certora

Security Assessment Report

aave

Aave Governance V3 MerklePatriciaTrieVerifier Bug Fix Review

March 2025

Prepared for:
Aave DAO

Code developed by:
BORED
GHOSTS
DEVELOPING

Table of contents

Project Summary

Project Scope

Project Name	Repository (link)	Commit Hash	Platform
DataWarehouse and merklePatriciaTrieVerifier	https://github.com/bgd-labs/aave-governance-v3	PR#66	EVM

Project Overview

This document describes the manual code review findings of **Aave Governance**. The following contract list is included in our scope:

- `src/contracts/voting/DataWarehouse.sol`
- `src/contracts/voting/libs/MerklePatriciaProofVerifier.sol`

The work was undertaken from **March 16, 2025,** to **March 19, 2025**. During this time, Certora's security researchers performed a manual audit of the Solidity contracts with the results being summarized in the subsequent section.

Protocol Overview

This report reviews the Aave Governance v3 fix for a proof-handling bug in `src/contracts/voting/libs/MerklePatriciaProofVerifier.sol` that let a truncated inclusion proof end at a branch while key nibbles remained, returning empty bytes and being misread as an exclusion. The vulnerability allowed an attacker to reset the voting power of any account, effectively preventing them from participating in governance. The patch adds a guard in the branch case to revert if the proof ends before descending to the child, restoring correct inclusion and exclusion semantics. We also reviewed the integration in `src/contracts/voting/DataWarehouse.sol`, which now requires `exists == true` before persisting roots or slots, preventing false exclusions from being recorded.

Audit Goals

1. The code should ensure the last element of the provided stack is **not** a branch node to prevent a truncated inclusion proof from being interpreted as an exclusion proof. When processing a branch node, the logic should verify that the end of the proof stack has not been reached yet and that there are remaining nodes to be processed in the stack.
2. DataWarehouse only saves results when `exists == true`, so no zero roots or zero slot values get stored.

Coverage and Conclusions

1. When dealing with a branch node, the code reverts in case the end of the stack has been reached. This effectively forbids the last element of the stack to be a branch node and thus prevents a **truncated** inclusion proof from being validated and interpreted as a proof of exclusion.
2. We verified that `processStorageRoot` and `processStorageSlot` only store results when `exists == true`. Valid proofs save the correct values and emit events, while invalid or truncated proofs revert and leave the state unchanged, ensuring only legitimate data is persisted in the `DataWarehouse` storage.

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.