Compound Governor Bravo Audit

OPENZEPPELIN SECURITY | FEBRUARY 12, 2021

Security Audits

The Compound team engaged us to audit a new governance mechanism called Governor Bravo. The audited commit is `f86c247f6f81e14f8e0fd78402653a0b8371266a` in the proposed pull request, and the following files were included in scope:

- `GovernorBravoDelegate.sol`
- `GovernorBravoDelegator.sol`
- `GovernorBravoInterfaces.sol`

Any files that are not listed as “in scope” have not been audited in this engagement.

High-level overview of the system

`GovernorBravoDelegate` is the logic contract of the new governance. It inherits many of the mechanism from the previous `GovernorAlpha` contract with some differences:

- `proposalThresold`, `votingDelay` and `votingPeriod` are now non-fixed parameters that can be set.
- Voters can now vote to `abstain` and add a `reason` along with the vote.
- Governance contract is now upgradeable.

For a complete list of changes here there’s a summary.

address) and the implementation contract (to set governance parameters or to change `admin`).

Finally, the `GovernorBravoInterfaces.sol` file contains several auxiliary contracts to define storage layouts and events to be emitted.

To update the current governance system to `Governor Bravo` the following steps will be executed:

- The `GovernorBravoDelegate` contract is deployed, and its address is passed as a constructor parameter to deploy the `GovernorBravoDelegator` contract, which will set the `admin` to the `Timelock` contract and will `initialize` the implementation contract.
- A governance proposal is submitted to the current governance contract `GovernorAlpha` to change the `admin` of the `Timelock` to the `GovernorBravoDelegator` contract and to call the Governor Bravo `_initiate` function.

Overall we are happy with the changes introduced. The code is clear and easy to read, no major issues have been found and we are pleased that incremental changes are being made to the sensitive governance structure. These contracts were audited by two auditors during the course of 5 days. Here we present our findings, in order of importance.

Update: *The Compound team has reviewed the issues and published fixes for them. The fixes can be found as pull requests in the following, separate github repository: https://github.com/Arr00-Blurr/compound-protocol.*

Critical severity

None.

High severity

None.

Medium severity

parameters, allowing things like `admin` to be set to `address(0)`, the addresses of `comp` or `timelock` to be incorrectly set, or `proposalThreshold` to be set arbitrarily despite the `MIN_PROPOSAL_THRESHOLD`. Additionally, functions such as `_setImplementation`, `_setVotingDelay` and `_setVotingPeriod` contain no input checks and allow sensitive values to be set to any arbitrary value.

Consider bounding inputs to reasonable ranges and excluding certain values, such as `address(0)` or `uint256(0)` from being successfully passed in. This will reduce the surface for error when using these functions.

***Update**: Partially fixed in pull request #5. There are still no input checks on the `_setImplementation` function.*

[M02] Proposals cannot contain duplicate transactions

When queueing a proposal in the `Timelock` contract, a check is done for each proposed action which verifies that this action is not being done already in this proposal or with the same `eta`.

Although this design appears to be intentional, consider documenting this behavior explicitly. It must be made apparent to future development efforts that any functions which are intended to be called by governance can only be called once with the same parameters per proposal. Developers should understand to design functions such that multiple identical calls are unneeded.

Low severity

[L01] Unexecuted proposals from Governor Alpha will be frozen

When upgrading the governance system from Governor Alpha to Governor Bravo, any proposals that are currently unexecuted will not be executable. Whenever the Governor Bravo is transitioned to, the `proposalCount` variable is set to the `proposalCount` value of Governor Alpha at that moment. At this point, executing any proposal will require it to be in the `Queued` state. During the execution, when checking the state of the proposal, Governor Alpha proposal IDs will be all less than or equal to `initialProposalId`, and in those cases, the execution will fail.

consider implementing some method to execute previously queued proposals in the contract.

[L02] `initialize` can be called multiple times

The `initialize` function is intended to be delegate-called during the `GovernorBravoDelegator`'s construction. However, since it is public and contains no checks to stop future calls, it can be called again.

It is unclear whether this is intended or not. On the one hand, `initialize` is currently the only method available which can change the `comp` or `timelock` addresses. On the other hand, there exist functions for changing the other three values set in `initialize`, suggesting that it is not intended for `comp` or `timelock` to change after initialization.

Consider implementing some check so that `initialize` can be called only once, like the `initializer` modifier of the OpenZeppelin `Initializable` contract. Alternatively, consider documenting why `initialize` is left open for future calls. If it is desired to be able to change `comp` or `timelock` while the contract is being used, consider implementing separate functions for setting just those variables, in order to better separate changes to sensitive values within the Governor Bravo system.

***Update**: Fixed in pull request #4.*

[L03] Missing docstrings

Many important functions in the Governor Bravo code base lack documentation. This hinders reviewers' understanding of the code's intention, which is fundamental to correctly assess not only security, but also correctness. Additionally, docstrings improve readability and ease maintenance. They should explicitly explain the purpose or intention of the functions, the scenarios under which they can fail, the roles allowed to call them, the values returned and the events emitted. Functions such as `propose`, `queue`, and `execute` lack explanatory docstrings.

Consider thoroughly documenting all functions (and their parameters) that are part of the contracts' public API. Functions implementing sensitive functionality, even if not public, should be clearly

Update: *Fixed in pull request #10.*

[L04] Superfluous condition in `require` statement

Within the `_acceptAdmin` function of the `GovernorBravoDelegate` contract, there is a `require` statement which checks that `msg.sender != address(0)`. It is a common assumption, within Ethereum, that the private key for `address(0)` will never be known, and therefore it will be impossible for `msg.sender == address(0)`. Although it may seem that this conditional will prevent `pendingAdmin == address(0)`, the first condition in the require (`msg.sender == pendingAdmin`) also takes care of this if we assume that `msg.sender` will never be `address(0)`.

