



Cyberscope

Audit Report

Opulence OPEC

April 2022

Type ERC20

Network AVAX

Address 0x7f03D24d2fC79fECF668485e1c4e4C4B21044B42

Audited by © cyberscope

Table of Contents

Table of Contents	1
Contract Review	3
Source Files	4
Audit Updates	4
Contract Analysis	5
ST - Stop Transactions	6
Description	6
Recommendation	6
ELFM - Exceed Limit Fees Manipulation	7
Description	7
Recommendation	7
MT - Mint Tokens	8
Description	8
Recommendation	8
BC - Blacklisted Contracts	9
Description	9
Recommendation	9
Contract Diagnostics	10
L01 - Public Function could be Declared External	11
Description	11
Recommendation	11
L02 - State Variables could be Declared Constant	12
Description	12
Recommendation	12
L04 - Conformance to Solidity Naming Conventions	13
Description	13

Recommendation	13
L06 - Missing Events Access Control	14
Description	14
Recommendation	14
L09 - Dead Code Elimination	15
Description	15
Recommendation	15
L11 - Unnecessary Boolean equality	16
Description	16
Recommendation	16
L13 - Divide before Multiply Operation	17
Description	17
Recommendation	17
Contract Functions	18
Contract Flow	23
Domain Info	24
Summary	25
Disclaimer	26
About Cyberscope	27

Contract Review

Contract Name	OPEC_V2
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	2000 runs
Licence	MIT
Explorer	https://snowtrace.io/address/0x7f03D24d2fC79fECF668485e1c4e4C4B21044B42
Symbol	OPEC
Decimals	18
Total Supply	6,000,000
Domain	opulence.network

Source Files

Filename	SHA256
contract.sol	25385e490bbc6d5fd2e6dd5be01aefe43f3b1832266b6532b7c0ff57059188ba

Audit Updates

Initial Audit	17th 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	critical
Location	contract.sol#L1142,1154

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `_liquidity_sell_tax` to a high value. This will cause the contract to operate as a honeypot.

```
if (!_isExcludedFromFee[from] && to == address(_uniswapV2Pair)) {  
    fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10);  
    amount = amount.sub(fees);  
}
```

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 `_balanceLimit` to zero.

```
if (!_isExcludedFromFee[to] && _isMaxLimit) {  
    require(amount + balanceOf(to) <= _balanceLimit, "OPEC: TRANSFER RECIPIENT  
BALANCE LIMIT");  
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` 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.

Read about the fees manipulation in the [Exceed Limit Fees Manipulation](#) section.

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L1115

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 `setUintParameter` function with `PROTOCOL_PARAMETER.SELL_TAX` and a high percentage value.

```
function setUintParameter(PROTOCOL_PARAMETER _parameter, uint256 _value)
external onlyManager {
    if (_parameter == PROTOCOL_PARAMETER.SELL_TAX) {
        _liquidity_sell_tax = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_AMOUNT) {
        _balanceLimit = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_BUY) {
        _maxBuyLimit = _value;
    }
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L1087

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `setTaxFeePercent` function. As a result the contract tokens will be highly inflated

```
function mint(address account_, uint256 amount_) external onlyPolicy {  
    _mint(account_, amount_);  
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1099

Description

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

```
require(!_isBlacklisted[from] && !_isBlacklisted[to], "OPEC: TRANSFER  
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
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L06	Missing Events Access Control
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

L01 - Public Function could be Declared External

Criticality

minor

Location

contract.sol#L658,666,683,690,715,728,745,768,797,824 and 3 more

Description

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

```
transferOwnership  
renounceOwnership  
owner  
decreaseAllowance  
increaseAllowance  
transferFrom  
approve  
allowance  
transfer  
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality

minor

Location

contract.sol#L1033

Description

Constant state variables should be declared constant to save gas.

```
_isSwapping
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L8,238,240,271,340,1091,1095,1099,1107,1115 and 22 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.

```
_liquidity_sell_tax  
_maxBuyLimit  
_balanceLimit  
_launchTime  
_isLaunched  
_isMaxLimit  
_isSwapping  
_isTaxable  
_policy  
...
```

Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>

L06 - Missing Events Access Control

Criticality

minor

Location

contract.sol#L1091

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.

```
_policy = _address
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality

minor

Location

contract.sol#L902

Description

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

```
_burn
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality

minor

Location

contract.sol#L1043

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)(_managers[msg.sender] == true,NOT MANAGER)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality

minor

Location

contract.sol#L1136

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
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	✓	-
	swapExactAVAXForTokensSupporting	External	Payable	-

	FeeOnTransferTokens			
	swapExactTokensForAVAXSupporting FeeOnTransferTokens	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	✓	-
IOPEC	Interface			
	setCheckParameter	External	✓	-
	setBooleanParameter	External	✓	-
	setUintParameter	External	✓	-
	setAddressParameter	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		

	div	Internal		
	mod	Internal		
	mod	Internal		
ERC20	Implementation	IERC20, IERC20Meta data		
	<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	✓	
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
OPEC	Implementation	IOPEC, ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	mint	External	✓	onlyPolicy

	setPolicy	External	✓	onlyOwner
	setManager	External	✓	onlyOwner
	setCheckParameter	External	✓	onlyManager
	setBooleanParameter	External	✓	onlyManager
	setUintParameter	External	✓	onlyManager
	setAddressParameter	External	✓	onlyManager
	setLaunch	External	✓	onlyManager
	_transfer	Internal	✓	

Contract Flow



Domain Info

Domain Name	opulence.network
Registry Domain ID	ca60e761bf374ab8b76e8e0a6c6442e3-DONUTS
Creation Date	2021-11-28T19:20:22Z
Updated Date	2021-12-03T19:21:08Z
Registry Expiry Date	2022-11-28T19:20:22Z
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 5 months before the creation of the audit. It will expire in 7 months.

