MixBytes()

Lido EasyTrack stVaults Security Audit Report

APRIL 21, 2026

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 audited contracts are Easy Track EVMScript factories for managing tiers and groups in Lido's OperatorGrid. Three factories exist: `AlterTiersInOperatorGrid` modifies existing tier parameters, `RegisterTiersInOperatorGrid` registers new tiers for existing groups, and `RegisterGroupsInOperatorGrid` atomically creates new groups together with their initial tiers. All three factories validate input parameters and produce EVMScripts that are executed through Aragon's EasyTrack governance mechanism.

The codebase was audited in 1 day by 4 auditors using a combination of manual review and automated tooling.

The diff introduces a single conceptual change applied across all three contracts: tier `shareLimit` validation is decoupled from the on-chain group `shareLimit` and instead checked against a hardcoded constant `MAX_SHARE_LIMIT = 10_000_000 * 1e18`. Previously, the factories required each tier's `shareLimit` to be at most its parent group's `shareLimit`, which was read from OperatorGrid at script creation time. The new approach relaxes this factory-level constraint, relying on the OperatorGrid's runtime enforcement in `onMintedShares()` to cap actual minting at both the tier and group level.

In `AlterTiersInOperatorGrid`, there is also a secondary change: tier existence is now checked via `_tierIds[i] < tiersCount` instead of calling `operatorGrid.tier()` and relying on a revert. This avoids an unnecessary storage read and makes the existence check explicit.

The audit verified the following vectors: that `MAX_SHARE_LIMIT` fits within `uint96` (the type OperatorGrid uses internally), so no overflow can occur when the value is eventually cast; that the constant is consistent across all three factories; that OperatorGrid's `onMintedShares()` still enforces both tier-level and group-level share limits at runtime regardless of what the factories allow; that the `tiersCount`-based existence check is sound because OperatorGrid uses a push-only array with sequential IDs starting from 0; and that the default tier (ID=0) still uses the separate immutable `defaultTierMaxShareLimit`, unaffected by this change.

The OperatorGrid contract itself and its internal enforcement logic were not in scope for this audit.

The code is clean and well-structured. The change is minimal and self-contained. No issues were found.

1.3 Project Overview

Summary

Title	Description
Client	Lido
Category	Liquid Staking
Project	EasyTrack stVaults
Type	Solidity
Platform	EVM
Timeline	19.03.2026 - 20.03.2026

Scope of Audit

File	Link
contracts/EVMScriptFactories/vaultFactories/AlterTiersInOperatorGrid.sol	contracts/EVMScriptFactories/vaultFactories/AlterTiersInOperatorGrid.sol
contracts/EVMScriptFactories/vaultFactories/RegisterGroupsInOperatorGrid.sol	contracts/EVMScriptFactories/vaultFactories/RegisterGroupsInOperatorGrid.sol
contracts/EVMScriptFactories/vaultFactories/RegisterTiersInOperatorGrid.sol	contracts/EVMScriptFactories/vaultFactories/RegisterTiersInOperatorGrid.sol

Versions Log

Date	Commit Hash	Note
19.03.2026	1330bcff9dec5d0c800af9b7f8347268e6743181	Initial Commit

