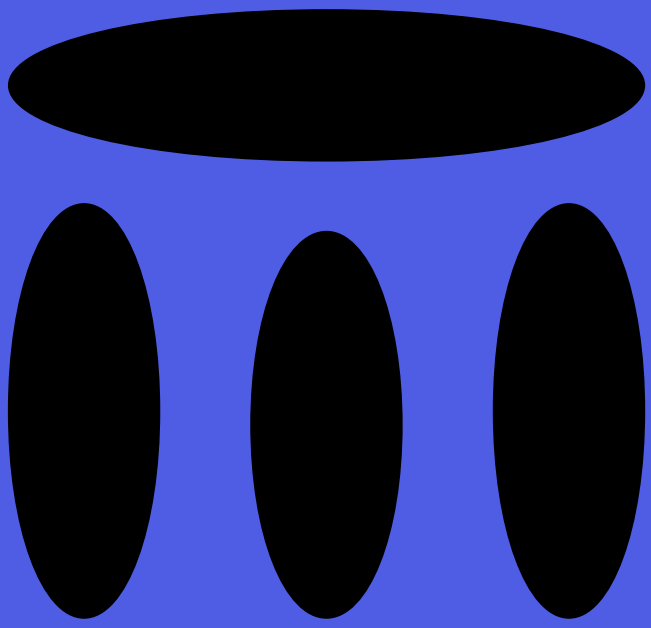
OpenZeppelin | security

Agora Optimism Governance Audit



November 22, 2023

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-10-16 To 2023-10-23
Languages	Solidity

Total Issues	27 (17 resolved, 1 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (1 resolved)
Low Severity Issues	5 (4 resolved)
Notes & Additional Information	21 (12 resolved, 1 partially resolved)

Scope

We audited the Agora Optimism Governance V6 repository at the a22052d commit.

In scope were the following contracts:

```
src
├── OptimismGovernorV6.sol
├── ProposalTypesConfigurator.sol
├── VotableSupplyOracle.sol
└── alligator
    └── AlligatorOP_V5.sol
```

System Overview

The Agora Optimism Governance V6 protocol has undergone a series of significant updates, broadening its capabilities and enhancing its utility. Among these developments, it has incorporated support for external voting modules and partial voting through Alligator. Additionally, a votable supply oracle has been added, offering valuable insights into supply dynamics for more informed governance decisions. The protocol now accommodates various proposal types, ensuring adaptability to a wide range of scenarios. The integration of the liquid delegation protocol, `Alligator V5`, enhances delegation capabilities. The `ProposalTypesConfigurator` empowers the `OptimismGovernorV6` manager to customize proposal types, and the `VotableSupplyOracle` allows the contract owner to efficiently manage the votable supply used by the `OptimismGovernorV6`.

Security Model & Trust Assumptions

Privileged Roles

The protocol implements multiple privileged roles. The following list has been limited only to the roles and actions that can be executed within the contracts in scope.

The `manager` role of `OptimismGovernorV6` can:

- Create proposals.
- Edit proposal type.
- Set proposal type via `ProposalTypesConfigurator` contract.

The `owner` of `Alligator_V5` contract can:

- Upgrade the implementation contract.
- Pause the contract.

The `owner` of `VotableSupplyOracle` contract can:

- Update votable supply at any block

Trust Assumptions

- The logic of `OptimismGovernorV6` relies on the out-of-scope `OptimismGovernorV5` contract and a set of modified OpenZeppelin contracts. Thus, it is assumed that the mentioned contracts are secure and behave according to their specifications.
- The holders of the privileged roles `manager` and `owner` are expected to be non-malicious, and act in the protocol's best interest.

Medium Severity

M-01 Incorrect Implementation of EIP-712

The `AlligatorOP_V5` contract incorrectly implements EIP-712. The `BALLOT_TYPEHASH` does not include parameters that are used in the hashing of multiple parameters in `castVoteBySig`, `castVoteWithReasonAndParamsBySig` and `limitedCastVoteWithReasonAndParamsBatchedBySig`.

This can lead to an exploitation path by reusing the same signature for `castVoteWithReasonAndParamsBySig` and passing parameters that concatenated together will result in the same bytes data that is expected to be hashed. The `MAX_VOTING_POWER` value of `0x5` will be interpreted in `castVoteWithReasonAndParamsBySig` as a length of the `authority` array.

Since there are three completely different methods to cast a vote there should be three different typehashes used, which include the relevant parameters that are being used for creating a hash.

