OpenZeppelin | security

Governor Bravo Signature Voting Audit

Compound

September 8, 2023

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-08-21 To 2023-08-29
Languages	Solidity

Total Issues	2 (1 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	1 (0 resolved)
Notes & Additional Information	1 (1 resolved)

Scope

We audited the new and changed functions added to `contracts/GovernorBravoDelegate.sol` in the compound-finance/compound-governance repository at the c965703 commit.

- File: `contracts/GovernorBravoDelegate.sol`
- Contract: `GovernorBravoDelegate`
- Methods:
  - `proposeInternal`
  - `propose`
  - `proposeBySig`
  - `castVoteWithReasonBySig`

System Overview

Governor Bravo is Compound's governance system, allowing users to create proposals and vote on them. The logic for how to tabulate votes is implemented in the `GovernorBravoDelegate` contract. COMP holders can currently call functions like `castVote`, `castVoteBySig`, or `castVoteWithReason` to cast votes on proposals. This audit reviewed a new function, `castVoteWithReasonBySig`, that allows users to sign votes off-chain like `castVoteBySig` but with a reason attached to the data, similarly to `castVoteWithReason`. The logic in the function is similar to the logic already implemented in `castVoteBySig` in that the function expects the caller to sign an EIP-712-compliant message which the contract will verify and then tabulate the message's vote. Alongside this function, `proposeBySig` was also added to allow users to create proposals using the same EIP-712 scheme. This new proposal function led to some refactoring with `propose` now utilizing a `proposeInternal` function. All three affected/new functions were in scope.

Low Severity

L-01 Proposal Creation Can Be Front-Run

Within the `proposeBySig` function, there is a requirement that the `proposalId` be the "next" proposal. This value is passed in, and is included in the signed message. Thus if this `proposalId` is consumed, the signature will have to be regenerated.

This allows an attacker to front-run `proposeBySig` calls by simply creating a new proposal, invalidating the signature. This attack will not prevent proposing forever but could be used to effectively stall the Compound protocol and significantly reduce the usefulness of `proposeBySig`. This attack also requires that the attacker is whitelisted or has at least a `proposalThreshold` amount of COMP, which is currently set to 25000 COMP.

Consider modifying `proposeBySig` not to require a specific `proposalId` to be passed in. Since this functionality prevents reusing signed messages, consider implementing some structure to flag signatures that have been used to prevent them from being used again. Alternatively, consider documenting this functionality to the community, specifically in relation to creating signatures for `proposeBySig` and changing the `proposalThreshold`.

Update: *Acknowledged, not resolved. The developer stated:*

> *I am reluctant to believe that the suggested front-running would ever be an issue given that the front-runner must have enough votes delegated to create a proposal as well as the drive to censor governance. Above all, this censorship is expensive while creating new signatures is free. Finally, the user proposing by sig could fallback to a normal proposal if the issue persists.*

Notes & Additional Information

N-01 Lack of Security Contact

Providing a specific security contact (such as an email or ENS) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice proves beneficial as it permits the code's owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. Additionally, if the contract incorporates third-party libraries and a bug surfaces in these, it becomes easier for the creators of those libraries to make contact, inform the code's owners about the problem, and provide mitigation instructions.

The `GovernorBravoDelegate` contract does not have a security contact.

Consider adding a NatSpec comment on top of the contract definition with a security contact. Using the `@custom:security-contact` convention is recommended as it has been adopted by the Openzeppelin Wizard and the ethereum-lists.

Update: *Resolved in commit bae4447*

Recommendations

Monitoring Recommendations

Technical

- Monitor for successful calls to `proposeBySig`, `castVoteBySig`, and `castVoteWithReasonBySig`. Then, validate the signature independently off-chain. This may help identify a problem with the signature verification within `GovernorBravoDelegate.sol`.
- Monitor for failed calls to `proposeBySig`, `castVoteBySig`, and `castVoteWithReasonBySig`. This may indicate a misuse of EIP-712 by the community, within `GovernorBravoDelegate.sol` or some error in wallet signing libraries.

Suspicious Activity

- Monitor for calls to `proposeBySig` which fail due to having an invalid `proposalId`. This may indicate front-running as mentioned above.

Conclusion

One low-severity issue and a few notes were reported to improve the overall quality of the codebase. The in-scope code changes were relatively simple and encapsulated, which makes assuring security easier. Additionally, the changes did not change the functionality of the code substantially, which assists in assessing the security of the changes. We were pleased to see many cosmetic changes that improved the quality of the code, such as adding line breaks in long lines and improving comments.