Security Assessment Report

AAVE-CAPO

02/24

Prepared for
Aave

Table of contents

Project Summary

Project Scope

Repo Name	Repository	Commits	Compiler version	Platform
aave-capo	Github Repository	bb84843	0.8.21	multichain

Project Overview

This document describes the verification of the **Aave-CAPO** using manual code review findings. The work was undertaken from **28 January 2024** to **6 February 2024**.

The following contract list is included in our scope:

- `PriceCapAdapterBase`
- `PriceCapAdapterStable`

The team performed a manual audit of all the Solidity contracts**.** During the audit, the Certora team discovered the findings listed below.

Protocol Overview

The contracts introduce a Correlated-asset price oracle, an adapter smart contract introducing extra upper price protections for assets highly correlated with an underlying, like LSTs (Liquid Staking Tokens) or stablecoins.

Audit Goals

Verify that the above contracts act as they are supposed to. More Specifically:

1. Check that the implementation is in accordance with the high-level rationale.
2. Check that the calculations are matching the description given in the README file.
3. Read the tests of the project and look for possible issues in them.

Coverage

1. We verified that critical functions such as *setPriceCap()* and *setCapParameters()* have proper access-control.
2. For both adapters, we checked that the price returned by the *latestAnswer()* function has the same units as the *basePrice*.
3. In the *priceCapAdapterBase* contract, we made sure that the *_snapshotTimestamp* variable could never decrease.
4. In the *priceCapAdapterBase* contract, we made sure that any update of the *_snapshotTimestamp* variable occurs at least MINIMUM_SNAPSHOT_DELAY seconds after the time represented in that new value.
5. In the *priceCapAdapterBase* contract, we went over the mathematical calculation of the maximal ratio and approved that it is according to the specification.
6. We made sure that the returned price is capped either by the *priceCap* (in the case of the stable adapter) or by the *basePrice* times *maxRatio* (in the case of the base adapter).
7. We ran the tests against a mutated code, and verified that indeed the tests failed. We did so for every test of every file in the directory *tests/*.

Findings Summary

We didn't find any bugs/issues with the above mentioned contracts.

Disclaimer

The Certora Prover takes a contract and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification and the Certora Prover does not check any cases not covered by the specification.

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.