



Cyberscope

Audit Report

StackPro

August 2022

Type BEP20

Network BSC TESTNET

Address 0xB04015ACC35a5233Ae10E7560c372eEb5dC58c37

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
ULTW - Unlimited Liquidity to Team Wallet	7
Description	7
Recommendation	8
BC - Blacklisted Contracts	9
Description	9
Recommendation	9
Contract Diagnostics	10
BLC - Business Logic Concern	11
Description	11
Recommendation	11
MAL - Misused Algorithmic Logic	12
Description	12
Recommendation	12
MTS - Manipulate Total Supply	13
Description	13
Recommendation	13
L02 - State Variables could be Declared Constant	14
Description	14

Recommendation	14
L04 - Conformance to Solidity Naming Conventions	15
Description	15
Recommendation	15
L05 - Unused State Variable	16
Description	16
Recommendation	16
L13 - Divide before Multiply Operation	16
Description	16
Recommendation	16
L14 - Uninitialized Variables in Local Scope	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	SPRO
Compiler Version	v0.7.4+commit.3f05b770
Optimization	1000 runs
Explorer	https://testnet.bscscan.com/address/0xB04015ACC35a5233Ae10E7560c372eEb5dC58c37
Symbol	SPRO
Decimals	18
Total Supply	400,000
Domain	https://stackpro.finance

Source Files

Filename	SHA256
contract.sol	b469b3d718f978e5cfe3aa40b71079800a1d03b110e87700ee9e552a5c1dde5c

Audit Updates

Initial Audit	5th August 2022
Corrected	10th August 2022

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#L810,814

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 `maxSellTransactionAmount` to one. As a result, the users will essentially not be able to sell their holdings.

```
require(amount <= maxSellTransactionAmount, "Error amount");
```

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 `onePercent` to one. As a result, the users will be able to sell only 1% of their holdings every one day.

```
uint256 onePercent = balanceOf(sender).mul(sellLimit).div(100); //Should use variable
require(amount <= onePercent, "ERR: Can't sell more than set %");

if( blkTime > tradeData[sender].lastTradeTime + TwentyFourhours) {
    tradeData[sender].lastTradeTime = blkTime;
    tradeData[sender].tradeAmount = amount;
}
else if( (blkTime < tradeData[sender].lastTradeTime + TwentyFourhours) && ((
blkTime > tradeData[sender].lastTradeTime)) ){
    require(tradeData[sender].tradeAmount + amount <= onePercent, "ERR: Can't sell more than 1% in One day");
    tradeData[sender].tradeAmount = tradeData[sender].tradeAmount + amount;
}
```

Recommendation

The contract could embody a check for not allowing setting the 'maxSellTransactionAmount' and 'onePercent' 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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L1074,1418

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 `withdrawAllToTreasury` and `payBiggestBuyerOutside` methods.

```
function withdrawAllToTreasury() external swapping onlyOwner {
    uint256 amountToSwap = _gonBalances[address(this)].div(
        _gonsPerFragment
    );
    require(
        amountToSwap > 0,
        "There is no spro token deposited in token contract"
    );
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = router.WETH();
    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        amountToSwap,
        0,
        path,
        treasuryReceiver,
        block.timestamp
    );
}
...
function payBiggestBuyerOutside(uint256 _hour) external onlyOwner {
    _checkAndPayBiggestBuyer(_hour);
}
```


Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token's price.

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.

BC - Blacklisted Contracts

Criticality	minor
Location	contract.sol#L636

Description

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

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

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	BLC	Business Logic Concern
●	MAL	Misused Algorithmic Logic
●	MTS	Manipulate Total Supply
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L05	Unused State Variable
●	L13	Divide before Multiply Operation
●	L14	Uninitialized Variables in Local Scope

BLC - Business Logic Concern

Criticality	minor
Location	contract.sol#L1318

Description

The firePitFee fee is conceptually meaning that the accumulated funds will be transferred to the dead wallet. The contract can change the 'firePit' address. As a result, the fees may not transfer to the dead addresses as expected.

```
firePit = 0x00000000000000000000000000000000dEaD;  
..  
firePit = _firePit;
```

Recommendation

The firePit address should not be able to change.

