certora

Security Assessment

ether.fi

Ether-Fi - v2.49

February 2025

Prepared for EtherFi

Table of Contents

Project Summary

Project Scope

Project Name	Repository (link)	Commit Hashes	Platform
EtherFi smart contracts	etherfi-protocol/smart-contracts	Audit start: 19503982 Audit end: 3e9f54ec Latest version reviewed: abc96405	EVM

Project Overview

This document describes the manual code review of PR 230 related to v2.49.
The work was a 5 day-effort undertaken from **10/02/2025** to **14/02/2025.**

The following contract list is included in our scope:

1. `src/EtherFiAdmin.sol`
2. `src/LiquidityPool.sol`
3. `src/RoleRegistry.sol`
4. `src/WeETH.sol`

The team performed a manual audit of all 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.

Note 1: A late-stage revision (PR 240) was also reviewed on **11/03/2025** with a 1 day-effort. The additional findings can be found in the following section: **Appendix 1: Late-stage revision - PR240**

Note 2: Some late-stage improvements were reviewed on **14/03/2025** with a 1 day-effort. There were no findings. See the following section: **Appendix 2: Late-stage improvements**

Findings Summary

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

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	1	1	1
Medium	1	1	1
Low	-	-	-
Total	2	2	2

Severity Matrix

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

Medium Severity Issues

M-01 Value sent with LiquidityPool.batchDeposit is lost

Severity: **Medium**	Impact: **High**	Likelihood: **Low**
Files: [LiquidityPool.sol#L297]()	Status: Fixed	

Description:

Within the changes in scope, LiquidityPool has been modified to always operate as BNFT holder. Within this context, the `batchDeposit` function remained `payable`, but the `msg.value` provided by callers is no longer accounted for:

```javascript
File: LiquidityPool.sol
292:     function batchDeposit(uint256[] calldata _candidateBidIds, uint256 _numberOfValidators, uint256 _validatorIdToShareSafeWith) public payable whenNotPaused returns (uint256[] memory) {
293:         address tnftHolder = address(this);
294:         address bnftHolder = address(this);
295:
296:         require(validatorSpawner[msg.sender].registered, "Incorrect Caller");
297:         require(totalValueInLp + msg.value >= 32 ether * _numberOfValidators, "Not enough balance");
298:
299:         uint256[] memory newValidators = stakingManager.batchDepositWithBidIds(_candidateBidIds, _numberOfValidators, msg.sender, tnftHolder, bnftHolder, SourceOfFunds.EETH, restakeBnftDeposits, _validatorIdToShareSafeWith);
300:         numPendingDeposits += uint32(newValidators.length);
301:
302:         return newValidators;
303:     }
```

We recommend making the `batchDeposit` function not payable because any value sent is a donation to the protocol which also cannot be used because it would increase the contract balance but not `totalValueInLp`.

ether.fi's response: Fixed in commit b7a8d04d

Certora's response: Fix confirmed

Informational Severity Issues

I-01. _sendFunds call can be removed from batchCancelDeposit

Affected code:

- src/LiquidityPool.sol:L384

The `batchCancelDeposit` function keeps track of a cumulative `returnAmount` which is then sent at the end of the iteration for canceling deposits.
However, `returnAmount` is never increased so the `_sendFund` call is useless.

We recommend removing `returnAmount` and the `_sendFund` call from the `batchCancelDeposit` function.

ether.fi's response: Fixed in commit b7a8d04d

Certora's response: Fix confirmed

I-02. Outbound amounts are unnecessarily re-calculated

Affected code:

- src/LiquidityPool.sol:L328
- src/LiquidityPool.sol:L366

The contract `LiquidityPool`, in the `batchRegister` and `batchApproveRegistration` functions, calculates native amounts to be sent out and stores this value in the `outboundEthAmountFromLp` variable.
However, later, when transferring this amount, it recalculates the amount instead of reusing the pre-calculated `outboundEthAmountFromLp`:

```JavaScript
File: LiquidityPool.sol
325:        uint256 outboundEthAmountFromLp = 1 ether * _validatorIds.length;
326:        _accountForEthSentOut(outboundEthAmountFromLp);
327:
328:        stakingManager.batchRegisterValidators{value: 1 ether *
_validatorIds.length}(...);
329:
```

```
---
        uint256 outboundEthAmountFromLp = 31 ether * _validatorIds.length;
        _accountForEthSentOut(outboundEthAmountFromLp);

        stakingManager.batchApproveRegistration{value: 31 ether * _validatorIds.length}(,,,);
    }
```

ether.fi's response: Fixed in commit 3e9f54ec

Certora's response: Fix confirmed

I-03. executeValidatorManagementTask does not follow CEI pattern

Affected code:

- src/EtherFiAdmin.sol:L228

The function `executeValidatorManagementTask` in `EtherFiAdmin` allows for asynchronous execution of validator actions after oracle report delivery. This function makes several external calls; however, a management task is marked as executed only after the external calls, thus violating the check-effect-interaction pattern

ether.fi's response: Fixed in commit b7a8d04d

Certora's response: Fix confirmed

I-04. setValidatorTaskBatchSize allows for setting batch size at zero

Affected code:

- src/EtherFiAdmin.sol:L171

The setValidatorTaskBatchSize misses a sanity check for the new batch size being strictly greater than zero. When the size is set to zero, oracle delivery can fail for a division-by-zero revert at L281

ether.fi's response: Fixed in commit 3e9f54ec

Certora's response: Fix confirmed, it is recommended to add an extra check for the new values set in `setValidatorTaskBatchSize`

I-05. validator tasks can be executed out of order or skipped entirely

Affected code:

- src/EtherFiAdmin.sol

With the validator tasks executed asynchronously, oracle report delivery only stores hashes of tasks to be executed later. This means that, despite the intention of the code of using a queued mechanism, validator tasks can be executed out of order, skipped altogether, or even executed after tasks from a following oracle report.

ether.fi's response: We don't think this is an issue even if it happens out of order. Do you see any issue with doing them out of order?

Appendix 1: Late-stage revision - PR240

Update partialWithdraw to sweep ETH from EtherFiNode contracts

The ETH validators earn the staking rewards and they are sent to the `EtherFiNode` contracts.
The process of sweeping those ETH is called `partialWithdraw`.
Currently, `EtherFiNodesManager.partialWithdraw` can't process the withdrawal beyond 16 ETH because the `EtherFiNodesManager._getTotalRewardsPayoutsFromSafe` reverts if the contract's balance >= 16 ETH.
This constraint was added in the past due to the complexity in handling the distribution of staking rewards and principal after exit (= 32 ETH); while the earned staking rewards were distributed to (T-NFT, B-NFT, NodeOperator, Treasury), the principal (32 ETH) were sent to (B-NFT, T-NFT).
This complexity was removed while ago and now the below constraints are true:

- for all validators, its T-NFT holder == B-NFT holder
- `stakingRewardsSplit.tnft` is set to `SCALE` (100%); stakingRewardsSplit.{treasury, nodeOperator, bnft} are 0.
    - The staking rewards distribution to (treasury, node operator) is handled with eETH (instead of ETH from `EtherFiNode` contracts) by Oracle minting eETH

Therefore, all ETH in (EtherFiNode) contracts belongs to the T-NFT holder (= LiquidityPool).
This allows us to remove the constraint on the withdrawal amount.

The file affected by this merge is:

- `src/EtherFiNodesManager.sol`

The first review commit hash is ffd0b87a

The last reviewed commit hash is 260ec8bb

Findings

H-01 Unintentional Removal of _distributePayouts in partialWithdraw()		
Severity: **High**	Impact: **Medium**	Likelihood: **High**
Files: EtherFiNodesManager.sol	Status: Fixed	

Description: `_distributePayouts()` was unintentionally removed, meaning `partialWithdraw()` now calculates payout amounts but does not distribute funds.

ether.fi's response: Fixed in commit ffd0b87a

I-06. Typo in comment

Affected code:

- EtherFiNodesManager.sol

```JavaScript
    /// @dev This function is will be deprecated in the future for simpler operations using the advanced rewards distribution
```

ether.fi's response: Fixed in this PR which is now merged to `v2.49`:
https://github.com/etherfi-protocol/smart-contracts/pull/230/commits/260ec8bb443401d873e0e55a18be76ac5ccfbf9f

Appendix 2: Late-stage improvements

Adding MAX_ROLE

The reviewed commit hash is a1dec631.

The Ether.Fi team noticed that, while the transaction does not fail without defining MAX_ROLE, there is still an “Error in Internal Transaction” every time grantRole() is called (see https://etherscan.io/address/0x13555cba55155796c70A5C8CC424E9ab6750A29F).
This improvement aims to have cleaner transactions.

The code review didn’t reveal any bugs.

Renaming roles

The reviewed PR is PR 243, with the latest commit hash being abc96405.

The changes were fairly trivial and didn’t reveal any bugs.

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.