Cyberscope

Audit Report

Pepperbird

April 2022

Github https://github.com/pepperbird/PepperbirdContracts

Commit 974c3f394418a0fa9d000269779af10bec212db8

Audited by © cyberscope

Table of Contents

Contract Review

Github	https://github.com/pepperbird/PepperbirdContracts
Commit	974c3f394418a0fa9d000269779af10bec212db8
Domain	pepperbird.finance

Source Files

Filename	**SHA256**
contract.sol	1699712f9d6484a792ba532500b4c9f8b9a08d3b7682a211a0ab992ca843226e

Audit Updates

Initial Audit	18th 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

ST - Stop Transactions

Criticality	medium
Location	contract.sol#L1363

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it by setting the `maxWalletToken` to zero.

```
if ((heldTokens + amount) > maxWalletToken) {
    revert WalletLimitReached({ walletBalance: heldTokens,
proposedWalletBalance: (heldTokens + amount), walletMaxBalance: maxWalletToken
});
}
```

Recommendation

The contract could embody a check for not allowing setting the maxWalletToken less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

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
	CO	Code Optimization
	CR	Code Repetition
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L11	Unnecessary Boolean equality

CO - Code Optimization

Criticality	minor
Location	contract.sol#L777

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 the following loop, once the state turns to false, then it will never change again.

```
function customReflectionsExist(address[] memory _reflectionAddresses) internal
view returns (bool) {
    bool state = true;
    uint256 arrayLength = _reflectionAddresses.length;
    for (uint256 i = 0; i < arrayLength; i++) {
        if (!distributorsMapping[_reflectionAddresses[i]].exists) {
            state = false;
        }
    }

    return state;
}
```

Recommendation

The function could return false at the first time that the expression fulfils.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L806

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

For instance, in the following segment the customReflectionMapping loop is executing the same logic independent of the branch.

```
if (!customReflectionMapping[_owner].exists) {
    customReflectionArrayOfKeys.push(_owner);
    if (customReflectionArrayOfKeys.length != 0) {
        customReflectionMapping[_owner].index =
customReflectionArrayOfKeys.length - 1;
    } else {
        customReflectionMapping[_owner].index = 0;
    }
    customReflectionMapping[_owner].exists = true;
    for (uint256 i = 0; i < arrayLength; i++) {

customReflectionMapping[_owner].reflection_tokens.push(_reflectionAddresses[i]);
    }
} else {
    for (uint256 i = 0; i < arrayLength; i++) {

customReflectionMapping[_owner].reflection_tokens.push(_reflectionAddresses[i]);
    }
}
```

Recommendation

The customReflectionMapping could be removed from both branches and implemented once at the end.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L308,315,875,894,951,955,959,967,1190,1258,1321,1578,1637

Description

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

```
getDefaultReflections
setFees
transferOwnership
getPairContract
getUnpaidEarnings
isCustomReflectionActive
getMaxUserReflections
getTotalDistributers
getDistributor
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L570,576,1040

Description

Constant state variables should be declared constant to save gas.

```
timeToClearNewOwnershipAddress
pancakeSwapV2Router
dividendsPerShareAccuracyFactor
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L347,601,547,555,557,737,744,777,789,798,825,830,831,832,863,867,871,875,879,898,951,963,976,977,978,734,735,1161,1162,1163,1173,1178,1182,1186,1190,1218,1226,1230,1312,1536,1537,1538,1539,1579,1580,1581,1582,1583,1584,1585,1612,1613,1614,1615,1623,1628,1633,1642,1643,1644,996,1074

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.

```
DOMAIN_SEPARATOR
_decimals
_minDistribution
_minPeriod
_BEP_TOKEN
_defaultReflectionAddresses
_denominator
_target
_amount
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L601,963,1535,1549,1623,1628

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.

