Cyberscope

# Audit Report

# Lava Token

April 2022

| | |
|---|---|
| File | LavaToken.sol |
| Commit | d59617e3ac107eea6d7601aac6e73e7f45ee00eb |
| Github | https://github.com/lavafinancial/LavaContracts |
| Audited by | © cyberscope |

# Table of Contents

Table of Contents 1
Contract Review 3
Audit Updates 3
Source Files 4
Contract Analysis 5
ST - Stop Transactions 6
Description 6
Recommendation 7
ELFM - Exceed Limit Fees Manipulation 8
Description 8
Recommendation 8
BC - Blacklisted Contracts 9
Description 9
Recommendation 9
Contract Diagnostics 10
L01 - Public Function could be Declared External 11
Description 11
Recommendation 11
L04 - Conformance to Solidity Naming Conventions 12
Description 12
Recommendation 12
Unit Test 13
Contract Functions 14
Contract Flow 18
Summary 19
Disclaimer 20

About Cyberscope 21

# Contract Review

| | |
|---|---|
| **Github** | LavaToken |
| **commit** | d59617e3ac107eea6d7601aac6e73e7f45ee00eb |
| **File** | LavaFinance.sol |

# Audit Updates

| | |
|---|---|
| **Initial Audit** | 9th April 2022 |
| **Corrected** | |

# Source Files

| Filename | SHA256 |
| --- | --- |
| @openzeppelin/contracts/access/Ownable.sol | 75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9 |
| @openzeppelin/contracts/security/Pausable.sol | 5b6abc290190f46b9941c674594eee083a3fe6b92d1828d0cfefacc94d1cac9a |
| @openzeppelin/contracts/token/ERC20/ERC20.sol | f7831910f2ed6d32acff6431e5998baf50e4a00121303b27e974aab0ec637d79 |
| @openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol | af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990 |
| @openzeppelin/contracts/token/ERC20/IERC20.sol | c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6 |
| @openzeppelin/contracts/token/ERC721/IERC721.sol | a88e8e63c7a737436f7ec62542620609ab07bb9a772e77146ed4dc98539e03d3 |
| @openzeppelin/contracts/utils/Context.sol | 1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a |
| @openzeppelin/contracts/utils/introspection/IERC165.sol | 701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5 |
| contracts/interfaces/JoeRouter.sol | dd1c237f78c9eea8d6378ebdaa91c3002e85bb0a1b70517413e158c5f4bf9b48 |
| contracts/LavaToken.sol | 5cc4866f96efb8f73fd7e21393aafe5eb432902d96856c2d786162aed7d69d71 |

# Contract Analysis

| 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#L157,152,131 |

## Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `salesTax` to a high value. This action may cause the contract to operate as a honeypot.

```
if (lpPairs[to] && !whitelistedTax[from] && from != address(this)) {
    // Sales Tax
    uint totalSalesTax = salesTax;
    if (nodeSalesTax > 0) {
        totalSalesTax += nodeSalesTax * IERC721(lavaFinance).balanceOf(from);
    }
    uint feeAmount = (amount * totalSalesTax) / 1e4;
    amount -= feeAmount;
    if (feeAmount > 0) {
        _transfer(from, address(this), feeAmount);
        swapLavaForTokens(feeAmount);
    }
}
```

The contract owner has the authority to stop transactions for all users after their initial transaction. The owner may take advantage of it by setting the `liquidityCooldown` to a high value.

```
if (block.number < liquidityCooldown + additionalLiquidityCooldown &&
blacklisted[to] == 0 && !whitelisted[to]) {
    blacklisted[to] = block.number;
    botsCount++;
}
```

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 `maxPerWallet` to a very low value.

```
require(!paused() || whitelisted[from] && whitelisted[to], "Paused");
if (!whitelisted[to]) {
```

```
    require(balanceOf(to) + amount <= maxPerWallet, "Max wallet amount
reached");
}
```

## Recommendation

The contract could embody checks for not allowing the contract owner to manipulate the configurable properties more 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.

# ELFM - Exceed Limit Fees Manipulation

| | |
|---|---|
| **Criticality** | critical |
| **Location** | contract.sol#L96 |

## 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 `setTaxes` function with a high percentage value.

```
function setTaxes(uint _transferTax, uint _salesTax, uint _nodeSalesTax)
external onlyOwner {
    transferTax = _transferTax;
    salesTax = _salesTax;
    nodeSalesTax = _nodeSalesTax;
}
```

## 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.

# BC - Blacklisted Contracts

| | |
|---|---|
| **Criticality** | critical |
| **Location** | contract.sol#L128 |

## Description

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

```
require(!isBlacklisted(from), "Not allowed");
```

## 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 |
| --- | --- | --- |
| | L01 | Public Function could be Declared External |
| | L04 | Conformance to Solidity Naming Conventions |

# L01 - Public Function could be Declared External

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contracts/LavaToken.sol#L119 |

## Description

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

```
burn
```

## Recommendation

Use the external attribute for functions never called from the contract

# L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contracts/LavaToken.sol#L65,78,83,90,96,102,107 |

## 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.

```
_cooldownBlocks
_maxPerWallet
_nodeSalesTax
_salesTax
_transferTax
_status
_lpPair
_lpRouter
_usdce
...
```

## Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

# Unit Test

- Test blacklist
- Test pausable
✓ Test max wallet (52ms)
✓ Test transfer tax (93ms)
- Test sales tax
✓ Test lp buy (120ms)

