



LYOPAY

SMART CONTRACT AUDIT



March 15th 2022 | v. 2.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.



TECHNICAL SUMMARY

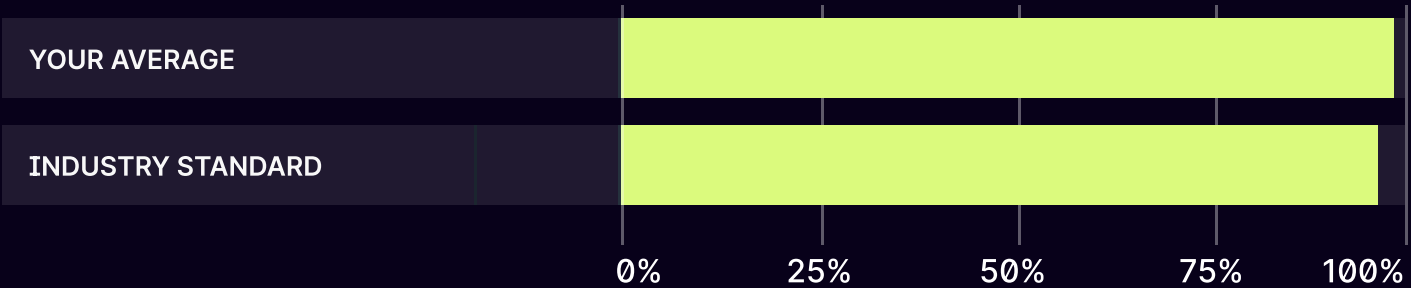
This document outlines the overall security of the LYOPAY smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the LYOPAY smart contract codebase for quality, security, and correctness.

Contract Status



Testable Code



98% of the code is testable, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network’s fast-paced and rapidly changing environment, we recommend that the LYOPAY team put in place a bug bounty program to encourage further active analysis of the smart contract.

Table of Contents

Auditing Strategy and Techniques Applied	3
Executive Summary	4
Structure and Organization of the Document	5
Protocol Overview	6
Complete Analysis	7
Code Coverage and Test Results for all files written by Zokyo Security	10

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the LYOPAY repository:
contracts are delivered as archive:

archive sha256:

84822bbf139690b9cb86160d066ed91cc9af3001d9e9aa0b723b4b7c341c09e3

LYOv2.sol sha256:

8cecf0a88b15e6db6dcec2837460fcf9369d5bb332fc2828a38d6d4d96eaa7e3

LYOv4.sol sha256:

72cf6a835c3cfeee2d4bb3e67fa921ac7db3858e8664ec103da5264625811a4f

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- contracts\LYOv4.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of LYOPAY smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Hardhat testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.	03	Testing contract logic against common and uncommon attack vectors.
02	Cross-comparison with other, similar smart contracts by industry leaders.	04	Thorough manual review of the codebase line by line.

Executive Summary

Zokyo Security team reviewed the LYO Credit token against the standard violations, against potential backdoors, and the correct roles system and verified all modifications to the ERC20 logic. From the business logic perspective, contracts act as expected, roles are assigned correctly, and no standard functions are violated. Also, the LYOPAY team verified the correctness of the Burn logic. However, auditors noted that unlimited burn may still cause issues, so it is recommended to provide sanitizing policy against token usage.

The main issue is the upgradeability of the token. Despite the verification from the LYOPAY team, Zokyo auditors still classify upgradeability as a controllable backdoor. Even if its usage is inevitable, it still influences the security of the token contract. Therefore the contract fails the appropriate checkpoint in the standard checklist. Also, the LYOPAY team verified that the token would have the new deployment, so no upgrade will be performed.

From other points of view, the LYOV3 token acts as expected. The only unresolved issue is the unrecommended notation for the Solidity version. Since no tests/deployment scripts were provided, auditors cannot verify the correct version which will be used during the deployment.


For the 4th version of the token, LYOPAY provided major changes.

- LYOV4 is not upgradeable anymore (therefore, the controllable backdoor is closed)
- The team removed Pauser and Frozen roles.
- The team removed the inheritance of OpenZeppelin contracts which included the ERC20 standard and replaced it with a custom implementation of BEP20
- the team utilized 0.8.18 Solidity version (therefore the issue with the correct version is closed)


The token is still ERC20 compatible; auditors checked the correctness of the standard implementation. LYOPAY team also presumed the burn() functionality (which is available only for the owner now) and custom decimals (8 decimals).

STRUCTURE AND ORGANIZATION OF THE DOCUMENT


For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

 **Critical**


The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

 **High**


The issue affects the ability of the contract to compile or operate in a significant way.

 **Medium**

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

 **Low**

The issue has minimal impact on the contract's ability to operate.

 **Informational**

The issue has no impact on the contract's ability to operate.

Protocol Overview

LYO Credit is a token that implements the ERC20 standard. The most important thing to note is that the contract is upgradeable.

- Token implements Burnable and Pausable functionality.
- The token has 8 decimals.
- Initializer gets the token name, symbol, the token's owner, and the token's initial supply.
- The initial supply is minted to the owner of the token.

