Cyberscope

Audit Report

Stepout

June 2022

Type	BEP20
Network	BSC
Address	0x20e53d8081f135e7332382501798b57544fc6327
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	StepOut
Compiler Version	v0.8.12+commit.f00d7308
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x20e53d8081f135e7332382501798b57544fc6327
Symbol	Sto
Decimals	18
Total Supply	100,000,000
Domain	step-out.app

Source Files

Filename	**SHA256**
contract.sol	4fbe8f9131f4188b62d84a1a9587ca6b25d40025c577c3a4fbc965430c036ed2

Audit Updates

Initial Audit	4th June 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#L452

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The contract owner can convert the contract into a honeypot and prevent users from selling by increasing the selling taxes (sellTaxes.marketing, sellTaxes.liquidity and sellTaxes.burn).

```
if(recipient == pair) {
            fee = amount * (sellTaxes.marketing + sellTaxes.liquidity) /
100;
            burnAmt = amount * sellTaxes.burn / 100;
        }
        else {
            fee = amount * (taxes.marketing + taxes.liquidity) / 100;
            burnAmt = amount * taxes.burn / 100;
        }

        //set fee to zero if fees in contract are handled or exempted
        if (swapping || excludedFromFees[sender] ||
excludedFromFees[recipient]) {
            fee = 0;
            burnAmt = 0;
        }


        //send fees if threshold has been reached
        //don't do this on buys, breaks swap
        if (swapEnabled && !swapping && sender != pair && fee > 0)
swapForFees();

        super._transfer(sender, recipient, amount - fee - burnAmt);
```

Recommendation

The contract could embody a check for not allowing setting the total sellTaxes more than a reasonable amount.

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#L1

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 `setTaxesFeePercent` and `setSellTaxes` functions with a high percentage value.

```
function setTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn)
external onlyOwner{
    taxes = Taxes(_marketing, _liquidity, _burn);
}
```

```
function setSellTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn)
external onlyOwner{
    sellTaxes = Taxes(_marketing, _liquidity, _burn);
}
```

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#L540, 480, 567, 571

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 setting a high fee to the swapThreshold variable.

```
function setSwapThreshold(uint256 new_amount) external onlyOwner {
    swapThreshold = new_amount;
}
```

```
function swapForFees() private inSwap {
uint256 contractBalance = balanceOf(address(this));
if (contractBalance >= swapThreshold) {
```

The contract can also transfer funds by calling the rescueBEP20 and rescueBNB functions.

```
function rescueBEP20(address tokenAddress, uint256 amount) external onlyOwner{
    IERC20(tokenAddress).transfer(owner(), amount);
}
```

```
function rescueBNB(uint256 weiAmount) external onlyOwner{
        payable(owner()).sendValue(weiAmount);
    }
```

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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	MC	Missing Check
	L01	Public Function could be Declared External
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination
	L13	Divide before Multiply Operation
	L15	Local Scope Variable Shadowing

MC - Missing Check

Criticality	minor
Location	contract.sol#L470, 478, 491

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.

```
if (swapEnabled && !swapping && sender != pair && fee > 0) swapForFees();
```

In the above code segment, the contract guarantees that at least one of the sellTaxes.marketing or sellTaxes.liquidity variables will have a non zero value. So, the next code segment will be executed without any problem.

```
function swapForFees() private inSwap {
        uint256 contractBalance = balanceOf(address(this));
        if (contractBalance >= swapThreshold) {

            // Split the contract balance into halves
            uint256 denominator = (sellTaxes.marketing + sellTaxes.liquidity)
* 2;
            uint256 tokensToAddLiquidityWith = contractBalance *
sellTaxes.liquidity / denominator;
```

But, in the following code segment, the execution would be reverted if the sellTaxes.marketing has a zero value.

```
uint256 deltaBalance = address(this).balance - initialBalance;
uint256 unitBalance= deltaBalance / (denominator - sellTaxes.liquidity);
```

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#L83,91,115,134,142,153,171,193,212,356,360

Description

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

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

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L57,59,378,540,544,548,557,563

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.

```
_address
_pair
_router
_burn
_liquidity
_marketing
new_amount
WETH
_allowances
...
```

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#L540

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.

```
swapThreshold = new_amount
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L278

Description

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

```
_burn
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L478

Description

Performing divisions before multiplications may cause lose of prediction.

```
unitBalance = deltaBalance / (denominator - sellTaxes.liquidity)
```

Recommendation

The multiplications should be prior to the divisions.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contract.sol#L544,548

