Cyberscope

Audit Report

Sweep Stake Network

July 2022

Type	BEP20
Network	BSC
Address	0xD6CDa0B438C7eF62d7976d2EC5B08d956D439528
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	SweepStakeNetwork
Compiler Version	v0.7.4+commit.3f05b770
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0xd6cda0b438c7ef62d7976d2ec5b08d956d439528
Symbol	$SSN
Decimals	4
Total Supply	1,000,000,000
Domain	sweepstake.network

Source Files

Filename	**SHA256**
contract.sol	b3600f9928c32804ef7f7d5d31707ee12a70bd72c0ad99d516857c27b429381e

Audit Updates

Initial Audit	30th July 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#L510,517

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 `cooldownTimerInterval` to a high value.

```
if (sender == pair &&
    buyCooldownEnabled &&
    !isTimelockExempt[recipient]) {
    require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1 minute
between 2 buys");
    cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
}
```

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

```
if (!authorizations[sender] && recipient != address(this)  && recipient !=
address(DEAD) && recipient != pair && recipient != marketingFeeReceiver &&
recipient != autoLiquidityReceiver){
    uint256 heldTokens = balanceOf(recipient);
    require((heldTokens + amount) <= _maxWalletToken,"Total Holding is currently
limited, you can not buy that much.");}
```

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.

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.

OTUT - Owner Transfer User's Tokens

Criticality	critical
Location	contract.sol#L718

Description

The contract owner has the authority to transfer the balance of a user's contract to the owner's contract. The owner may take advantage of it by calling the `airdrop` function and providing the user's address as the first argument.

```
    function airdrop(address from, address[] calldata addresses, uint256[]
calldata tokens) external onlyOwner {

    uint256 SCCC = 0;

    require(addresses.length == tokens.length,"Mismatch between Address and
token count");

    for(uint i=0; i < addresses.length; i++){
        SCCC = SCCC + tokens[i];
    }

    require(balanceOf(from) >= SCCC, "Not enough tokens to airdrop");

    for(uint i=0; i < addresses.length; i++){
        _basicTransfer(from,addresses[i],tokens[i]);
        if(!isDividendExempt[addresses[i]]) {
            try distributor.setShare(addresses[i], _balances[addresses[i]]) {}
catch {}
        }
    }

    if(!isDividendExempt[from]) {
        try distributor.setShare(from, _balances[from]) {} catch {}
    }
}
```

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L585

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by setting the totalFee any value greater than ¼ of the feeDenominator rate. If the contract owner sets a rate greater than 1, then the transactions will revert since the user's balance will not be sufficient.

```
function setFees(uint256 _liquidityFee, uint256 _buybackFee, uint256
_reflectionFee, uint256 _marketingFee, uint256 _treasuryFee, uint256
_feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    treasuryFee = _treasuryFee;
    buybackFee = _buybackFee;
    totalFee =
_liquidityFee.add(_buybackFee).add(_reflectionFee).add(_marketingFee).add(_treas
uryFee);
    feeDenominator = _feeDenominator;
    require(totalFee <= 15, "Total fee is over 15%");
}
```

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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L596

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `clearStuckBalance`.

```
function clearStuckBalance(uint256 amountPercentage) external authorized {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer(amountBNB * amountPercentage / 100);
}
```

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
●	ZD	Zero Division
●	CO	Code Optimization
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L05	Unused State Variable
●	L07	Missing Events Arithmetic
●	L09	Dead Code Elimination

ZD - Zero Division

Criticality	critical
Location	contract.sol#L570,609

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

```
uint256 feeAmount = amount.mul(totalFee).mul(multiplier).div(feeDenominator *
100);
//
uint256 amountToLiquify =
swapThreshold.mul(dynamicLiquidityFee).div(totalFee).div(2);
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L505

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The tradingOpen is always true, as a result the require expression will never fail.

```
if(!authorizations[sender] && !authorizations[recipient]){
    require(tradingOpen,"Trading not open yet");
}
```

Recommendation

The entire statement could be eliminated.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L112,117,132,601

Description

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

```
cooldownEnabled
transferOwnership
unauthorize
authorize
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L212,225,370,372,371,373,379,414,401,415

Description

Constant state variables should be declared constant to save gas.

```
tradingOpen
sellMultiplier
launchedAt
_totalSupply
ZERO
WBNB
DEAD
BUSD
dividendsPerShareAccuracyFactor
...
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L147,251,203,211,212,585,601,679,686,691,696,370,371,372,373,375,376,377,379,381,382,383,385,386

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.

```
_allowances
_balances
_maxWalletToken
_maxSellTxAmount
_maxBuyTxAmount
_totalSupply
_decimals
_symbol
_name
...
```

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#L370

Description

There are segments that contain unused state variables.

