Cyberscope

Audit Report

Liko Protocol

April 2023

Network	BSC
Address	0x16dbe5a6626402a04d6f4f6fadc2826780409f39
Audited by	© cyberscope

Table of Contents

Review

Contract Name	Token
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x16dbe5a6626402a04d6f4f6fadc2826780409f39
Address	0x16dbe5a6626402a04d6f4f6fadc2826780409f39
Network	BSC
Symbol	LIKO
Decimals	18
Total Supply	10,000,000,000

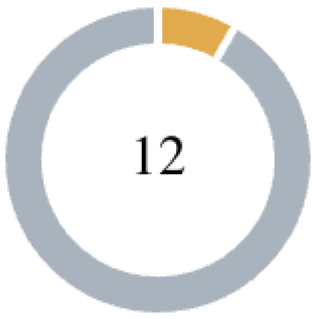
Audit Updates

Initial Audit	22 Apr 2023

Source Files

Filename	SHA256
Token.sol	5e8fcbf8e3d081f0ca2c482b55104367cf09e8117bc3e18f64ef6f9d84ea4b31

Findings Breakdown



●	Critical	0
●	Medium	1
●	Minor / Informative	11

	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	11	0	0	0

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	Unresolved
🔵	MT	Mints Tokens	Passed
🔵	BT	Burns Tokens	Passed
🟠	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Minor / Informative
Location	Token.sol#L553,560
Status	Unresolved

Description

The contract owner has the authority to stop the buys and transfers. The owner may take advantage of it by setting the `_maxWalletToken` to zero.

```
if (!ownerLeaders[sender] && recipient != address(this) &&
recipient != address(DEAD) && recipient != pair
    && recipient != marketingFeeReceiver && recipient !=
autoLiquidityReceiver){
    uint256 heldTokens = balanceOf(recipient);
    require((heldTokens + amount) <= _maxWalletToken,"Total Holding
is currently limited, you can not buy that much.");
}
```

The contract owner has the authority to limit the sales up to 1 every 10 minutes. The owner may take advantage of it by setting the `cooldownTimerInterval` to 599.

```
if (recipient == pair && buyCooldownEnabled &&
!isTimelockExempt[sender]) {
    require(cooldownTimer[sender] < block.number,"No consecutive
sells allowed. Please wait.");
    cooldownTimer[sender] = block.number + cooldownTimerInterval;
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxWalletToken` 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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	Token.sol#L723
Status	Unresolved

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 the `totalFee` to zero. As a result, the contract will swap the contract's accumulated tokens and send them to the `marketingFeeReceiver` address.

```
(bool tmpSuccess,) = payable(marketingFeeReceiver).call{value:
amountBNB, gas: 30000}("");
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Medium
Location	Token.sol#L502
Status	Unresolved

Description

The `ownerLeaded` role has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `devListAddress` function.

```
require(!_isDevlisted[recipient], "Devlisted address");
```

Recommendation

The team should carefully manage the private keys of the ownerLeaded roles. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	RTS	Reward Token Sanitisation	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

RTS - Reward Token Sanitisation

Criticality	Minor / Informative
Location	Token.sol#L535
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The reward token should not be allowed to be the native token address, since there is no pair between the BNB address and its self. Generally, the contract should not allow addresses that do not have a direct pair with the BNB token in the router's pair list.

```
function setRewardToken(address _rewardTokenAddress) external
onlyOwner {
    require(
        _rewardTokenAddress != DEAD
    && _rewardTokenAddress != pair
    && _rewardTokenAddress != owner
    );
    REWARD = _rewardTokenAddress;
    distributor.setRewardToken(_rewardTokenAddress);
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications. A recommended way is to check the router if the secific address has a valid pair with the BNB token address.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	Token.sol#L786
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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(bool _enabled, uint256 _amount)
external onlyOwner {
    swapEnabled = _enabled;
    swapThreshold = _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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Token.sol#L457,458,461
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
router
pair
distributor
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Token.sol#L224,384,385,386,387,394,424,425,437
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0x1cAcb4979dACAEF191b9A67341e739ce6997dEF5
uint256 _totalSupply = 10000000000* (10**_decimals)
uint256 public feeDenominator         = 100
uint256 public maxTxFee             = 20
uint256 public launchedAt
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L146,202,210,211,251,272,277,281,383,384,385,386,387,390,391,392,394,397,400,402,403,405,531,646,651,658,759,766,776,782,787,792
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);
address _token
IBEP20 REWARD = IBEP20(0)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address _address
address public REWARD = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0x1cAcb4979dACAEF191b9A67341e739ce6997dEF5
string constant _name = "Liko Protocol"
string constant _symbol = "LIKO"
uint8  public constant _decimals = 18

...
```

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	Token.sol#L252,734,762,767,788
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.

```
minPeriod = _minPeriod
_maxTxAmount = maxTxAmount
transferFee = _transferTaxRate
liquidityFee = _liquidityFee
targetLiquidity = _target
```

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.

L12 - Using Variables before Declaration

Criticality	Minor / Informative
Location	Token.sol#L703
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.

```
bool tmpSuccess
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Token.sol#L282,777,778
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.

