GHO Wrapper Security Review

Pashov Audit Group

Conducted by: sashik-eth, Said, ast3ros

February 7th 2025 - February 8th 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **aave/gho-wrapper** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About GHO Wrapper

GHO Wrapper is a smart contract that wraps the ERC20 token GHO, enabling deposits, withdrawals, and permit-based approvals for seamless integration. The token metadata (name and symbol) is configurable, allowing the deployer to set custom values during initialization.

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - 57fcab6d61b20eea96f5fce1e88b2f7d4438cbf8

Scope

The following smart contracts were in scope of the audit:

- `WrappedGhoToken`

7. Executive Summary

Over the course of the security review, sashik-eth, Said, ast3ros engaged with Avara to review GHO Wrapper. In this period of time a total of **1** issues were uncovered.

Protocol Summary

Protocol Name	GHO Wrapper
Repository	https://github.com/aave/gho-wrapper
Date	February 7th 2025 - February 8th 2025
Protocol Type	Token wrapper

Findings Count

Severity	**Amount**
Low	1
Total Findings	**1**

Summary of Findings

ID	**Title**	**Severity**	**Status**
[L-01]	Add emergency permit signature cancellation mechanism	Low	Acknowledged

8. Findings

8.1. Low Findings

[L-01] Add emergency permit signature cancellation mechanism

WrappedGhoToken uses `ERC20PermitUpgradeable` for gasless token approvals. While users can technically invalidate previous permit signatures by creating new ones with the same nonce and submitting a transaction to invalidate the old one, this process requires signature generation and transaction submission, which may not be suitable in urgent situations. In emergency scenarios, a faster and simpler cancellation mechanism would enhance security.

Consider adding a direct `nonce` invalidation function to cancel signatures that users can call in emergency situations:

```
function invalidatePermitNonce() external returns (uint256) {
        return _useNonce(msg.sender);
    }
```