Cyberscope

Audit Report

Belifex

February 2023

Type	BEP20
Network	BSC
Address	0x75b2fdd95418e093fca7db858b36549e5e412076
Audited by	© cyberscope

Table of Contents

Review

Contract Name	DividendTokenWithAntibot
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Explorer	https://bscscan.com/address/0x75b2fdd95418e093fca7db858b36549e5e412076
Address	0x75b2fdd95418e093fca7db858b36549e5e412076
Network	BSC
Symbol	BEFX
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	19 Feb 2023

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	f0cbb88e6cbc994b565645eabd4320d27d529c7f1f4b3abb5fc263f3961c0a24
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	6e058aaee8c641107b209b62c34d484f2f125a44ecb66f7204a701614dfc1d68
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	a439a162881f7f36131b1fe307aa2a8dc98ac3f01ac121ff92fbbc25d0d216b5
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	68bcca423fc72ec9625e219c9e36306c726a347e43f3711467c579bd3f6500c8
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	db1d80b38061ba675444e6ad861a621d99666042950278d6cdeae9a108afdd17
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	44edc4d7099c781d11421cea2d82a52948e738f5f6191c8ad01dfc0f9858549c
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
@openzeppelin/contracts/proxy/Clones.sol	59fb8ba872adf90edb8a2e37ba16199c9a7625b8a155016e2b6aed0c592284d4
@openzeppelin/contracts/token/ERC20/ERC20.sol	f7831910f2ed6d32acff6431e5998baf50e4a00121303b27e974aab0ec637d79
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

@openzeppelin/contracts/utils/math/SafeMath.sol	15941f3904992a62ed117e93d9e2d5c4c22bd09a7ff97fdd5f49273cf09703ac
contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol	e29bf0b95c709f7c564e72de2b8ac8574692e1dbcfb7306c6908141b06efc2a6
contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol	8c1ad0ef06f5b7feb10d271e1e94ba01dc25f55cf4c53092ecf25a6065d14fe0
contracts/Tokens/interfaces/IUniswapV2Factory.sol	a63a844ad84f9df76c9822e73adf2f66b6e0c100eece0c4af5103734cb3f4698
contracts/Tokens/interfaces/IUniswapV2Pair.sol	d2a719db1ef447e334a57cba344e05ea85ec3fb6766ef717b4448fc4a5032634
contracts/Tokens/interfaces/IUniswapV2Router02.sol	5a8d4b8843a4ceb469d39bc30a3e7d528d8fabee8aa2f75fc00365d2c7bf90bb
contracts/Tokens/libs/IterableMapping.sol	1372c0015c643617e2ca65a6aaeccfba2839dcb0be3a58e96aa69fbd6046eaad
contracts/Tokens/libs/SafeMathInt.sol	7f1fe79198fb6d30490321221e656bbf8f767f724339abfb3c23469a7c1b571d
contracts/Tokens/libs/SafeMathUint.sol	22e1e7ff4e22dc8a599c9ed317acc2c70e213da90029ffc936eac5febf2caa34

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ST - Stops Transactions

Criticality	Medium
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L245
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the `maxWallet` or `maxTransactionAmount` to a very low value like 1. As a result, the transaction limit will be so decreased that it essentially not allow the user to trade.

```
function updateMaxWallet(uint256 _maxWallet) external onlyOwner {
    require(_maxWallet>0, "maxWallet>0");
    emit UpdateMaxWallet(_maxWallet, maxWallet);
    maxWallet = _maxWallet;
}

function updateMaxTransactionAmount(uint256 _maxTransactionAmount)
    external
    onlyOwner
{
    require(_maxTransactionAmount>0, "maxTransactionAmount>0");
    maxTransactionAmount = _maxTransactionAmount;
    emit UpdateMaxTransactionAmount(_maxTransactionAmount,
maxTransactionAmount);
}
```

Recommendation

The contract could embody a check for not allowing setting the `maxWallet` or `maxTransactionAmount` 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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol#L319
Status	Unresolved

Description

The contract sends funds to a `_marketingWalletAddress` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
function setMarketingWallet(address payable wallet) external onlyOwner {
    require(_marketingWalletAddress!=wallet, "already");
    emit MarketingWalletUpdated(_marketingWalletAddress, wallet);
    _marketingWalletAddress = wallet;
}
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be archived by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol#L239
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount > 0, "swapTokensAtAmount > 0");
    emit UpdateSwapTokensAtAmount(amount, swapTokensAtAmount);
    swapTokensAtAmount = amount;
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L30,57,245,251,285,325,332,333,354,355,376,377 contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol#L80,100,101,102,103,162,169,181,195,370
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
IPancakeCaller public constant pancakeCaller =
        IPancakeCaller(0x617715A9Bf6dD62D1Beb70F29914Fcf821933B39)
address public _marketingWalletAddress
uint256 _maxWallet
uint256 _maxTransactionAmount
address _baseTokenForPair
address _tokenForMarketingFee
uint16 _sellLiquidityFee
uint16 _buyLiquidityFee
uint16 _sellMarketingFee
uint16 _buyMarketingFee
uint16 _sellRewardFee
uint16 _buyRewardFee
uint256 internal constant magnitude = 2**128

