Cyberscope

Audit Report

POPO

May 2023

Network	BSC
Address	0x2cFC69C980fcFC41b741cBB1A7FB916913A04a69
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CSD	Circulating Supply Discrepancy	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	TPP	Token Pair Prevalidation	Unresolved
●	RE	Redundant Events	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved

Table of Contents

Review

Contract Name	POPO
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x2cfc69c980fcfc41b741cbb1a7fb916913a04a69
Address	0x2cfc69c980fcfc41b741cbb1a7fb916913a04a69
Network	BSC
Symbol	$POPO
Decimals	18
Total Supply	1.000.000.000

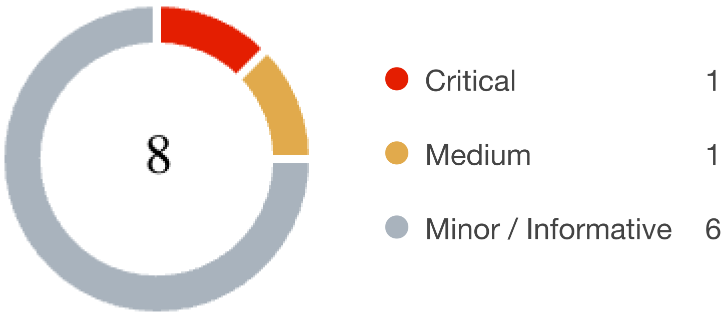
Audit Updates

Initial Audit	29 May 2023

Source Files

Filename	SHA256
POPO.sol	e056cb5c92a04cb95c157d0b2c21379e514bd0e5b06efe28321b4d6ea384bf19

Findings Breakdown



Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	POPO.sol#L328
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the owner address and the liquidity address. The owner can enable the transactions for all users. Once the transactions are enabled, the owner will not be able to disable them again.

```
if (isLimitedAddress(from,to)) {
  require(isTradingEnabled,"Trading is not enabled");
}
```

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. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

CSD - Circulating Supply Discrepancy

Criticality	Medium
Location	POPO.sol#L158
Status	Unresolved

Description

According to the ERC20 specification, the `totalSupply()` function should return the total supply of the token. The total supply should always equal the sum of the balances. The contract does not return the `totalSupply()`. Instead, the function returns the `totalSupply()` minus the amount that has been moved to the dead address. This amount is the circulating supply of the token. Many decentralized applications and tools are calculating many indicators like the circulating supply and market cap based on the `totalSupply()`. As a result, these applications will produce misleading results.

```
function totalSupply() external view override returns (uint256) { if
(_totalSupply == 0) { revert(); } return _totalSupply -
balanceOf(address(0xdead)); }
```

Recommendation

The `totalSupply()` should always equal the sum of the holder's balances. The contract should comply with this convention so that the decentralized applications will

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	POPO.sol#L273
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 updates the `_noFee` mappinng even if its current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function setNoFeeWallet(address account, bool enabled) public
onlyOwner {
    _noFee[account] = enabled;
}
```

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.

TPP - Token Pair Prevalidation

Criticality	Minor / Informative
Location	POPO.sol#L311
Status	Unresolved

Description

The variable `newPair` can be any address. Additionally, the contract does not validate if a token pair exists between the following four addresses. This lack of validation can lead to unintended behavior and potential security vulnerabilities.

```
function changeLpPair(address newPair) external onlyOwner {
    isLpPair[newPair] = true;
    emit _changePair(newPair);
}
```

Recommendation

It is recommended to perform a prevalidation check on the contract addresses used for swapping, to ensure a smooth transaction flow within the contract. This validation should confirm that the addresses have valid pair address values associated with them.

RE - Redundant Events

Criticality	Minor / Informative
Location	POPO.sol#L212,214,215
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 `_changeThreshold` , `_changeFees` , and `SwapAndLiquify` events are not utilized in the contact's implementation. Hence, they are redundant.

