HackSafe

# Smart Contract Security Audit Report

## Omlira

October 2022

Security Status

★★★★☆

www.hacksafe.io

# Audit Details

**Audited project**
Omlira

**Deployer address**
0x07cC8Fd23927AE95758Fa56C9B67A07Ac84C74ef

**Client contacts**
Omlira Team

**Blockchain**
Avalanche

**Website**
https://omlira.com/

# Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (HackSafe) owe no duty of care towards you or any other person, nor does HackSafe make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and HackSafe hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HackSafe hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HackSafe, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

# Procedure

**Step 1 - In-Depth Manual Review**

Manual line-by-line code reviews to ensure the logic behind each function is sound and safe from various attack vectors. This is the most important and lengthy portion of the audit process (as automated tools often cannot find the nuances that lead to exploits such as flash loan attacks).

**Step 2 - Automated Testing**

Simulation of a variety of interactions with your Smart Contract on a test blockchain leveraging a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

**Step 3 – Leadership Review**

The engineers assigned to the audit will schedule meetings with our leadership team to review the contracts, any comments or findings, and ask questions to further apply adversarial thinking to discuss less common attack vectors.

**Step 4 - Resolution of Issues**

Consulting with the team to provide our recommendations to ensure the code's security and optimize its gas efficiency, if possible. We assist project team's in resolving any outstanding issues or implementing our recommendations.

**Step 5 - Published Audit Report**

Boiling down results and findings into an easy-to-read report tailored to the project. Our audit reports highlight resolved issues and any risks that exist to the project or its users, along with any remaining suggested remediation measures. Diagrams are included at the end of each report to help users understand the interactions which occur within the project.

# Background

**HackSafe was commissioned by Omlira to perform an audit of smart contracts:**

- https://snowtrace.io/address/0x979fFD8eEd7a43629eA29581DF4Bfe2b3F224e47#code

**The purpose of the audit was to achieve the following:**

- Ensutre that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be understand the risk exposure of the  smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were  identified.

# Contract Details

## Token contract details for 10.10.2022

| | |
|---|---|
| **Token Type** | : ERC20 |
| **Contract name** | : LToken |
| **Contract address** | : 0x979fFD8eEd7a43629eA29581DF4Bfe2b3F224e47 |
| **Total supply** | : 418,322.192432 |
| **Token ticker** | : OML |
| **Decimals** | : 18 |
| **Token holders** | : 34 |
| **Transactions count** | : 1,354 |
| **Compiler version** | : v0.8.7+commit.e28d00a7 |
| **Contract deployer address** | : 0x07cC8Fd23927AE95758Fa56C9B67A07Ac84C74ef |
| **Owner address** | : 0x07cC8Fd23927AE95758Fa56C9B67A07Ac84C74ef |

# Social profiles

| | |
|---|---|
| **Telegram profile** | : https://t.me/omlira |
| **Coinmarketcap profile** | : https://coinmarketcap.com/currencies/omlira/ |
| **Coingecko profile** | : https://www.coingecko.com/en/coins/omlira/ |

# Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are “**Secure**”. This token contract does contain owner control, which do not make it fully decentralized as owner does have control over the smart contract.

| Insecure | Poor secured | Secure | Well-secured |
|---|---|---|---|

You are here 

We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 0 medium and 1 low and some very low-level issues.