Update: *Resolved in pull request #16 at commit 4229289.*

Low Severity

L-01 Floating Pragma

Pragma directives should be fixed to clearly identify the Solidity version with which the contracts will be compiled.

Throughout the codebase, there are multiple floating pragma directives. For instance:

- The `AlligatorOP_V5.sol` file has the `solidity ^0.8.19` floating pragma directive.
- The `OptimismGovernorV6.sol` file has the `solidity ^0.8.19` floating pragma directive.

- The `ProposalTypesConfigurator.sol` file has the `solidity ^0.8.19` floating pragma directive.
- The `VotableSupplyOracle.sol` file has the `solidity ^0.8.19` floating pragma directive.

Consider using a fixed pragma version throughout the codebase.

Update: *Resolved in pull request #16 at commit 0b40e39.*

L-02 Missing Docstrings

Throughout the codebase, there are several parts that do not have docstrings. For instance:

- Events, constant values and the `initialize` function in the `AlligatorOP_V5` contract
- Events, constant values and the `propose`, `proposeWithModule`, and `hasVoted`, `COUNTING_MODE` functions in the `OptimismGovernorV6` contract
- The `proposalTypes` function in the `ProposalTypesConfigurator` contract
- The `_updateVotableSupply` and `_updateVotableSupplyAt` functions in the `VotableSupplyOracle` contract

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Acknowledged, will resolve. The Agora team stated:*

> *Docstrings will be updated in the future before deployment.*

L-03 Missing Signature Validation Checks

The `AlligatorOP_V5` contract allows casting votes by using signatures. The signatures are validated using the `ecrecover` function but are missing essential security checks such as those against signature malleability.

Consider using OpenZeppelin's ECDSA library to validate signatures.

Update: *Resolved in pull request #16 at commit 383292a.*

L-04 Lack of Safety Check for `_voteSucceeded` Function

Within the `OptimismGovernorV6.sol` contract, the `_voteSucceeded()` function does not check `totalVotes != 0` before using it as a denominator. This edge case could happen if a proposal only has `abstainVotes`.

Consider adding this check to the `_voteSucceeded()` function.

Update: *Resolved in pull request #16 at commit acba358.*

L-05 Missing `quorum` Parameter Validation

The `ProposalTypesConfigurator` contract allows setting proposal types through the `setProposalType` function. The value of `quorum` is validated to not exceed `10000`, which is considered `100%`.

However, this process does not check for non-zero values. The `OptimismGovernorV6` contract relies on the fact that the correct proposal type has the value of `quorum` not equal to `0`. This is checked in the `propose`, `proposeWithModule` and `editProposalType` functions.

Consider adding a non-zero check to the value of `quorum`.

Update: *Resolved in pull request #16 at commits 1507cf3 and bd69f93.*

Notes & Additional Information

N-01 Missing Zero-Address Checks

Multiple contracts are missing zero-address checks for setting storage variables. Accidentally setting storage variable to address zero result in an incorrect configuration of the protocol.

- Missing check in the `constructor` of the `ProposalTypesConfigurator` contract for `governor.manager()` to ensure the `onlyManager` modifier works correctly.

- Missing check in `initialize` within the `AlligatorOPV5` contract for the `_initOwner` address parameter.
- Missing check in the `constructor` of the `VotableSupplyOracle` contract for the `_initOwner` address parameter.

Consider adding zero-address checks to the listed parameters.

Update: *Acknowledged, not resolved. The Agora team stated:*

> *We will not be adding zero-address checks to constructors and initializers.*

N-02 Inconsistent Use of Named Returns

To improve the readability of the contract, use the same return style in all of its functions. The `AlligatorOPV5` contract has an inconsistent usage of named returns within its functions.

Consider naming all return values of all functions.

Update: *Resolved in pull request #16 at commit b137b4d.*

N-03 Lack of Indexed Event Parameters

Throughout the codebase, several events do not have their parameters indexed. For instance:

- The `ProposalCreated` event of `OptimismGovernorV6.sol`
- The `ProposalCreated` event of `OptimismGovernorV6.sol`
- The `ProposalTypeUpdated` event of `OptimismGovernorV6.sol`

Consider indexing event parameters to improve the ability of off-chain services to search and filter for specific events.

Update: *Resolved in pull request #16 at commit bb82801.*

N-04 Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) 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 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 maintainers of those libraries to make contact with the appropriate person about the problem and provide mitigation instructions.

Throughout the codebase, there are several contracts that do not have a security contact. For instance:

- The `AlligatorOPV5` contract
- The `OptimismGovernorV6` contract
- The `ProposalTypesConfigurator` contract
- The `VotableSupplyOracle` contract

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

Update: *Acknowledged, not resolved. The Agora team stated:*

> *We may consider adding this in the future.*

N-05 Unused Error

In `AlligatorOP_V5.sol`, the `ProxyNotExistent` error is unused.