```
BUSD
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L251,490,494,585,686,691

Description

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

```
targetLiquidity = _target
swapThreshold = _amount
liquidityFee = _liquidityFee
_maxSellTxAmount = (_totalSupply * maxSellTxPercent) / 100
_maxBuyTxAmount = (_totalSupply * maxBuyTxPercent) / 100
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L546

Description

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

```
checkTxLimit
```

Recommendation

Remove unused functions.

Contract Functions

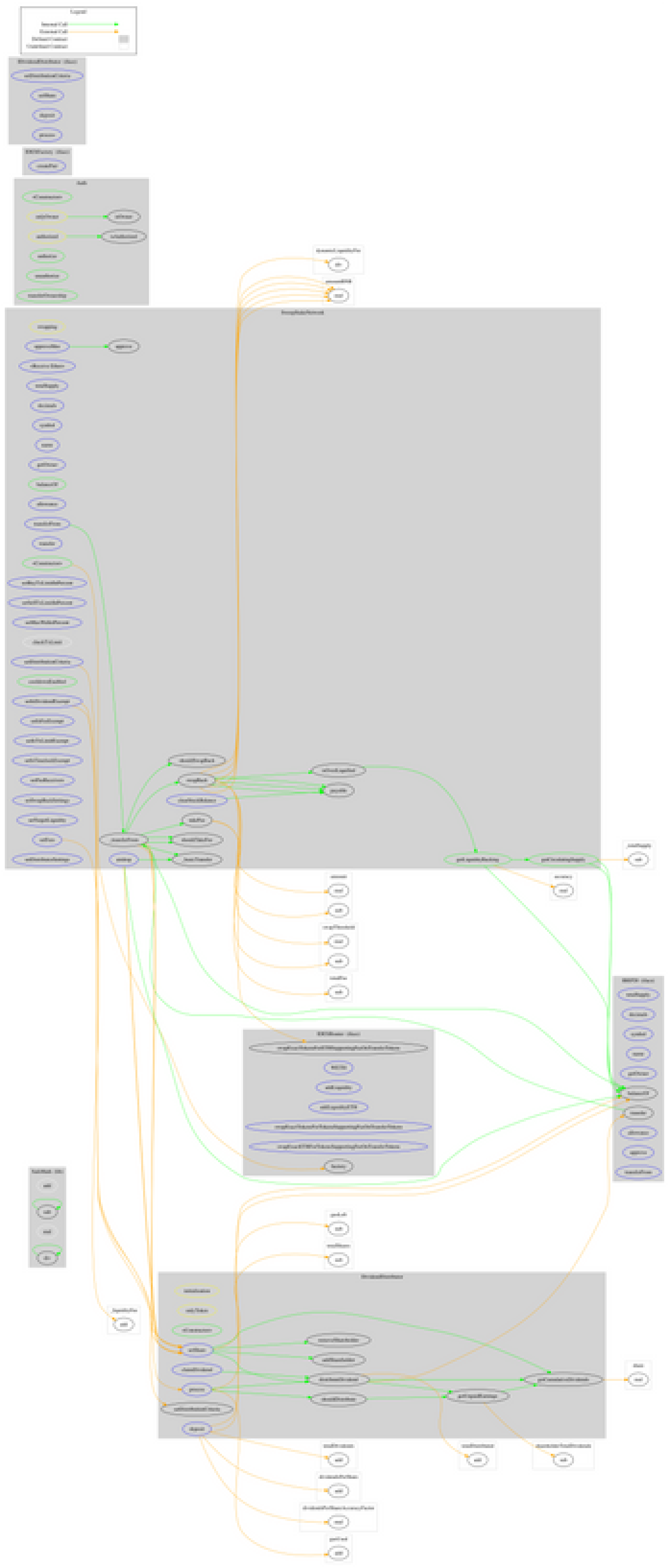
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
	<Constructor>	Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner
	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			

	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
SweepStakeNe	Implementation	IBEP20,		

twork		Auth		
	<Constructor>	Public	✓	Auth
	<Receive Ether>	External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setBuyTxLimitInPercent	External	✓	authorized
	setSellTxLimitInPercent	External	✓	authorized
	setMaxWalletPercent	External	✓	authorized
	_transferFrom	Internal	✓	
	checkTxLimit	Internal		
	_basicTransfer	Internal	✓	
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	setFees	External	✓	authorized
	clearStuckBalance	External	✓	authorized
	cooldownEnabled	Public	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setIsDividendExempt	External	✓	authorized
	setIsFeeExempt	External	✓	authorized
	setIsTxLimitExempt	External	✓	authorized
	setIsTimelockExempt	External	✓	authorized
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	setDistributionCriteria	External	✓	authorized

	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	airdrop	External	✓	onlyOwner

Contract Flow



Domain Info

Domain Name	sweepstake.network
Registry Domain ID	11bdd4dde11840ae9a9bc95af0f0b235-DONUTS
Creation Date	2022-07-12T16:07:39Z
Updated Date	2022-07-17T16:08:13Z
Registry Expiry Date	2023-07-12T16:07:39Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain has been created in 12 months before the creation of the audit.

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 stopping transactions, transferring the user's tokens, manipulating fees and transferring funds to the team's wallet. 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 provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io