



Cyberscope

Audit Report

ArmysGrok

December 2023

Network BSC

Address 0x749BA74C4cC8241251d3676165DdBB29504C5A52

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MEE	Missing Events Emission	Unresolved
●	AOI	Arithmetic Operations Inconsistency	Unresolved
●	DKO	Delete Keyword Optimization	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	4
Audit Updates	4
Source Files	4
Findings Breakdown	5
MEE - Missing Events Emission	6
Description	6
Recommendation	8
AOI - Arithmetic Operations Inconsistency	9
Description	9
Recommendation	9
DKO - Delete Keyword Optimization	10
Description	10
Recommendation	10
PVC - Price Volatility Concern	11
Description	11
Recommendation	12
FSA - Fixed Swap Address	13
Description	13
Recommendation	13
RSML - Redundant SafeMath Library	14
Description	14
Recommendation	14
IDI - Immutable Declaration Improvement	15
Description	15
Recommendation	15
L04 - Conformance to Solidity Naming Conventions	16
Description	16
Recommendation	16
L07 - Missing Events Arithmetic	18
Description	18
Recommendation	18
L09 - Dead Code Elimination	19
Description	19
Recommendation	20
L16 - Validate Variable Setters	21
Description	21

Recommendation	21
L17 - Usage of Solidity Assembly	22
Description	22
Recommendation	22
Functions Analysis	23
Inheritance Graph	30
Flow Graph	31
Summary	32
Disclaimer	33
About Cyberscope	34

Review

Contract Name	LiquidityGeneratorToken
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	200 runs
Explorer	https://bscscan.com/address/0x749ba74c4cc8241251d3676165ddbb29504c5a52
Address	0x749ba74c4cc8241251d3676165ddbb29504c5a52
Network	BSC
Symbol	ArmysGrok
Decimals	9
Total Supply	420,000,000,000,000

Audit Updates

Initial Audit	28 Dec 2023
---------------	-------------

Source Files

Filename	SHA256
LiquidityGeneratorToken.sol	1724eac36d3892bd38b0d2906dcc95930 443e7f66b307b3295bc1cad040ceac7

Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	12

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	12	0	0	0

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1186,1237
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.


```
function excludeFromReward(address account) public onlyOwner {
    // require(account !=
0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude
Uniswap router. ');
    require(!_isExcluded[account], "Account is already
excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] =
tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external
onlyOwner {
    ...
}

function setTaxFeePercent(uint256 taxFeeBps) external
onlyOwner {
    _taxFee = taxFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}

function setLiquidityFeePercent(uint256 liquidityFeeBps)
external
onlyOwner
{
    _liquidityFee = liquidityFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}

function setCharityFeePercent(uint256 charityFeeBps)
external onlyOwner {
    _charityFee = charityFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

AOI - Arithmetic Operations Inconsistency

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1240
Status	Unresolved

Description

The contract uses both the SafeMath library and native arithmetic operations. The SafeMath library is commonly used to mitigate vulnerabilities related to integer overflow and underflow issues. However, it was observed that the contract also employs native arithmetic operators (such as `+`, `-`, `*`, `/`) in certain sections of the code.

The combination of SafeMath library and native arithmetic operations can introduce inconsistencies and undermine the intended safety measures. This discrepancy creates an inconsistency in the contract's arithmetic operations, increasing the risk of unintended consequences such as inconsistency in error handling, or unexpected behavior.

```
_taxFee + _liquidityFee + _charityFee <= MAX_FEE,  
  
_tFeeTotal = _tFeeTotal.add(tAmount);
```

Recommendation

To address this finding and ensure consistency in arithmetic operations, it is recommended to standardize the usage of arithmetic operations throughout the contract. The contract should be modified to either exclusively use SafeMath library functions or entirely rely on native arithmetic operations, depending on the specific requirements and design considerations. This consistency will help maintain the contract's integrity and mitigate potential vulnerabilities arising from inconsistent arithmetic operations.

DKO - Delete Keyword Optimization

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1429
Status	Unresolved

Description

The contract resets variables to the default state by setting the initial values. Setting values to state variables increases the gas cost.