To improve the overall clarity, intentionality, and readability of the codebase, consider either using or removing any currently unused errors.

Update: *Resolved in pull request #16 at commit bc48b7e.*

N-06 Unused Event

In `AlligatorOP_V5.sol`, the `ProxyDeployed` event is unused.

To improve the overall clarity, intentionality, and readability of the codebase, consider emitting or removing any currently unused events.

Update: *Resolved in pull request #16 at commit bc48b7e.*

N-07 Unused Named Return Variables

Named return variables are a way to declare variables that are meant to be used within a function's body for the purpose of being returned as the function's output. They are an alternative to explicit in-line `return` statements.

Throughout the `AlligatorOP_V5` contract, there are multiple instances of unused named return variables. For instance:

- The `endBlock` return variable in the `_proposalEndBlock` function.
- The `snapshotBlock` return variable in the `_proposalSnapshot` function.
- The `weightCast` return variable in the `_weightCast` function.

Consider either using or removing any unused named return variables.

Update: *Resolved in pull request #16 at commit b137b4d.*

N-08 Use of Modified OpenZeppelin Contracts

The protocol currently utilizes a modified version of OpenZeppelin's governance contracts, which is considered an anti-pattern and not the intended way of using OpenZeppelin's contracts.

It is advisable to import the original OpenZeppelin contracts and override the necessary functions to align them with the protocol's specific requirements.

Update: *Acknowledged, not resolved. The Agora team stated:*

> *We had to use a modified version to edit the storage layout.*

N-09 Use Custom Errors

Since solidity version `0.8.4`, custom errors provide a cleaner and more cost-efficient way to explain to users why an operation failed.

There were instances of `require` and/or `revert` with error messages rather than custom errors in these files:

- `OptimismGovernorV6.sol`
- `ProposalTypesConfigurator.sol`

For conciseness and gas savings, consider replacing `require` and `revert` messages with custom errors.

Update: *Partially resolved. The Agora team stated:*

> *Left the `require` statements in `OptimismGovernorV6` for now to minimize changes from the inherited governor.*

N-10 Constants Not Using `UPPER_CASE` Format

Throughout the codebase, there are constants not using `UPPER_CASE` format. For instance:

- The `governor` constant declared on line 62 in `AlligatorOP_V5.sol`
- The `op` constant declared on line 64 in `AlligatorOP_V5.sol`
- The `alligator` constant declared on line 74 in `OptimismGovernorV6.sol`
- The `votableSupplyOracle` constant declared on line 77 in `OptimismGovernorV6.sol`
- The `proposalTypesConfigurator` constant declared on line 81 in `OptimismGovernorV6.sol`
- The `governor` constant declared on line 15 in `ProposalTypesConfigurator.sol`

According to the Solidity Style Guide, constants should be named with all capital letters with underscores separating words. For better readability, consider following this convention.

Update: *Resolved in pull request #16 at commit d3b7230.*

N-11 TODO Comments

During development, having well described TODO/Fixme comments will make the process of tracking and solving them easier. Without this information these comments might age and important information for the security of the system might be forgotten by the time it is released to production. These comments should be tracked in the project's issue backlog and resolved before the system's deployment.

Multiplies instances of TODO/Fixme comments were identified in the codebase. For instance:

- The `TODO` comment on line 73 in `OptimismGovernorV6.sol`.
- The `TODO` comment on line 76 in `OptimismGovernorV6.sol`.
- The `TODO` comment on line 80 in `OptimismGovernorV6.sol`.

Consider removing all instances of TODO/Fixme comments and tracking them in the issues backlog instead. Alternatively, consider linking each inline TODO/Fixme comment to the corresponding issues backlog entry.

Update: *Acknowledged, not resolved. The Agora team stated:*

> *We will not fix these for now.*

N-12 State Variable Visibility Not Explicitly Declared

Within `ProposalTypesConfigurator.sol`, the state variable `governor` lacks an explicitly declared visibility.

For clarity, consider always explicitly declaring the visibility of variables, even when the default visibility matches the intended visibility.

Update: *Resolved in pull request #16 at commit f07004c.*

N-13 Usage of Magic Numbers

Throughout the codebase, there are occurrences of literal values with unexplained meaning.

