Cyberscope

Audit Report

ARES

July 2022

Type	ERC20
Network	ETH
Address	0xE0c7094dBCf14E7b3e8908A52da0CaEa801B0674
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Contract Review 4
Source Files 4
Audit Updates 4
Contract Analysis 5
ST - Stop Transactions 6
Description 6
Recommendation 6
ULTW - Unlimited Liquidity to Team Wallet 7
Description 7
Recommendation 7
BT - Burn Tokens 8
Description 8
Recommendation 8
BC - Blacklisted Contracts 9
Description 9
Recommendation 9
Contract Diagnostics 10
ZD - Zero Division 11
Description 11
Recommendation 11
MC - Missing Check 12
Description 12
Recommendation 12
BLC - Business Logic Concern 13
Description 13

Recommendation 13
CR - Code Repetition 14
Description 14
Recommendation 14
L01 - Public Function could be Declared External 15
Description 15
Recommendation 15
L02 - State Variables could be Declared Constant 16
Description 16
Recommendation 16
L04 - Conformance to Solidity Naming Conventions 17
Description 17
Recommendation 17
L07 - Missing Events Arithmetic 18
Description 18
Recommendation 18
L08 - Tautology or Contradiction 19
Description 19
Recommendation 19
L13 - Divide before Multiply Operation 20
Description 20
Recommendation 20
L15 - Local Scope Variable Shadowing 21
Description 21
Recommendation 21
Contract Functions 22
Contract Flow 25
Summary 26

Disclaimer 27
About Cyberscope 28

Contract Review

Contract Name	GODOFCRYPTO
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Licence	MIT
Explorer	https://etherscan.io/token/0xE0c7094dBCf14E7b3e8908A52da0CaEa801B0674
Symbol	ARES
Decimals	18
Total Supply	10,000,000,000
Domain	

Source Files

Filename	**SHA256**
contract.sol	6f19b4689e82b8f78cb703ab6eb41b17b1a6d3c67eb950a839a042415d626b80

Audit Updates

Initial Audit	21st 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	medium
Location	contract.sol#L674,L676

Description

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 `blockPenalty` to maximum amount.

```
if(takeFee){
        // bot/sniper penalty.  Tokens get transferred to marketing wallet to allow potential refund.
        if(isPenaltyActive() && automatedMarketMakerPairs[from]){
```

The owner can also stop buying transactions by setting sellTotalFees to zero.

```
tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees;
tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
tokensForDev += fees * sellDevFee / sellTotalFees;
```

Recommendation

It is recommended to add the necessary checks to keep blockPenalty and to sellTotalFees 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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L934

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 `withdrawStuckETH` method.

```
// withdraw ETH if stuck before launch
    function withdrawStuckETH() external onlyOwner {
        require(!tradingActive, "Can only withdraw if trading hasn't started");
        bool success;
        (success,) = address(msg.sender).call{value: address(this).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.

BT - Burn Tokens

Criticality	medium
Location	contract.sol#L874

Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `manualBurnLiquidityPairTokens` function. As a result the targeted contract address will lose the corresponding tokens.

The function `manualBurnLiquidityPairTokens` can burn up to 10% lp tokens every one hour.

```
function manualBurnLiquidityPairTokens(uint256 percent) external onlyOwner {
        require(block.timestamp > lastManualLpBurnTime + manualBurnFrequency , "Must wait for
cooldown to finish");
        require(percent <= 1000, "May not nuke more than 10% of tokens in LP");
        lastManualLpBurnTime = block.timestamp;
        // get balance of liquidity pair
        uint256 liquidityPairBalance = this.balanceOf(lpPair);
        // calculate amount to burn
        uint256 amountToBurn = liquidityPairBalance * percent / 10000;
        if (amountToBurn > 0){
            super._transfer(lpPair, address(0xdead), amountToBurn);
        }
        //sync price since this is not in a swap transaction!
        IDexPair pair = IDexPair(lpPair);
        pair.sync();
        emit ManualNukeLP(amountToBurn);
    }
```

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#L942

Description

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

```
function addBotToBlackList(address account) external onlyOwner() {
    require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not blacklist 
Uniswap router.');
    require(!_isBlackListedBot[account], "Account is already blacklisted");
    _isBlackListedBot[account] = true;
    _blackListedBots.push(account);
}
```

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
●	MC	Missing Check
●	BLC	Business Logic Concern
●	CR	Code Repetition
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L07	Missing Events Arithmetic
●	L08	Tautology or Contradiction
●	L13	Divide before Multiply Operation
●	L15	Local Scope Variable Shadowing

ZD - Zero Division

Criticality	minor
Location	contract.sol#L676

Description

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

The variable sellTotalFees can be set to zero with the function updateSellFees.

```
if(isPenaltyActive() && automatedMarketMakerPairs[from]){
            fees = amount * 99 / 100;
            tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
            tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees;
            tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
            tokensForDev += fees * sellDevFee / sellTotalFees;
}
```

Recommendation

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

MC - Missing Check

Criticality	minor
Location	contract.sol#L674,L850

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 blockPenalty can be set to the maximum and as a result the transaction will be reverted.

```
function isPenaltyActive() public view returns (bool) {
        return tradingActiveBlock >= block.number - blockPenalty;
}
/**
  *  transfer
  */
if(takeFee){
            // bot/sniper penalty.  Tokens get transferred to marketing wallet to allow potential refund.
            if(isPenaltyActive() && automatedMarketMakerPairs[from]){
```

Recommendation

It is recommended to add the necessary check to keep blockPenalty lower than block.number.

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

BLC - Business Logic Concern

Criticality	minor
Location	contract.sol#L674

Description

The business logic seems peculiar. The implementation may not follow the expected behaviour.

On buy transaction if there is an active penalty, sell fees are applied.

```
if(isPenaltyActive() && automatedMarketMakerPairs[from]){
    fees = amount * 99 / 100;
    tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
    tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees;
    tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
    tokensForDev += fees * sellDevFee / sellTotalFees;
}
// on sell
else if (automatedMarketMakerPairs[to] && sellTotalFees > 0){
    fees = amount * sellTotalFees / 100;
    tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
    tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees;
    tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
    tokensForDev += fees * sellDevFee / sellTotalFees;
}
// on buy
else if(automatedMarketMakerPairs[from] && buyTotalFees > 0) {
     fees = amount * buyTotalFees / 100;
     tokensForLiquidity += fees * buyLiquidityFee / buyTotalFees;
    tokensForBuyBack += fees * buyBuyBackFee / buyTotalFees;
    tokensForMarketing += fees * buyMarketingFee / buyTotalFees;
    tokensForDev += fees * buyDevFee / buyTotalFees;
}
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L675,L691,L800,L835

Description

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

```
fees = amount * 99 / 100;
tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees;
tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
tokensForDev += fees * sellDevFee / sellTotalFees;

address[] memory path = new address[](2);
path[0] = dexRouter.WETH();
path[1] = address(this);

// make the swap
dexRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amountInWei}(
    0, // accept any amount of Ethereum
    path,
    address(0xdead),
    block.timestamp
);
emit BuyBackTriggered(amountInWei);
```

Recommendation

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

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L182,156,203,577,152,168,198,555,148,935,177,173

Description

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

```
allowance
approve
isBot
name
setAutomatedMarketMakerPair
increaseAllowance
transfer
symbol
isExcludedFromFees
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L379

Description

Constant state variables should be declared constant to save gas.

```
manualBurnFrequency
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L307,891,541,532,818,141,810,140,411,352

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.

```
_frequencyInSeconds
deadAddress
_isExcludedmaxTxnAmount
_liquidityFee
_devFee
_buyBackAmount
_name
_marketingFee
_autoBuyBackEnabled
...
```

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#L810,818,891,518,541,532,511

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.

```
swapTokensAtAmount = newAmount
buyMarketingFee = _marketingFee
sellMarketingFee = _marketingFee
maxTxnAmount = newNum * (10 ** 18)
blockPenalty = _blockPenalty
lpBurnFrequency = _frequencyInSeconds
autoBuyBackFrequency = _frequencyInSeconds
```

Recommendation

Emit an event for critical parameter changes.

L08 - Tautology or Contradiction

Criticality	minor
Location	contract.sol#L818

Description

Detects expressions that are tautologies or contradictions. For instance, an uint variable will always be greater than or equal to zero.

```
require(bool,string)(_percent <= 1000 && _percent >= 0,Must set auto LP burn percent between 1% and 10%)
```

Recommendation

Fix the incorrect comparison by changing the value type or the comparison.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L581

Description

Performing divisions before multiplications may cause lose of prediction.

```
tokensForDev += fees * sellDevFee / sellTotalFees
tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees
fees = amount * buyTotalFees / 100
tokensForMarketing += fees * sellMarketingFee / sellTotalFees
tokensForBuyBack += fees * sellBuyBackFee / sellTotalFees
...
```

Recommendation

The multiplications should be prior to the divisions.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contract.sol#L451