Description

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

```
_burn
```

Recommendation

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

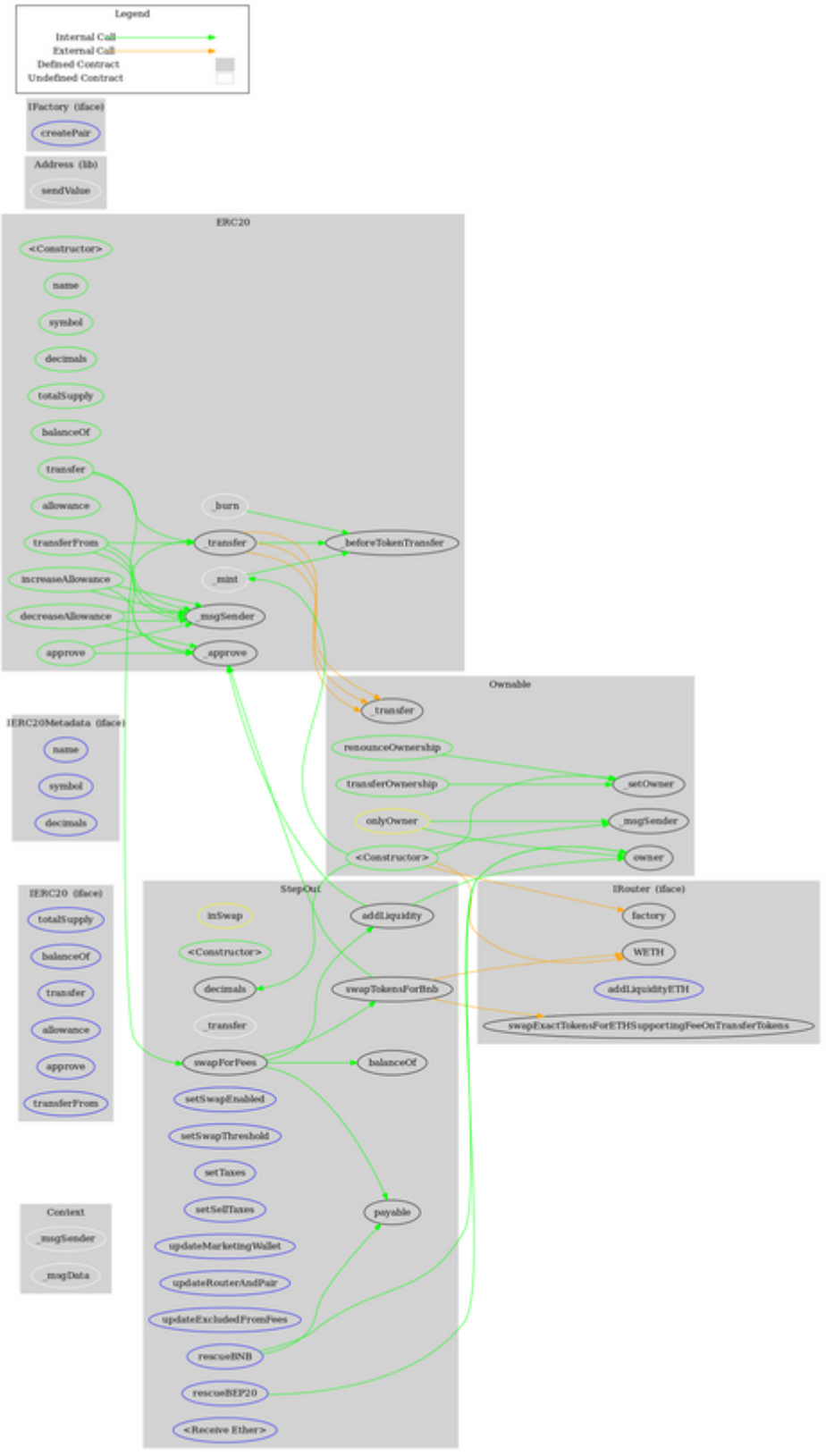
Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-

	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
Address	Library			
	sendValue	Internal	✓	
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
IFactory	Interface			
	createPair	External	✓	-
IRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
StepOut	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	decimals	Public		-
	_transfer	Internal	✓	

	swapForFees	Private	✓	inSwap
	swapTokensForBnb	Private	✓	
	addLiquidity	Private	✓	
	setSwapEnabled	External	✓	onlyOwner
	setSwapThreshold	External	✓	onlyOwner
	setTaxes	External	✓	onlyOwner
	setSellTaxes	External	✓	onlyOwner
	updateMarketingWallet	External	✓	onlyOwner
	updateRouterAndPair	External	✓	onlyOwner
	updateExcludedFromFees	External	✓	onlyOwner
	rescueBEP20	External	✓	onlyOwner
	rescueBNB	External	✓	onlyOwner
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	step-out.app
Registry Domain ID	490AADDF4-APP
Creation Date	2022-06-01T08:39:50Z
Updated Date	2022-06-01T08:40:19Z
Registry Expiry Date	2023-06-01T08:39:50Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	Namecheap Inc.
Registrar IANA ID	1068

The domain has been created 3 days before the creation of the audit. It will expire in 12 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 stopping transactions, 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