certora

Security Assessment & Formal Verification Report

Custom CCIP TokenPool contracts for GHO

May-2024

Table of content

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Modified CCIP	aave-ccip	d6cb9da	EVM/Solidity 0.8

Project Overview

This document describes the specification and verification of the **Modified CCIP** using the Certora Prover and manual code review findings. The work was undertaken from **8 May 2024** to **11 June 2024**.

The scope of our review is the modifications that were done in the following contracts:

- UpgradeableLockReleaseTokenPool
- UpgradeableBurnMintTokenPool
- UpgradeableBurnMintTokenPoolAbstract
- UpgradeableLockReleaseTokenPool
- UpgradeableTokenPool

For more information about the modifications please refer to the following PR.

The Certora Prover demonstrated that the implementation of the Solidity contracts above is correct with respect to the formal rules written by the Certora team. In addition, the team performed a manual audit of all the Solidity contracts**.** During the verification process and the manual audit, no bug was discovered. (Anyhow we have 2 informational issues that we list below.)

Protocol Overview

The contracts under review are customized versions of the Chainlink Cross-Chain Interoperability Protocol (CCIP) contracts, specifically designed for the GHO cross-chain strategy. These include the UpgradeableBurnMintTokenPool and UpgradeableLockReleaseTokenPool, which are based on the CCIP's BurnMintTokenPool and LockReleaseTokenPool from the version 2.8.0 release, respectively. The modifications made are as follows:

1. Soften Solidity compiler version of all base contracts of BurnMintTokenPool and LockReleaseTokenPool to ensure compatibility with the gho-core contracts.

2. Addition of upgradeability mechanism using the battle-tested transparent proxy pattern.
3. Introduction of a bridge limit on UpgradeableLockReleaseTokenPool to regulate the maximum amount of tokens that can be transferred out (either burned or locked) from Ethereum to other chains, thereby enhancing UX by enabling the precise calculation of GHO liquidity available for minting on remote networks.

Coverage

1. We wrote several rules for the contract UpgradeableLockReleaseTokenPool. See more information later.
2. With respect to manual auditing we went over the ARFC and the code and we saw that the implementation matches the described changes. We mainly focused on the bridged limit which sets a limit to the amount of token that can be bridged from Ethereum to other chains.

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	2		
Total			

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
I-01	Superfluous message sender checking.	Informational	Acknowledged
I-02	s_bridgeLimit can be smaller than s_currentBridged.	Informational	Acknowledged

Informational Severity Issues

I-01. Superfluous message sender checking

Description: In the contract UpgradeableLockReleaseTokenPool the function provideLiquidity() requires rebalancer permissions even though anyone can provide liquidity to the contract, simply by transferring tokens to the contract. The only difference between an unauthorized transfer of tokens to an authorized call to provideLiquidity() function would be the addition of the LiquidityAdded() event in the latter.

Aave Labs response: Acknowledged. Given the aim to minimize differences between this contract and the standard CCIP implementation, we prefer to maintain the current logic.

I-02. s_bridgeLimit can be smaller than s_currentBridged.

Description: In the contract UpgradeableLockReleaseTokenPool the function setBridgeLimit() allows the owner to set the limit of bridged amount to any value, even if it is already smaller than the current bridged amount.

Recommendation: None

Aave Labs response: Modifications to the bridge limit undergo DAO assessment and review prior to implementation. It's essential to evaluate general aspects of GHO liquidity and related controls, such as GHO Token Facilitator Bucket Capacity, to ensure the configuration is appropriate.

A comment has been added to caution about the implications of altering the bridge limit: https://github.com/aave/ccip/pull/3/commits/64cc73d6312fb302d65a668fd7dd1c6602edbfc8

Formal Verification

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counter-example exists that violates one of the assertions of the rule.

Formal Verification Properties

In the table below we specify all the formally verified rules that we wrote for the contract UpgradeableLockReleaseTokenPool, and give a detailed description for them. A link to the Certora's prover report can be found here.

P-01. currentBridge_LEQ_bridgeLimit	
Status: Verified	

Rule Name	Status	Description	Rule Assumptions
currentBridge_LEQ_bridgeLimit	Verified	The value of the variable s_currentBridged is less or equal to the value of the variable s_bridgeLimit.	This rule can be violated if one calls the function setBridgeLimit(...) with a value smaller than s_currentBridged.

P-02. withdrawLiquidity_correctness

Status: Verified

Rule Name	Status	Description
withdrawLiquidity_correctness	Verified	The rule checks that while calling the function withdrawLiquidity(), the balance of the contract changes accordingly.

P-03. provideLiquidity_correctness

Status: Verified

Rule Name	Status	Description
provideLiquidity_correctness	Verified	The rule checks that while calling the function provideLiquidity(), the balance of the contract changes accordingly.

P-04. only_bridgeLimitAdmin_or_owner_can_call_setBridgeLimit

Status: Verified

Rule Name	Status	Description
only_bridgeLimitAdmin_or_owner_can_call_setBridgeLimits	Verified	The rule checks that only the bridgeLimitAdmin or the owner can call the function setBridgeLimits().

P-05. only_lockOrBurn_can_increase_currentBridged

Status: Verified

Rule Name	Status	Description
only_lockOrBurn_can_increase_currentBridged	Verified	The rule checks that the only function that can increase the value of s_currentBridged is lockOrBurn().

P-06. only_releaseOrMint_can_decrease_currentBridged

Status: Verified

Rule Name	Status	Description
only_releaseOrMint_can_decrease_currentBridged	Verified	The rule checks that the only function that can decrease the value of s_currentBridged is releaseOrMint().

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.