SMART CONTRACTS REVIEW

zokyo

October 6th 2025 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passed a security audit.

ZOKYO AUDIT SCORING LAB

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: 0 points

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 0 Medium issues: 0 points deducted
- 0 Low issues: 0 points deducted
- 0 Informational issues: 0 points deducted

Thus, the score is 100

TECHNICAL SUMMARY

This document outlines the overall security of the LAB smart contract/s evaluated by the Zokyo Security team.

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

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s 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 LAB team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the LAB repository:
Repo: https://github.com/lab-pro-official/lab-token/

Last commit - 07ef8dfbc77bd672acf29f1cd3420093e3880317

Contracts under the scope:

- LabToken.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 LAB smart contract/s. 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 Foundry 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.
02	Cross-comparison with other, similar smart contract/s by industry leaders.
03	Thorough manual review of the codebase line by line.
04	Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team has performed a security audit of the provided codebase. Detailed findings from the audit process are outlined in the "Complete Analysis" section.

LabToken is a standard ERC20-compliant token with burnable features, designed as a utility or governance token for a potential lab-related project or ecosystem.

The codebase for the audit consists of a contract called LabToken which mints an initial supply of 1,000,000,000 LAB tokens to the deployer.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the LAB team and the LAB team is aware of it, but they have chosen to not solve it. 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.

COMPLETE ANALYSIS

DURING THE AUDITING PROCESS (BOTH MANUAL PART AND TESTING PART)
NO ISSUES WERE IDENTIFIED.

	LabToken.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 are written by Zokyo Secured team

As part of our work assisting Lab in verifying the correctness of their contract/s code, our team was responsible for writing integration tests using the Foundry testing framework.

Tests were based on the functionality of the code, as well as a review of the Lab contract/s requirements for details about issuance amounts and how the system handles these.

Ran 42 tests for test/LabToken.t.sol:LabTokenTest
[PASS] testFuzz_Approve(address,uint256) (runs: 259, μ: 36271, ~: 36733)
[PASS] testFuzz_Burn(uint256) (runs: 259, μ: 26428, ~: 26344)
[PASS] testFuzz_BurnFrom(address,uint256) (runs: 259, μ: 61728, ~: 61551)
[PASS] testFuzz_MultipleTransfers(uint256) (runs: 259, μ: 84349, ~: 84506)
[PASS] testFuzz_Transfer(address,uint256) (runs: 259, μ: 46691, ~: 46640)
[PASS] testFuzz_TransferFrom(address,address,uint256) (runs: 259, μ: 75573, ~: 75596)
[PASS] testInvariant_TotalSupplyAfterBurn() (gas: 41663)
[PASS] testInvariant_TotalSupplyEqualsBalances() (gas: 73848)
[PASS] test_AllowanceDecrement() (gas: 72598)
[PASS] test_ApproveToSelf() (gas: 33744)
[PASS] test_Approve_MaxUint256() (gas: 35899)
[PASS] test_Approve_Overwrite() (gas: 40923)
[PASS] test_Approve_Success() (gas: 40912)
[PASS] test_Approve_ZeroAddress() (gas: 8732)
[PASS] test_Approve_ZeroAmount() (gas: 20957)
[PASS] test_BurnFrom_InsufficientAllowance() (gas: 15641)
[PASS] test_BurnFrom_InsufficientBalance() (gas: 29286)
[PASS] test_BurnFrom_Success() (gas: 41892)
[PASS] test_BurnFrom_WithMaxApproval() (gas: 52502)
[PASS] test_Burn_EntireBalance() (gas: 13312)
[PASS] test_Burn_InsufficientBalance() (gas: 13294)
[PASS] test_Burn_MultipleUsers() (gas: 64197)
[PASS] test_Burn_Success() (gas: 26930)
[PASS] test_Burn_ZeroAmount() (gas: 21186)
[PASS] test_ConsecutiveBurns() (gas: 32697)
[PASS] test_Constructor_Decimals() (gas: 5616)
[PASS] test_Constructor_InitialSupply() (gas: 7708)
[PASS] test_Constructor_OwnerBalance() (gas: 9973)
[PASS] test_Constructor_TokenName() (gas: 12002)

```
[PASS] test_Constructor_TokenSymbol() (gas: 12066)
[PASS] test_TransferFromSelfToSelf() (gas: 30304)
[PASS] test_TransferFrom_InsufficientAllowance() (gas: 17905)
[PASS] test_TransferFrom_InsufficientBalance() (gas: 31216)
[PASS] test_TransferFrom_Success() (gas: 57836)
[PASS] test_TransferFrom_ToZeroAddress() (gas: 27567)
[PASS] test_TransferFrom_WithMaxApproval() (gas: 72113)
[PASS] test_TransferToSelf() (gas: 14019)
[PASS] test_Transfer_EntireBalance() (gas: 37154)
[PASS] test_Transfer_InsufficientBalance() (gas: 15754)
[PASS] test_Transfer_Success() (gas: 47032)
[PASS] test_Transfer_ToZeroAddress() (gas: 8815)
[PASS] test_Transfer_ZeroAmount() (gas: 23055)
Suite result: ok. 42 passed; 0 failed; 0 skipped; finished in 31.55ms (108.77ms CPU time)
```

We are grateful for the opportunity to work with the Lab 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 Lab team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo