certora

Security Assessment

ether.fi

ether.fi – Core Contracts Combined Audit Report

June - July 2025

Prepared for ether.fi

Table of contents

Project Summary

Project Scope

Project Name	Initial Commit Hash	Latest Commit Hash	Platform	Start Date	End Date
V3-Prelude minor updates	Hash	Hash	EVM	09/07/2025	11/07/2025
V3-Prelude storage shift fix	Hash	Hash	EVM	23/07/2025	23/07/2025

Project Overview

This document describes the manual code review of several modules and changes to the core contracts repository.

The work was a 3 day effort undertaken between **09/07/2025** and **23/07/2025**

The team performed a manual audit of the Solidity smart contracts. During the manual audit, the Certora team discovered bugs in the Solidity smart contracts code, as listed on the following page.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	-	-	-
Medium	-	-	-
Low	-	-	-
Informational	3	2	2
Total	-	-	-

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
V3-Prelude minor updates			
I-01	Check for 0 balances before emitting events	Informational	Fixed
I-02	Reset pending merkle proofs upon finalization	Informational	Acknowledged
I-03	Potential storage collision on WeEth contract	Informational	Acknowledged
V3-Prelude storage shift fix			
-	-	-	-

V3-Prelude minor updates

Project Overview

This report presents the findings of a manual code review for the **V3-Prelude minor updates** audit within the **EtherFi** core contracts. The work was undertaken from **July 9th to July 11th 2025**

The following contract list is included in the scope of this audit:

- `src/CumulativeMerkleRewardsDistributor.sol`
- `src/EtherFiAdmin.sol`
- `src/EtherFiNode.sol`
- `src/EtherFiNodesManager.sol`
- `src/LiquidityPool.sol`
- `src/StakingManager.sol`
- `src/WeETH.sol`
- `src/WithdrawRequestNFT.sol`
- `src/interfaces/ICumulativeMerkleRewardsDistributor.sol`
- `src/interfaces/IEtherFiNode.sol`
- `src/interfaces/IEtherFiNodesManager.sol`
- `src/interfaces/IEtherFiOracle.sol`
- `src/interfaces/IStakingManager.sol`
- `src/interfaces/IWeETH.sol`
- `src/libraries/DepositDataRootGenerator.sol`

The code modifications examined during this review were implemented in the following pull request - PR#270

Informational Issues

I-01. Check for 0 balances before emitting events

Description: The sweepFunds() and completeQueuedETHWithdrawals() calls inside EtherFiNodesManager always emit events, even when the balances are 0 and no operation was actually executed by the underlying EtherFiNode

Recommendations: Check that the balances are not 0 before emitting events

Customer's response: Fixed in commit dba7a58

Fix Review: Fixed

I-02. Reset pending merkle proofs upon finalization

Description: Inside CumulativeMerkleRewardsDistributor an admin calls setPendingMerkleRoot() to schedule a new merkle root. Once the delay expires the same admin calls finalizeMerkleRoot() to update the root used for reward claims.

It is considered a good practice to always clear pending values when they get set as the current values of a variable. Currently once the claimableMerkleRoots is updated the pendingMerkleRoots & lastPendingMerkleUpdatedToTimestamp variable values are not cleared, meaning that finalizeMerkleRoot() can be called again with the already applied pending root value, since it continues to exist in storage.

Recommendations: Reset pendingMerkleRoots & lastPendingMerkleUpdatedToTimestamp variables after finalizing them

Customer's response: Acknowledged

Fix Review: Acknowledged

I-03. Potential storage collision on WeEth contract

Description: One of the main changes to the WeETH contract is that the roleRegistry public state variable has been modified to an immutable one.

WeETH is an upgradeable contract, which means that storage slots should be handled carefully. The initially defined public state variable roleRegistry has been removed in the new version of the weETH contract and replaced by an immutable var.

In this particular case the above change is not a problem, because the old version was never deployed as upgrade to the proxy, but it is important to still be aware of the potential risks involved with storage slots changes in future upgrades

Recommendations: Currently there is no risk, since the team did not deploy the initial version on the blockchain. It is essential that storage slot pointers (roleRegistry) in upgradeable contracts never get removed once added - this ensures predictable and manageable behaviour.

Customer's response: "The previous version of weETH including the roleRegistry storage variable has not been deployed, so the changes will not cause storage collisions"

Fix Review: Acknowledged

V3-Prelude storage shift fix

Project Overview

This report presents the findings of a manual code review for the **V3-Prelude storage shift fix** audit within the **EtherFi** core contracts. The work was undertaken on **July 23rd 2025**

The following contract list is included in the scope of this audit:

- `src/AssetRecovery.sol`
- `src/EETH.sol`
- `src/interfaces/IEtherFiNodesManager.sol`
- `src/interfaces/ILiquidityPool.sol`

The code modifications examined during this review were implemented in the following pull request - PR#274

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.