Cyberscope

Audit Report

Potions

April 2022

Address 0e0643e326edf9e8e7c5a3b50e0bea71f79370b384caaae8f4edfce6c77f1c99

Audited by © cyberscope

Table of Contents

Source Files

Filename	SHA256
contract.sol	0e0643e326edf9e8e7c5a3b50e0bea71f79370b384caaae8f4edfce6c77f1c99

Audit Updates

Initial Audit	17th April 2022
Corrected	

Contract Analysis

The contract is based on two contracts. The token and the potion manager. The token transfers are not affected by the potions. The transfers are pure ERC20 implementation.

The portion can be created by the token or the contract owner.

The users can buy potions according to a specific price. The price is in tokens. The users take rewards proportionally to the potions that they own. The tokens from the payments are accumulated to the contract.

The token mainly handles the payments and delegates calls to potionRewardManager

The contract tokens are used in order to send funds to the cauldronPool, distributionPool, auto-generated liquidity pool and user’s wallet.

The contract owner has the authority to manipulate the proportions that each address will receive.

The contract owner has the authority to withdraw all the remaining funds from the contract.

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#L2396

Description

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

```
require(
    !_isBlacklisted[from] && !_isBlacklisted[to],
)
```

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
	CR	Code Repetition
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L05	Unused State Variable
	L06	Missing Events Access Control
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination
	L13	Divide before Multiply Operation
	L14	Uninitialized Variables in Local Scope
	L15	Local Scope Variable Shadowing

CR - Code Repetition

Criticality	minor
Location	contract.sol#L2059,2978,2104,2130

Description

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

```
for (uint256 i = 1; i < potionsCount; i++) {
    _potion = potions[i];

    _creationTimes = string(
        abi.encodePacked(
            _creationTimes,
            separator,
            uint2str(_potion.creationTime)
        )
    );
}
```

Recommendation

Create an internal function that contains the code segment and remove it from all the sections.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L259,263,274,282,973,982,1132,1140,1157,1189,1202,1219,1242,1271,1298,1639,1661,1687,1695,1719,2190,2315,2359,2456,2520,2549,2582,2586,2590,2599,2608,2612,2616,2620,2624,2628,2632,2636,2640,2644,2648,2652,2661,2670,2679,2688,2700,2704,2708

Description

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

```
getTotalCreatedPotions
getTotalStakedReward
publiDistriRewards
distributeRewards
getPotionsLastClaims
getPotionsRewards
getPotionsCreationTime
getPotionsNames
getDistriCount
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L1800,2227

Description

Constant state variables should be declared constant to save gas.

```
deadWallet
lastDistributionCount
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L576,810,812,843,1942,1962,1979,2000,2019,2038,2048,2066,2092,2118,2144,2170,2174,2178,2182,2186,2190,2198,2202,1786,1789,2241

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.

```
_isBlacklisted
PotionPrice
PotionOwners
_distributeRewards
_isPotionOwner
_getPotionNumberOf
_changeGasDistri
_changeAutoDistri
_changeClaimTime
...
```

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#L27

Description

There are segments that contain unused state variables.

```
MAX_INT256
```

Recommendation

Remove unused state variables.

L06 - Missing Events Access Control

Criticality	minor
Location	contract.sol#L1822

Description

Detected missing events for critical access control parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
token = token_
```

Recommendation

Emit an event for critical parameter changes.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L2178,2324,2336,2341,2346,2351,2355

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.

```
rwSwap = value
cashoutFee = value
cauldronFee = value
liquidityPoolFee = value
rewardsFee = value
swapTokensAmount = newVal
claimTime = newTime
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L403,435,515,533,479,498,1376,1917,1481,1515,1499,1462,82,87,12

Description

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

```
toInt256Safe
toUint256Safe
abs
safeTransferFrom
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
_burn
functionStaticCall
...
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L2144,2456

Description

Performing divisions before multiplications may cause lose of prediction.

```
rewardsPoolTokens = contractTokenBalance.mul(rewardsFee).div(100)
temp = (48 + uint8(_i - (_i / 10) * 10))
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contract.sol#L2538,2566,1933

