FAIRYPROOF

Lucrar Token

AUDIT REPORT

Version 1.0.0

Serial No. 2022021000022013

Presented by Fairyproof

Feb 10, 2022

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Lucrar project's token contract.

Audit Start Time:

Feb 8, 2022

Audit End Time:

Feb 9, 2022

Project Token's Name:

Lucrar

Audited Source Files' Onchain Address:

https://bscscan.com/address/0x1510211e6dc81f5724a1beca33c5ac70dcca6ce0#code

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire code-base horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the Lucrar team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from off-chain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

FAIRYPROOF

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyper-linked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— The Lucrar Team's Consent/Acknowledgement:

The audited materials of the project including but not limited to the documents, home site, source code, etc are all developed, deployed, managed, and maintained outside Mainland CHINA.

The members of the team, the foundation, and all the organizations that participate in the audited project are not Mainland Chinese residents.

The audited project doesn’t provide services or products for Mainland Chinese residents.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The code-base was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Fairyproof describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following sources of truth about how the token issurance function should work:

https://whitepaper.lucrar.pt/

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Lucrar team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2022021000022013	Fairyproof Security Team	Feb 8, 2022 - Feb 9, 2022	Passed

Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, no issues were discovered.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Major functions of audited code

The audited code mainly implements a token issurance function and here are the details:

Token Name: Lucrar

Token Symbol: LCR

Token Precision: 9

Max Supply: 100,000,000

Mint/Burn: No additonal mintage, no burn

Transfer Pause/Freeze: Transfer cannot be paused or frozen.

Transaction Charge in Transfer: No

Misc: No

04. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Re-entrancy Attack
- Replay Attack
- Reordering Attack
- Miner's Advantage
- Rollback Attack
- DDos Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Integer Overflow/Underflow
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Tx.origin
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issurance
- Admin Rights
- Inappropriate Proxy Design
- Inappropriate Use of Slots
- Asset Security
- Contract Upgrade/Migration
- Code Improvement
- Misc

05. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

06. Major areas that need attention

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Integer Overflow/Underflow

We checked all the code sections, which had arithmetic operations and might introduce integer overflow or underflow if no safe libraries were used. All of them used safe libraries.

In our initial review we found neither line 402 `_totalSupply += amount;` nor line 403 `_balances[account] += amount;` used safemath library. This might cause overflow. Here was the code section:

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);
    _totalSupply += amount;
    _balances[account] += amount;
    emit Transfer(address(0), account, amount);

    _afterTokenTransfer(address(0), account, amount);
}
```

Recommendation:

Consider using SafeMath.

Status:

It has been confirmed by the Lucrar team, however the contract has been deployed and the coin will not have additional mintage, therefore the issue will not be triggered.

We reviewed again and didn't find new issues or risks in these functions or areas at the time of writing.

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by "owner".

We didn't find issues or risks in these functions or areas at the time of writing.

- Token Issuance

We checked whether or not the contract files could mint tokens at will.

We didn't find issues or risks in these functions or areas at the time of writing.

- State Update

We checked some key state variables which should only be set at initialization.

We didn't find issues or risks in these functions or areas at the time of writing.

- Asset Security

We checked whether or not all the functions that transfer assets were safely hanlded.

We didn't find issues or risks in these functions or areas at the time of writing.

- Unused Variables or Functions

FAIRYPROOF

We checked whether or not there were unused variables or functions which could be removed.

In our initial review we found the modifier defined in line 584 was unused. Here was the code section:

```
modifier zeroBalanceAndInsufficientBalance(uint256 value) {
    uint256 lcrBalance = balanceOf(address(msg.sender));
    require (lcrBalance > 0, "zero balance");
    require (lcrBalance >= value, "insufficient balance");
    _;
}
```

Recommendation:

Consider removing the modifier。

Status:

It has been confirmed by the Lucrar team, however the contract has been deployed and the coin has been issued, therefore the issue will not be triggered.

This was not a risk.

- Code Improvement

We checked whether or not there was code which could be improved to reduce gas consumption.

In our initial review we found in line 558 `if (i < 80) releaseDone[i] = false;`, the default value of the array was `false`, therefore this line could be removed. Here was the code section:

```
for (uint i= 0; i < 81; i++)
{
    releasePeriods[i] = startBlock + i * intervalBlocksQuarter;
    if (i < 80) releaseDone[i] = false;
}
```

Recommendation:

It has been confirmed by the Lucrar team, however the contract has been deployed and the coin has been issued, therefore the issue will not be triggered.

This was not a risk.

- Miscellaneous

The Fairyproof team didn't find issues or risks in other functions or areas at the time of writing.

07. List of issues by severity

-N/A

FAIRYPROOF

08. Issue descriptions

- N/A

FAIRYPROOF

09. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- N/A

FAIRYPROOF

FAIRYPROOF

FAIRYPROOF

FAIRYPROOF

https://medium.com/@FairyproofT

https://twitter.com/FairyproofT

https://www.linkedin.com/company/fairyproof-tech

https://t.me/Fairyproof_tech

Reddit: https://www.reddit.com/user/FairyproofTech