# Contract Functions

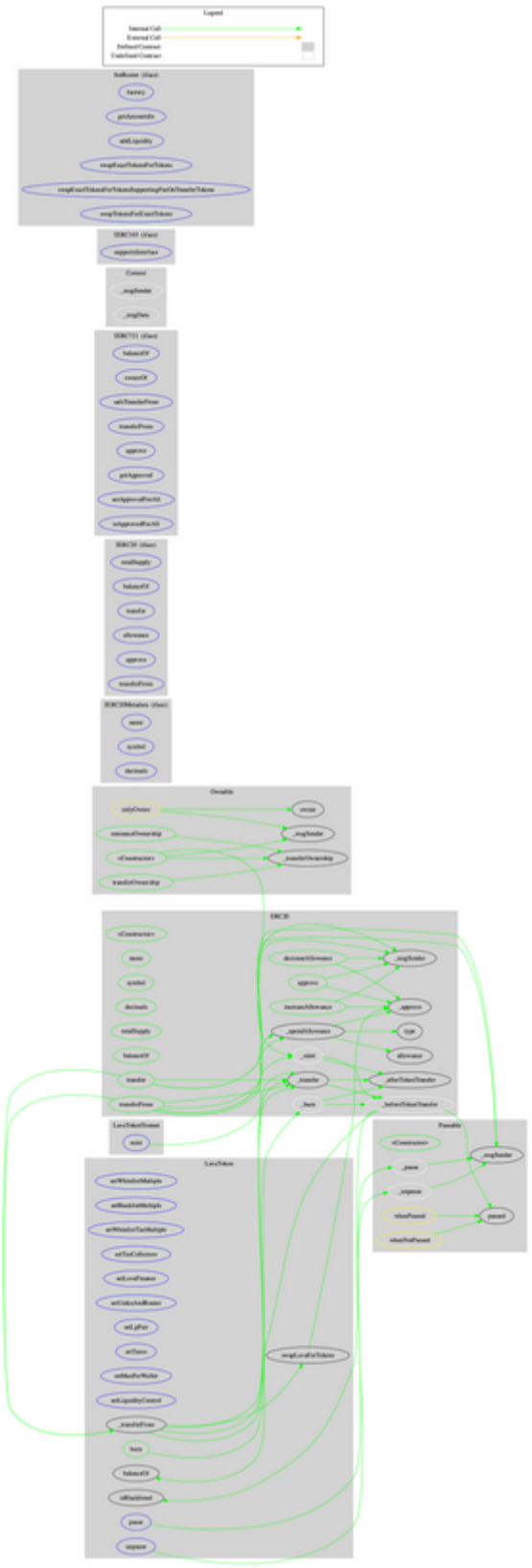
| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | <Constructor> | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |
| | | | | |
| **Pausable** | Implementation | Context | | |
| | <Constructor> | Public | ✓ | - |
| | paused | Public | | - |
| | _pause | Internal | ✓ | whenNotPaused |
| | _unpause | Internal | ✓ | whenPaused |
| | | | | |
| **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 | ✓ | |
| | _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 | ✓ | - |
| | | | | |
| **IERC721** | Interface | IERC165 | | |
| | balanceOf | External | | - |
| | ownerOf | External | | - |
| | safeTransferFrom | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | approve | External | ✓ | - |
| | getApproved | External | | - |
| | setApprovalForAll | External | ✓ | - |
| | isApprovedForAll | External | | - |
| | safeTransferFrom | External | ✓ | - |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |

| | | | | |
|---|---|---|---|---|
| **IERC165** | Interface | | | |
| | supportsInterface | External | | - |
| | | | | |
| **JoeRouter** | Interface | | | |
| | factory | External | | - |
| | getAmountsIn | External | | - |
| | addLiquidity | External | ✓ | - |
| | swapExactTokensForTokens | External | ✓ | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapTokensForExactTokens | External | ✓ | - |
| | | | | |
| **LavaToken** | Implementation | ERC20, Ownable, Pausable | | |
| | <Constructor> | Public | ✓ | ERC20 |
| | setWhitelistMultiple | External | ✓ | onlyOwner |
| | setBlacklistMultiple | External | ✓ | onlyOwner |
| | setWhitelistTaxMultiple | External | ✓ | onlyOwner |
| | setTaxCollectors | External | ✓ | onlyOwner |
| | setLavaFinance | External | ✓ | onlyOwner |
| | setUsdceAndRouter | External | ✓ | onlyOwner |
| | setLpPair | External | ✓ | onlyOwner |
| | setTaxes | External | ✓ | onlyOwner |
| | setMaxPerWallet | External | ✓ | onlyOwner |
| | setLiquidityControl | External | ✓ | onlyOwner |
| | pause | External | ✓ | onlyOwner |
| | unpause | External | ✓ | onlyOwner |
| | burn | Public | ✓ | - |
| | isBlacklisted | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | transferFrom | Public | ✓ | - |
| | transfer | Public | ✓ | - |
| | _transferFrom | Internal | ✓ | |
| | swapLavaForTokens | Private | ✓ | |

| | | | | |
|---|---|---|---|---|
| **LavaTokenTestnet** | Implementation | LavaToken | | |
| | mint | External | ✓ | - |

# Contract Flow



# Summary

There are some functions that can be abused by the owner, like manipulating fees, stopping transactions and blacklisting wallets. 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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io