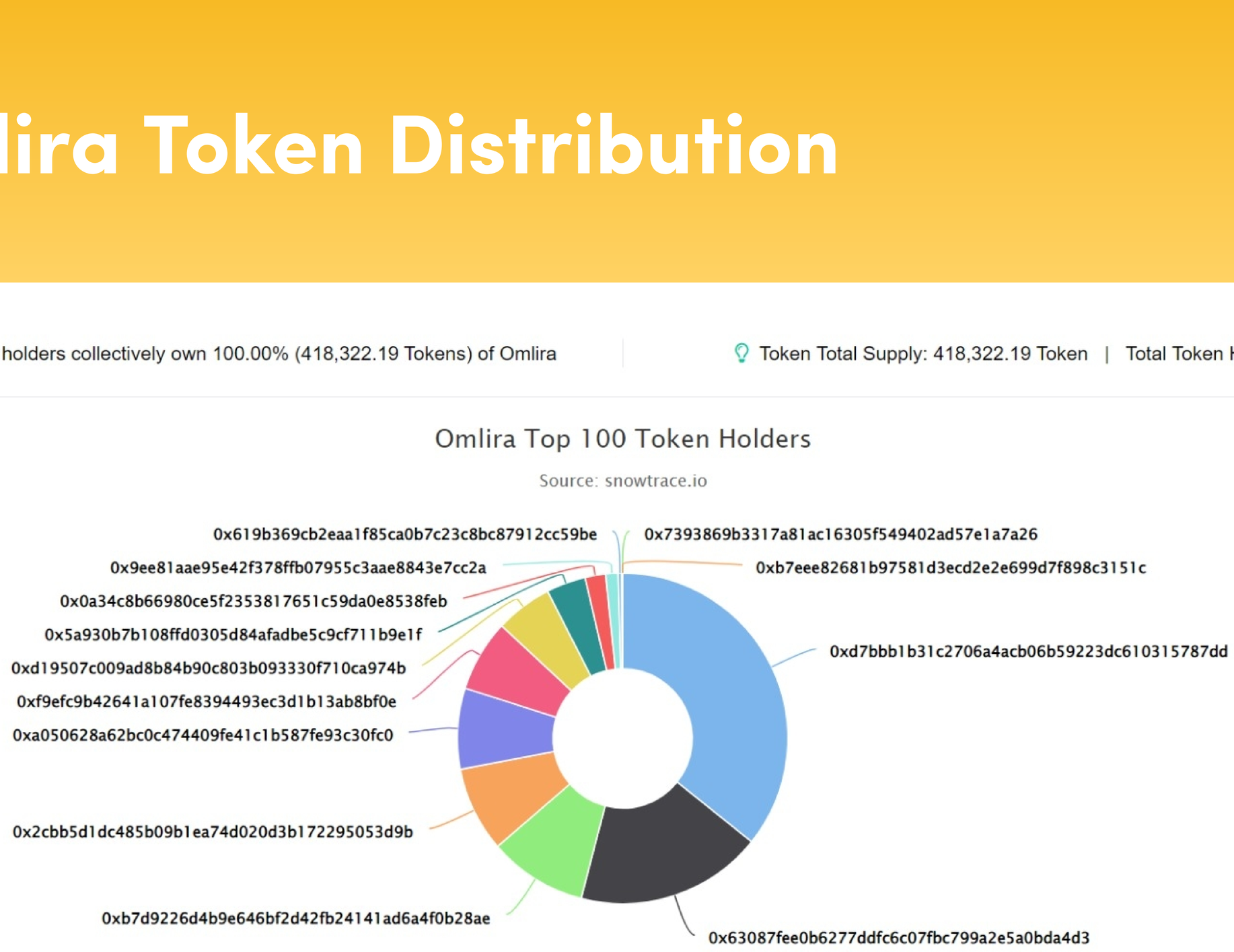
# Omlira Token Distribution

The top 100 holders collectively own 100.00% (418,322.19 Tokens) of Omlira

Token Total Supply: 418,322.19 Token | Total Token Holders: 34

Omlira Top 100 Token Holders

Source: snowtrace.io




## Omlira Top 20 Token Holders

(A total of 418,322.19 tokens held by the top 100 accounts from the total supply of 418,322.19 token)

| Rank | Address | Quantity (Token) | Percentage |
|---|---|---|---|
| 1 | 0xd7bbb1b31c2706a4acb06b59223dc610315787dd | 149,599.293436510556688467 | 35.7617% |
| 2 | 0x63087fee0b6277ddfc6c07fbc799a2e5a0bda4d3 | 76,469.778038115593596887 | 18.2801% |
| 3 | 0xb7d9226d4b9e646bf2d42fb24141ad6a4f0b28ae | 40,326.678059384439043685 | 9.6401% |
| 4 | 0x2cbb5d1dc485b09b1ea74d020d3b172295053d9b | 34,756.66645393849341406 | 8.3086% |
| 5 | 0xa050628a62bc0c474409fe41c1b587fe93c30fc0 | 33,171.069957983359935399 | 7.9296% |
| 6 | 0xf9efc9b42641a107fe8394493ec3d1b13ab8bf0e | 29,301.771344956659737509 | 7.0046% |
| 7 | 0xd19507c009ad8b84b90c803b093330f710ca974b | 23,313.9 | 5.5732% |
| 8 | 0x5a930b7b108ffd0305d84afadbe5c9cf711b9e1f | 16,130.018302330239085207 | 3.8559% |
| 9 | 0x0a34c8b66980ce5f2353817651c59da0e8538feb | 8,349.301273066585870346 | 1.9959% |
| 10 | 0x9ee81aae95e42f378ffb07955c3aae8843e7cc2a | 5,176.785519541211320338 | 1.2375% |
| 11 | 0x619b369cb2eaa1f85ca0b7c23c8bc87912cc59be | 1,384.984479029087702746 | 0.3311% |
| 12 | 0x3cf34d1f6b7839bc7b60b4f52dea328e186cc188 | 100 | 0.0239% |
| 13 | 0x76bc33a47af9c4fe11d9e7f1b2d2edfe969be968 | 92.682750475941761979 | 0.0222% |
| 14 | 0x777336ae2cef9ddc261a61a97cbfb4e0aa7d1329 | 61.812767453416978865 | 0.0148% |
| 15 | 0xcab005f52204903defaf35a3b7273850cd213467 | 60.72304855555980714 | 0.0145% |
| 16 | 0x49b19c98743341f6b1a7765f53aaf6374135ee3b | 15 | 0.0036% |
| 17 | 0xd3668aa1c016d87a4b3fbdfc33184c742d02dd61 | 5.61488366117824845 | 0.0013% |
| 18 | 0x94432ac892521e7470bc6124e3bd29d86647cca6 | 0.922676245468463181 | 0.0002% |
| 19 | 0x3ae0337a42e67669a1903c1a3dd0023b35a0eb9c | 0.869967102654210281 | 0.0002% |
| 20 | 0x144a5ee72ab076cfe564d3d2162b5387973ee70c | 0.80703504077098598 | 0.0002% |

# Contract functions details

+[Int] IAccessControl
    -[Ext] hasRole
    -[Ext] getRoleAdmin
    -[Ext] grantRole
    -[Ext] revokeRole
    -[Ext] renounceRole

+ Context
    -[Int] _msgSender
    -[Int] _msgData

+[Lib] Strings
    -[Int] toString
    -[Int] toHexString
    -[Int] toHexString

+[Int] IERC165
    -[Ext] supportsInterface

+ERC165 (IERC165)
    -[Pub] supportsInterface

+AccessControl (Context, IAccessControl, ERC165)
    -[Pub] supportsInterface
    -[Pub] hasRole
    -[Int] _checkRole
    -[Pub] getRoleAdmin
    -[Pub] grantRole #
      -modifiers: onlyRole
    -[Pub] revokeRole #
      -modifiers: onlyRole
    -[Pub] renounceRole #
    -[Int] _setupRole #
    -[Int] _setRoleAdmin #
    -[Pvt] _grantRole #
    -[Pvt] _revokeRole #

+Guarded (AccessControl)
    -<constructor>

# Contract functions details

-[Pub] pause #
  -modifiers: onlyAdmin nonPaused
-[Pub] unpause #
  -modifiers: onlyAdmin paused
  -[Pub] isPaused
  -[Pub] transferOwner #
    -modifiers: onlyOwner
  -[Pub] setRoleAdmin #
    -modifiers: onlyOwner

+[Int] IERC20
  -[Ext] totalSupply
  -[Ext] balanceOf
  -[Ext] transfer
  -[Ext] allowance
  -[Ext] approve
  -[Ext] transferFrom

+[Lib] Address
  -[Int] isContract
  -[Int] sendValue
  -[Int] functionCall
  -[Int] functionCall
  -[Int] functionCallWithValue
  -[Int] functionCallWithValue
  -[Int] functionStaticCall
  -[Int] functionStaticCall
  -[Int] functionDelegateCall
  -[Int] functionDelegateCall
  -[Int] verifyCallResult

+[Lib] SafeERC20
  -[Int] safeTransfer
  -[Int] safeTransferFrom
  -[Int] safeIncreaseAllowance
  -[Int] safeDecreaseAllowance
  -[Pvt] _callOptionalReturn