```
event _changeThreshold(uint256 newThreshold);
event _changeFees(uint256 buy, uint256 sell);
event SwapAndLiquify();
```

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. It is recommended to remove redundant events.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	POPO.sol#L219,235
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
swapRouter
lpPair
```

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	POPO.sol#L75,174,175,176,188,189,191,204,205,206,207,208,209,210,286,291,296,426
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.

```
on WETH() external pure returns (address);

6 constant private buyfee = 10;

...

 constant private _symbol = "$POPO";

constant private _decimals = 18;

...
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	POPO.sol#L296
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
on is_transfer(address ins, address out) internal view returns
(bool) {
        bool _is_transfer = !isLpPair[out] && !isLpPair[ins];
        return _is_transfer;
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

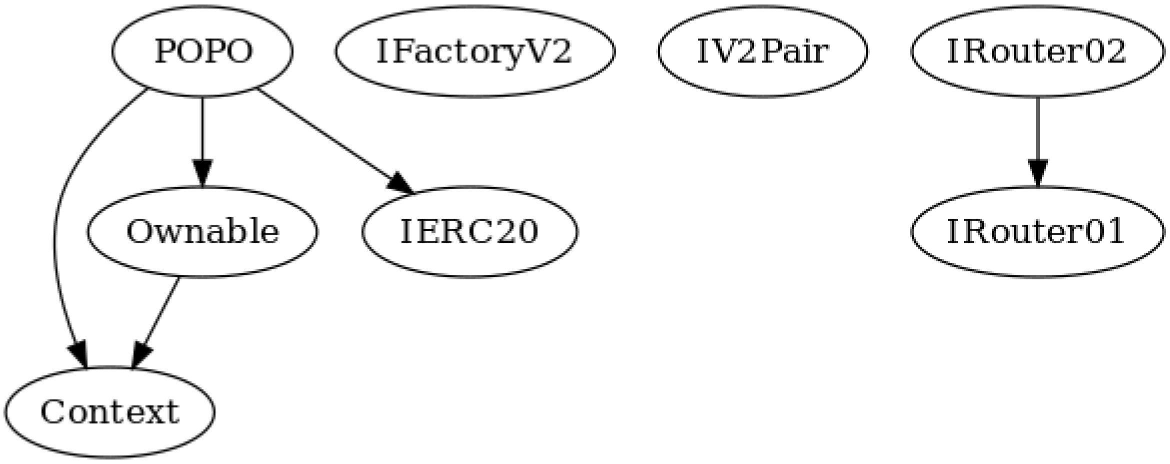
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-

	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-

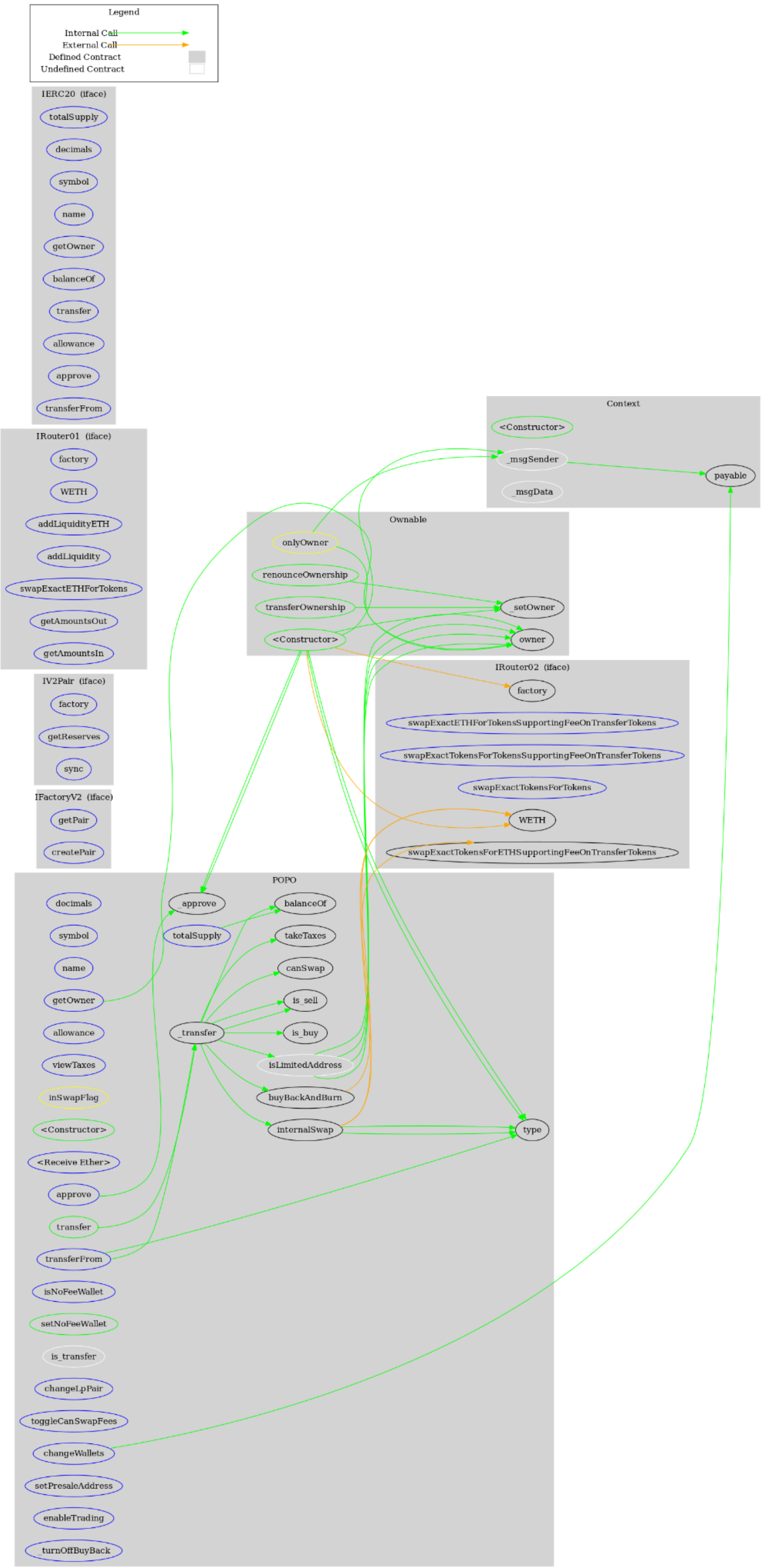
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
POPO	Implementation	Context, Ownable, IERC20		
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-
	viewTaxes	External		-
		Public	✓	-
		External	Payable	-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	transferFrom	External	✓	-
	isNoFeeWallet	External		-