Mainnet Deployments

We verified that the bytecode of the AlterTiersInOperatorGrid, RegisterGroupsInOperatorGrid, and RegisterTiersInOperatorGrid contracts audited in this report matches the bytecode compiled from the commit referenced in the audit scope. The initialization parameters of these new implementations are consistent with those currently used by the corresponding production mainnet instances (AlterTiersInOperatorGrid at 0x73f80240ad9363d5d3C5C3626953C351cA36Bfe9, RegisterGroupsInOperatorGrid at 0xE73842AEbEC99Dacf2aAEec61409fD01A033f478, and RegisterTiersInOperatorGrid at 0x5292A1284e4695B95C0840CF8ea25A818751C17F).
The three audited factories are activated by Aragon DAO vote https://dao.lido.fi/vote/199. The vote's script includes six actions directed at EasyTrack (0xF0211b7660680B49De1A7E9f25C65660F0a13Fea) that jointly swap the old Phase-2/3 factories for the redeployed ones audited in this report:
Three EasyTrack.removeEVMScriptFactory(...) calls target the currently-registered production factories - 0xE73842AEbEC99Dacf2aAEec61409fD01A033f478 (old RegisterGroupsInOperatorGrid), 0x5292A1284e4695B95C0840CF8ea25A818751C17F (old RegisterTiersInOperatorGrid), and 0x73f80240ad9363d5d3C5C3626953C351cA36Bfe9 (old AlterTiersInOperatorGrid) - removing their EVMScript execution rights.
EasyTrack.addEVMScriptFactory(0x17305dB55c908e84C58BbDCa57258A7D1f7eEa7c, permissions) registers the new RegisterGroupsInOperatorGrid with permissions encoding the OperatorGrid address (0xC69685E89Cefc327b43B7234AC646451B27c544D) followed by the registerGroup selector 0xe37a7c0b and the registerTiers selector 0x552b91da.
EasyTrack.addEVMScriptFactory(0x6b535F441F95046562406F4E2518D9AD7Db2dc0D, permissions) registers the new RegisterTiersInOperatorGrid with a single permission entry targeting OperatorGrid's registerTiers selector 0x552b91da.
EasyTrack.addEVMScriptFactory(0x37d9B09EDA477a84E3913fCB4d032EFb0BF9B62E, permissions) registers the new AlterTiersInOperatorGrid with a single permission entry targeting OperatorGrid's alterTiers selector 0x54544bcb.
Each factory is granted the minimum set of OperatorGrid entry-points its scripts actually invoke when executed: RegisterGroupsInOperatorGrid is the only factory whose scripts include registerGroup, and every script it produces pairs each registerGroup call with a subsequent registerTiers call for the same operator, so both selectors appear in its permission set; RegisterTiersInOperatorGrid and AlterTiersInOperatorGrid each receive exactly one selector, matching their single-action scripts. No additional target contract beyond OperatorGrid is whitelisted for any of the three, so the factories cannot be repurposed to invoke other Lido contracts through EasyTrack.

File	Address
AlterTiersInOperatorGrid	0x37d9B09EDA477a84E3913fCB4d032EFb0BF9B62E
RegisterGroupsInOperatorGrid	0x17305dB55c908e84C58BbDCa57258A7D1f7eEa7c
RegisterTiersInOperatorGrid	0x6b535F441F95046562406F4E2518D9AD7Db2dc0D

1.4 Security Assessment Methodology

Project Flow

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

Stage	Scope of Work
	Consolidation of Auditors' Reports: · Cross-check findings among auditors · Discuss identified issues · Issue an interim audit report for client review OBJECTIVE: COMBINE INTERIM REPORTS FROM ALL AUDITORS INTO A SINGLE COMPREHENSIVE DOCUMENT.
Re-audit	**Bug Fixing & Re-Audit:** · The client addresses the identified issues and provides feedback · Auditors verify the fixes and update their statuses with supporting evidence · A re-audit report is generated and shared with the client 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:** · Verify the final code version against recommendations and their statuses · Check deployed contracts for correct initialization parameters · Confirm that the deployed code matches the audited version · Issue a public audit report, published on our official GitHub repository · Announce the successful audit on our official X account 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	0

2. Findings Report

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

Not Found

3. About MixBytes

MixBytes is a leading provider of smart contract auditing and blockchain security research, helping Web3 projects enhance their security and reliability.

Since its inception, MixBytes has been dedicated to safeguarding innovation in **DeFi** through rigorous audits and cutting-edge technical research.

Our team brings together experienced engineers, security auditors, and blockchain researchers with deep expertise in smart contract security and protocol design.

With years of proven experience in Web3, MixBytes combines in-depth technical excellence with a proactive, security-first approach.

Why MixBytes

- **Proven Track Record:** Trusted by leading blockchain protocols such as **Lido**, **Aave**, **Curve**, **1inch**, **Fluid**, **Gearbox**,**Resolv**, and others. MixBytes has successfully secured billions of dollars in digital assets.
- **Technical Expertise:** Our auditors and researchers hold advanced degrees in cryptography, cybersecurity, and distributed systems, backed by years of hands-on experience.
- **Innovative Research:** MixBytes actively contributes to blockchain security research and open-source tools, sharing insights with the global community.

Our Services

- **Smart Contract Audits:** Comprehensive security assessments of DeFi protocols to detect and mitigate vulnerabilities before deployment.
- **Blockchain Research:** In-depth technical studies, economic and protocol-level modeling for Web3 projects.
- **Custom Security Solutions:** Tailored frameworks and advisory for complex decentralized systems and blockchain ecosystems.

Contact Information

 https://mixbytes.io/

 https://github.com/mixbytes/audits_public

 https://x.com/MixBytes

 hello@mixbytes.io