPercentage
removeCharityWalletByAddress
<Constructor>
getLockedTokens
_transfer
lockTokens
Context
_msgSender
_msgData
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

LopeCoin 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 owner like stopping transactions and manipulating the fees. 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