Blackthorn

Security Review For **Aave v3.4**

Collaborative Audit Prepared For:	**Aave DAO**
Code Developed By:	**BGD Labs**
Lead Security Expert(s):	**bughuntoor**
	mstpr-brainbot
	pkqs90
Date Audited:	**May 27 - June 8, 2025**

Introduction

Aave v3.4 is an upgrade to the Aave v3 protocol, currently running on v3.3 in production across all networks.
It includes the following changes and improvements:

- Migration of the custom GHO logic and a/v tokens to a model the same as any other asset, to simplify overall the codebase and its reasoning.
- Addition of Multicall support on the Pool contract.
- Introduction of a Position Manager role for users to assign to other addresses, allowing them to do a subset of actions on their behalf: switching Liquid modes, and enabling/disabling an asset as collateral.
- Removal of the unused BridgeLogic and the concept of “unbacked” in the protocol, never used either.
- Make different variables immutables, to align with the high-level nature of never-to-be-changed.
- Refactor the Error logic to use Error signatures instead of error codes.
- In addition to the migration of GHO from custom to standard, unification of aTokens’ storage and implementation for all assets (aAAVE was different from others due to its role in governance voting).
- Upgrade the compilation version to 0.8.27 to improve overall compatibility with tooling and dependencies.
- Multiple minor misc improvements and optimizations.

An exhaustive explanation of all changes included can be found on
Aave-v3.4-features.md

Scope

Repository: aave-dao/aave-v3-origin

Audited Commit: bb66c4b0e3374bd4fc4c6ccd6381e659048ff6be

Final Commit: bb66c4b0e3374bd4fc4c6ccd6381e659048ff6be

Files:

- src/contracts/helpers/AaveProtocolDataProvider.sol
- src/contracts/helpers/L2Encoder.sol
- src/contracts/instances/ATokenInstance.sol
- src/contracts/instances/ATokenWithDelegationInstance.sol
- src/contracts/instances/L2PoolInstance.sol
- src/contracts/instances/VariableDebtTokenInstance.sol
- src/contracts/misc/DefaultReserveInterestRateStrategyV2.sol
- src/contracts/misc/aave-upgradeability/VersionedInitializable.sol
- src/contracts/protocol/configuration/ACLManager.sol
- src/contracts/protocol/configuration/PoolAddressesProviderRegistry.sol
- src/contracts/protocol/libraries/configuration/EModeConfiguration.sol
- src/contracts/protocol/libraries/configuration/ReserveConfiguration.sol
- src/contracts/protocol/libraries/configuration/UserConfiguration.sol
- src/contracts/protocol/libraries/helpers/Errors.sol
- src/contracts/protocol/libraries/logic/BorrowLogic.sol
- src/contracts/protocol/libraries/logic/ConfiguratorLogic.sol
- src/contracts/protocol/libraries/logic/EModeLogic.sol
- src/contracts/protocol/libraries/logic/FlashLoanLogic.sol
- src/contracts/protocol/libraries/logic/GenericLogic.sol
- src/contracts/protocol/libraries/logic/IsolationModeLogic.sol
- src/contracts/protocol/libraries/logic/LiquidationLogic.sol
- src/contracts/protocol/libraries/logic/PoolLogic.sol

- src/contracts/protocol/libraries/logic/ReserveLogic.sol
- src/contracts/protocol/libraries/logic/SupplyLogic.sol
- src/contracts/protocol/libraries/logic/ValidationLogic.sol
- src/contracts/protocol/libraries/math/MathUtils.sol
- src/contracts/protocol/libraries/types/ConfiguratorInputTypes.sol
- src/contracts/protocol/libraries/types/DataTypes.sol
- src/contracts/protocol/pool/L2Pool.sol
- src/contracts/protocol/pool/Pool.sol
- src/contracts/protocol/pool/PoolConfigurator.sol
- src/contracts/protocol/pool/PoolStorage.sol
- src/contracts/protocol/tokenization/AToken.sol
- src/contracts/protocol/tokenization/ATokenWithDelegation.sol
- src/contracts/protocol/tokenization/VariableDebtToken.sol
- src/contracts/protocol/tokenization/base/DebtTokenBase.sol
- src/contracts/protocol/tokenization/base/DelegationMode.sol
- src/contracts/protocol/tokenization/base/IncentivizedERC20.sol
- src/contracts/protocol/tokenization/base/MintableIncentivizedERC20.sol
- src/contracts/protocol/tokenization/base/ScaledBalanceTokenBase.sol
- src/contracts/protocol/tokenization/delegation/BaseDelegation.sol

---

Repository: bgd-labs/protocol-v3.4-upgrade-blackthorn

Audited Commit: 597c064ebabb3db1c6196bedd807ceb39828727b

Final Commit: d37769db892b03211f1e5a567265f5e6b5fd8af0

Files:

- src/ATokenMainnetInstanceGHO.sol
- src/CustomInitialize.sol
- src/L2PoolInstanceWithCustomInitialize.sol
- src/MainnetCorePoolInstanceWithCustomInitialize.sol
- src/PoolConfiguratorWithCustomInitialize.sol
- src/PoolInstanceWithCustomInitialize.sol
- src/UpgradePayload.sol
- src/UpgradePayloadMainnet.sol
- src/VariableDebtTokenMainnetInstanceGHO.sol

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	0

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Security Experts Dedicated to This Review

@bughuntoor

[illegible]

@mstpr-brainbot

Tapir

@pkqs90

Pkqs90

Disclaimers

Blackthorn does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.

