MixBytes()

# Lido LDO Revesting Security Audit Report

DECEMBER 20, 2025

# Table of Contents

# 1. Introduction

## 1.1 Disclaimer

The audit makes no statements or warranties regarding the utility, safety, or security of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other claims about the fitness of the contracts for a particular purpose or their bug-free status.

## 1.2 Executive Summary

The **LDORevesting** contract is an Ownable contract that integrates with Aragon's TokenManager to re-issue LDO tokens in a single consecutive operation with new vesting schedules. The contract performs two sequential operations: re-issuing tokens via `TokenManager.burn()` and `TokenManager.issue()`, and assigning a new vesting schedule via `TokenManager.assignVested()` with a 1-year cliff and 2-year vesting period.

Security focus was placed on verifying the spendable balance invariant before and after revesting operations. The audit thoroughly examined potential attack vectors related to:

- **Balance Consistency**: Ensuring that the user's total LDO balance remains unchanged (burn + issue = net zero)
- **Spendable Balance Reduction**: Verifying that spendable balance correctly decreases by the revested amount
- **Total Supply Preservation**: Confirming that the total token supply remains constant throughout the operation
- **Edge Cases**: Testing scenarios with users who have multiple existing vesting schedules to ensure calculations remain accurate

The contract implements comprehensive assertions to verify these invariants, and the audit confirmed that the implementation correctly handles all tested scenarios, including complex cases with overlapping vesting schedules.

## 1.3 Project Overview

### Summary

| Title | Description |
|---|---|
| **Client** | Lido |
| **Category** | Liquid Staking |

| Title | Description |
| --- | --- |
| **Project** | LDO Revesting |
| **Type** | Solidity |
| **Platform** | EVM |
| **Timeline** | 20.12.2025 - 20.12.2025 |

## Scope of Audit

| File | Link |
| --- | --- |
| **src/LDORevesting.sol** | LDORevesting.sol |

## Versions Log

| Date | Commit Hash | Note |
| --- | --- | --- |
| **20.12.2024** | 6d7e4cb9e971dc4006927ac9ba5447254f9db7a8 | Initial Commit |

## Mainnet Deployments

All initial parameters of the LDORevesting contract were verified during the deployment verification. The contract owner is set to the Lido Token Reward Program (TRP) multisig (https://docs.lido.fi/multisigs/committees/#24-token-reward-program-trp-association-multisig). Four addresses were verified and added to the disallowed list during contract initialization. These addresses correspond to top LDO holders that are related to centralized exchanges (CEXes) and DAO Agent: 0xF977814e90dA44bFA03b6295A0616a897441aceC, 0x3e40D73EB977Dc6a537aF587D48316feE66E9C8c, 0x8Fa129F87B8a11ee1ca35Abd46674F8b66984d4a, 0x611f7bF868a6212f871e89F7e44684045DdFB09d

| File | Address | Blockchain |
| --- | --- | --- |
| **LDORevesting.sol** | 0xC2F50D3277539fbD54346278E7b92Faa76DC7364 | Ethereum |

# 1.4 Security Assessment Methodology

## Project Flow

| Stage | Scope of Work |
| --- | --- |
| Interim audit | **Project Architecture Review:**<br><br>· Review project documentation<br>· Conduct a general code review<br>· Perform reverse engineering to analyze the project's architecture based solely on the source code<br>· Develop an independent perspective on the project's architecture<br>· Identify any logical flaws in the design<br><br>OBJECTIVE: UNDERSTAND THE OVERALL STRUCTURE OF THE PROJECT AND IDENTIFY POTENTIAL SECURITY RISKS. |
| | **Code Review with a Hacker Mindset:**<br><br>· Each team member independently conducts a manual code review, focusing on identifying unique vulnerabilities.<br>· Perform collaborative audits (pair auditing) of the most complex code sections, supervised by the Team Lead.<br>· Develop Proof-of-Concepts (PoCs) and conduct fuzzing tests using tools like Foundry, Hardhat, and BOA to uncover intricate logical flaws.<br>· Review test cases and in-code comments to identify potential weaknesses.<br><br>OBJECTIVE: IDENTIFY AND ELIMINATE THE MAJORITY OF VULNERABILITIES, INCLUDING THOSE UNIQUE TO THE INDUSTRY. |
| | **Code Review with a Nerd Mindset:**<br><br>· Conduct a manual code review using an internally maintained checklist, regularly updated with insights from past hacks, research, and client audits.<br>· Utilize static analysis tools (e.g., Slither, Mythril) and vulnerability databases (e.g., Solodit) to uncover potential undetected attack vectors.<br><br>OBJECTIVE: ENSURE COMPREHENSIVE COVERAGE OF ALL KNOWN ATTACK VECTORS DURING THE REVIEW PROCESS. |