Consider removing the second condition from the mentioned `require`. This will make the code more easily understandable and reduce gas consumption.

[L05] Unused return value

The `delegateTo` function is `internal` and is only called once within the codebase. When the function is called, the return value is ignored.

To simplify the codebase, consider removing the returned value from `delegateTo`. Alternatively, consider utilizing the returned value when `delegateTo` is called.

Update: *Fixed in pull request #6.*

Notes & Additional Information

[N01] Inconsistent style

Within the Governor Bravo contracts, there are a few instances of inconsistency in coding style. We have identified the following:

- Some `external` or `public` functions begin with underscores, such as `_initiate`, while others do not, such as `castVote`. Additionally, some `internal` functions begin with underscores, such as `_queueOrRevert`, while some do not, such as `getChainId`. Often, leading underscores are used for `internal` functions only, to

- To improve readability, lines [illegible] of the [illegible] contract should be converted into the equivalent used in line 95 or 79.
- The constant `name` and the constant `MIN_PROPOSAL_THRESHOLD` are declared differently than the `pure` functions `quorumVotes` and `proposalMaxOperations`, but they all behave as constants within the code. Consider documenting the discrepancy in declaration style, or changing the declarations of all four to be consistent.

***Update**: Fixed in pull request #7.*

[N02] Misleading revert messages

The error message returned on line 90 of `GovernorBravoDelegate` implies that queueing has failed because there is already a proposal with the chosen `eta`. However, it is possible to queue multiple proposals with the same `eta`. This `require` will fail only when the exact same action has already been queued.

Additionally, the error message returned on line 288 of `GovernorBravoDelegate` implies only the `admin` is allowed to call the `acceptAdmin` function. In actuality, the `pendingAdmin` role is the only one allowed to call this function. Since `pendingAdmin` and `admin` are two distinct roles, this error message should be changed to match this distinction.

Error messages are intended to notify users about failing conditions, and should provide enough information so that the appropriate corrections needed to interact with the system can be applied. Uninformative error messages greatly damage the overall user experience, thus lowering the system's quality. Therefore, consider not only fixing the specific issues mentioned, but also reviewing the entire codebase to make sure every error message is informative and user-friendly enough. Furthermore, for consistency, consider reusing error messages when extremely similar conditions are checked.

***Update**: Partially fixed in pull request 8. The error message on line 90 of `GovernorBravoDelegate` has not been changed. Consider updating this message for greater clarity.*

[N03] Overflow protection unneeded

Consider upgrading to Solidity 0.8 and removing instances of `add256` and `sub256`. Utilizing built-in checks reduces code size and therefore the surface for error.

[N04] Declare `uint` as `uint256`

In the audited contracts, there is a general use of unsigned integer variables declared as `uint`.

To favor explicitness, consider replacing all instances of `uint` to `uint256`.

[N05] Upgradeable proxy design can be improved

The new governance implementation has been designed to have the `GovernorBravoDelegator` acting as a proxy for the `GovernorBravoDelegate` contract.

In this sense, the `GovernorBravoDelegate` contract is just the logic implementation layer, and the proxy is in charge of forwarding any call through a `delegatecall` instruction, and of mantaining the storage of the proxied contract.

Separating storage and logic into two separate contracts is a common pattern to follow when upgradeability of the logic layer is needed, but there are some downsides that must be addressed when implementing it.

Storage collision or function clashing are two of them.

In the audited contracts, we did not identify any possible storage collisions, but there's a lack of control against function clashing. Moreover, the `GovernorBravoDelegateStorageV1` in the `GovernorBravoInterfaces.sol` file defines variables of different types in mixed order and this is prone to errors when contract upgrades change the storage layout of the contracts.

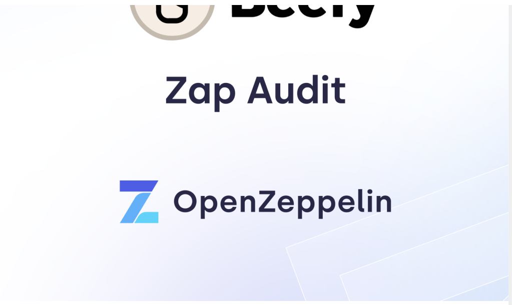
Consider using the OpenZeppelin *unstructured storage* proxy pattern to improve design robustness and have proper control over possible function clashes and storage collisions. Alternatively, consider documenting how storage collisions and function clashes will be handled in

Update: *Fixed in pull request #9. Storage variables within `GovernorBravoInterfaces` are now declared in a predictable order.*

Conclusion

No critical or high severity issues were found. Recommendations were made to improve code quality and usability, as well as improve the basic structure of the new governance system. We found that typical issues with proxy implementations, such as storage layout management issues, were not present here, although we still reccomend usage of OpenZeppelin's proxy implementations rather than "rolling your own". Overall, we found the codebase to be well-written and were happy with the fact that only incremental changes were made from Governor Alpha.

Related Posts



Beefy Zap Audit

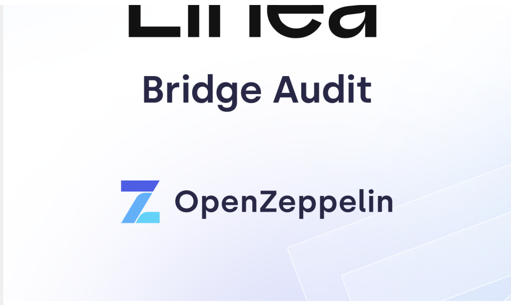
BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits



Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits