Compound Governance

Security Assessment

February 28, 2020

Prepared For:
Robert Leshner | *Compound Labs*
robert@compound.finance

Geoff Hayes | *Compound Labs*
geoff@compound.finance

Prepared By:
Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

Executive Summary

From February 13 through February 28, 2020, Compound Labs engaged with Trail of Bits to review the security of the Compound Governance Token (“COMP”) and GovernorAlpha contracts. Trail of Bits conducted this assessment over the course of 11 person-days with one engineer working from version `2.5-rc1` from the `compound-protocol-alpha` repository.

The first week of the assessment focused on gaining a high-level understanding of the codebase through the specification. We began our initial manual review of the codebase and performed static analysis using Slither. During the final week of the assessment, we concluded our manual review of the contracts with a focus on identifying any edge cases not covered by the specification, as well as the EIP712 implementations and the search algorithm used to find specific vote checkpoints. We also conducted fuzzing of the built-in safe arithmetic functions and ERC20 functionality of the `COMP` token with Echidna.

This review resulted in three informational-severity findings. The first finding, TOB-COMP-001: Ignored return values in GovernorAlpha functions, pertains to several instances of unchecked return values from external calls to the `Timelock` contract. In our second finding, TOB-COMP-002: Variability in time between blocks may drastically alter voting period length, variability in time between blocks may impact the outcome of proposals. The final finding, TOB-COMP-003: Unnecessary calls to Timelock when cancelling unqueued proposals, describes a situation in which gas may be wasted when cancelling operations.

In addition to the security findings, we discuss code quality issues not related to any particular vulnerability or discrepancy between the code and specification in Appendix B.

Project Dashboard

Application Summary

Name	Compound Governance System & Token
Version	`55729b31b85220ab42e26d7b6db0ff6c0eb0dd23`
Type	Solidity
Platforms	Ethereum

Engagement Summary

Dates	February 13 - February 28
Method	Whitebox
Consultants Engaged	1
Level of Effort	11 person-days

Vulnerability Summary

Total High-Severity Issues	0	
Total Medium-Severity Issues	0	
Total Low-Severity Issues	0	
Total Informational-Severity Issues	3	■ ■ ■
Total Undetermined-Severity Issues	0	
Total	3	

Category Breakdown

Data Validation	2	■ ■
Timing	1	■
Hardening	0	
Path Traversal	0	
Total	3	

Engagement Goals

The engagement was scoped to provide a security assessment of the `COMP` token as well as the GovernorAlpha contract.

Specifically, we sought to answer the following questions:

- Can balances be manipulated to get more voting rights than intended?
- Are delegations handled properly when tokens are transferred or delegates are changed?
- Can users delay, block, or unfairly influence proposals?
- Can proposals be transitioned to inconsistent states?
- Are there any inconsistencies between the Governance specification and the implementation?

Coverage

This review included the contracts comprising the Compound Governance system, specifically `Comp.sol` and `GovernorAlpha.sol`, as well as their interactions with the `Timelock.sol` contract. The specific version of the codebase used for the assessment came from commit hash `55729b31b85220ab42e26d7b6db0ff6c0eb0dd23` of `compound-protocol-alpha` on Github.

Trail of Bits reviewed the contracts for common design flaws, such as front-running, denial of service, race conditions, arithmetic issues, and protocol manipulations. Further, contracts were reviewed with special consideration for the proposal state machine in the GovernorAlpha contract and the complex voting mechanisms, which snapshots balances periodically and allows for voting rights to be delegated to other users.

Recommendations Summary

This section aggregates all the recommendations made during the engagement. Short-term recommendations address the immediate causes of issues. Long-term recommendations pertain to the development process and long-term design goals.

