Cyberscope

Audit Report

Goat Wolf

August 2022

Type	BEP20
Network	BSC
Address	0x549305c52C6984057D299F51df20BdEdC2133BE2
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	GoatWolf
Compiler Version	v0.8.15+commit.e14f2714
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x549305c52C6984057D299F51df20BdEdC2133BE2
Symbol	GOATWOLF
Decimals	18
Total Supply	1,000,000,000

Source Files

Filename	**SHA256**
contract.sol	b55f531de45b0d8501ad30bb655ca5fcce5e711f1970a870ac69498fdaeb4d87

Audit Updates

Initial Audit	4th August 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#L558,561

Description

The contract owner has the authority to stop selling transactions for all users including the owner. The owner may take advantage of it by setting the liquidityFee and projectFee to values greater than the totalFee. As a result the methods swapExactTokensForETHSupportingFeeOnTransferTokens will revert and turn the contract into a honeypot.

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal  override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }
       uint256 contractTokenBalance = balanceOf(address(this));
        if( !swapping && contractTokenBalance > 0 && !automatedMarketMakerPairs[from])
        {
            swapping = true;

            uint256 liquidityTokens = (contractTokenBalance * liquidityFee) / totalFee;
            swapAndLiquidity(liquidityTokens);

            uint256 projectTokens = (contractTokenBalance * projectFee) / totalFee;
            swapAndSendMarketing(projectTokens);

            swapping = false;
        }
```

Recommendation

The contract could embody a check for not allowing setting the liquidityFee and projectFee to values greater than the totalFee.

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#L584

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 setFeeWallet function with a high percentage value.

```
function setFeeWallet(uint256 _liquidityFee, uint256 _projectFee) public onlyOwner
    {
        require(_liquidityFee >= 1, "Liquidity Fee be more than 1.");
        require(_projectFee >= 5, "Project Fee be more than 5.");
        liquidityFee = _liquidityFee;
        projectFee = _projectFee;
    }
```

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#L513

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 `claimStuckTokens` method.

```
function claimStuckTokens(address token) external onlyOwner {
        require(token != address(this), "Owner can not claim native tokens");
        if (token == address(0x0)) {
            payable(msg.sender).transfer(address(this).balance);
            return;
        }
        IERC20 ERC20token = IERC20(token);
        uint256 balance = ERC20token.balanceOf(address(this));
        ERC20token.transfer(msg.sender, balance);
}
```

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
	STC	Succeeded Transfer Check
	FSA	Fixed Swap Address
	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
	L11	Unnecessary Boolean equality

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L513

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function claimStuckTokens(address token) external onlyOwner {
        require(token != address(this), "Owner can not claim native tokens");
        if (token == address(0x0)) {
            payable(msg.sender).transfer(address(this).balance);
            return;
        }
        IERC20 ERC20token = IERC20(token);
        uint256 balance = ERC20token.balanceOf(address(this));
        ERC20token.transfer(msg.sender, balance);
    }
```

Recommendation

The contract should check if the result of the transfer methods is successful.

FSA - Fixed Swap Address

Criticality	minor
Location	contract.sol#L498

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.

```
 constructor () ERC20("Goat Wolf", "GOATWOLF")
   {
       IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
       uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
           .createPair(address(this), _uniswapV2Router.WETH());
```

Recommendation

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

MC - Missing Check

Criticality	critical
Location	contract.sol#L592

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.

The percentages can be set to values greater than 100.

```
function setPercentWallet(uint256[3] memory _pCent) public onlyOwner
    {
        pCent = _pCent;
    }
```

Recommendation

The contract should embody a check for not allowing setting the sum of percentages to be greater than 100%.

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

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L157,530,58,580,131,180,127,143,175,63,123,588,135,152,148

Description

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

```
allowance
approve
totalSupply
setPercentWallet
name
transferOwnership
increaseAllowance
transfer
symbol
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L473,474,476,472,481

Description

Constant state variables should be declared constant to save gas.