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

Summary

There are some functions that can be abused by the owner, like manipulating fees, blacklisting addresses, stopping transactions and minting tokens. 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. 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>



Cyberscope

Audit Report

Opulence Management

April 2022

Type ERC20

Network AVAX

SHA256 178a995c82d9c510646c4fd040baaa59111231bd6a9240da3eb08314033ff13c

Audited by © cyberscope

Table of Contents

Table of Contents	1
Source Files	3
Audit Updates	3
Contract Analysis	4
Contract Diagnostics	5
MC - Missing Check	6
Description	6
Recommendation	7
CR - Code Repetition (1/2)	8
Description	8
Recommendation	8
CR - Code Repetition (2/2)	9
Description	9
Recommendation	9
DN - Duplicate Namings	10
Description	10
Recommendation	10
L01 - Public Function could be Declared External	11
Description	11
Recommendation	11
L04 - Conformance to Solidity Naming Conventions	12
Description	12
Recommendation	12
L07 - Missing Events Arithmetic	13
Description	13
Recommendation	13

L11 - Unnecessary Boolean equality	14
Description	14
Recommendation	14
L13 - Divide before Multiply Operation	15
Description	15
Recommendation	15
Contract Functions	16
Contract Flow	18
Summary	19
Disclaimer	20
About Cyberscope	21

Source Files

Filename	SHA256
contract.sol	178a995c82d9c510646c4fd040baaa59111231bd6a9240da3eb08314033ff13c

Audit Updates

Initial Audit	16th April 2022
Corrected	

Contract Analysis

The Management Contract is responsible for creating nodes and delivering rewards. The role `OnlyManager` can interact with the contract and has access to the following functionality:

- Create New Node for a user and set his rewards per day
- Calculate available rewards for all the Nodes of a specific user
- Get Nodes of specific user
- Get the Node names of a specific user
- Get the Node created time of a specific user
- Get the last time a specific user claimed rewards
- Get the daily Node rewards of a specific user
- Get all the Node rewards of a specific user
- Cashout the rewards for a specific user
- Compound the rewards of a specific user

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	MC	Missing Check
●	CR	Code Repetition
●	DN	Duplicate Namings
●	L01	Public Function could be Declared External
●	L04	Conformance to Solidity Naming Conventions
●	L07	Missing Events Arithmetic
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

MC - Missing Check

Criticality	minor
Location	contract.sol#L275

Description

The claimInterval is used as a divisor in the reward calculation. Since the contract owner can arbitrary change it, there are 2 potential issues:

Zero division

If the contract owner sets the claimInterval to zero, then the reward calculation will revert.

Rewards Calculation Underflow

If the contract owner sets a small number to claimInterval, the reward will yield a relatively big number.

Later, if the contract owner sets a big number to claimInterval the expression `(block.timestamp - node.createTime).div(claimInterval).mul(node.rewardPerDay)` may yield a smaller number than the previous `rewardedAmount` value. As a result the entire expression will underflow.

```
function setClaimInterval(uint256 _interval) public onlyOwner {  
    claimInterval = _interval;  
}
```

```
reward = (block.timestamp - node.createTime)  
        .div(claimInterval)  
        .mul(node.rewardPerDay)  
        .sub(node.rewardedAmount);  
  
totalRewards += reward;  
node.lastClaimTime = block.timestamp;  
node.rewardedAmount = node.rewardedAmount + reward;
```

Recommendation

The contract should guarantee that the `claimInterval` will not revert the rewards calculation.

CR - Code Repetition (1/2)

Criticality	minor
Location	contract.sol#L297,323

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily. The functions `airdropNode` and `createNode` have the exact same logic, one of them can be removed.

```
function createNode(
    address account,
    string memory _name,
    uint256 _rewardPerDay
) external onlyManager {
    require(
        nodeCountOfUser[account] < nodeLimit,
        "MANAGEMENT: CREATE NODE LIMIT ERROR"
    );

    nodesOfUser[account].push(
        NodeInfo({
            name: _name,
            createTime: block.timestamp,
            lastClaimTime: block.timestamp,
            rewardPerDay: _rewardPerDay,
            rewardedAmount: 0
        })
    );

    nodeCountOfUser[account] += 1;
    totalCount += 1;

    emit CREATE_NODE(account, _name, _rewardPerDay, block.timestamp);
}
```

Recommendation

Remove one of these duplicate functions.

CR - Code Repetition (2/2)

Criticality	minor
Location	contract.sol#L399,404

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily. The line of code that sets the node's last claim to block timestamp can be moved before the if statement as it will execute under both circumstances.

```
if (returnValue + reward < amount) {  
    node.lastClaimTime = block.timestamp;  
    node.rewardedAmount = node.rewardedAmount + reward;  
  
    returnValue += reward;  
} else {  
    node.lastClaimTime = block.timestamp;  
    node.rewardedAmount = node.rewardedAmount + amount - returnValue;  
  
    returnValue += amount - returnValue;  
}
```

Recommendation

Move the logic of setting node's lastClaimTime before the if statement.