Description

The are variables that are defined in the local scope and are not initialized.

```
validIndex
feeAmount
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contract.sol#L2264,2456

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
name
shares
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMathUint	Library			
	toInt256Safe	Internal		
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
	toUint256Safe	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-

Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
IJoeRouter01	Interface			
	factory	External		-
	WAVAX	External		-
	addLiquidity	External	✓	-
	addLiquidityAVAX	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityAVAX	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityAVAXWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactAVAXForTokens	External	Payable	-
	swapTokensForExactAVAX	External	✓	-
	swapExactTokensForAVAX	External	✓	-
	swapAVAXForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-

IJoeRouter02	Interface	IJoeRouter01		
	removeLiquidityAVAXSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityAVAXWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactAVAXForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForAVAXSupportingFeeOnTransferTokens	External	✓	-
IUniswapV2Pair	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	✓	-
IJoeFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	migrator	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
	setMigrator	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

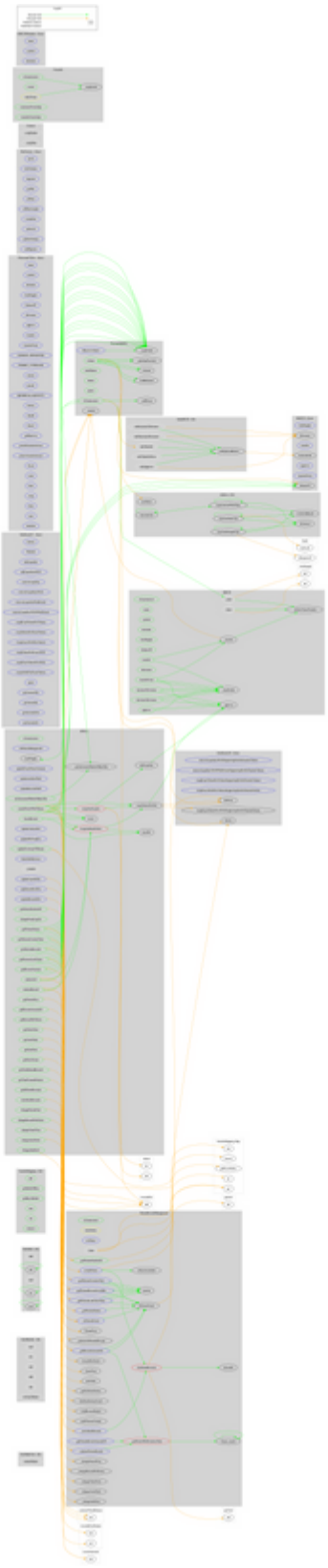
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
PaymentSplitte	Implementation	Context		

r				
	<Constructor>	Public	Payable	-
	<Receive Ether>	External	Payable	-
	totalShares	Public		-
	totalReleased	Public		-
	totalReleased	Public		-
	shares	Public		-
	released	Public		-
	released	Public		-
	payee	Public		-
	release	Public	✓	-
	release	Public	✓	-
	_pendingPayment	Private		
	_addPayee	Private	✓	
PotionReward Management	Implementation			
	<Constructor>	Public	✓	-
	setToken	External	✓	onlySentry
	distributeRewards	Private	✓	
	createPotion	External	✓	onlySentry
	isNameAvailable	Private		
	_burn	Internal	✓	
	_getPotionWithCreationTime	Private		
	binary_search	Private		
	_cashoutPotionReward	External	✓	onlySentry
	_cashoutAllPotionsReward	External	✓	onlySentry
	claimable	Private		
	_getRewardAmountOf	External		-
	_getRewardAmountOf	External		-
	_getPotionRewardAmountOf	External		-
	_getPotionsNames	External		-
	_getPotionsCreationTime	External		-
	_getPotionsRewardAvailable	External		-
	_getPotionsLastClaimTime	External		-
	uint2str	Internal		

	_changePotionPrice	External	✓	onlySentry
	_changeRewardPerPotion	External	✓	onlySentry
	_changeClaimTime	External	✓	onlySentry
	_changeAutoDistri	External	✓	onlySentry
	_changeGasDistri	External	✓	onlySentry
	_getPotionNumberOf	Public		-
	isPotionOwner	Private		
	_isPotionOwner	External		-
	_distributeRewards	External	✓	onlySentry
SPELL	Implementation	ERC20, Ownable, PaymentSplitter		
	<Constructor>	Public	✓	ERC20 PaymentSplitter
	setPotionManagement	External	✓	onlyOwner
	updateUniswapV2Router	Public	✓	onlyOwner
	updateSwapTokensAmount	External	✓	onlyOwner
	updatecauldronWall	External	✓	onlyOwner
	updateRewardsWall	External	✓	onlyOwner
	updateRewardsFee	External	✓	onlyOwner
	updateLiquiditFee	External	✓	onlyOwner
	updatecauldronFee	External	✓	onlyOwner
	updateCashoutFee	External	✓	onlyOwner
	updateRwSwapFee	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	blacklistMalicious	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	_transfer	Internal	✓	
	swapAndSendToFee	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	createPotionWithTokens	Public	✓	-
	cashoutReward	Public	✓	-

	cashoutAll	Public	✓	-
	boostReward	Public	✓	onlyOwner
	changeSwapLiquify	Public	✓	onlyOwner
	getPotionNumberOf	Public		-
	getRewardAmountOf	Public		onlyOwner
	getRewardAmount	Public		-
	changePotionPrice	Public	✓	onlyOwner
	getPotionPrice	Public		-
	changeRewardPerPotion	Public	✓	onlyOwner
	getRewardPerPotion	Public		-
	changeClaimTime	Public	✓	onlyOwner
	getClaimTime	Public		-
	changeAutoDistri	Public	✓	onlyOwner
	getAutoDistri	Public		-
	changeGasDistri	Public	✓	onlyOwner
	getGasDistri	Public		-
	getDistriCount	Public		-
	getPotionsNames	Public		-
	getPotionsCreationTime	Public		-
	getPotionsRewards	Public		-
	getPotionsLastClaims	Public		-
	distributeRewards	Public	✓	onlyOwner
	publiDistriRewards	Public	✓	-
	getTotalStakedReward	Public		-
	getTotalCreatedPotions	Public		-

Contract Flow



Domain Info

Domain Name	potion.financial
Registry Domain ID	8cf1c8e07a2c4af580eff5ea28934d45-DONUTS
Creation Date	2022-02-14T22:41:10Z
Updated Date	2022-02-19T22:41:35Z
Registry Expiry Date	2023-02-14T22:41:10Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

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

Potions is an interesting project that has a friendly and growing community. The contract implements a novel approach that combines an ERC20 token with a dividend tracker related mechanism. This audit focuses on the business logic, security concerns and potential improvements.

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