



Security Assessment



Ether-Fi

February 2025

Prepared for EtherFi

[Table of content](#)

| | |
|--|-----------|
| Project Summary | 3 |
| Project Scope | 3 |
| Project Overview | 3 |
| Findings Summary | 4 |
| Severity Matrix | 4 |
| Detailed Findings | 5 |
| Medium Severity Issues | 6 |
| M-01 CREATE3 will not work in zkSync | 6 |
| Low Severity Issues | 8 |
| L-01 BeaconFactory misses implementation for UpgradeableBeacon::upgradeTo | 8 |
| L-02 Anyone can take ownership of TopUp's implementation | 8 |
| L-03 Bridging native tokens is not possible | 9 |
| Informational Severity Issues | 11 |
| I-01. Internal upgradeable functions should have the onlyInitializing modifier | 11 |
| I-02. UpgradeableProxy::_setRoleRegistry is not used anywhere | 11 |
| I-03. UpgradeableProxy Comments Are Wrong | 11 |
| I-04. Use of Storage for Constant ETH Token Address | 12 |
| I-05. Unused Error In EtherFiOFTBridgeAdapter | 13 |
| I-06. Considering Indexing Some Event Parameters | 13 |
| I-07. Compute minAmount in TopUpFactory Rather Than In Bridge Adapters | 14 |
| Disclaimer | 15 |
| About Certora | 15 |

Project Summary

Project Scope

| Project Name | Repository (link) | Latest Commit Hash | Platform |
|-------------------------|---|--|----------|
| EtherFi TopUp contracts | https://github.com/etherfi-protocol/cash-v2/tree/dev | c0ae4e5d286df90bd d612a532eeab3135 32eb6c2 | EVM |

Project Overview

This document describes the specification and verification of **EtherFi TopUp contracts** using manual code review. The work was undertaken from **24/02/2025** to **03/03/2025**

The following contract list is included in our scope:

src/*

Excluding:

src/mocks/*

src/top-up/bridge/BridgeAdapterBase.sol

src/top-up/bridge/EtherFiOFTBridgeAdapter.sol

src/top-up/bridge/StargateAdapter.sol

The team performed a manual audit of all the Solidity contracts. During this review, the Certora team discovered bugs in the Solidity 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 | 1 | - | - |
| Low | 3 | 3 | 3 |
| Informational | 7 | 5 | 5 |
| Total | 11 | 8 | 8 |

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 |
|------|---|----------|--------------|
| M-01 | CREATE3 will not work in zkSync | Medium | Acknowledged |
| L-01 | BeaconFactory misses implementation for <code>UpgradeableBeacon::upgradeTo</code> | Low | Fixed |
| L-02 | Anyone can take ownership of TopUp's implementation | Low | Fixed |
| L-03 | Bridging native tokens is not possible | Low | Fixed |

Medium Severity Issues

M-01 CREATE3 will not work in zkSync

| Severity: Medium | Impact: Medium | Likelihood: Medium |
|-----------------------------|-----------------------|---------------------------|
| Files: BeaconFactory.sol | Status: Acknowledged | |

Description: The `BeaconFactory` contract relies on Solady's `CREATE3` to deploy `BeaconProxy` contracts. However, on zkSync Era, `CREATE2` address derivation differs from Ethereum's, causing the computed deterministic address to mismatch the actual deployed address. As a result, `_deployBeacon()` reverts when comparing `getDeterministicAddress(salt)` with the deployed address. Since zkSync modifies deployment logic and bytecode handling, using `CREATE3` for deterministic deployments on zkSync is unsafe and should be replaced with a zkSync-compatible approach. You can see the difference in how the addresses are computed and deployed when using `CREATE2` [here](#)

JavaScript

```
function _deployBeacon(bytes32 salt, bytes memory initData) internal returns (address) {
    address expectedAddr = this.getDeterministicAddress(salt);
    address deployedAddr =
address(CREATE3.deployDeterministic(abi.encodePacked(type(BeaconProxy).creationCode,
abi.encode(beacon(), initData)), salt));
    if (expectedAddr != deployedAddr) revert DeployedAddressDifferentFromExpected();

    emit BeaconProxyDeployed(deployedAddr);
    return deployedAddr;
}
```



Recommendations: For zkSync use a different approach to compute the address and deploy the proxy

Customer's response: Acknowledged

Low Severity Issues

L-01 BeaconFactory misses implementation for `UpgradeableBeacon::upgradeTo`

| Severity: Low | Impact: Low | Likelihood: Low |
|-----------------------------|--------------------|------------------------|
| Files: BeaconFactory.sol | Status: Fixed | |

Description: During the initialization of `BeaconFactory` in `__BeaconFactory_initialize`, the protocol deploys an `UpgradeableBeacon` and assigns `BeaconFactory` as its owner. However, `UpgradeableBeacon` has a function that allows only the owner to update the implementation, but this function is missing from the `BeaconFactory`. As a result, `BeaconFactory` cannot upgrade the beacon's implementation.

Recommendations: In `BeaconFactory` implement a function that calls `UpgradeableBeacon::upgradeTo`

Customer's response: Fixed in [branch](#)

Fix Review: Fix confirmed

L-02 Anyone can take ownership of TopUp's implementation

| Severity: Low | Impact: Low | Likelihood: Low |
|----------------------|--------------------|------------------------|
| Files: TopUp.sol | Status: Fixed | |

Description: The `TopUp` implementation contract includes an `initialize` function that only checks if `owner() != address(0)`, ensuring it can only be executed once. When `TopUpFactory` deploys the beacon proxy and calls `initialize`, the `owner()` is set, but the ownership state is stored in the beacon proxy's storage, not the implementation.

Since `TopUp` lacks a `_disableInitializers` call in its constructor, the `initialize` function remains callable on the implementation contract itself. This allows anyone to call `TopUp::initialize` and become the owner of the implementation contract. If the protocol later introduces logic that depends on `owner()`, an attacker could exploit this flaw to execute owner-restricted functions.

JavaScript

```
function initialize(address _owner) external {
  if (owner() != address(0)) revert AlreadyInitialized();
  _initializeOwner(_owner);
}
```

Recommendations: To prevent this, `_disableInitializers()` should be called in the constructor of `TopUp` to permanently lock the initialization function.

Customer's response: Fixed in commit [d02fdc4](#)

Fix Review: Fix confirmed

L-03 Bridging native tokens is not possible

Severity: **Low**

Impact: **Low**

Likelihood: **Low**

Files:
TopUpFactory.sol

Status: Fixed

Description: The `TopUpFactory` contract does not support bridging native tokens, nor does it provide a way to recover them if mistakenly sent.

1. Bridging Failure:

- The `TopUp` contract uses the address `0xEeeeeEeeeEeEeeEeEeEEEEEEEEEEEEEEEEEE` to represent native tokens.
- When calling `TopUpFactory::bridge` with this address, the call reverts due to `token.balanceOf()` since native tokens do not implement this function.
- As a result, native tokens cannot be bridged.

2. Unrecoverable Stuck Funds:

- The `recoverFunds` function attempts to transfer tokens using `token.safeTransfer()`, which does not work for native tokens.
- This means any native tokens sent to `TopUpFactory` will be stuck.

3. Leftover Bridging Fees Stuck in Proxy:

- Bridging fees are paid in native tokens by specifying a `msg.value` when calling `TopUpFactory::bridge`.
- If the fee sent is greater than what the bridge requires, the excess is returned to the factory.
- Since it lacks functionality to recover or use these excess funds, they remain inaccessible.

Recommendations: Allow bridging of native tokens by checking if the specified token address is the native token and take `address(this).balance` instead of `token.balanceOf()`

Customer's response: Fixed in commit [5d02ded](#)

Fix Review: Fix confirmed, there's no need to leave 0.01 ether for fees since `msg.value` is expected to cover them.

Informational Severity Issues

I-01. Internal upgradeable functions should have the `onlyInitializing` modifier

Description: `__BeaconFactory_initialize` and `__UpgradeableProxy_init` miss the `onlyInitializing` modifier that is usually used in internal initializing functions. Right now they can be called outside initialize function which should not be possible

Recommendation: Add the `onlyInitializing` modifier to these two functions

Customer's response: Fixed in commit [476ed1f](#)

Fix Review: Fix confirmed

I-02. `UpgradeableProxy::_setRoleRegistry` is not used anywhere

Description: `UpgradeableProxy::_setRoleRegistry` is not used anywhere. Moreover the role registry doesn't need an update because it will be a proxy so the implementation can be updated any time.

Recommendation: Remove that function

Customer's response: Acknowledged

I-03. `UpgradeableProxy` Comments Are Wrong

Description: The contract-level comments on `UpgradeableProxy` seem to be copy-pasted from `BeaconProxy.sol` and do not describe the contract correctly.

Check [here](#)

JavaScript

```
/**
 * @notice Factory contract for deploying beacon proxies with deterministic addresses
 * @dev This contract uses CREATE3 for deterministic deployments and implements UUPS
 * upgradeability pattern
```



Recommendation: Update the comments

Customer's response: Fixed in commit [7332433](#)

Fix Review: Fix confirmed

I-04. Use of Storage for Constant ETH Token Address

Description: The address value that designates the native ETH token is written to storage in `BridgeAdaperBase`, which means that every use of it will result in an SLOAD operation; it could be made constant (or better yet, obtained from the `Constants.sol` file when needed).

Check it [here](#)

```
JavaScript
    /// @notice Special address used to represent native ETH in token operations
    /// @dev Standard ETH placeholder address commonly used in DeFi
    address ETH = 0xEeeeeEeeeEeEeeEeEeEeEeEEeEEEEeEEEEeEEeE;
```

Recommendation: Take the ETH address from `Constants.sol`

Customer's response: Fixed in commit [485c569](#)

Fix Review: Fix confirmed

I-05. Unused Error In EtherFiOFTBridgeAdapter

Description: This `AmountOutOfFTLimit` error in `EtherFi0FTBridgeAdapter.sol` is unused.

JavaScript

```
/// @notice Error thrown when the bridged amount exceeds OFT limits
error AmountOutOfOFTLimit();
```

Recommendation: Consider removing that error if you don't plan to use it

Customer's response: Fixed in commit [695b01a](#)

Fix Review: Fix confirmed

I-06. Considering Indexing Some Event Parameters

Description: Consider making the token parameter indexed in these events to allow more easily filtering for all bridge events involving a certain token.

Check the [first](#) event and the [second](#)

JavaScript

```
event BridgeOFT(address token, uint256 amount, MessagingReceipt messageReceipt, OFTReceipt
oftReceipt);
event BridgeViaStargate(address token, uint256 amount, Ticket ticket);
```

Recommendation: Consider adding indexed parameters

Customer's response: Fixed in commit [485c569](#)

Fix Review: Fix confirmed

I-07. Compute minAmount in TopUpFactory Rather Than In Bridge Adapters

Description: Rather than computing a `minAmount` in every bridge adapter from the maximum slippage, a `minAmount` could be computed directly in the `TopUpFactory` and passed to



`adapter.bridge()` instead of `maxSlippageInBps`. This would reduce code duplication and make the adapters simpler.

Recommendation: Consider calculating the `minAmount` in the `TopUpFactory`

Customer's response: Acknowledged

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.