Security Review Process

Attack Surfaces/areas explored and areas verified by written tests:

- Multicall is now used, explored the potential of any new attack surfaces due to it. The process yielded no results as `msg.value` is not used anywhere, anything that's currently feasible has been this way without multicall too.
- Isolation mode logic is now moved from directly within BorrowLogic, to `IsolationModeLogic.increaseIsolatedDebtIfIsolated` - logic needs to be identical to the one in `BorrowLogic` and it is.
- `__deprecatedInterestRateStrategyAddress` will remain with the current interest rate on the currently active reserves. New reserves will not have it. Explored the potential of an impact here. There is none. Protocol should be careful if they decide to use the same storage slot for something else.
- GenericLogic switched from iterating through all reserves, to just checking the necessary through the bitmaps. Checked against the correct logic implementation of the iteration (w/ flags and bitmap) and found no issues.
- All variables in the new GenericLogic iteration were checked if calculated the same way as previously. No issues found.
- Checked against the new PositionManager functions being used for accidental or purposeful direct instant loss of funds. Found that the worst case scenario is choosing eMode / collateral assets such as borrow up to LTV and waiting until LT (Liquidation Threshold). This was found as being an unrealistic scenario that cannot practically happen by accident without intervening on time.
- Double checked that msg.sender is correctly replaced with user/borrower (and not

onBehalfOf) everywhere.

- Verified all existing assets have a zero `.backed` field.
- The updated formula in `MathUtils.calculateCompoundedInterest` is virtually impossible to overflow. The discrepancy between the current formula and the original is ~0.033% over 7 years at an 8% annual rate (174.60693% vs. 174.54868%).
- Invariants after the Gho upgrade:
  - `vGho.totalSupply` does not change.
  - Treasury correctly receives aGho after calling `Pool.mintToTreasury([Gho])`.
  - Treasury can transfer aGho to other users, and users can correctly withdraw Gho from aGho.
  - Gho cannot be supplied due to a supply cap of 1.
- Gho can be flash-loaned as expected.
- Executing the upgrade during a flash loan always reverts, whether:
  - the flash loan repays the Gho (it reverts because the aToken lacks an `IAToken.handleRepayment()` interface), or
  - the flash loan creates a borrow position (it reverts due to changed return values in the upgraded `IVariableDebtToken.mint()`).
- The storage layout for aGho, vGho, ATokenWithDelegationInstance remains identical before and after the upgrade.
- vGho's new `accruedTreasury` correctly covers all pending interest.

Other observations:

1. Gho cannot be supplied, so there’s no easy way to obtain aGho. Currently, liquid aGho only exists via treasury fees (aGho held by GhoDirectMinter is non-transferable). This may hinder deficit elimination since only aTokens can be used for repayment.
2. Some contracts weren’t upgraded (e.g., `PoolAddressProviderRegistry`, `DefaultReserveInterestRateStrategyV2`, `ACLManager`). Given their minimal differences, leaving them unchanged makes sense.

3. In `ATokenWithDelegation.sol::_transfer()`, the delegation update amount was corrected from amount to `amount.rayDiv(index)` (old code vs. new code). This change aligns with an issue the team was aware of, but had decided to keep that way because the Aave Governance side assumes index 1. The change has been made to align naturally since the team is now touching teh component and generalizing it.

4. In V3.4, the protocol switched to using Multicall + Context libraries. Now, `_msgSender()` is used instead of `msg.sender`. While this is not a problem on its own, protocol team must be careful to not implement TrustedForwarder-like logic which could change the value of `_msgSender()` in order not to introduce the Critical issue of Multicall + ERC2771.

5. Although it was not explicitly mentioned in the changes readme, in V3.4 the protocol changed its logic when opening a borrow position to always set the borrowing flag, instead of to rely on whether its a first borrow, based on the returned value of the debt token. This approach is much safer, as if somehow (e.g. due to rounding + special implementation) a user is able to remain with borrowing flag off + 1 wei of debt token, they will now not be able to steal more funds from the protocol. (Previously, they'd be able to borrow the entire reserve and the flag would remain off).

6. The protocol team was concerned about the case where the upgrade tx is executed within a user flashloan and data is written to wrong storage slot. They correctly noted that it would revert, due to `handleRepayment` being removed. However, there's also a case where the flashloan would not be repaid, but instead a borrow should be opened. This case would also revert, due to the change in the return values of debt token's mint method.

7. While it is assumed that the protocol has their own reasons to execute their upgrades in a permisionless way, this is an extra risk by the protocol. Currently, the only function with a callback is the flashloan. If the protocols wants to / has to keep upgrades permissionless, it is advised they disable flashloans for a very short period of time during upgrade.