DN - Duplicate Namings

Criticality	minor
Location	contract.sol#L344,369

Description

Duplicate named functions can be problematic and should be avoided according to solidity naming conventions. Use declarative naming patterns.

```
function calculateAvailableReward(address account)
    external
    view
    onlyManager
    returns (uint256)
{
    uint256 totalRewards = 0;

    for (uint256 i = 0; i < nodeCountOfUser[account]; i++) {
        totalRewards += _calculateAvailableReward(account, i);
    }
    return totalRewards;
}

function calculateAvailableReward(address account, uint256 _index)
    external
    view
    onlyManager
    returns (uint256)
{
    uint256 reward;
    reward = _calculateAvailableReward(account, _index);
    return reward;
}
```

Recommendation

Rename one of the functions above to a unique name.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L179,198,207,260,264,268,272

Description

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

```
setClaimInterval  
setNodeLimit  
removeManager  
addManager  
transferOwnership  
renounceOwnership  
owner
```

Recommendation

Use the external attribute for functions never called from the contract.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L237,243,260,264,268,272,299,300,325,326,364,443,644

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.

```
_i  
_index  
_rewardPerDay  
_name  
_interval  
_limit  
_manager  
AIREDROP_NODE  
CREATE_NODE  
...
```

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#L268,272

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.

```
claimInterval = _interval  
nodeLimit = _limit
```

Recommendation

Emit an event for critical parameter changes.

L11 - Unnecessary Boolean equality

Criticality

minor

Location

contract.sol#L251

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)(managers[msg.sender] == true,MANAGEMENT: NOT MANAGER)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L377,414,443,616,644

Description

Performing divisions before multiplications may cause lose of prediction.

```
temp = (48 + uint8(_i - (_i / 10) * 10))  
reward = (block.timestamp -  
node.createTime).div(claimInterval).mul(node.rewardPerDay).sub(node.rewardedAmount)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Management	Implementation	Ownable		
	<Constructor>	Public	✓	-
	addManager	Public	✓	onlyOwner
	removeManager	Public	✓	onlyOwner
	setNodeLimit	Public	✓	onlyOwner
	setClaimInterval	Public	✓	onlyOwner
	getNodeLimit	External		onlyManager
	getClaimInterval	External		onlyManager
	getTotalCount	External		onlyManager
	getNodesCountOfUser	External		onlyManager
	createNode	External	✓	onlyManager
	airdropNode	External	✓	onlyManager
	calculateAvailableReward	External		onlyManager

	calculateAvailableReward	External		onlyManager
	compoundNode	External	✓	onlyManager
	cashoutAllReward	External	✓	onlyManager
	cashoutReward	External	✓	onlyManager
	getNodeNames	External		onlyManager
	getNodeCreateTime	External		onlyManager
	getNodeLastClaimTime	External		onlyManager
	getNodeRewardPerDay	External		onlyManager
	getNodeAvailableReward	External		onlyManager
	isNodeOwner	Internal		
	_calculateAvailableReward	Internal		
	uint2str	Internal		

Contract Flow



Summary

Opec Management is the reward mechanism contract for the Opec protocol. This audit focuses on the correct functionality, the security concerns and some performance 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.

Cyberscope 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>



Cyberscope

Audit Report

Opulence Migrate

March 2022

SHA256 e602bb51636f9d09cfa6186295e012525e3567b215a9649aa0d20c507e34e430

Audited by © cyberscope

Table of Contents

Table of Contents	1
Source Files	2
Audit Updates	2
Contract Analysis	3
Contract Diagnostics	4
MC - Missing Check	5
Description	5
Recommendation	5
L01 - Public Function could be Declared External	6
Description	6
Recommendation	6
L02 - State Variables could be Declared Constant	7
Description	7
Recommendation	7
L04 - Conformance to Solidity Naming Conventions	8
Description	8
Recommendation	8
L09 - Dead Code Elimination	9
Description	9
Recommendation	9
Contract Functions	10
Contract Flow	12
Summary	13
Disclaimer	14
About Cyberscope	15

Source Files

Filename	SHA256
contract.sol	e602bb51636f9d09cfa6186295e012525e3567b215a9649aa0d20c507e34e430

Audit Updates

Initial Audit	28th March 2022
Corrected	

Contract Analysis

The migrate contract is responsible for transforming an legacy token to a new one.

The old and the new tokens are created once in the constructor and cannot be changed.

When a users calls the “migrate” method, the amount of his legacy tokens are moved to the migrator contract and the migrator contract sends him the corresponding amount of the new tokens.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	MC	Missing Check
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination

MC - Missing Check

Criticality	medium
Location	contract.sol#L365

Description

The contract should check if the amount of the new token is enough to cover the old user's amount. Otherwise, the user may transfer his tokens without receiving the corresponding amount.

```
uint256 oldBalance;  
oldBalance = oldToken.balanceOf(address(msg.sender));  
  
oldToken.transferFrom(msg.sender, address(this), oldBalance);  
newToken.transfer(msg.sender, oldBalance);
```

Recommendation

The contract should properly check the variables according to the required specifications

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L277,296,305,364,372

Description

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

```
burnToken  
migrate  
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#L349

Description

Constant state variables should be declared constant to save gas.

```
_burnAddress
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L356,357,349

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.