```
WBNB = _address
autoLiquidityReceiver = _buyBackReceiver
marketingFeeReceiver = _marketingFeeReceiver
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Token.sol#L274,342,653
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.

```
REWARD.transfer(_address, balance)
REWARD.transfer(shareholder, amount)
IBEP20(_tokenAddress).transfer(address(msg.sender), _tokenAmount)
```

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**
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	ownerLeade	External	✓	ownerLeaded
	unOwnerLeade	External	✓	ownerLeaded
	isOwner	Public		-

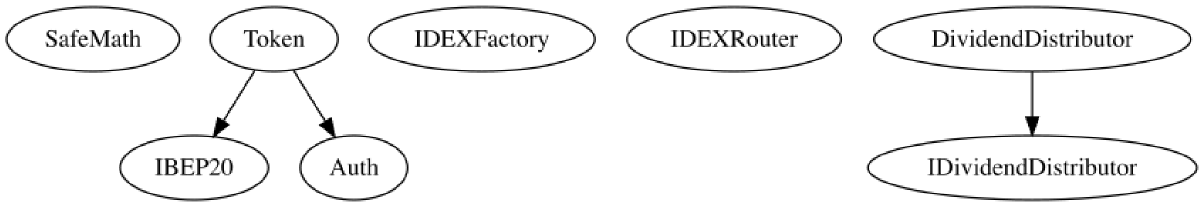
	isOwnerLeaded	Public		-
	transferOwnership	Public	✓	ownerLeaded
	renounceOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-

DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	clearStuckDividends	External	✓	onlyToken
	setRewardToken	External	✓	onlyToken
	setWBNB	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
Token	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-

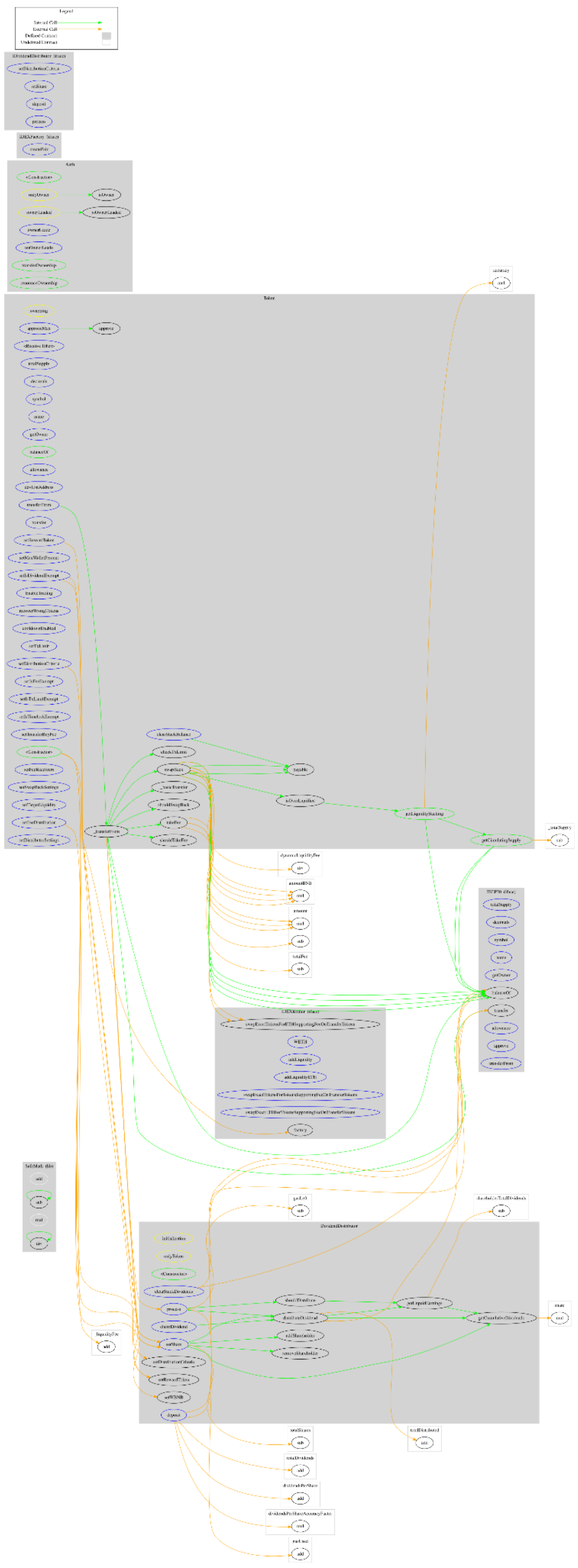
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	devListAddress	External	✓	ownerLeaded
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent	External	✓	onlyOwner
	setRewardToken	External	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	ownerLeaded
	EnableTrading	External	✓	onlyOwner
	recoverWrongTokens	External	✓	ownerLeaded
	cooldownEnabled	External	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setTxLimit	External	✓	onlyOwner

	setIsDividendExempt	External	✓	onlyOwner
	setIsFeeExempt	External	✓	onlyOwner
	setIsTxLimitExempt	External	✓	onlyOwner
	setIsTimelockExempt	External	✓	onlyOwner
	setTransferBuyFee	External	✓	onlyOwner
	setFeeDistribution	External	✓	onlyOwner
	setFeeReceivers	External	✓	ownerLeaded
	setSwapBackSettings	External	✓	onlyOwner
	setTargetLiquidity	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-

Inheritance Graph



Flow Graph



Summary

LIKO PROTOCOL contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stopping transactions and transferring funds to the team's wallet but not in a critical manner. 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

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