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol#L214
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _transfer(
        address from,
        address to,
        uint256 value
    ) internal virtual override {
        require(false);
...
            .toInt256Safe();
        magnifiedDividendCorrections[from] =
magnifiedDividendCorrections[from]
            .add(_magCorrection);
        magnifiedDividendCorrections[to] =
magnifiedDividendCorrections[to].sub(
            _magCorrection
        );
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L12 - Using Variables before Declaration

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L653,654,655
Status	Unresolved

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or if the variable has been declared in a different scope. It is not a good practice to use a local variable before it has been declared.

```
uint256 iterations
uint256 claims
uint256 lastProcessedIndex
```

Recommendation

By declaring local variables before using them, contract ensures that it operates correctly. It's important to be aware of this rule when working with local variables, as using a variable before it has been declared can lead to unexpected behavior and can be difficult to debug.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L607,608,609,653,654,655
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 _liquidityFee
uint256 _marketingFee
uint256 _rewardFee
uint256 iterations
uint256 claims
uint256 lastProcessedIndex
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L287
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
baseTokenForPair=_baseTokenForPair
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenDividendTracker.sol#L2
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.13;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/Tokens/DividendTokens/DividendTokenWithAntibot.sol#L677,687
Status	Unresolved

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.

```
IERC20(rewardToken).transfer(_marketingWalletAddress, newBalance)
IERC20(baseTokenForPair).transfer(_marketingWalletAddress, newBalance)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal	✓	onlyInitializing
	__Ownable_init_unchained	Internal	✓	onlyInitializing
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
	_isConstructor	Private		
ERC20Upgradeable	Implementation	Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable		
	__ERC20_init	Internal	✓	onlyInitializing
	__ERC20_init_unchained	Internal	✓	onlyInitializing
	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	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20MetadataUpgradeable	Interface	IERC20Upgradeable		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20Upgradeable	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
AddressUpgradeable	Library			

	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Clones	Library			
	clone	Internal	✓	
	cloneDeterministic	Internal	✓	
	predictDeterministicAddress	Internal		
	predictDeterministicAddress	Internal		

ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-

	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
DividendPayingTokenInterface	Interface			
	dividendOf	External		-
	withdrawDividend	External	✓	-

DividendPayingTokenOptionalInterface	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