```
_burnAddress  
_oldToken  
_newToken
```

Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>

L09 - Dead Code Elimination

Criticality

minor

Location

contract.sol#L248

Description

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

```
_msgSender
```

Recommendation

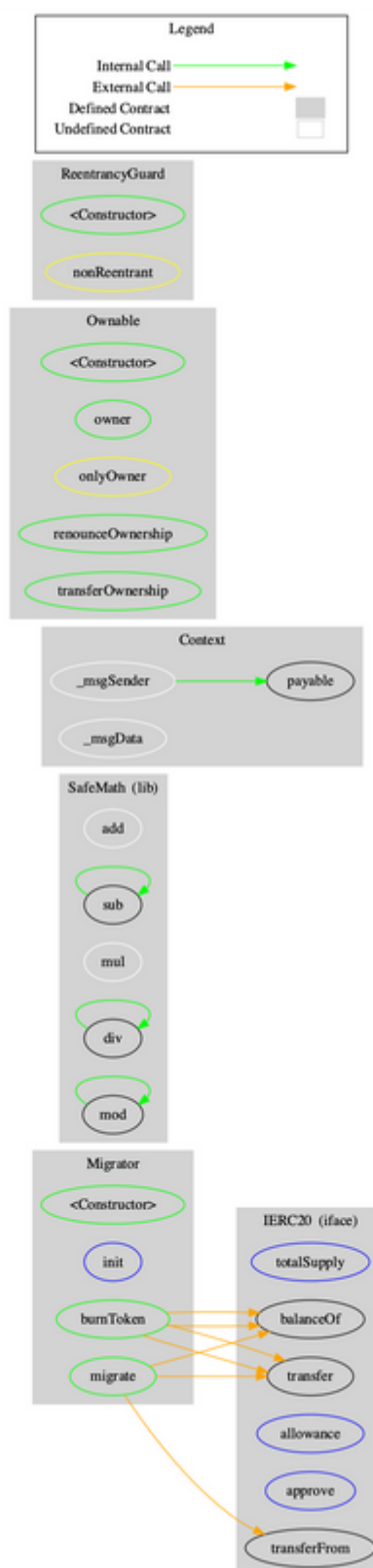
Remove unused functions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
Migrator	Implementation	Context, Ownable, Reentrancy Guard		
	<Constructor>	Public	✓	-
	init	External	✓	onlyOwner
	migrate	Public	✓	-
	burnToken	Public	✓	onlyOwner

Contract Flow



Summary

The migrator is a contract that is responsible for converting a legacy to a new token. This audit focuses on the correct functionality, the security concerns and some performance 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.

Cyberscope 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>



Cyberscope

Audit Report

Opulence OPEC

March 2022

Type ERC20

Network AVAX

Address 0x283366bb42ef49a994913BAF22263c6562e588a4

Audited by © cyberscope

Table of Contents

Table of Contents	1
Contract Review	3
Source Files	3
Audit Updates	3
Contract Analysis	4
ST - Stop Transactions	5
Description	5
Recommendation	6
ELFM - Exceed Limit Fees Manipulation	7
Description	7
Recommendation	7
MT - Mint Tokens	8
Description	8
Recommendation	8
BC - Blacklisted Contracts	9
Description	9
Recommendation	9
Contract Diagnostics	10
FSA - Fixed Swap Address	11
Description	11
Recommendation	11
L01 - Public Function could be Declared External	12
Description	12
Recommendation	12
L02 - State Variables could be Declared Constant	13
Description	13

Recommendation	13
L04 - Conformance to Solidity Naming Conventions	14
Description	14
Recommendation	14
L06 - Missing Events Access Control	15
Description	15
Recommendation	15
L09 - Dead Code Elimination	16
Description	16
Recommendation	16
L11 - Unnecessary Boolean equality	17
Description	17
Recommendation	17
L13 - Divide before Multiply Operation	18
Description	18
Recommendation	18
Contract Functions	19
Contract Flow	24
Summary	25
Disclaimer	26
About Cyberscope	27

Contract Review

Contract Name	OPEC
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	2000 runs
Licence	MIT
Explorer	https://snowtrace.io/token/0x283366bb42ef49a994913BAF22263c6562e588a4
Symbol	OPEC
Decimals	18
Total Supply	6,000,000

Source Files

Filename	SHA256
contract.sol	1f62057224f30668eb99d33dddc5a6eaf36895a0cf82f7f87c0ccae9d0c544f9

Audit Updates

Initial Audit	28th March 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

critical

Location

contract.sol#L1142,1154,1161

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `_liquidity_sell_tax` to a high value. This can cause the contract to operate as a honeypot.

```
if (!_isExcludedFromFee[from] && to == address(_uniswapV2Pair)) {  
    fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10);  
    amount = amount.sub(fees);
```

The contract owner has the authority to stop the buys and transfers for all users excluding the owner. The owner may take advantage of it by setting the `_balanceLimit` to zero.

```
if (!_isExcludedFromFee[to] && _isMaxLimit) {  
    require(amount + balanceOf(to) <= _balanceLimit, "OPEC: TRANSFER RECIPIENT  
BALANCE LIMIT");  
}
```

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

```
if (block.timestamp - _launchTime < 10 minutes) {  
    require(amount <= 10 * 1e18, "OPEC: TRANSFER BUY LIMIT ERROR");  
} else {  
    require(amount <= _maxBuyLimit, "OPEC: TRANSFER BUY LIMIT ERROR");  
}
```

Recommendation

The contract could embody a check for not allowing setting the `_balanceLimit` and `_balanceLimit` 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.

Read more in the [fees manipulation section](#).

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L1115

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 `setUintParameter` function with `SELL_TAX` and a high percentage value.

