certora

Security Assessment Report

aave

UmbrellaBatchHelper

March-2025

Prepared for:
Aave DAO

Code developed by:
BORED
GHOSTS
DEVELOPING

certora

aave

Table of content

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
UmbrellaBatchHelper	Github Repository	a05713e	EVM

Project Overview

This document describes the verification of **UmbrellaBatchHelper** code using manual code review. The work was undertaken from **February 25** to **March 4, 2025**.

The following contracts are considered in scope for this review:

- `src/contracts/helpers/UmbrellaBatchHelper.sol`
- `src/contracts/helpers/interfaces/IUmbrellaBatchHelper.sol`
- `src/contracts/helpers/interfaces/IUniversalToken.sol`

The team performed a manual audit of all the solidity contracts. During the audit, Certora didn't find any significant issues in the code.

Protocol Overview

The **UmbrellaBatchHelper** is a utility contract designed to facilitate users' interaction with umbrella by consolidating multiple transactions into a single one. It simplifies deposits, withdrawals, restaking, and cooldown activation for holders of multiple StakeTokens.

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	-	-	-
Total	-	-	-

Severity Matrix

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

Detailed Findings

Audit Goals

1. Initialization of stake tokens is completely permissionless, so it's crucial to ensure the registration and configuration of the token in the contract are independent of the caller—e.g., sender address, input, etc.

2. User actions should be permitted and facilitated only on official stake tokens.

3. Calls with permits should handle replay griefing.

4. The main functions that move assets around - `deposit`, `redeem` and `claimRewards` - transfer the entire sums as specified without accumulating or holding any funds at the end of the execution.

5. Since the contract uses `token.BalanceOf`, it's worth focusing on its usage and checking that donations cannot be used to manipulate the system.

Coverage and Conclusions

1. Initialization populates the token's data regardless of the caller's identity or the calling context. It is treated in the following way - if the stake token exists and is registered as a distributor in the official rewards controller.
    a. An initialized staked token cannot be reinitialized or reconfigured within the helper contract.
    b. The helper verifies against the official Aave rewards controller that the token requested for initialization is indeed an official stake token that is eligible for incentives.

   c. All the data is extracted from the staked token through a chain of calls to view functions of the stake token itself (trusted due to point b. above) and its underlying asset.

2. The contract validates that the token is initialized before it allows any deposit/redemption/claim and even uses the initialization agnosticism to the caller identity to simply initialize a token if it's official (1.b).

3. Function with permit either uses a best practice of calling through try-catch flow or, in some cases, is exempt from it due to a clever construct of the permit digest.

4. For static balanced tokens, the transfers are exact and do not under/over transfer assets when passed between the user <-> helper <-> stake token, or rewards controller <-> helper <-> user/stake token. However, dynamically balanced tokens, e.g. aTokens, are handled with logic that ensures:
   a. The successful execution of the operation
   b. Approvals are fully utilized, i.e., transfers are made on the entire approved sum, so at the end of execution, all approvals end at 0.
   c. No major accumulation of assets to the helper contract.

   In the scenario where the transfer into the helper contract rounds up the specified value, a small dust amount equal to the rounding excess stays at the helper contract. Since the approvals are nullified in every execution branch and transfers are made at most to the specified amount, this dust will keep accumulating, and unauthorized users will not be able to use it.
   It's important to note that we're talking about extremely low sums on each round-up scenario of aToken, in the magnitude of 0.5 aToken index * smallest unit of the token.

5. Following the conclusion from the previous point (4), it's apparent that donations are treated exactly as round-ups. Donations to the system do not affect the proper functioning of the contract. These assets can be rescued at any time with an authorized function.

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.