Security Assessment & Formal Verification Report

Gho Direct Minter

December-2024

Prepared for:
Aave DAO

Code developed by:
BORED GHOSTS DEVELOPING

Table of content

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
GhoDirectMinter	Github	eb0af96	EVM/Solidity 0.8

Project Overview

This document describes the specification and verification of the **GhoDirectMinter** using manual code review. The work was undertaken from **16December 2024 to 19 December 2024**.

The following contract list is included in our scope:

- GhoDirectMinter.sol

During our manual audit, no bugs or issues were found.

Protocol Overview

This contract is a Gho facilitator that will be used in block chains other than Etherium. Its usage is supplying Gho liquidity to the pool.

Coverage

We checked the following aspects of the contract:

1. **Use of ReserveDataLegacy in Constructor**

Observation: The constructor fetches reserve data using *pool.getReserveData(...)* and references the *ReserveDataLegacy* structure, rather than the updated *ReserveData*.

Analysis: This decision is intentional and focuses on efficiency. The contract only requires the *aTokenAddress* from the reserve data, which is available in *ReserveDataLegacy*. Using this minimal structure avoids fetching unnecessary fields, thereby optimizing gas consumption.

2. **_disableInitializers(...) Call in Constructor**

Observation: The *_disableInitializers(...)* function is called in the constructor, raising questions about its necessity.
Analysis: The call to *_disableInitializers(...)* is critical to ensure that the contract cannot be re-initialized after deployment. Placing this call in the constructor guarantees that initialization is conclusively sealed and prevents inherited contracts from accidentally enabling initialization logic. This defensive approach fortifies the contract's security posture against potential misconfigurations or malicious interactions.
3. **Supply Cap Handling in mintAndSupply(...)**
Observation: The *mintAndSupply(...)* function temporarily sets the *supplyCap* to zero before supplying liquidity to the pool.
Analysis: This mechanism is crucial for enabling the contract to supply liquidity even when the reserve is close to its cap. The Aave protocol's *validateSupply(...)* logic does not fail when the *supplyCap* is zero, allowing the operation to proceed. By temporarily disabling the cap, the function ensures that the contract can execute its mint-and-supply operations as intended. Restoring the original *supplyCap* immediately after the operation maintains the system's integrity. This design is both intentional and necessary to achieve the desired functionality.
4. **Preventing Excessive Burns in withdrawAndBurn(...)**
Observation: The *withdrawAndBurn(...)* function withdraws *GHO* tokens from the pool and burns them. Concerns were raised about whether it can burn more tokens than it minted.
Analysis: The GHO token's internal mechanics prevent excessive burns. Specifically, the *IGhoToken.burn(...)* function ensures that facilitators cannot burn more tokens than they have minted. This safeguard is enforced within the GHO token contract itself, eliminating the risk of over-burning. This design choice ensures that token supply remains consistent with the facilitator's activities.
5. **Configuration of GHO_A_TOKEN in transferExcessToTreasury(...)**
Observation: The *transferExcessToTreasury(...)* function uses *GHO_A_TOKEN*, which must be correctly configured to point to the default AToken address for *GHO*.
Analysis: The default configuration ensures that *GHO_A_TOKEN* correctly represents the AToken address for GHO. This setup allows functions like *IERC20.transfer(...)* to work seamlessly, as they rely on standard AToken behavior.

Findings Summary

We didn't find any bugs or issues. The contract is well-prepared for its role as a *GHO* facilitator within the Aave ecosystem, with no vulnerabilities or inefficiencies identified.

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.