# Contract functions details

+TokenRecover (Guarded)
  -[Pub] recoverERC20 #
    -modifiers: onlyOwner

+Blacklistable (Guarded)
  -[Pub] isBlacklisted
  -[Pub] blacklist #
    -modifiers: onlyBlacklister
  -[Pub] unBlacklist #
    -modifiers: onlyBlacklister
  -[Ext] updateBlacklister #
    -modifiers: onlyOwner

+[Int] IERC20Metadata (IERC20)
  -[Ext] name
  -[Ext] symbol
  -[Ext] decimals

+ERC20 (Context, IERC20, IERC20Metadata)
  -<constructor>
  -[Pub] name
  -[Pub] symbol
  -[Pub] decimals
  -[Pub] totalSupply
  -[Pub] balanceOf
  -[Pub] transfer #
  -[Pub] allowance
  -[Pub] approve #
  -[Pub] transferFrom #
  -[Pub] increaseAllowance #
  -[Pub] decreaseAllowance #
  -[Int] _transfer #
  -[Int] _mint #
  -[Int] _burn #
  -[Int] _approve #
  -[Int] _beforeTokenTransfer #
  -[Int] _afterTokenTransfer #

# Contract functions details

+[Int] ILToken (IERC20)
  -[Ext] masterMint
  -[Ext] mintTo
  -[Ext] burnFrom

+LToken (Guarded, TokenRecover, Blacklistable, ERC20, ILToken)
  -<constructor>
  -[Pub] decimals
  -[Pub] masterMint #
    -modifiers: onlyOwner
  -[Pub] mintTo #
    -modifiers: onlyMinter, nonPaused, notBlacklisted
  -[Pub] burnFrom #
    -modifiers: nonPaused
  -[Pub] transfer #
    -modifiers: nonPaused
  -[Pub] transferFrom #
    -modifiers: nonPaused

($) = payable function
# = non-constant function

# Issues Checking Status

| No. | Title | Status |
|---|---|---|
| 1. | Unlocked Compiler Version | Low issue |
| 2. | Missing Input Validation | Passed |
| 3. | Race conditions and Reentrancy. Cross-function race conditions. | Passed |
| 4. | Possible delays in data delivery | Passed |
| 5. | Oracle calls. | Passed |
| 6. | Timestamp dependence. | Passed |
| 7. | Integer Overflow and Underflow | Passed |
| 8. | DoS with Revert. | Passed |
| 9. | DoS with block gas limit. | Passed |
| 10. | Methods execution permissions. | Passed |
| 11. | Economy model of the contract. | Passed |
| 12. | Private use data leaks. | Passed |
| 13. | Malicious Event log. | Passed |
| 14. | Scoping and Declarations. | Passed |
| 15. | Uninitialized storage pointers. | Passed |
| 16. | Arithmetic accuracy. | Passed |
| 17. | Design Logic. | Passed |
| 18. | Safe Open Zeppelin contracts implementation and usage. | Passed |
| 19. | Incorrect Naming State Variable | Passed |
| 20. | Too old version | Passed |

# Severity Definitions

| Risk Level | Description |
| --- | --- |
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution. |

# Security Issues

✓ **Critical Severity Issues**

No critical severity issue found.

✓ **High Severity Issues**

No high severity issues found.

✓ **Medium Severity Issues**

No medium severity issue found.

✓ **Low Severity Issues**

One low severity issue found..

## 1. Unlocked Compiler Version.

- **Description**

  The contract utilizes an unlocked compiler version. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

- **Recommendation**

  It is advisable that the compiler version is alternatively locked at the lowest version possible so that the contract can be compiled. For example, for version 0.8.0 the contract should contain the following line:

  pragma solidity 0.8.7;

# Centralization

## Owner Privileges :

- Omlira Contract:
    - Owner can transfer ownership.
    - Owner can mint new tokens.
    - Owner can set admin role.
    - Owner can recover ERC20.
    - Owner can update blacklist.

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble as smart contract ownership has not been renounced. Following are Admin functions:

- Transferowner
- Setroleadmin
- Recovererc20
- Updateblacklister
- Mastermint

# Conclusion

Smart contract contains low severity issues! The further transfer and operations with the fund raised are not related to this particular contract.

HackSafe note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.