Cyberscope

Audit Report

Phoenix Finance

April 2022

Type	BEP20
Network	BSC
Address	0x3b981e78cb270C4F5e35bA3D1faEfB222f6433A9
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	PHF
Compiler Version	v0.7.4+commit.3f05b770
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0x3b981e78cb270C4F5e35bA3D1faEfB222f6433A9
Symbol	PHF
Decimals	5
Total Supply	200,000
Domain	phoenix.finance

Source Files

Filename	**SHA256**
contract.sol	3a7425311b679598f0c4e7ab6e0d8188994da1011aa82af3a63728a0b75067ac

Audit Updates

Initial Audit	4th April 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L944

Description

The contract owner has the authority to stop contracts from transactions. The owner may take advantage of it by calling the `setBotBlacklist` function.

```
require(!blacklist[sender] && !blacklist[recipient], "in_blacklist");
```

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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	MTS	Manipulate Total Supply
	CO	Code Optimization
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L05	Unused State Variable
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination
	L13	Divide before Multiply Operation

MTS - Manipulate Total Supply

Criticality	minor
Location	contract.sol#L890

Description

The total supply increased proportional to the time that has elapsed since the contract creation. This change will have a direct impact on the token price and Market Cap. This is a common feature in smart contracts called “rebase”.

```
for (uint256 i = 0; i < times; i++) {
    _totalSupply = _totalSupply
        .mul((10**RATE_DECIMALS).add(rebaseRate))
        .div(10**RATE_DECIMALS);
}
```

CO - Code Optimization

Criticality	minor
Location	contract.sol#L507

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.

```
if (deltaTimeFromInit <= (100 days)) {
    rebaseRate = 828
;
} else if (deltaTimeFromInit <= (100 days)) {
    rebaseRate = 609;
} else if (deltaTimeFromInit <= (200 days)) {
    rebaseRate = 392;
} else {
    rebaseRate = 45;
}
```

Recommendation

The second “if” statement will never be reached hence can be removed. If the contract wants to take account more cases then the logic should be rewritten.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L702,715,720,746,750,754,1279

Description

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

```
getLiquidityBacking
decimals
symbol
name
transferOwnership
renounceOwnership
owner
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L509,788,789,783,786,779,781,782,799,780

Description

Constant state variables should be declared constant to save gas.

```
treasuryFee
swapEnabled
sellFee
safuDividendFee
liquidityFee
feeDenominator
autofirePitFee
ZERO
DEAD
...
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L158,160,191,237,534,535,484,492,1152,1161 and 19 more

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
_lastAddLiquidityTime
_lastRebasedTime
_initRebaseStartTime
_autoAddLiquidity
_autoRebase
ZERO
DEAD
_isFeeExempt
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L7

Description

There are segments that contain unused state variables.

```
MAX_INT256
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L533

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.

```
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L35

Description

Functions that are not used in the contract, and make the code's size bigger.

```
abs
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L871,989,1279

Description

Performing divisions before multiplications may cause lose of prediction.

```
liquidityBalance = _gonBalances[pair].div(_gonsPerFragment)
_gonBalances[autoLiquidityReceiver] =
_gonBalances[autoLiquidityReceiver].add(gonAmount.div(feeDenominator).mul(liquid
ityFee))
_gonBalances[address(this)] =
_gonBalances[address(this)].add(gonAmount.div(feeDenominator).mul(_treasuryFee.a
dd(safuDividendFee)))
_gonBalances[autofirePit] =
_gonBalances[autofirePit].add(gonAmount.div(feeDenominator).mul(autofirePitFee))
feeAmount = gonAmount.div(feeDenominator).mul(_totalFee)
times = deltaTime.div(300)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	transfer	External	✓	-
	approve	External	✓	-
	transferFrom	External	✓	-
IPancakeSwap Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-

