MixBytes()

Resolv Treasury Extension Security Audit Report

JULY 17, 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

Resolv is a protocol that issues USR, a USD-pegged stablecoin backed by ETH, using a combination of ETH collateral, short perpetual futures for hedging, and an insurance pool (RLP) to maintain stability. The Resolv treasury is an onchain contract used to manage part of the protocol's assets. Admins and managers can interact with external protocols through modular extensions.

In this audit, we reviewed the EtherFiTreasuryExtension, which allows the treasury to stake ETH into the EtherFi protocol and unstake it back to the treasury. We focused on ensuring correct access control and verifying the safety of ETH transfers to and from EtherFi, including behavior related to rebasing and rounding.

We also went through our detailed checklist, covering other aspects such as business logic, common ERC20/ERC4626 issues, interactions with external contracts, integer overflows, reentrancy, access control boundaries, typecast risks, rounding precision, and potential edge cases.

No significant vulnerabilities were found.

1.3 Project Overview

Summary

Title	Description
Client Name	Resolv
Project Name	Treasury
Type	Solidity
Platform	EVM

Title	Description
Timeline	28.05.2025-03.06.2025

Scope of Audit

File	Link
contracts/EtherFiTreasuryExtension.sol	EtherFiTreasuryExtension.sol

Versions Log

Date	Commit Hash	Note
28.05.2025	ae2b6873a369e3eaa107a269dbeb2e1eb63eb74b	Commit for the audit
03.06.2025	625ed9eef96efc6c0167f74497c0e3129d730f47	Commit for the re-audit

Mainnet Deployments

File	Address	Blockchain
EtherFiTreasuryExtension.sol	0x0dbD42...A0beF998	Ethereum Mainnet

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	2

Findings Statuses

ID	Finding	Severity	Status
L-1	`LIQUIDITY_POOL.amountForShare(1 wei)` may round down to zero	Low	Fixed
L-2	Inconsistent inheritance pattern in EtherFiTreasuryExtension	Low	Fixed

2. Findings Report

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	LIQUIDITY_POOL.amountForShare(1 wei) may round down to zero		
Severity	Low	**Status**	Fixed in 625ed9ee

Description
This issue has been identified within the depositETHForEETH() and depositETHForWeETH() functions of the EtherFiTreasuryExtension.
- EtherFiTreasuryExtension.sol#L46
- EtherFiTreasuryExtension.sol#L58

While the underlying _deposit(...) function of the LIQUIDITY_POOL ensures that zero shares are never minted (i.e., share == 0 triggers a revert), the subsequent computation:

```
uint256 eEthAmount = LIQUIDITY_POOL.amountForShare(eETHShares);
```

may still result in eEthAmount == 0 even for eETHShares > 0.
This is due to the implementation of:

```
function amountForShare(uint256 _share) public view returns (uint256) {
    uint256 totalShares = eETH.totalShares();
    if (totalShares == 0) {
        return 0;
    }
    return (_share * getTotalPooledEther()) / totalShares;
}
```

If the pool is in early stages or under unusual liquidity conditions (e.g., small getTotalPooledEther() or very high totalShares), this division may truncate to zero for _share == 1.

As a result:

- A non-zero share amount is minted.
- But zero tokens are transferred to the treasury.
- Emitted events (e.g. Deposited(…)) will indicate a completed operation, but the treasury received no assets.

This may result in inconsistent accounting, noisy events, or even broken assumptions downstream.

Recommendation

We recommend adding an explicit check in EtherFiTreasuryExtension to ensure that eEthAmount > 0 after calling amountForShare(...):

```
require(eEthAmount > 0, "Deposit too small to mint usable amount");
```

Client's Commentary:

PR-348

L-2	Inconsistent inheritance pattern in EtherFiTreasuryExtension		
Severity	Low	**Status**	Fixed in 625ed9ee

Description
The EtherFiTreasuryExtension contract inherits from both ERC721HolderUpgradeable and AccessControlDefaultAdminRules:
• EtherFiTreasuryExtension.sol#L16

This is inconsistent as one parent is upgradeable while the other is not.
The inheritance pattern should be consistent - either both parents should be upgradeable (using AccessControlDefaultAdminRulesUpgradeable) or both should be non-upgradeable (using ERC721Holder).
This inconsistency could potentially lead to initialization issues or unexpected behavior during upgrades.

Recommendation
Choose one of the following approaches:
1. Use upgradeable versions of both contracts:

```
ERC721HolderUpgradeable, AccessControlDefaultAdminRulesUpgradeable
```

2. Use non-upgradeable versions of both contracts:

```
ERC721Holder, AccessControlDefaultAdminRules
```

Client's Commentary:
PR-347

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