| Stage | Scope of Work |
| --- | --- |
| | **Consolidation of Auditors' Reports:**<br><br>· Cross-check findings among auditors<br>· Discuss identified issues<br>· Issue an interim audit report for client review<br><br>OBJECTIVE: COMBINE INTERIM REPORTS FROM ALL AUDITORS INTO A SINGLE COMPREHENSIVE DOCUMENT. |
| Re-audit | **Bug Fixing & Re-Audit:**<br><br>· The client addresses the identified issues and provides feedback<br>· Auditors verify the fixes and update their statuses with supporting evidence<br>· A re-audit report is generated and shared with the client<br><br>OBJECTIVE: VALIDATE THE FIXES AND REASSESS THE CODE TO ENSURE ALL VULNERABILITIES ARE RESOLVED AND NO NEW VULNERABILITIES ARE ADDED. |
| Final audit | **Final Code Verification & Public Audit Report:**<br><br>· Verify the final code version against recommendations and their statuses<br>· Check deployed contracts for correct initialization parameters<br>· Confirm that the deployed code matches the audited version<br>· Issue a public audit report, published on our official GitHub repository<br>· Announce the successful audit on our official X account<br><br>OBJECTIVE: PERFORM A FINAL REVIEW AND ISSUE A PUBLIC REPORT DOCUMENTING THE AUDIT. |

# 1.5 Risk Classification

## Severity Level Matrix

| Severity | Impact: High | Impact: Medium | Impact: Low |
|---|---|---|---|
| Likehood: High | Critical | High | Medium |
| Likehood: Medium | High | Medium | Low |
| Likehood: Low | Medium | Low | Low |

## Impact

- **High** – Theft from 0.5% OR partial/full blocking of funds (>0.5%) on the contract without the possibility of withdrawal OR loss of user funds (>1%) who interacted with the protocol.
- **Medium** – Contract lock that can only be fixed through a contract upgrade OR one-time theft of rewards or an amount up to 0.5% of the protocol's TVL OR funds lock with the possibility of withdrawal by an admin.
- **Low** – One-time contract lock that can be fixed by the administrator without a contract upgrade.

## Likelihood

- **High** – The event has a 50–60% probability of occurring within a year and can be triggered by any actor (e.g., due to a likely market condition that the actor cannot influence).
- **Medium** – An unlikely event (10–20% probability of occurring) that can be triggered by a trusted actor.
- **Low** – A highly unlikely event that can only be triggered by the owner.

## Action Required

- **Critical** – Must be fixed as soon as possible.
- **High** – Strongly advised to be fixed to minimize potential risks.
- **Medium** – Recommended to be fixed to enhance security and stability.
- **Low** – Recommended to be fixed to improve overall robustness and effectiveness.

## Finding Status

- **Fixed** – The recommended fixes have been implemented in the project code and no longer impact its security.
- **Partially Fixed** – The recommended fixes have been partially implemented, reducing the impact of the finding, but it has not been fully resolved.
- **Acknowledged** – The recommended fixes have not yet been implemented, and the finding remains unresolved or does not require code changes.

## 1.6 Summary of Findings

### Findings Count

| Severity | Count |
| --- | --- |
| Critical | 0 |
| High | 0 |
| Medium | 0 |
| Low | 1 |

### Findings Statuses

| ID | Finding | Severity | Status |
| --- | --- | --- | --- |
| L-1 | Missing Spendable Balance Validation in `_revest` Function | Low | Acknowledged |

# 2. Findings Report

## 2.1 Critical

Not Found

## 2.2 High

Not Found

## 2.3 Medium

Not Found

## 2.4 Low

| L-1 | Missing Spendable Balance Validation in `_revest` Function | | |
|---|---|---|---|
| **Severity** | Low | **Status** | Acknowledged |

**Description**
The `_revest` function does not validate that the `amount` parameter is less than or equal to the account's spendable balance before executing the revesting operation. The function directly proceeds to subtract `amount` from `spendableBalanceBefore`, which can cause underflow revert.
**Recommendation**
Add an explicit check at the beginning of the `_revest` function to ensure `amount <= spendableBalanceBefore`. This will provide early validation and better error handling.

# 3. About MixBytes

MixBytes is a leading provider of smart contract audit and research services, helping blockchain projects enhance security and reliability. Since its inception, MixBytes has been committed to safeguarding the Web3 ecosystem by delivering rigorous security assessments and cutting-edge research tailored to DeFi projects.

Our team comprises highly skilled engineers, security experts, and blockchain researchers with deep expertise in formal verification, smart contract auditing, and protocol research. With proven experience in Web3, MixBytes combines in-depth technical knowledge with a proactive security-first approach.

## Why MixBytes

- **Proven Track Record:** Trusted by top-tier blockchain projects like Lido, Aave, Curve, and others, MixBytes has successfully audited and secured billions in digital assets.
- **Technical Expertise:** Our auditors and researchers hold advanced degrees in cryptography, cybersecurity, and distributed systems.
- **Innovative Research:** Our team actively contributes to blockchain security research, sharing knowledge with the community.

## Our Services

- **Smart Contract Audits:** A meticulous security assessment of DeFi protocols to prevent vulnerabilities before deployment.
- **Blockchain Research:** In-depth technical research and security modeling for Web3 projects.
- **Custom Security Solutions:** Tailored security frameworks for complex decentralized applications and blockchain ecosystems.

MixBytes is dedicated to securing the future of blockchain technology by delivering unparalleled security expertise and research-driven solutions. Whether you are launching a DeFi protocol or developing an innovative dApp, we are your trusted security partner.

## Contact Information


https://mixbytes.io/

https://github.com/mixbytes/audits_public

hello@mixbytes.io

https://x.com/mixbytes