Security Assessment

Ether-Fi – Berachain Native Minting Contracts

January 2025

Prepared for EtherFi

Table of content

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
EtherFi smart contracts	etherfi-protocol/weETH-cross-chain	Audit start: 9154247f Latest fix review: d5820774	EVM

Project Overview

This document describes the manual code review of PR 40.
The work was a 1 day-effort undertaken from **24/01/2025** to **27/01/2025.**

The following contract list is included in our scope:

1. `NativeMinting/L2SyncPoolContracts/HydraSyncPoolETHUpgradeable.sol`
2. `NativeMinting/ReceiverContracts/L1HydraReceiverETHUpgradeable.sol`

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	1	1	1
Total	1	1	1

Severity Matrix

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

Low Severity Issues

L-01 The excessive fee isn't refunded back

Severity: **Low**	Impact: **Medium**	Likelihood: **Low**
Files: HydraSyncPoolETHUpgradeable.sol	Status: Fixed	

Description:

`refundAddress` is `address(0x0)` in `stargate.sendToken()`

The excessive tokens will be lost:

```
File: HydraSyncPoolETHUpgradeable.sol
159:         (MessagingReceipt memory _msgReceipt, OFTReceipt memory
oftReceipt,) = stargate.sendToken{ value: hydraFee.nativeFee }(sendParam,
hydraFee, address(0x0));
```

ether.fi's response: The exact amount should always be supplied, but I can just add the sender for best practice: PR 43

Certora's response: Be aware that if `msg.sender` is a contract that lacks a `receive` function then this will revert.

ether.fi's response: This is automatically called by an EOA in our backend.

Informational Severity Issues

I-01. The returns parameter and the return statement are using different values

This is more about code confusion as the final `return` statement is correct and is what will be returned.

```
File: HydraSyncPoolETHUpgradeable.sol
147:     ) internal virtual override returns (MessagingReceipt memory
msgReceipt) {
...
160:         msgReceipt = _msgReceipt;
...
166:         return receipt;
```

ether.fi's response: `msgReceipt = _msgReceipt;` is an unnecessary line of code and will be removed. See PR 42

I-02. Frontrunnable initializers

The initializations are frontrunnable:

```
File: DeployMockNativeMintingL1.s.sol
32:         address receiver = address(new
TransparentUpgradeableProxy(receiverImpl, delegate, ""));
33:
34:
L1HydraReceiverETHUpgradeable(payable(receiver)).initialize(address(mockPool),
lzEndpoint, deployer);
```

and

```
File: DeployMockNativeMintingL2.s.sol
42:         HydraSyncPoolETHUpgradeable poolProxy =
HydraSyncPoolETHUpgradeable(address(new TransparentUpgradeableProxy(poolImpl,
deployer, "")));
43:
44:         poolProxy.initialize(address(rateProvider), address(0x0),
address(token), l1Eid, stargateOFTETH, l1Receiver, deployer);
```

Consider a deployment with a payload (passing a data field instead of "" will trigger `upgradeToAndCall()` which can atomically initialize the contracts)

ether.fi's response: These are just the scripts to deploy and test on testnet and will not be used to deploy these contracts to production. Aware of the possible front run issue and we always use an encoded deployment payload for our production scripts

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.