MAL - Misused Algorithmic Logic

Criticality	minor
Location	contract.sol#L704

Description

The algorithmic flow does not follow the required business logic. According to the statements, the forth branches will never executed since `deltaTimeFromInit` will always be greater than or equal to `7 * 365 days`

```
if (deltaTimeFromInit < (365 days)) {
    rebaseRate = rebaseRateFirstYear;
} else if (
    deltaTimeFromInit >= (365 days) &&
    deltaTimeFromInit < ((15 * 365 days) / 10)
) {
    rebaseRate = 2110000;
} else if (
    deltaTimeFromInit >= ((15 * 365 days) / 10) &&
    (deltaTimeFromInit >= (7 * 365 days))
) {
    rebaseRate = 140000;
} else if (deltaTimeFromInit >= (7 * 365 days)) {
    rebaseRate = 20000;
}
```

Recommendation

The algorithm should be reshaped so it will match to the business logic.

MTS - Manipulate Total Supply

Criticality	minor
Location	contract.sol#L721

Description

Owner is able to manipulate total supply. This change will have a direct impact on the token price and Market Cap.

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

Recommendation

The contract owner should carefully manage the adjustment of the circulating supply (increases or decreases), according to the token's price fluctuations.

L02 - State Variables could be Declared Constant

Criticality

minor

Location

contract.sol#L560,601,573,606,605

Description

Constant state variables should be declared constant to save gas.

```
MAX_SELL_LIMIT  
MIN_SELL_LIMIT  
DEAD  
CHECK_PRICE_TIME  
MAX_TOTAL_FEE
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L542,543,1322,1228,625,1300,601,734,1230,1131,1252,1263,621,627,1301,150,623,1447,1346,622,1303,1293,1135,585,541,1302,576,1144,1229,225,1418,1367,595,626,560,148,1299,181,1287,1248,569,1207,605,573,546,566,1368,1227,606,624,1267,608

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.

```
TwentyFourhours
_address
_initRebaseStartTime
_flag
MIN_SELL_LIMIT
_autoLiquidityReceiver
_rebaseRateFirstYear
rewardBuyerFee
_isFeeExempt
...
```

Recommendation

Follow the Solidity naming convention.

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

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L6

Description

There are segments that contain unused state variables.

```
MAX_INT256
```

Recommendation

Remove unused state variables.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L695,1018,1238,910

Description

Performing divisions before multiplications may cause lose of prediction.

```
times = deltaTime.div(rebaseCycle * 60)
_gonBalances[autoLiquidityReceiver] =
_gonBalances[autoLiquidityReceiver].add(gonAmount.mul(liquidityFee.sub(rewardB
uyerFee).div(feeDenominator)))
liquidityBalance = _gonBalances[pair].div(_gonsPerFragment)
amountToSwapTrUSD = treasuryFeeCollect.div(_gonsPerFragment)
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality

minor

Location

contract.sol#L698

Description

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

```
rebaseRate
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

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

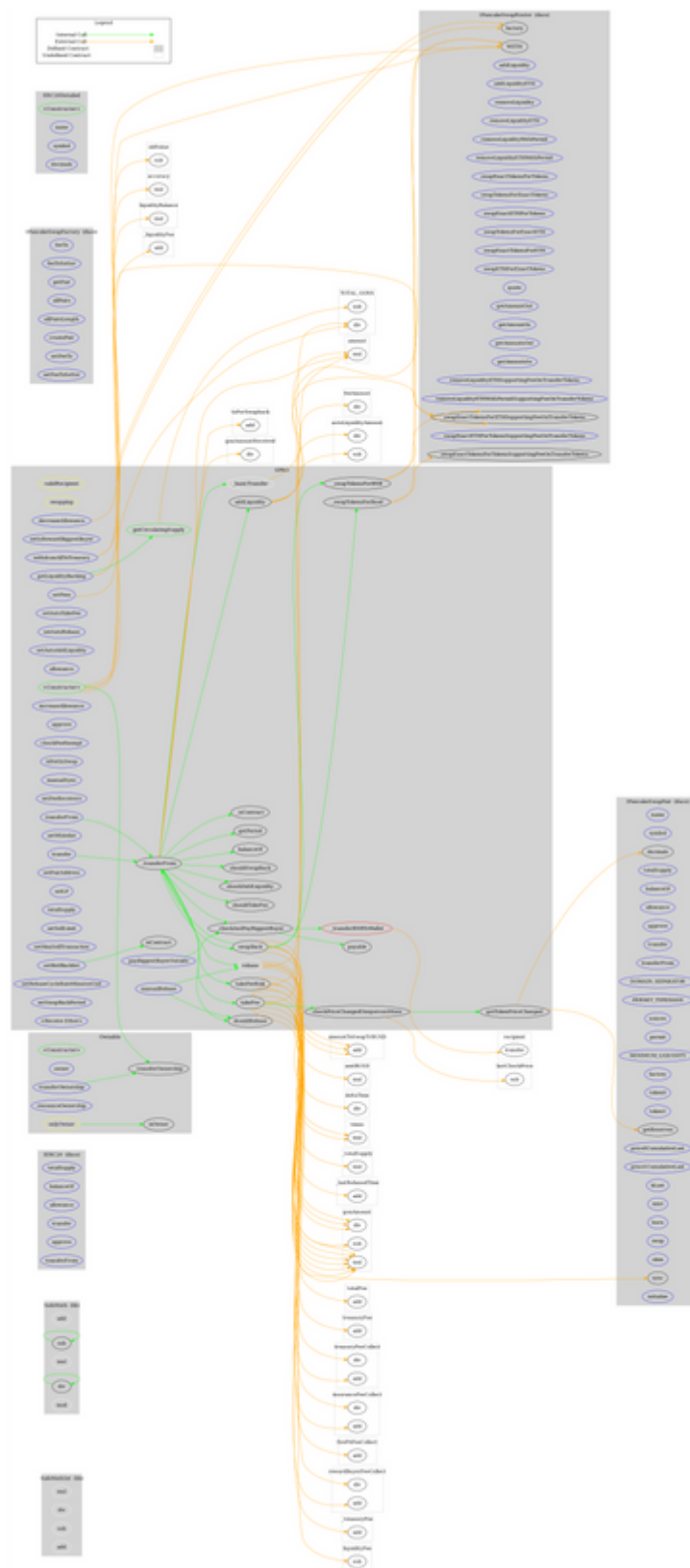
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External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

	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IPancakeSwapFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	External		-
	isOwner	Public		-
	renounceOwnership	External	✓	onlyOwner

	transferOwnership	External	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20Detailed	Implementation	IERC20		
	<Constructor>	Public	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
SPRO	Implementation	ERC20Detailed, Ownable		
	<Constructor>	Public	✓	ERC20Detailed
	rebase	Internal	✓	
	setIsRewardBiggestBuyer	External	✓	onlyOwner
	_isContract	Internal		
	transfer	External	✓	validRecipient
	transferFrom	External	✓	validRecipient
	_basicTransfer	Internal	✓	
	_transferFrom	Internal	✓	
	takeFeeRisk	Internal	✓	
	takeFee	Internal	✓	
	addLiquidity	Internal	✓	swapping
	swapBack	Internal	✓	swapping
	withdrawAllToTreasury	External	✓	swapping onlyOwner
	shouldTakeFee	Internal		
	shouldRebase	Internal		
	shouldAddLiquidity	Internal		
	shouldSwapBack	Internal		
	setAutoTakeFee	External	✓	onlyOwner
	setAutoRebase	External	✓	onlyOwner
	setAutoAddLiquidity	External	✓	onlyOwner
	allowance	External		-

	decreaseAllowance	External	✓	-
	increaseAllowance	External	✓	-
	approve	External	✓	-
	checkFeeExempt	External		-
	getCirculatingSupply	Public		-
	isNotInSwap	External		-
	manualSync	External	✓	-
	setFeeReceivers	External	✓	onlyOwner
	getLiquidityBacking	External		-
	setWhitelist	External	✓	onlyOwner
	setBotBlacklist	External	✓	onlyOwner
	setPairAddress	External	✓	onlyOwner
	setLP	External	✓	onlyOwner
	totalSupply	External		-
	balanceOf	Public		-
	isContract	Internal		
	setSellLimit	External	✓	onlyOwner
	setMaxSellTransaction	External	✓	onlyOwner
	setFees	External	✓	onlyOwner
	checkPriceChangedOnepercentMore	Internal	✓	
	getTokenPriceChanged	Internal		
	manualRebase	External	✓	-
	setRebaseCycleRateMinutesUnit	External	✓	onlyOwner
	_swapTokensForBNB	Private	✓	
	_swapTokensForBusd	Private	✓	
	_transferBNBToWallet	Private	✓	
	getPeriod	Public		-
	payBiggestBuyerOutside	External	✓	onlyOwner
	_checkAndPayBiggestBuyer	Private	✓	
	setSwapBackPeriod	External	✓	onlyOwner
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	stackpro.finance
Registry Domain ID	00f9830a709f4a66b01b7a616a239baf-DONUTS
Creation Date	2022-06-05T15:25:32Z
Updated Date	2022-07-01T17:52:29Z
Registry Expiry Date	2023-06-05T15:25:32Z
Registrar WHOIS Server	whois.porkbun.com
Registrar URL	http://porkbun.com
Registrar	Porkbun LLC
Registrar IANA ID	1861

The domain has been created in 10 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 funds to the team's wallet and blacklisting addresses. 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>