```
function setUintParameter(PROTOCOL_PARAMETER _parameter, uint256 _value)
external onlyManager {
    if (_parameter == PROTOCOL_PARAMETER.SELL_TAX) {
        _liquidity_sell_tax = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_AMOUNT) {
        _balanceLimit = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_BUY) {
        _maxBuyLimit = _value;
    }
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L1087

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result the contract tokens will be highly inflated

```
function mint(address account_, uint256 amount_) external onlyPolicy {  
    _mint(account_, amount_);  
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1138

Description

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

```
require(!_isBlacklisted[from] && !_isBlacklisted[to], "OPEC: TRANSFER  
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
●	FSA	Fixed Swap Address
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L06	Missing Events Access Control
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

FSA - Fixed Swap Address

Criticality	minor
Location	contract.sol#L1074

Description

The swap address is assigned once in the constructor and it can not be changed. The decentralized swaps sometimes create a new swap version or abandon the current. A contract that cannot change the swap address may not be able to catch-up the upgrade.

```
_uniswapV2Router = IJoeRouter02(uniV2Router);  
  
_uniswapV2Pair =  
IUniswapV2Pair(IJoeFactory(_uniswapV2Router.factory()).createPair(address(this),  
_uniswapV2Router.WAVAX()));
```

Recommendation

It could be better to allow the swap address mutation in case of future swap updates.

L01 - Public Function could be Declared External

Criticality

minor

Location

contract.sol#L658,666,683,690,715,728,745,768,797,824 and 3 more

Description

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

```
transferOwnership  
renounceOwnership  
owner  
decreaseAllowance  
increaseAllowance  
transferFrom  
approve  
allowance  
transfer  
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality

minor

Location

contract.sol#L1033

Description

Constant state variables should be declared constant to save gas.

```
_isSwapping
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L8,238,240,271,340,1091,1095,1099,1107,1115 and 22 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.

```
_liquidity_sell_tax  
_maxBuyLimit  
_balanceLimit  
_launchTime  
_isLaunched  
_isMaxLimit  
_isSwapping  
_isTaxable  
_policy  
...
```

Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>

L06 - Missing Events Access Control

Criticality

minor

Location

contract.sol#L1091

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.

```
_policy = _address
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality

minor

Location

contract.sol#L902

Description

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

```
_burn
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality

minor

Location

contract.sol#L1043

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)(_managers[msg.sender] == true,NOT MANAGER)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality

minor

Location

contract.sol#L1136

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
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	✓	-
	swapExactAVAXForTokensSupporting	External	Payable	-

	FeeOnTransferTokens			
	swapExactTokensForAVAXSupporting FeeOnTransferTokens	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	✓	-
IOPEC	Interface			
	setCheckParameter	External	✓	-
	setBooleanParameter	External	✓	-
	setUintParameter	External	✓	-
	setAddressParameter	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		

	div	Internal		
	mod	Internal		
	mod	Internal		
ERC20	Implementation	IERC20, IERC20Meta data		
	<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	✓	
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
OPEC	Implementation	IOPEC, ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	mint	External	✓	onlyPolicy

	setPolicy	External	✓	onlyOwner
	setManager	External	✓	onlyOwner
	setCheckParameter	External	✓	onlyManager
	setBooleanParameter	External	✓	onlyManager
	setUintParameter	External	✓	onlyManager
	setAddressParameter	External	✓	onlyManager
	setLaunch	External	✓	onlyManager
	_transfer	Internal	✓	

Contract Flow



Summary

There are some functions that can be abused by the owner, like manipulating fees, stopping transactions, minting tokens and blacklisting addresses. 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. 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>



Cyberscope

Audit Report

Opulence Reward Pool

March 2022

SHA256 13b5d6c1a1c43eb09a52360152b4ea88557da3126f6d9f42452e1e9f1282926c

Audited by © cyberscope

Table of Contents

Table of Contents	1
Source Files	2
Audit Updates	2
Contract Analysis	3
Contract Diagnostics	4
L01 - Public Function could be Declared External	5
Description	5
Recommendation	5
L04 - Conformance to Solidity Naming Conventions	6
Description	6
Recommendation	6
L09 - Dead Code Elimination	7
Description	7
Recommendation	7
L11 - Unnecessary Boolean equality	8
Description	8
Recommendation	8
Contract Functions	9
Contract Flow	11
Summary	12
Disclaimer	13
About Cyberscope	14

Source Files

Filename	SHA256
contract.sol	13b5d6c1a1c43eb09a52360152b4ea88557da3126f6d9f42452e1e9f1282926c

Audit Updates

Initial Audit	28th March 2022
Corrected	

Contract Analysis

Rewards pool accumulated OPEC tokens. The contract is responsible for transferring the accumulated OPEC tokens to the treasury and arbitrary addresses. The contract contains two roles.

Contract Owner

The contract owner role has the ability to set the treasury and OPEC token address. The role is also responsible for adding and removing addresses to the “manager” role.

Manager

The manager role has the ability to withdraw the rewards pool accumulated tokens to the treasury and other arbitrary addresses. Many addresses can have the manager role.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	L01	Public Function could be Declared External
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L111,130,139

Description

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

```
transferOwnership  
renounceOwnership  
owner
```

Recommendation

Use the external attribute for functions never called from the contract

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L615,633,634,653,658,663,621

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.

```
_managers  
_rewardAmount  
_account  
_treasury  
_opex  
RewardPool_V2
```

Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L228,257,317,327,290,300,203,404,428,419 and 1 more

Description

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

```
safeTransferFrom
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
sendValue
functionStaticCall
functionDelegateCall
functionCallWithValue
functionCall
...
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality

minor

Location

contract.sol#L623

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)(_managers[msg.sender] == true,Only managers can call this function)
```

Recommendation

Remove the equality to the boolean constant.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
SafeERC20	Library			
	safeTransfer	Internal	✓	

	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
RewardPool_V2	Implementation	Ownable		
	<Constructor>	Public	✓	-
	init	External	✓	onlyOwner
	addManager	External	✓	onlyOwner
	removeManager	External	✓	onlyOwner
	setOPECAddress	External	✓	onlyOwner
	setTreasuryAddress	External	✓	onlyOwner
	rewardTo	External	✓	onlyManager
	withdrawToken	External	✓	onlyManager

Contract Flow



Summary

The Opulence reward pool contract is responsible for transferring the accumulated tokens to addresses. The contract contains two different types of roles. The audit focuses on the role responsibilities and the core functionality of the contract.