DividendPayingToken	Implementation	ERC20Upgradeable, OwnableUpgradeable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
	__DividendPayingToken_init	Internal	✓	onlyInitializing
	distributeCAKEDividends	Public	✓	onlyOwner
	withdrawDividend	Public	✓	-
	_withdrawDividendOfUser	Internal	✓	
	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_setBalance	Internal	✓	
DividendTokenDividendTracker	Implementation	OwnableUpgradeable, DividendPayingToken		
		Public	✓	-
	initialize	External	✓	initializer

	_transfer	Internal		
	withdrawDividend	Public		-
	excludeFromDividends	External	✓	onlyOwner
	isExcludedFromDividends	Public		-
	updateClaimWait	External	✓	onlyOwner
	updateMinimumTokenBalanceForDividends	External	✓	onlyOwner
	getLastProcessedIndex	External		-
	getNumberOfTokenHolders	External		-
	getAccount	Public		-
	getAccountAtIndex	Public		-
	canAutoClaim	Private		
	setBalance	External	✓	onlyOwner
	process	Public	✓	-
	processAccount	Public	✓	onlyOwner
IGemAntiBot	Interface			
	setTokenOwner	External	✓	-
	onPreTransferCheck	External	✓	-
IPancakeCaller	Interface			
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
DividendTokenWithAntibot	Implementation	ERC20, Ownable		
		Public	Payable	ERC20
		External	Payable	-
	setUsingAntiBot	External	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner
	updateMaxWallet	External	✓	onlyOwner

	updateMaxTransactionAmount	External	✓	onlyOwner
	updateDividendTracker	Public	✓	onlyOwner
	updateUniswapV2Pair	External	✓	onlyOwner
	updateUniswapV2Router	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	decimals	Public		-
	setMarketingWallet	External	✓	onlyOwner
	updateTokenForMarketingFee	External	✓	onlyOwner
	updateLiquidityFee	External	✓	onlyOwner
	updateMarketingFee	External	✓	onlyOwner
	updateRewardFee	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	excludeFromMaxTransactionAmount	External	✓	onlyOwner
	updateGasForProcessing	Public	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getClaimWait	External		-
	updateMinimumTokenBalanceForDividends	External	✓	onlyOwner
	getMinimumTokenBalanceForDividends	External		-
	getTotalDividendsDistributed	External		-
	isExcludedFromFees	Public		-
	withdrawableDividendOf	Public		-
	dividendTokenBalanceOf	Public		-
	excludeFromDividends	External	✓	onlyOwner
	isExcludedFromDividends	Public		-
	getAccountDividendsInfo	External		-
	getAccountDividendsInfoAtIndex	External		-
	processDividendTracker	External	✓	-
	claim	External	✓	-

	getLastProcessedIndex	External		-
	getNumberOfDividendTokenHolders	External		-
	_transfer	Internal	✓	
	swapAndSendToFee	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForBaseToken	Private	✓	
	swapTokensForCake	Private	✓	
	addLiquidity	Private	✓	
	swapAndSendDividends	Private	✓	
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	✓	-

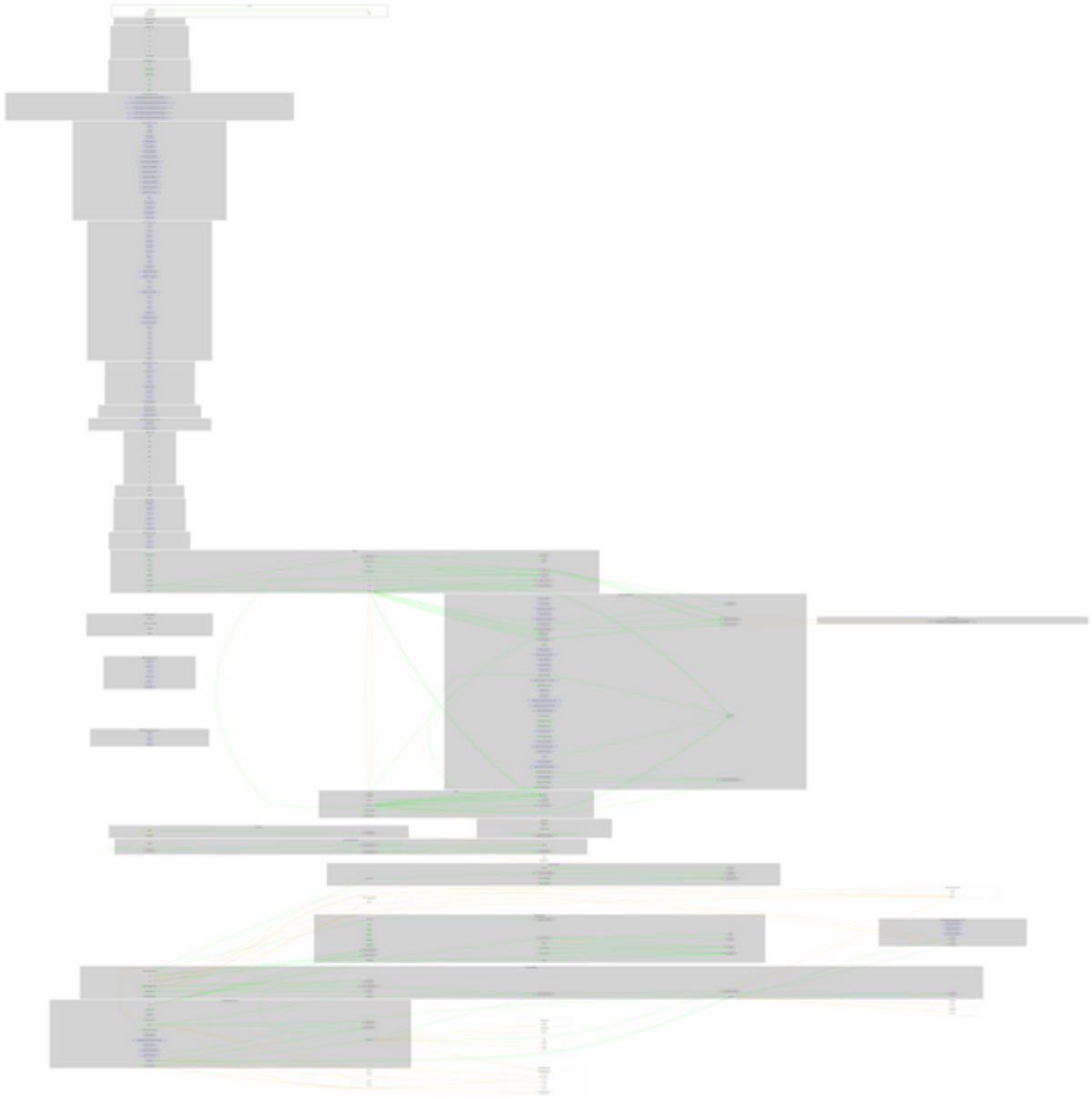
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	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	✓	-
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-

SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
	toUint256Safe	Internal		
SafeMathUint	Library			
	toInt256Safe	Internal		

Inheritance Graph

Flow Graph



Summary

There are some functions that can be abused by the owner like stop transactions. 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% fees.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io