```
function removeAllFee() private {  
    _previousTaxFee = _taxFee;  
    _previousLiquidityFee = _liquidityFee;  
    _previousCharityFee = _charityFee;  
  
    _taxFee = 0;  
    _liquidityFee = 0;  
    _charityFee = 0;  
}
```

Recommendation

The team is advised to use the `delete` keyword instead of setting variables. This can be more efficient than setting the variable to a new value, using delete can reduce the gas cost associated with storing data on the blockchain.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1264,1471
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `numTokensSellToAddToLiquidity` 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 setSwapBackSettings(uint256 _amount) external
onlyOwner {
    require(
        _amount >= totalSupply().mul(5).div(10**4),
        "Swapback amount should be at least 0.05% of total
supply"
    );
    numTokensSellToAddToLiquidity = _amount;
    emit SwapAndLiquifyAmountUpdated(_amount);
}

...
uint256 contractTokenBalance = balanceOf(address(this));

bool overMinTokenBalance = contractTokenBalance >=
    numTokensSellToAddToLiquidity;

...
contractTokenBalance = numTokensSellToAddToLiquidity;
swapAndLiquify(contractTokenBalance);
```

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 exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1020
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
constructor() {  
    ...  
    IUniswapV2Router02 _uniswapV2Router =  
        IUniswapV2Router02(router_);  
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())  
        .createPair(address(this), _uniswapV2Router.WETH());  
  
    uniswapV2Router = _uniswapV2Router;  
    ...  
}
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L999,1001,1010,1016,1022
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_decimals  
_tTotal  
_charityAddress  
swapAndLiquifyEnabled  
uniswapV2Pair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L641,942,945,948,953,1264,1403,1407,1415
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.

```
function WETH() external pure returns (address);
uint256 public _taxFee
uint256 public _liquidityFee
uint256 public _charityFee
address public _charityAddress
uint256 _amount
```

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>.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1238,1249,1257
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_taxFee = taxFeeBps  
_liquidityFee = liquidityFeeBps  
_charityFee = charityFeeBps
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L441,469,494,504,523,537,556,566,583,593,610
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 isContract(address account) internal view returns
(bool) {
    // This method relies on extcodesize, which returns 0
    for contracts in
    // construction, since the code is only stored at the
    end of the
    // constructor execution.

    uint256 size;
    assembly {
        size := extcodesize(account)
    }
    return size > 0;
}

...
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L1041
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.

```
payable(serviceFeeReceiver_).transfer(serviceFee_)
```

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.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	LiquidityGeneratorToken.sol#L447,622
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {  
    size := extcodesize(account)  
}  
  
assembly {  
    let returndata_size := mload(returndata)  
    revert(add(32, returndata),  
    returndata_size)  
}
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	

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		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		

	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
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		-

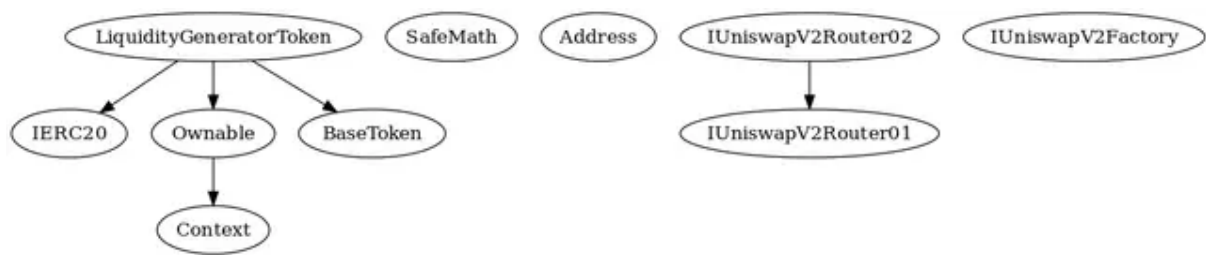
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
BaseToken	Implementation			

LiquidityGeneratorToken	Implementation	IERC20, Ownable, BaseToken		
		Public	Payable	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner

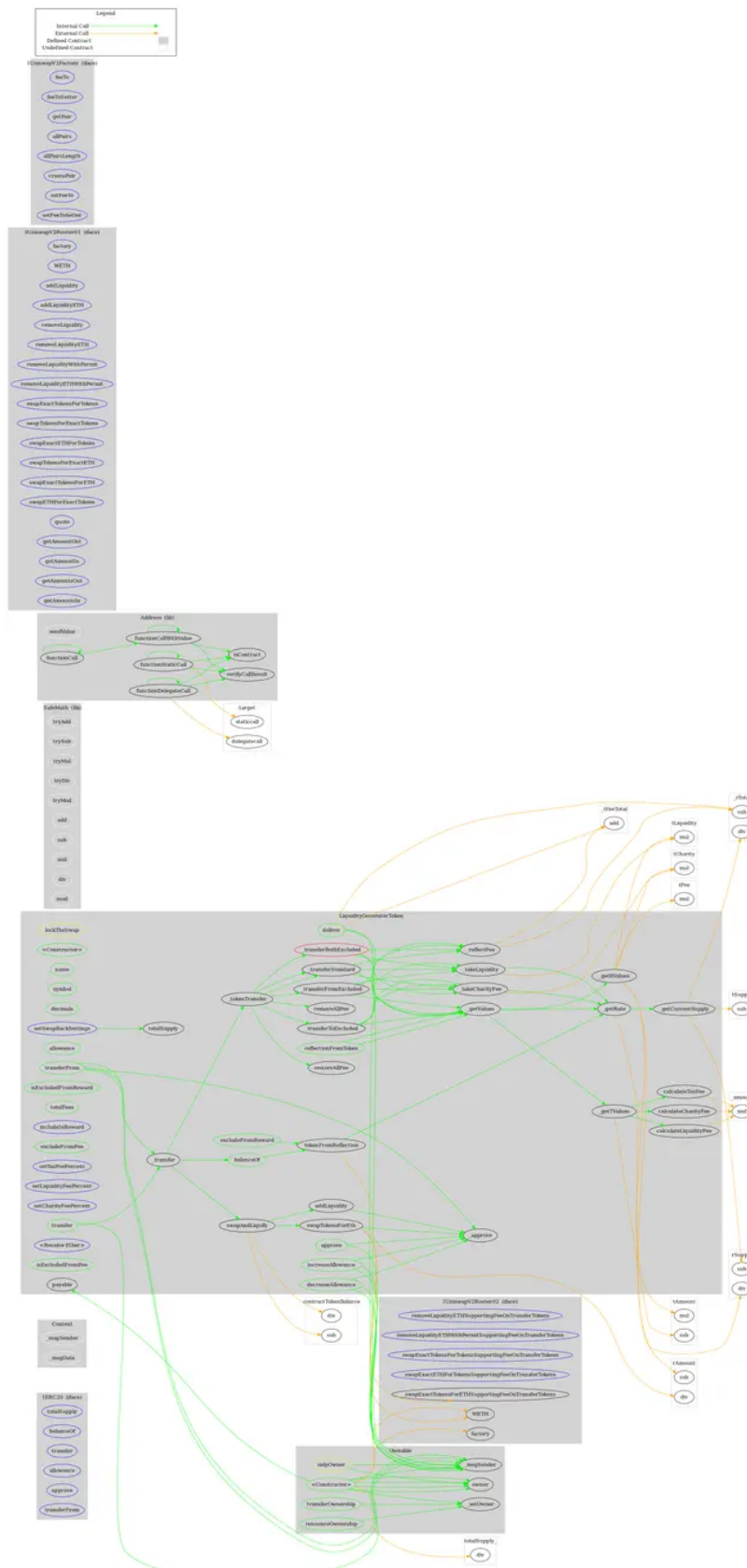
	setLiquidityFeePercent	External	✓	onlyOwner
	setCharityFeePercent	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
		External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTVValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	_takeCharityFee	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	calculateCharityFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	

	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Inheritance Graph



Flow Graph



Summary

ArmysGrok contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. ArmysGrok is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% 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>