The contract contains two roles: Pauser and Frozen role. The owner of the token immediately gets default admin and Pauser roles. Pauser role can pause transfers. When the Frozen role is assigned to the user, the user loses the ability for tokens transfer (both `transfer()` and `transferFrom()`). Also, frozen users cannot renounce the role, and Admin can freeze any user with no restrictions.

The token also has the functionality to recover tokens stuck in the contract.

After the major code update from LYOPAY team, LYOV3 became LYOV4:

- the token is no upgradeable
- the token inherits custom BEP20 implementation instead of OZ one (as it was for LYOV3)
- the token is still ERC20 compatible
- the token has `burn()` function available for the owner only
- The token has 8 decimals.
- Constructor gets the token's initial supply.
- The initial supply is minted to the owner of the token, which is set from `msg.sender`
- Token's name and symbol are "LYO Credit" and "LYO"

COMPLETE ANALYSIS

MEDIUM-1 | RESOLVED

Use fixed Solidity version.

Currently, LYOV2.sol utilizes the ^0.8 version. Since there are no deployment scripts, it is unknown which version will be used. It is crucial because several releases of solc0.8 contained bugs (which affected optimization and correct work). Therefore there is a chance to compile and deploy the contract with a vulnerable version. It is a general recommendation to use a strict declaration of the Solidity version and use the latest stable version (0.8.17 by this time).

Recommendation:

Use *pragma solidity 0.8.17* OR provide deployment scripts with exact version utilized.

Post-audit:

The code utilizes the latest stable version now.

INFO-1 | RESOLVED

Upgradeable token.

LYOV2.sol is implemented as an upgradeable contract; therefore, upgradeability creates a controllable but still a backdoor. So, this point fails the check against the existence of backdoors in the token code. Usually, upgradeability creates no problems for the contract code. But things are different for tokens since upgradeability affects listings and operations on DEXes and creates uncertainty for token users. The issue is marked as info because upgradeability may serve as an intended business logic solution. But since it violates the criteria for backdoors in the code, the issue will influence the security rating.

Recommendation:

Remove token upgradeability or verify that it is an intended business logic solution.

Post-audit:

The LYOPAY team has verified that it is an intended logic. Nevertheless, auditors should note that such an approach leaves the backdoor in the token's logic (even if it is controllable).

Note: The team removed the upgradeability for the LYOV4

Burnable token.

LYOv2.sol is implemented as a burnable token with no restrictions. Therefore the token holder can burn his tokens in any possible amount. It creates a potential loophole when custom contracts can burn tokens stored on them. The issue is marked as Info because it may be an intended business logic solution. Also, since regular contracts (Uniswap or other DEXes) cannot burn users' tokens, the chance of the exploit is low and limited to protocols with upgradeable contracts. Still, since it violates the criteria for backdoors in the code, the issue may influence the security rating.

It is especially crucial, since the token has no mint functionality - initial supply is the only minted.

Recommendation:

Verify the correctness of the Burn logic or verify which protocols will use LYO token.

Post-audit

LYOPAY team verified, that Burn logic works as expected. Nevertheless, auditors still recommend to provided the sanitizing policy against the protocols or contracts where the token may be used.

Note: The team left burn() function available for the owner only

Unused library

LYOv4.sol imports SafeMath library, but it is both obsolete because of 0.8 solc and unused in the code. Thus it just increases the size of the contract with no useful effect

Recommendation:

Remove unused SafeMath library

contracts\LYOv2.sol	
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting LYOPAY in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Hardhat testing framework.

The tests were based on the functionality of the code, as well as a review of the LYOPAY contract requirements for details about issuance amounts and how the system handles these.

Contract: LYO

Standard ERC20 functions

- ✓ Correct name (110ms)
- ✓ Correct symbol (39ms)
- ✓ Correct totalSupply (46ms)
- ✓ Correct transfer (62ms)
- ✓ Correct approve (61ms)
- ✓ Correct increase allowance (57ms)
- ✓ Correct decrease allowance (74ms)
- ✓ Correct transfer from (92ms)

Specific LYO functions

- ✓ Correct Initialization
- ✓ Should not repeat Initialization
- ✓ Correct return decimals (39ms)
- ✓ PAUSER should pause (49ms)
- ✓ Not PAUSER should not pause (38ms)
- ✓ PAUSER should unpause (52ms)
- ✓ Not PAUSER should not unpause (45ms)
- ✓ Not FROZEN should renounce his existing role (38ms)
- ✓ FROZEN should not renounce his existing role (55ms)
- ✓ Not FROZEN should not renounce not his role (38ms)
- ✓ ADMIN should recover tokens (49ms)
- ✓ Not ADMIN should not recover tokens (42ms)
- ✓ FROZEN should not transfer (57ms)
- ✓ Not FROZEN should not transfer from FROZEN (76ms)

22 passing (2s)

FILE	% STMTS	% BRANCH	% FUNCS
LYOv2.sol	100	90	100

We are grateful for the opportunity to work with the LYOPAY team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the LYOPAY team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