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>



Cyberscope

Audit Report

Opulence User Helper

March 2022

SHA256 c397f881b5c847b734b0ce56a5e78d333f3793aa71d35d0d1eb4a51077ed1aca

Audited by © cyberscope

Table of Contents

Table of Contents	1
Source Files	3
Audit Updates	3
Contract Analysis	4
Create Nodes	4
management	4
Contract Diagnostics	5
FSA - Fixed Swap Address	6
Description	6
Recommendation	6
CR - Code Repetition	7
Description	7
Recommendation	7
MC - Missing Check	8
Description	8
Recommendation	8
L01 - Public Function could be Declared External	9
Description	9
Recommendation	9
L02 - State Variables could be Declared Constant	10
Description	10
Recommendation	10
L04 - Conformance to Solidity Naming Conventions	11
Description	11
Recommendation	11
L07 - Missing Events Arithmetic	12

Description	12
Recommendation	12
L09 - Dead Code Elimination	13
Description	13
Recommendation	13
Contract Functions	14
Contract Flow	18
Summary	19
Disclaimer	20
About Cyberscope	21

Source Files

Filename	SHA256
contract.sol	c397f881b5c847b734b0ce56a5e78d333f3793aa71d35d0d1eb4a51077ed1aca

Audit Updates

Initial Audit	28th March 2022
Corrected	

Contract Analysis

UserHelper is a wrapper of the management contract. The management contract is responsible for the essential functionality of the create node. Additionally, it is the medium to guarantee the safety of the process.

The auditing of the “**Management**” contract is **out of the scope of this audit**.

Create Nodes

- A user has the ability to create nodes.
- The users can give a name to the created node.
- The cost of a node can be configurable by the contract owner.
- When the user creates a node, the funds are transferred to the contract's address.
- If the contract's accumulated amount of tokens is more than a threshold, the amount is distributed proportionally to the liquidity pool, treasury address and burned address.
- The threshold can be configurable by the contract owner.
- The distribution portions can be configured by the contract owner.
- The entire UserHelper functionality can be paused by the contract owner.

management

- The user has the ability to claim the rewards.
- The management contract is responsible for checking if the depositor is applicable to claim the rewards.
- The awarded amount is taxed with a "claimTax".
- The claimTax configurable by the contract owner.
- The claimTax amount is converted to the native coin and transferred to the treasury and dev wallet proportionally.
- The proportions of treasury and dev amounts can be paused by the contract owner.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	FSA	Fixed Swap Address
●	CR	Code Repetition
●	MC	Missing Check
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L07	Missing Events Arithmetic
●	L09	Dead Code Elimination

FSA - Fixed Swap Address

Criticality	minor
Location	contract.sol#L899

Description

The swap address is assigned once in the constructor and it can not be changed. The decentralized swaps sometimes create a new swap version or abandon the current. A contract that cannot change the swap address may not be able to catch-up the upgrade.

```
uniswapV2Router = IJoeRouter02(_router);
```

Recommendation

It could be better to allow the swap address mutation in case of future swap updates.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L937,1003

Description

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

```
if (!distributing && checkBalance) {
    distributing = true;

    uint256 amount = opec.balanceOf(address(this));

    uint256 liquidityAmount = amount * create_liquidity / 100;
    uint256 treasuryAmount = amount * create_treasury / 100;
    uint256 burnAmount = amount * create_burn / 100;

    swapAndLiquify(liquidityAmount);

    swapAndSendToFee(treasury, treasuryAmount);

    opec.safeTransfer(burn, burnAmount);

    distributing = false;
}
```

Recommendation

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

MC - Missing Check

Criticality	medium
Location	contract.sol#L942,967

Description

The contract is processing variables that have not properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

There are some properties like `create_liquidity`, `create_treasury`, `create_burn` that are used in order to get proportional fees from the amount. These fees are assumed to be a percentage, but in the code there is no guarantee that the fees can be configured to be more than 100%.

```
uint256 liquidityAmount = amount * create_liquidity / 100;
uint256 treasuryAmount = amount * create_treasury / 100;
uint256 burnAmount = amount * create_burn / 100;
//
swapAndSendToFee(treasury, cashoutAmount * claim_treasury / 100);
swapAndSendToFee(dev, cashoutAmount * claim_dev / 100);
//
```

Recommendation

The contract should properly check the variables according to the required specifications

L01 - Public Function could be Declared External

Criticality

minor

Location

contract.sol#L816,835,844,907,911,916,930,958,973,988 and 10 more

Description

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

```
getNodeAvailableReward  
getNodeRewardPerDay  
getNodeLastClaimTime  
getNodeCreateTime  
getNodeNames  
getNodeCount  
getTotalCount  
getClaimInterval  
getNodeLimit  
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality

minor

Location

contract.sol#L860

Description

Constant state variables should be declared constant to save gas.

