certora

Security Assessment

Compound Governor - Whitelist

October 2025

Prepared for Compound

certora

Table of content

certora

Project Summary

Project Scope

Project Name	Repository	Latest Commit Hash	Platform
Compound Governor Upgrade	PR 5	9c0b254	EVM

Project Overview

This document describes the manual code review of **Compound Governor Update PR**

The work was a 1 week effort undertaken from **14/10/2025** to **21/10/2025**

The following contract list is included in our scope:

- CompoundGovernor.sol
- GovernorBravoDelegate.sol

The team performed a manual audit of all the Solidity smart contracts. During the manual audit, the Certora team discovered no serious vulnerabilities that would put in jeopardy the protocol's users, funds or reputation. A couple informational issues and a list of remarks were added that reflect the researchers' thoughts on the scope and some of the design choices made.

Audit Goals

The audit scope concentrated on a logic upgrade and new features of governance-related contracts, `CompoundGovernor, GovernorBravoDelegate`. The goals of the upgrade were:

- **Confirm correct whitelist implementation**
  - The upgrade aims to introduce a proposer whitelist functionality responsible for granting and revoking proposer rights for a limited duration.
- **Handle proposal guardian expiration**
  - Once expired, only proposals for resubmitting a guardian can be made

Coverage

- **Proposer members can both grant and revoke whitelist rights**
  - We verified that proposal rights have a working feature to be revoked without having to wait till expiration. This is enforced through `setWhitelistAccountExpiration` by setting a timestamp in the past.
- **Minimum proposers is enforced at all times**
  - The upgrade introduces `MIN_PROPOSERS` variable which is properly enforced on each action that changes the proposer count. Some design inconsistencies were found, none of them security related.
- **Proposal guardian expiration handled correctly**
  - After guardian expiration, the only proposals that can be submitted are to set the new one, by ensuring the target and calldata can point only to the governor contract, calling the setProposalGuardian selector.

Remarks

The upgrade of the Compound Governor contracts follows best security practices. All new functions added are properly constrained to be used only by specific actors. Their in-code implementation matches the intended functionality. Some remarks:

- `MAX_TEMPORARY_PROPOSER_LIFETIME` is set as a constant of 365 days. This value is not modifiable and applies to all whitelisted proposers. A more foolproof approach would be to give a proposal allowance e.g 3 proposals in total, which goes down after each proposal.
- `SET_PROPOSAL_GUARDIAN_SELECTOR` = 0xb80d105a is correct and is the output of `bytes4(keccak256("setProposalGuardian((address,uint96))"))`. The dev comment states that it's setProposalGuardian(ProposalGuardian), presumably for simplicity. The double brackets around the inputs are required for correct output whenever the input is a struct.
- Proposal guardian won't be the first item in the enumerable set after its first expiration and if a new guardian is submitted.
- Setting whitelist expiration was previously an allowed action for the `_executor,` but in this upgrade it's not (unless executor is one of the initial allowed proposers).

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	3	3	1
Total	-	-	-

Severity Matrix

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

I-01: Guardian can add 1 proposer more than intended

Severity: Informational	Impact: Low	Likelihood: Low
Files: CompoundGovernor.sol		Status: Fixed

Description:
Whenever the proposal guardian is adding a proposer it is checked against the minimum to ensure it doesn't surpass it.

```
            if (allowedProposers.length() > MIN_PROPOSERS) {
                revert MinProposersReached();
            }
```

However, in the current state, it allows one extra due to using > instead of >=. On the 5th proposer added, the method will pass one more time, reaching 6 proposers although we can see from the Error message that this intends to limit it up to the minimum.

Recommendations:
Unless intentional, change the operator to >=

Customer response:
Fix confirmed [3a4ab447bc100ef435598337e07393b931a0e83a]

I-02: EIP712 non-compliancy

Severity: Informational	Impact: Low	Likelihood: Low
Files: GovernorBravoDelegate.sol		Status: Acknowledged

Description:
Some typehashes used in EIP712 signing seem to be misconfigured. This poses no security risk and is only a design inconsistency. `PROPOSAL_TYPEHASH` follows the Proposal struct found in CompoundGovernorTest rather than GovernorBravoInterfaces. It also includes 2 extra fields, not seen in the struct, `proposalId & signatures`

Recommendations:
If EIP712 compliance is required, this needs to be addressed. Otherwise, it poses no security risks since correct hashes can be produced and `proposeBySig` will work correctly either way

Customer response:
This issue is not part of our current update scope. The EIP-712 non-compliance existed in the original implementation and was not introduced by our changes. We have chosen to maintain the existing behavior to ensure backward compatibility and avoid breaking changes to the current governance system.

Acknowledged

I-03: The proposal guardian won't be the first in the enumerable set after it expires once

Severity: Informational	Impact: Low	Likelihood: Low
Files: CompoundGovernor.sol		Status: Acknowledged

Description:
During initialization in `batchWhitelist`, the proposal guardian is required to be the very first item in the set.

```
        if (_initProposers[0] != proposalGuardian.account) {
            revert FirstMustBeProposalGuardian(_initProposers[0]);
        }
```

However after it expires once and it gets replaced, the new proposal guardian will be last in the set

```
        if (currentProposalGuardian != newProposalGuardian) {
            allowedProposers.remove(currentProposalGuardian);
            allowedProposers.add(newProposalGuardian);
        }
```

During remove, the enum set uses swap and pop method to edit the item, while add pushes the new one. This goes against the logic in `batchWhitelist`, but poses no security risks.

Recommendations:

This needs to be addressed only if the proposal guardian is required to be the first item in the set at all times.

Customer response:
Acknowledged

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.