Short Term

❑ **Add return statements to `_queueOrRevert`, `execute`, `__queueSetTimelockPendingAdmin`, `__executeSetTimelockPendingAdmin`** so the return values from the Timelock contract can be exposed to users instead of being silently dropped. TOB-COMP-001

❑ **Document the variability in voting period length**. Consider the historical range of block times and whether it will allow for an acceptable voting period. TOB-COMP-002

❑ **Add a check to `GovernorAlpha.cancel` so that it only calls `Timelock.cancelTransaction` if the proposal is in the queued state**. This will prevent wasting gas if a proposal is not queued in the Timelock yet. TOB-COMP-003

Long Term

❑ **Use Slither or Crytic as part of your CI/CD pipeline** to detect issues such as ignored return values. TOB-COMP-001

❑ **Consider comparing block timestamps instead of block numbers** to create a fixed-length voting period. TOB-COMP-002

❑ **Ensure state-dependent functionality correctly verifies the object state** to prevent triggering undefined behavior in the protocol. TOB-COMP-003

Findings Summary

#	Title	Type	Severity
1	Ignored return values in GovernorAlpha functions	Data Validation	Informational
2	Variability in time between blocks may drastically alter voting period length	Timing	Informational
3	Unnecessary calls to Timelock when canceling unqueued proposals	Data Validation	Informational

1. Ignored return values in GovernorAlpha functions

Severity: Informational
Difficulty: Low
Type: Data Validation
Finding ID: TOB-COMP-001
Target: `contracts/Governance/GovernorAlpha.sol`

Description
The `GovernorAlpha` contract allows successful proposals to be queued and executed through the Timelock contract. The corresponding functions in `Timelock` return data that may be useful for users, but the data is ignored by the `GovernorAlpha` contract and is not returned to end users. This may make it more difficult for users to debug and understand whether calls were successful.

The relevant functions in the `GovernorAlpha` contract are:

- `_queueOrRevert`,
- `execute`,
- `__queueSetTimelockPendingAdmin`,
- `__executeSetTimelockPendingAdmin`.

Exploit Scenario
Alice submits a proposal to the `GovernorAlpha` contract. The proposal passes and she calls `queue` to get her proposal into the Timelock contract's queue. `queue` calls `_queueOrRevert` internally, which queues the proposal in the `Timelock` contract. `Timelock` returns a `txHash` but it is ignored by the `GovernorAlpha` and not returned to Alice. Though the transaction succeeds, there is no confirmation that the proposal was queued successfully.

Recommendation
Short term, Add return statements to `_queueOrRevert`, `execute`, `__queueSetTimelockPendingAdmin`, `__executeSetTimelockPendingAdmin` so that the return values from the Timelock contract can be exposed to users instead of being silently dropped.

Long term, use Slither or Crytic as part of your CI/CD pipeline to detect these types of issues.

2. Variability in time between blocks may drastically alter voting period length

Severity: Informational
Difficulty: Low
Type: Timing
Finding ID: TOB-COMP-002
Target: `contracts/Governance/GovernorAlpha.sol`

Description
The `GovernorAlpha` contract contains a function, `votingPeriod()`, that returns `17280`, which corresponds to three days in blocks (according to comments). This is true with an average block time of 15 seconds. However, since March 2019, outside of the difficulty bomb increases, the average block time has been closer to 13 seconds. A 13-second block time represents only 87% of the expected voting period, or approximately 2.6 days.

Historically, daily average block times have varied from 13 seconds to 30 seconds. Due to this wide variability, it may be advantageous to compare block timestamps to create a fixed-length voting period instead. Note, however, that as block timestamps are miner-controlled to some degree, this would introduce a very small amount of miner influence over the exact block where the voting period would end.

Exploit Scenario
Alice submits a proposal to the `GovernorAlpha` contract. Bob, believing that he has 72 hours to submit his vote, delays submitting his vote until what he believes are the final hours of the proposal voting period. Due to the shorter block time, his votes are not counted toward this proposal.

Recommendation
Short term, document this variability in voting period length. Consider the historical range of block times and whether this will allow for an acceptable voting period.