```
DEAD
marketingWallet
walletToWalletTransferWithoutFee
charityWallet
developerWallet
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L588,295,312,294,580,332,481

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.

```
DEAD
_liquidityFee
WETH
_projectFee
DOMAIN_SEPARATOR
MINIMUM_LIQUIDITY
PERMIT_TYPEHASH
_pCent
```

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#L580

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.

```
liquidityFee = _liquidityFee
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L224

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#L525

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)(isExcludedFromFees[account] != true,Account is already excluded)
```

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	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		

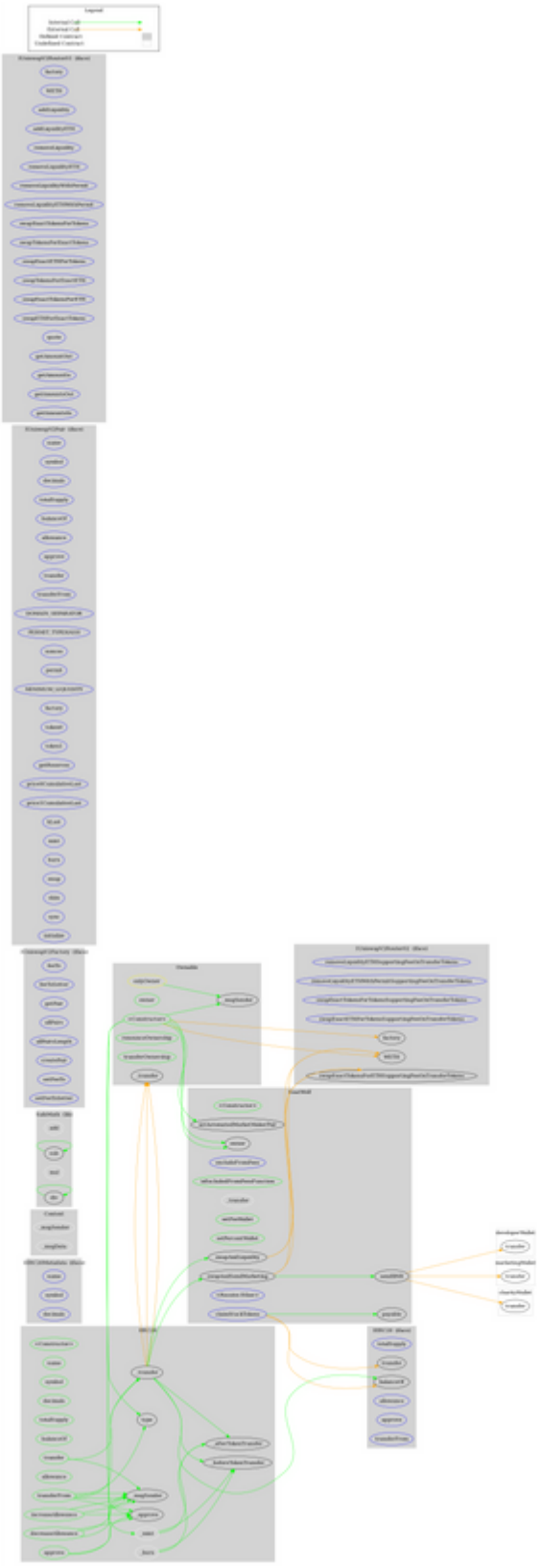
	div	Internal		
	div	Internal		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	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	✓	-
IUniswapV2Router01	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		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
GoatWolf	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	claimStuckTokens	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Private	✓	
	excludeFromFees	External	✓	onlyOwner
	isExcludedFromFeesFunction	Public		-

	_transfer	Internal	✓	
	setFeeWallet	Public	✓	onlyOwner
	setPercentWallet	Public	✓	onlyOwner
	swapAndLiquidity	Private	✓	
	swapAndSendMarketing	Private	✓	
	sendBNB	Private	✓	
	<Receive Ether>	External	Payable	-

Contract Flow



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