Security Assessment

Item 9 Internal Report

February 2026

Prepared for ether.fi

Disclaimer

This document is intended for **internal** use only.

Initial Commit Hash	Final Commit Hash
e24c1c2	54150cc

Informational Issues

I-01. Inaccurate Natspec documentation for redeemFraxToUsdc regarding error handling

Description: The `redeemFraxToUsdc` function in `SettlementDispatcherV2` includes Natspec documentation stating that it throws an `InvalidValue` error. However, the function body does not contain a check that would revert with this specific error (e.g., checking if the amount is zero). It only checks for configuration presence, balance sufficiency, and return amount sufficiency.

Recommendation: Update the Natspec documentation to accurately reflect the errors thrown by the function, or implement the missing check for zero amount if it was intended.

Customer's response: Fixed in commit 54150cc

Fix Review: Fixed

I-02. Lack of output asset verification in Frax configuration

Description: The `setFraxConfig` function allows an admin to configure a Frax custodian address, and `redeemFraxToUsdc` utilizes this custodian to redeem Frax tokens. However, the system does not verify that the configured custodian actually outputs USDC. Since `redeemFraxToUsdc` implies a specific output asset (USDC) by its name and intended logic, a misconfiguration could lead to unexpected behavior if the custodian returns a different asset.

Recommendation: In `setFraxConfig`, verify that the configured custodian's output asset is USDC, or document that this validation is admin's responsibility.

Customer's response: Fixed in commit 54150cc

Fix Review: Fixed

I-03. Discrepancy in validation between quoteAsyncFraxRedeem and redeemFraxAsync

Description: The `quoteAsyncFraxRedeem` function provides fee quotes for asynchronous Frax redemptions but lacks input validation present in the execution function `redeemFraxAsync`. Specifically, it does not check if the amount is non-zero or if it meets the `DUST_THRESHOLD` requirement (multiples of 1e12). This can lead to the function returning valid quotes for amounts that would inevitably revert when attempting the actual redemption.

Recommendation: Align the input validation in `quoteAsyncFraxRedeem` with `redeemFraxAsync` by adding checks for zero amounts and dust thresholds.

Customer's response: Acknowledged - *"This is fine. Doesn't add any additional risk as the `redeemFraxAsync` function will revert when necessary"*

Fix Review: Acknowledged

I-04. Missing pausable check in redeemFraxAsync

Description: The `bridge` function in `SettlementDispatcherV2` is protected by the `whenNotPaused` modifier, allowing the protocol to halt bridging operations in emergencies. However, `redeemFraxAsync`, which performs a similar cross-chain bridging operation for Frax, lacks this modifier. This inconsistency means that even if the contract is paused, Frax bridging operations could still be executed.

Recommendation: Add the `whenNotPaused` modifier to the `redeemFraxAsync` function to ensure consistent pause functionality across all bridging operations.

Customer's response: Fixed in commit 54150cc

Fix Review: Fixed

I-05. Potential zero output in redeemFraxToUsdc for small amounts

Description: The `redeemFraxToUsdc` function facilitates the conversion of FraxUSD (18 decimals) to USDC (6 decimals). Due to the decimal precision loss during conversion in `RemoteCustodianUsdcScroll` (dividing by 1e12), input amounts smaller than 1e12 wei will result in 0 USDC output. While the function has a `minReceive` check, passing `minReceive` as 0 would allow the transaction to succeed while burning non-0 FraxUSD for 0 USDC.

Recommendation: Enforce a minimum input amount of `1e12` inside `redeemFraxToUsdc` to prevent caller from accidentally redeeming amounts that result in zero output.

Customer's response: Fixed in commit 54150cc

Fix Review: Fixed

I-06. Typo in IFraxCustodian interface parameter name

Description: There is a typo in the `IFraxCustodian` interface definition where the parameter `receiver` is misspelled as `reciever`. While this does not affect the contract's functionality or compilation (as Solidity uses types for function selectors), correcting it improves code readability and professionalism.

Recommendation: Correct the spelling of `reciever` to `receiver` in the `IFraxCustodian` interface.

Customer's response: Fixed in commit 54150cc

Fix Review: Fixed

I-07. Inability to disable Midas redemption vaults

Description: The `setMidasRedemptionVault` function in `SettlementDispatcherV2` enforces a check that prevents setting the `redemptionVault` address to `address(0)`. This restriction makes it impossible to disable or remove a configured Midas redemption vault once it has been set, potentially locking a configuration even if a vault is deprecated.

Recommendation: Remove the check ensuring `redemptionVault != address(0)` to allow administrators to disable vaults by setting them to the zero address.

Customer's response: Acknowledged

Fix Review: 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.