

Security Assessment Report



June-2025

Prepared for:

Aave DAO

Code developed by:







Project Summary	
Project Scope	
Project Overview	
Protocol Overview	
Findings Summary	
Severity Matrix	
Detailed Findings	
Audit Goals	
Coverage and Conclusions	5
Disclaimer	
About Certora	8





Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Aave v3.4 AIP	Github Repository	<u>e958112</u>	EVM

Project Overview

This document describes the verification of **Aave v3.4 AIP** code using manual code review. The work was undertaken from **May 8** to **June 11, 2025**.

The following contracts are considered in scope for this review:

- src/ATokenMainnetInstanceGHO.sol
- src/CustomInitialize.sol
- src/L2PoolInstanceWithCustomInitialize.sol
- src/PoolConfiguratorWithCustomInitialize.sol
- src/PoolInstanceWithCustomInitialize.sol
- src/UpgradePayload.sol
- src/UpgradePayloadMainnet.sol
- src/VariableDebtTokenMainnetInstanceGHO.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 **Aave v3.4 AIP** is the upgrade payload dedicated to the transition of Aave v3.3 to v3.4. It consists of AToken and VToken contracts designed specifically for the GHO token which requires special handling, the Pool instances and the Payload contract for Mainnet as well as the Payload contract for other networks. The upgrade changes the behavior of all the reserves and pools across all chains and the steps are performed in a specific order to maintain pre-existing states.



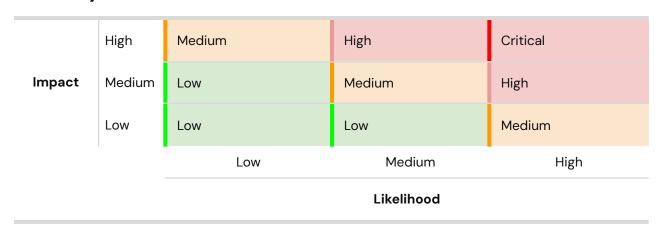


Findings Summary

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







Detailed Findings

Audit Goals

- 1. The aGHO implementation should delete deprecated variables slots to align with the common aToken implementation.
- 2. The vGHO implementation should delete deprecated variables slots to align with the common vToken implementation.
- 3. The upgrade of aGHO should first distribute fees to the treasury.
- 4. The new GHODirectMinter should have the same bucket capacity as aGHO currently has.
- 5. The total amount of GHO minted should remain the same after the upgrade.
- 6. aGHO contract should be removed from facilitators with no residual privileges.
- 7. aGHO and aAave tokens must be upgraded differently than other ATokens.
- 8. All existing reserves should be upgraded with the new AToken and VToken versions
- 9. GHO reserve's supply cap and reserve factor should be updated accordingly in the pool to align with the new design.
- 10. Deprecated reserve fields stored in the pool should be correctly shifted to prevent any clash.
- 11. Any GHO allowance given by the DAO to Umbrella should be converted from GHO allowance to aGHO allowance, to support virtual accounts of GHO.

Coverage and Conclusions

1. Inheritance chain is respected and GhoAToken slots are aligned with v3.4 AToken. The only difference was GhoAToken had 2 additional variables declared _ghoVariableDebtToken and





_ghoTreasury which are correctly deleted during ATokenMainnetInstanceGHO::initialize() as per the storage layout. These variables are marked as deprecated to prevent unexpected use.

2. The special vGHO implementation correctly deletes storage slots dedicated to the deprecated discount model with the exception of the _ghoUserState mapping which can't be realistically cleared.

The BGD team is aware of this behavior and will take it into account in future vGHO upgrades.

- 3. The payload responsible for the upgrade on Mainnet starts by distributing fees before updating the aGHO implementation. This effectively transfers the entire contract balance of GHO to the treasury resulting in a O token balance.
- 4. The payload correctly retrieves the aGHO current bucket capacity and assigns it to the new facilitator, namely GHODirectMinter contract.
- 5. The amount of GHO minted by the new facilitator is used to burn an equivalent amount from the old facilitator, keeping the correct amount of minted GHO when the upgrade is done. Additionally, all GHO is accounted for whatever needs to be swept into the treasury is, and all previously unaccounted profits are accounted for. Additionally any deficit is closed as a preprocessing step to the upgrade.
- 6. After reducing aGHO bucket level to O, the contract address is removed from the GHO facilitators by the payload contract.
- 7. There is a special logic on Mainnet's payload that prevents Aave and GHO reserves from being upgraded like the other reserves.
- 8. The payload loops through the entire reserves list retrieved from the pool and updates the corresponding AToken and VToken contracts.
- 9. GHO supply cap is set to 1, effectively preventing the reserve from being supplied in the protocol. Its reserve factor is set to 100%, effectively redirecting all interest generated from debt to the treasury.





10. The payload correctly:

- upgrades the Pool implementation to v3.4
- takes, for each reserve, the <u>__deprecatedVirtualUnderlyingBalance</u> field (located at the last field of the v3.4 struct) and copies the value to <u>virtualUnderlyingBalance</u> field (now located at the 3rd last field of the v3.4 struct).
- sets the __deprecatedVirtualUnderlyingBalance field to 0 (since it won't be used anymore).

11. All DAO GHO allowance is converted to aGHO allowance in step 1 of the execute 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.