	setNoFeeWallet	Public	✓	onlyOwner
	isLimitedAddress	Internal		
	is_buy	Internal		
	is_sell	Internal		
	is_transfer	Internal		
	canSwap	Internal		
	changeLpPair	External	✓	onlyOwner
	toggleCanSwapFees	External	✓	onlyOwner
	_transfer	Internal	✓	
	changeWallets	External	✓	onlyOwner
	takeTaxes	Internal	✓	
	buyBackAndBurn	Internal	✓	
	internalSwap	Internal	✓	inSwapFlag
	setPresaleAddress	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	_turnOffBuyBack	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
IRouter01 (iface)
factory
WETH
addLiquidityETH
addLiquidity
swapExactETHForTokens
getAmountsOut
getAmountsIn
IV2Pair (iface)
factory
getReserves
sync
IFactoryV2 (iface)
getPair
createPair
Context
<Constructor>
_msgSender
_msgData
payable
Ownable
onlyOwner
renounceOwnership
transferOwnership
<Constructor>
_setOwner
owner
IRouter02 (iface)
factory
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokens
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
POPO
decimals
symbol
name
getOwner
allowance
viewTaxes
inSwapFlag
<Constructor>
<Receive Ether>
approve
transfer
transferFrom
isNoFeeWallet
setNoFeeWallet
is_transfer
changeLpPair
toggleCanSwapFees
changeWallets
setPresaleAddress
enableTrading
_turnOffBuyBack
_approve
totalSupply
_transfer
balanceOf
takeTaxes
canSwap
is_sell
is_buy
isLimitedAddress
buyBackAndBurn
internalSwap
type

Summary

POPO 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. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. There is also a limit of max 1% fee.

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