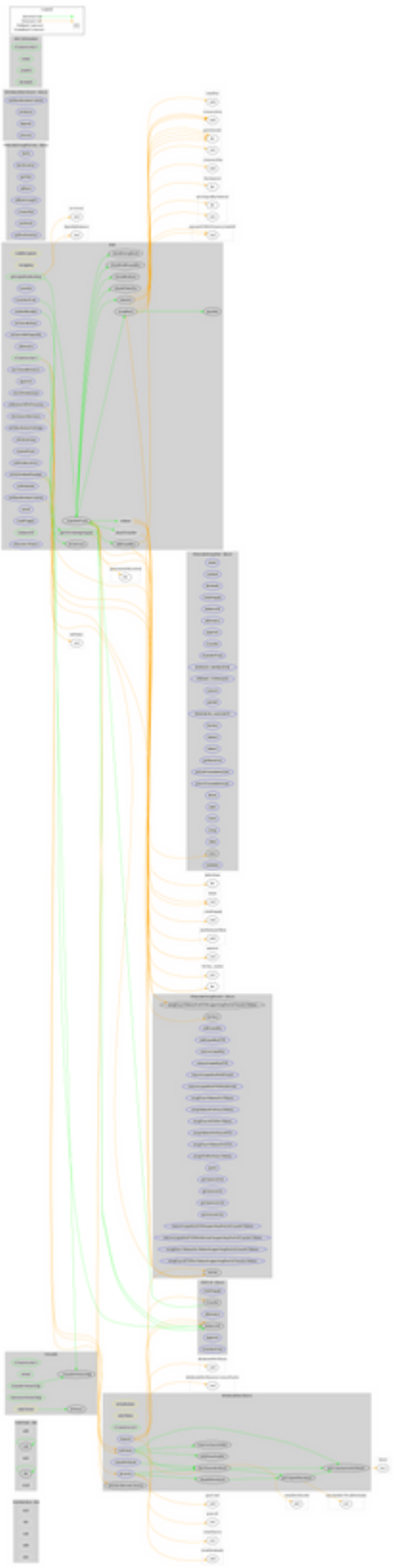
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IPancakeSwap Router	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-

	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IPancakeSwapFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-

DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	isOwner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20Detailed	Implementation	IERC20		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
PHF	Implementation	ERC20Detailed, Ownable		
	<Constructor>	Public	✓	ERC20Detailed Ownable
	rebase	Internal	✓	

	transfer	External	✓	validRecipient
	transferFrom	External	✓	validRecipient
	_basicTransfer	Internal	✓	
	_transferFrom	Internal	✓	
	takeFee	Internal	✓	
	addLiquidity	Internal	✓	swapping
	swapBack	Internal	✓	swapping
	withdrawAllToTreasury	External	✓	swapping onlyOwner
	shouldTakeFee	Internal		
	shouldRebase	Internal		
	shouldAddLiquidity	Internal		
	shouldSwapBack	Internal		
	setAutoRebase	External	✓	onlyOwner
	setAutoAddLiquidity	External	✓	onlyOwner
	allowance	External		-
	decreaseAllowance	External	✓	-
	increaseAllowance	External	✓	-
	approve	External	✓	-
	checkFeeExempt	External		-
	setIsDividendExempt	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	isNotInSwap	External		-
	manualSync	External	✓	-
	setFeeReceivers	External	✓	onlyOwner
	getLiquidityBacking	Public		-
	setWhitelist	External	✓	onlyOwner
	setBotBlacklist	External	✓	onlyOwner
	setLP	External	✓	onlyOwner
	totalSupply	External		-
	balanceOf	Public		-
	isContract	Internal		
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	phoenix.finance
Registry Domain ID	94df594a791c4fadb82a922648de942e-DONUTS
Creation Date	2020-08-21T15:50:19Z
Updated Date	2021-08-15T08:58:58Z
Registry Expiry Date	2022-08-21T15:50:19Z
Registrar WHOIS Server	whois.tucows.com
Registrar URL	http://www.tucows.com
Registrar	Tucows Domains Inc.
Registrar IANA ID	69

The domain has been created over 1 year before the creation of the audit. It will expire in 5 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The Smart Contract analysis reported one medium risk level issue. The contract owner can blacklist contracts. Apart from this the contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. The contract raises the total supply and the corresponding holdings proportionally to the time that has elapsed. The taxes are fixed to 15% for buys and 17% to sales and can't be changed. 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

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io