Long term, consider comparing block timestamps instead of block numbers to create a fixed-length voting period.

References
- Ethereum Average Block Time Chart

3. Unnecessary calls to Timelock when canceling unqueued proposals

Severity: Informational
Difficulty: Low
Type: Data Validation
Finding ID: TOB-COMP-003
Target: `contracts/Governance/GovernorAlpha.sol`

Description
At any time before a proposal is executed, it may be moved immediately to the canceled state by calling `cancel` in the `GovernorAlpha` contract. In the event that a proposal is already queued, then it must also be canceled in the `Timelock` contract. The `GovernorAlpha` appears to be missing a check that the proposal is in the queued state before calling out to the `Timelock` contract to cancel it there as well. For contracts in any state prior to queued, this results in wasted gas due to unnecessary calls to cancel non-existent actions in the `Timelock,` as well as the emission of superfluous cancellation events. Note that the specification correctly states that these calls to the `Timelock` are only required if a proposal has been queued.

Exploit Scenario
Alice creates a new proposal in the `GovernorAlpha` contract. She then sells enough of her `COMP` tokens to fall below the proposal threshold. Alice's proposal is still in the voting process and Bob attempts to cancel it. Bob successfully cancels the proposal but spends more gas than expected in the process.

Recommendation
Short term, add a check to `GovernorAlpha.cancel` so that it only calls `Timelock.cancelTransaction` if the proposal is in the queued state.

Long term, ensure state-dependent functionality correctly verifies the object state to prevent triggering undefined behavior in the protocol.

A. Vulnerability Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking, or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user’s information is at risk, exploitation would be bad for client’s reputation, moderate financial impact, possible legal

	implications for client
High	Large numbers of users, very bad for client’s reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details, or must discover other weaknesses in order to exploit this issue

B. Code Quality Recommendations

The following recommendations are not associated with specific vulnerabilities. However, they enhance code readability and may prevent the introduction of vulnerabilities in the future.

Throughout:

- The specification refers to addresses that have been delegated to as *delegates* (and in one case, *delegees*). The implementation instead calls them *delegatees*. All three are correct, but for the sake of clarity one should be chosen and used consistently throughout.

Comp:

- **Delegate.** The description of this function implies that it is valid to delegate votes to the zero address, but it neither describes the significance of this nor does it state that it is the default state for delegation. As a result, the user *must* delegate voting rights, even if they intend to vote on their own behalf. Additionally, the process of undelegating via delegating to the zero address is described in DelegateBySig, though it may be more appropriate to include it here.
- **TransferFrom**. In addition to the modifications from the EIP20 specification listed here, `COMP` also allows for infinite allowances to be set. This is documented in the code comments, but omitted from the specification.
- **VoteCheckpoint**. The description notes that this can be the automatic getter function generated for the corresponding state variable. However, the state variable is simply called `checkpoints` instead of `VoteCheckpoint`.

GovernorAlpha:

- **Cancel**. The intended purpose of the cancel function is not clearly defined. It is implied by the requirements that cancellation is intended for proposers who no longer meet the proposal threshold between the time of proposal and execution. This should be clarified in case users may expect that (for example) a proposer has the right to cancel their own proposal at any time (i.e., without having to transfer their `COMP` first).
- **ProposalApproved**. This function, marked internal, is not actually implemented anywhere in the GovernorAlpha contract. Instead, the logic appears to be baked into the `queue` function. Either the code or specification should be updated accordingly.
- **Queue, Execute, State**. Each of the Queue, Execute, and State function specifications refer to an explicit `queued` flag in a `Proposal` object. However, unlike the `executed` and `canceled` flags, the `Proposal` struct does not contain a boolean `queued` flag. Instead, the queued state is implied if no other state requirements are met when calling `state`. Either add a `queued` flag or update the specification to remove mentions of an explicit `queued` flag.