```
burn
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L134,930,958,988,871,872,873,876,877

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.

```
claim_dev  
claim_treasury  
create_burn  
create_treasury  
create_liquidity  
_name  
_index  
WAVAX
```

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#L916

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.

```
create_liquidity = parameters[0]
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L410,439,499,509,472,482,385,586,610,601

Description

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

```
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
sendValue
functionStaticCall
functionDelegateCall
functionCallWithValue
functionCall
...
```

Recommendation

Remove unused functions.

Contract Functions

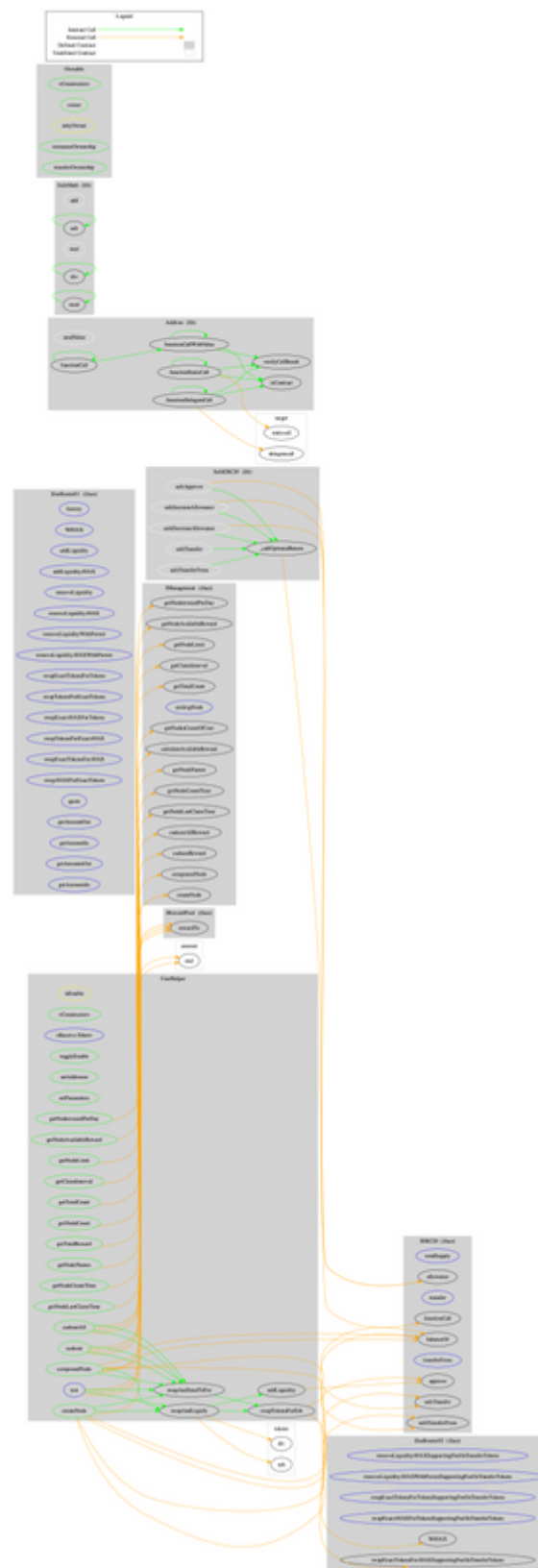
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IManagement	Interface			
	getNodeLimit	External		-
	getClaimInterval	External		-
	getTotalCount	External		-
	getNodesCountOfUser	External		-
	createNode	External	✓	-
	airdropNode	External	✓	-
	calculateAvailableReward	External		-
	calculateAvailableReward	External		-
	cashoutAllReward	External	✓	-
	cashoutReward	External	✓	-
	compoundNode	External	✓	-
	getNodeNames	External		-
	getNodeCreateTime	External		-
	getNodeLastClaimTime	External		-
	getNodeRewardPerDay	External		-
	getNodeAvailableReward	External		-
IRewardPool	Interface			
	rewardTo	External	✓	-

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	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	

	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
UserHelper	Implementation	Ownable		

	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	toggleEnable	Public	✓	onlyOwner
	setAddresses	Public	✓	onlyOwner
	setParameters	Public	✓	onlyOwner
	createNode	Public	✓	isEnabled
	cashout	Public	✓	isEnabled
	cashoutAll	Public	✓	isEnabled
	compoundNode	Public	✓	isEnabled
	getTotalReward	Public		-
	getNodeLimit	Public		-
	getClaimInterval	Public		-
	getTotalCount	Public		-
	getNodeCount	Public		-
	getNodeNames	Public		-
	getNodeCreateTime	Public		-
	getNodeLastClaimTime	Public		-
	getNodeRewardPerDay	Public		-
	getNodeAvailableReward	Public		-
	swapAndSendToFee	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	test	External	✓	onlyOwner

Contract Flow



Summary

UserHelper wraps the functionality of the Management contract. In this audit we focus on the wrapping functionality. The management contract is out of the scope. This audit mentions some performance improvements, security concerns and optimizations.

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.

Cyberscope 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>



Cyberscope

Audit Report

Opulence OPEC

March 2022

Type ERC20

Network AVAX

Address 0x283366bb42ef49a994913BAF22263c6562e588a4

Audited by © cyberscope

Table of Contents

Table of Contents	1
Contract Review	3
Source Files	4
Audit Updates	4
Contract Analysis	5
ST - Stop Transactions	6
Description	6
Recommendation	6
ELFM - Exceed Limit Fees Manipulation	7
Description	7
Recommendation	7
MT - Mint Tokens	8
Description	8
Recommendation	8
BC - Blacklisted Contracts	9
Description	9
Recommendation	9
Contract Diagnostics	10
L01 - Public Function could be Declared External	11
Description	11
Recommendation	11
L02 - State Variables could be Declared Constant	12
Description	12
Recommendation	12
L04 - Conformance to Solidity Naming Conventions	13
Description	13

Recommendation	13
L06 - Missing Events Access Control	14
Description	14
Recommendation	14
L09 - Dead Code Elimination	15
Description	15
Recommendation	15
L11 - Unnecessary Boolean equality	16
Description	16
Recommendation	16
L13 - Divide before Multiply Operation	17
Description	17
Recommendation	17
Contract Functions	18
Contract Flow	23
Domain Info	24
Summary	25
Disclaimer	26
About Cyberscope	27

Contract Review

Contract Name	OPEC
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	2000 runs
Licence	MIT
Explorer	https://snowtrace.io/token/0x283366bb42ef49a994913BAF22263c6562e588a4
Symbol	OPEC
Decimals	18
Total Supply	6,000,000
Domain	https://opulence.network

Source Files

Filename	SHA256
contract.sol	1f62057224f30668eb99d33dddc5a6eaf36895a0cf82f7f87c0ccae9d0c544f9

Audit Updates

Initial Audit	22nd March 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	critical
Location	contract.sol#L1142,1154

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `_liquidity_sell_tax` to a high value. This will cause the contract to operate as a honeypot.

```
if (!_isExcludedFromFee[from] && to == address(_uniswapV2Pair)) {  
    fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10);  
    amount = amount.sub(fees);  
}
```

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 `_balanceLimit` to zero.

```
if (!_isExcludedFromFee[to] && _isMaxLimit) {  
    require(amount + balanceOf(to) <= _balanceLimit, "OPEC: TRANSFER RECIPIENT  
BALANCE LIMIT");  
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` 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.

Read about the fees manipulation in the [Exceed Limit Fees Manipulation](#) section.

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L1115

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 `setUintParameter` function with `PROTOCOL_PARAMETER.SELL_TAX` and a high percentage value.

```
function setUintParameter(PROTOCOL_PARAMETER _parameter, uint256 _value)
external onlyManager {
    if (_parameter == PROTOCOL_PARAMETER.SELL_TAX) {
        _liquidity_sell_tax = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_AMOUNT) {
        _balanceLimit = _value;
    } else if (_parameter == PROTOCOL_PARAMETER.LIMIT_BUY) {
        _maxBuyLimit = _value;
    }
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L1087

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `setTaxFeePercent` function. As a result the contract tokens will be highly inflated

```
function mint(address account_, uint256 amount_) external onlyPolicy {  
    _mint(account_, amount_);  
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1099

Description

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

```
require(!_isBlacklisted[from] && !_isBlacklisted[to], "OPEC: TRANSFER  
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
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L06	Missing Events Access Control
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

L01 - Public Function could be Declared External

Criticality

minor

Location

contract.sol#L658,666,683,690,715,728,745,768,797,824 and 3 more

Description

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

```
transferOwnership  
renounceOwnership  
owner  
decreaseAllowance  
increaseAllowance  
transferFrom  
approve  
allowance  
transfer  
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality

minor

Location

contract.sol#L1033

Description

Constant state variables should be declared constant to save gas.

```
_isSwapping
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality

minor

Location

contract.sol#L8,238,240,271,340,1091,1095,1099,1107,1115 and 22 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.

```
_liquidity_sell_tax  
_maxBuyLimit  
_balanceLimit  
_launchTime  
_isLaunched  
_isMaxLimit  
_isSwapping  
_isTaxable  
_policy  
...
```

Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>

L06 - Missing Events Access Control

Criticality

minor

Location

contract.sol#L1091

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.

```
_policy = _address
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L902

Description

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

```
_burn
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality

minor

Location

contract.sol#L1043

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)(_managers[msg.sender] == true,NOT MANAGER)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality

minor

Location

contract.sol#L1136

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = (amount / 100) * (_liquidity_sell_tax - passedDays * 10)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
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	✓	-
	swapExactAVAXForTokensSupporting	External	Payable	-

	FeeOnTransferTokens			
	swapExactTokensForAVAXSupporting FeeOnTransferTokens	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	✓	-
IOPEC	Interface			
	setCheckParameter	External	✓	-
	setBooleanParameter	External	✓	-
	setUintParameter	External	✓	-
	setAddressParameter	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		

	div	Internal		
	mod	Internal		
	mod	Internal		
ERC20	Implementation	IERC20, IERC20Meta data		
	<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	✓	
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
OPEC	Implementation	IOPEC, ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	mint	External	✓	onlyPolicy

	setPolicy	External	✓	onlyOwner
	setManager	External	✓	onlyOwner
	setCheckParameter	External	✓	onlyManager
	setBooleanParameter	External	✓	onlyManager
	setUintParameter	External	✓	onlyManager
	setAddressParameter	External	✓	onlyManager
	setLaunch	External	✓	onlyManager
	_transfer	Internal	✓	

Contract Flow



Domain Info

Domain Name	opulence.network
Registry Domain ID	ca60e761bf374ab8b76e8e0a6c6442e3-DONUTS
Creation Date	2021-11-28T19:20:22Z
Updated Date	2021-12-03T19:21:08Z
Registry Expiry Date	2022-11-28T19:20:22Z
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 4 months before the creation of the audit. It will expire in 8 months.

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

Summary

There are some functions that can be abused by the owner, like manipulating fees, blacklisting addresses, stopping transactions and minting tokens. 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. 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>