

Smart Contract Audit

Sovryn Governance

Franklin Richards

23rd December 2020

Index

1. [Introduction](#)
2. [High Threats](#)
3. [Medium Threats](#)
4. [Low Threats](#)
5. [Optimization & Readability](#)
6. [Typo's & Comments](#)
7. [Suggestions](#)

Note: Some threat levels or headings might be empty if there is no vulnerability/updates/suggestions found.



Introduction

The contract audited here is/are:

<https://github.com/DistributedCollective/Sovryn-smart-contracts/blob/0c01598/contracts/governance/GovernorAlpha.sol>

and was shared by Sovryn for auditing purposes while working with them.

Based on the audit, we were able to find 0 High threats, 1 Medium threat, and 2 Low threat with some other changes in the optimization, readability, typos, and comments section.



High Threats

1. None

Medium Threats

1. The cancel() can be misused by anyone based on the increase of tokens staked.

Assumptions:

Total Staked: 1000

i.e. ProposalThreshold: 10

Block: 200

Proposer Stake: 10

Step 1:

Proposer makes a proposal and the voting starts.

Step 2: This step can be happening at any time between the start of a proposal and before executing the proposal. This person does not have to be malicious.

Someone stakes 100 tokens. Increasing the ProposalThreshold to 11

Step 3:

Voting Finishes and the proposal is approved.

Step 4:

Proposal Queued

Step 5:

Block: 300

A malicious user can now call the cancel(), which **checks the prior votes of the proposer with the current total voting power in block 300.**

And as the prior vote of the proposer is less than the proposalThreshold, the proposal can be cancelled, even after getting the majority at the time of creating the proposal.

Low Threats

1. Only allowing a person who stakes 1% of the total staked can result in only whales being able to propose changes. And if the project becomes really distributed, it might even be difficult to make any changes. (Bitcoin has no wallet with more than 1% holding, Ethereum has less than 10 wallets, with 1% holding, etc.). And if it is not really distributed, a small fraction of investors can control the entire system for the majority of users.
2. Assumption: Sovryn Token is Governed by Governance, and minting is still allowed, or a part of Sovryn Token is Governed by Governance. And there is a way to short the Sovryn Token (optional).

Step 1: If the minPercentage and votes for a quorum are low, the whales who hold Sovryn Token can propose a proposal to mint new tokens or transfer the reserve tokens from governance to some wallet.

Step 2: After the proposal is put forward, every whale/member who is going with this plan, except the proposer can remove the stake, because **the votes are determined from the block and timestamp of the proposal time, and not the current one.**

Step 3: Those tokens can then be converted into ETH/BTC or any other pair, which can be used for shorting.

Step 4: Even though the proposal might not pass, or if due to the quorum and minPercentage is low, the proposal gets passed, the damage is already done in Step 2 of mass sale and FOMO.

Recommendation: Well thought and researched values for both quorum and minPercentage. Fair distribution of tokens using any mechanism thought fit.

This is a far fetched scenario, but a possible one.

Optimization & Readability


1. Indexing the id and the proposer is recommended for front-end fetching as well as easier readability in explorers in [Line 130](#). The same indexing can be used for other events as well, with necessary parameters.
2. Optimization based on saving the current state of the proposal in the struct vs the current method. Both have its advantages and disadvantages. But for the voters, which can be hundreds, we can reduce the storage reads to at least 100 times. (Saved state reading vs Finding state by reading multiple storage values).

Typo's & Comments

1. Many functions are without any comments. Please make sure every function follows the NatSpec Format.
2. Removing unnecessary and deprecated comments is recommended. Ex [Line 163](#)

Suggestions

1. Hard coding values which might improve in the future based on less gas cost is not really recommended in [line 13](#).
2. Similar to above, hard-coding values is not a good technique when the values are immutable for the rest of the contract lifetime. Instances: [Line 16](#) and [Line 19](#).
3. [Line 173](#) can be written after the next two require in Line 174 and 175 because the next two requires takes precedence when it comes to saving gas after a revert.
4. It is already written in [Line 249](#), I think we should not allow removing the proposalThreshold stake if he has proposed something. Though he could be allowed to remove the amount higher than proposalThreshold.
5. ``quorumPercentageVotes`` and ``minPercentageVotes`` are set in the constructor and can't be changed after that. It is recommended to make the governance flexible depending on the scenario. As a safe side can keep a minimum which can be set for both values, to make sure no one hijacks the contract. But it should be allowed to increase or decrease.

- 
6. ``quorumVotes()`` was never used. If not used in frontend as well, then this function can be removed.
 7. ``proposalCanceled`` event should contain the address of the person who canceled (guardian or any other user).