```
targetLiquidity = _target
swapThreshold = _amount
buybackMultiplierNumerator = numerator
autoBuybackCap = _cap
maxCustomReflections = _maxReflections
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L898,914,1068

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool,string)(buyBacker[msg.sender] == true,Not a buybacker)
useCustomReflection(shareholder) == true
customReflectionsOn == false
customReflectionMapping[_shareholder].exists == false
```

Recommendation

Remove the equality to the boolean constant.

Contract Functions

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		
IERC20Extended	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IUniswapV2Factory	Interface			

	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
Auth	Implementation			
	<Constructor>	Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner
	isOwner	Public		-
	isAuthorized	Public		-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
IUniswapV2Router01	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	-

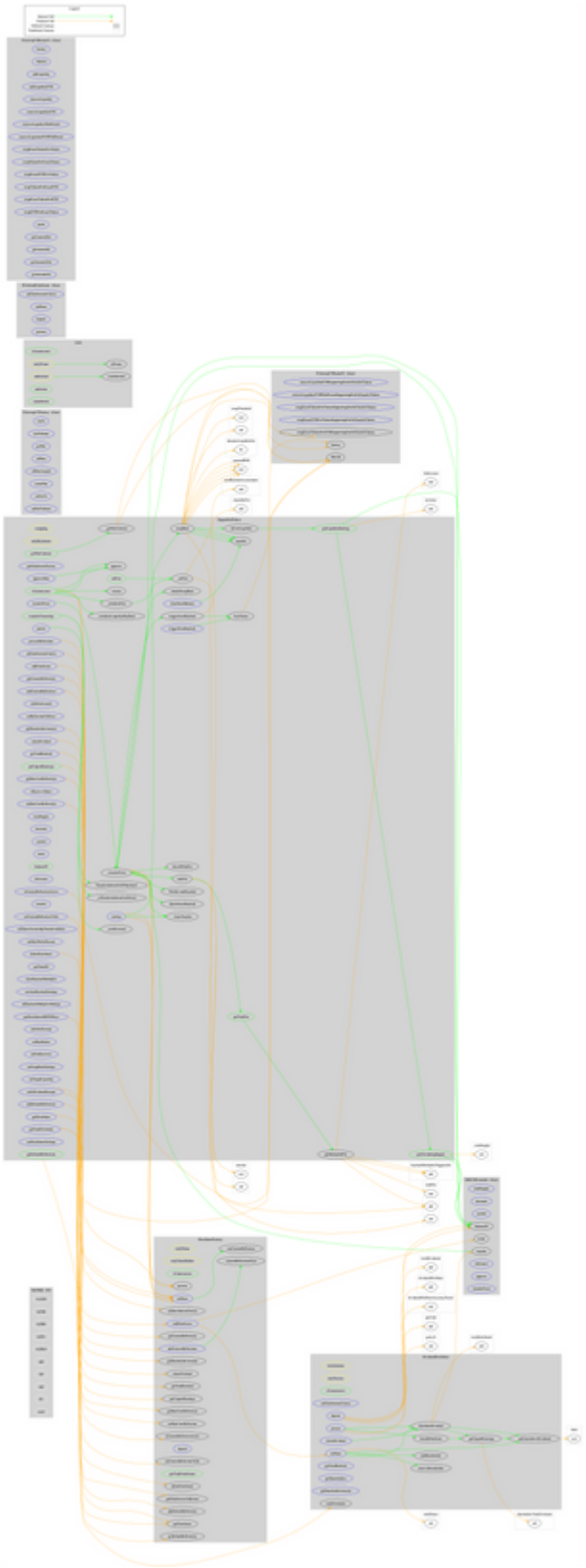
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyFactory
	setShare	External	✓	onlyFactory
	deposit	External	Payable	onlyFactory
	process	External	✓	onlyFactory
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getTotalRealized	External		-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	getShareholders	External		onlyFactory
	getShareholderAmount	External		-

	removeShareholder	Internal	✓	
DistributorFactory	Implementation			
	<Constructor>	Public	✓	-
	customReflectionsExist	Internal		
	addDefaultReflections	External	✓	onlyToken
	getDefaultReflections	External		-
	addCustomReflections	External	✓	-
	getCustomReflections	External		-
	addDistributor	External	✓	onlyToken
	getShareholderAmount	External		-
	claimDividend	External	✓	-
	getTotalRealized	External		-
	getUnpaidEarnings	Public		-
	deleteDistributor	External	✓	onlyToken
	getDistributorsAddresses	Public		-
	useCustomReflection	Internal		
	setShare	External	✓	onlyToken
	process	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	getDistributor	Public		-
	getTotalDistributers	Public		-
	getMaxUserReflections	Public		-
	setMaxUserReflection	External	✓	onlyToken
	isCustomReflectionActive	Public		-
	setCustomReflectionToOn	External	✓	onlyToken
	setDistributionCriteria	External	✓	onlyToken
PepperBirdToken	Implementation	IERC20Extended, Auth		
	<Constructor>	Public	Payable	Auth
	getDistributorFactory	External		-
	addDistributor	External	✓	authorized
	getCustomReflections	External		-
	addCustomReflections	External	✓	-

	getShareholderAmount	External		-
	claimDividend	External	✓	-
	getTotalRealized	External		-
	getUnpaidEarnings	Public		-
	getMaxUserReflections	External		-
	setMaxUserReflections	External	✓	authorized
	isCustomReflectionActive	External		-
	setIsPostLaunch	External	✓	authorized
	setReflectionOnTimer	External	✓	authorized
	setCustomReflectionToOn	External	✓	authorized
	deleteDistributor	External	✓	authorized
	getDistributersBEP20Keys	External		-
	getDistributer	External		-
	getTotalDividends	External		-
	_initializeFees	Internal	✓	
	_initializeLiquidityBuyBack	Internal	✓	
	<Receive Ether>	External	Payable	-
	getPairContract	Public		-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setFutureOwnershipTransferAddress	External	✓	onlyOwner
	setMaxWalletPercent	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_getPairContract	Internal		
	_isTransferAddressConfirmed	Internal		
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	

	shouldTakeFee	Internal		
	clearStuckBalance	External	✓	onlyOwner
	getTotalFee	Public		-
	version	Public		-
	getChainID	External		-
	getMultipliedFee	Public		-
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	swapBack	Internal	✓	swapping
	shouldAutoBuyback	Internal		
	triggerZeusBuyback	External	✓	authorized
	clearBuybackMultiplier	External	✓	authorized
	triggerAutoBuyback	Internal	✓	
	buyTokens	Internal	✓	swapping
	setAutoBuybackSettings	External	✓	authorized
	setBuybackMultiplierSettings	External	✓	authorized
	setIsDividendExempt	External	✓	authorized
	setIsFeeExempt	External	✓	authorized
	setBuyBacker	External	✓	authorized
	setFees	Public	✓	authorized
	_setFees	Internal	✓	
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	addDefaultReflections	External	✓	authorized
	getDefaultReflections	Public		-
	setDistributionCriteria	External	✓	authorized
	processReflections	External	✓	authorized
	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	airdrop	External	✓	onlyOwner
	_setAllowance	Internal	✓	
	permit	External	✓	-

Contract Flow



Domain Info

Domain Name	pepperbird.finance
Registry Domain ID	aaa0b285b7c44207a54e0ed383622a07-DONUTS
Creation Date	2022-02-13T08:50:11Z
Updated Date	2022-02-18T08:50:50Z
Registry Expiry Date	2023-02-13T08:50:11Z
Registrar WHOIS Server	whois.donuts.co
Registrar URL	http://domains.google.com
Registrar	Google Inc.
Registrar IANA ID	895

The domain has been created 2 months before the creation of the audit. It will expire in 10 months.

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

Summary

Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported one issue. The contract owner can stop the transactions. Other than that, the contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% fees.

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