Description

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

```
totalSupply
```

Recommendation

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

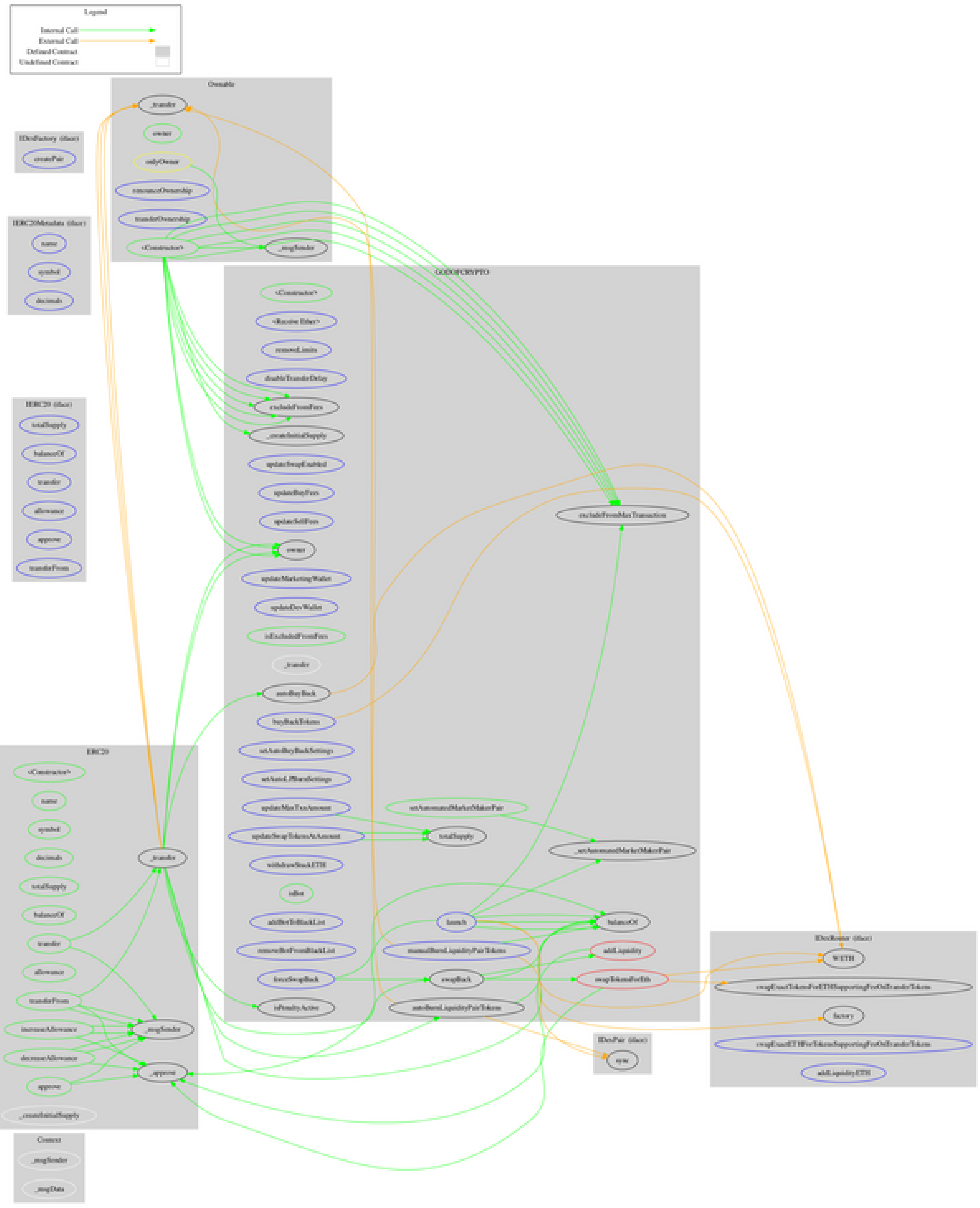
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IDexPair	Interface			
	sync	External	✓	-
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	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	External	✓	onlyOwner
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	addLiquidityETH	External	Payable	-
IDexFactory	Interface			
	createPair	External	✓	-
GODOFCRYPTO	Implementation	ERC20, Ownable		
	<Constructor>	Public	Payable	ERC20
	<Receive Ether>	External	Payable	-
	removeLimits	External	✓	onlyOwner
	disableTransferDelay	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner

	updateMaxTxnAmount	External	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	updateSwapEnabled	External	✓	onlyOwner
	updateBuyFees	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateMarketingWallet	External	✓	onlyOwner
	updateDevWallet	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	forceSwapBack	External	✓	onlyOwner
	buyBackTokens	External	✓	onlyOwner
	setAutoBuyBackSettings	External	✓	onlyOwner
	setAutoLPBurnSettings	External	✓	onlyOwner
	autoBuyBack	Internal	✓	
	isPenaltyActive	Public		-
	autoBurnLiquidityPairTokens	Internal	✓	
	manualBurnLiquidityPairTokens	External	✓	onlyOwner
	launch	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	isBot	Public		-
	addBotToBlackList	External	✓	onlyOwner
	removeBotFromBlackList	External	✓	onlyOwner

Contract Flow



Summary

There are some functions that can be abused by the owner like stop transactions, burn tokens, 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. There is also a limit of max 20% buy fees and a limit of max 25% sell fees.

Disclaimer

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