- Consider adding `PERCENT_DIVISOR = 10_000;` constant instead of using number `10000` in `ProposalTypesConfigurator` contract.
- Consider adding `PERCENT_DIVISOR = 10_000;` constant and use it in `quorum` and `_voteSucceeded` functions in `OptimismGovernorV6` contract.

To improve the code's readability and facilitate refactoring, consider defining a constant for every magic number, giving it a clear and self-explanatory name. For complex values, consider adding an inline comment explaining how they were calculated or why they were chosen.

Update: *Resolved in pull request #16 at commit 9b58c4c.*

N-14 Use of Deprecated OpenZeppelin's `Timers` Library

The `OptimismGovernorV6` contract uses OpenZeppelin's `Timers` library which has been deprecated as of OpenZeppelin version `4.9.0` and has been removed in `5.0.0`.

Consider switching to another solution.

Update: *Acknowledged, will resolve. The Agora team stated:*

> *We will change it once we upgrade the OpenZeppelin versions to v5, in the future.*

N-15 Missing Named Parameters in Mapping

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means mappings can take the form of `mapping(KeyType KeyName? => ValueType ValueName?)`. This updated syntax provides a more transparent representation of the mapping's purpose.

Following mappings have been identified that would benefit from named parameters:

- `_proposalTypes` in the `ProposalTypesConfigurator.sol` contract.
- `weightCast` in the `OptimismGovernorV6.sol` contract.
- Final value of `votesCast` in the `AlligatorOP_V5.sol` contract.

Consider adding named parameters to the listed mappings to improve readability and maintainability of the code.

Update: *Acknowledged, not resolved. The Agora team stated:*

> `weightCast` *already has all named parameters. The other two mentioned functions only have the last parameter missing, which is intended here.*

N-16 Code Is Not Consistent With Solidity Style Guide

There are several occurrences where the Solidity style guide is not followed, which makes the code more error-prone and difficult to read.

- Function `_updateVotableSupply` of `VotableSupplyOracle` contract is external and its name should not start with an underscore.
- Function `_updateVotableSupplyAt` of `VotableSupplyOracle` contract is external and its name should not start with an underscore.
- Function `_togglePause` of `AlligatorOP_V5` contract is external and its name should not start with an underscore.

To increase overall code readability, it is recommended to follow Solidity style guide across the entire codebase.

Update: *Acknowledged, not resolved. The Agora team stated:*

> *Will not fix, the underscore here is used to signal that only the manager can call the methods.*

N-17 Unused State Variable

Within the `OptimismGovernorV6` contract, the `MAX_VOTE_TYPE` state variable is unused.

To improve the overall clarity, intentionality, and readability of the codebase, consider removing any unused state variable.

Update: *Resolved in pull request #16 at commit 698f5f6.*

N-18 Lack of `gap` storage variables for upgradable contracts

The upgradeable `AlligatorOPV5` contract do not have a `__gap` variable. The `__gap` storage variable is used to allow for additional storage variables to be safely added when using inheritance, without it could cause future storage collision when new storage variables are introduced.

Similar issue is also identified in the `OptimismGovernorV5` contract and `OptimismGovernorV6` contract

Note current implementation will work fine as a versioned upgrade solution. If this is the intended behavior, consider documenting it in the codebase to avoid accidental storage collision in the future. Otherwise consider adding storage gaps to these contracts.

Update: *Acknowledged, not resolved. The Agora team stated:*

> *This is an intended behavior.*

N-19 Function could fail early

Function `propose` in contract `OptimismGovernorV6.sol` validates input parameters from line184 after hashing the `proposalId` in line 182.

In order to fail early and save gas, consider validating input parameters first.

Update: *Resolved in pull request #16 at commit 03ce194.*

N-20 Missing Initialization of Base Contracts

The AlligatorOP_V5 contract is an upgradeable contract that inherits OpenZeppelin's `UUPSUpgradeable`, `OwnableUpgradeable` and `PausableUpgradeable` contracts. The issue is that within the `initialize` only `PausableUpgradeable` contract is initialized while `UUPSUpgradeable` and `OwnableUpgradeable` are not.

Consider initializing all the inherited upgradeable contracts `UPPSUpgradeable` and `OwnableUpgradeable` to enhance readability and clarity of the code.

Update: *Resolved in pull request #16 at commit 4ea0462.*

N-21 Confusing Use of `VotableSupplyUpdated` Event

The `VotableSupplyOracle` contract emits `VotableSupplyUpdated` in case the votable supply has changed. The parameters of `VotableSupplyUpdated` are `blockNumber`, `oldVotableSupply`, `newVotableSupply`.

The issue is that in case a new checkpoint is being added either in `constructor` or via `_updateVotableSupply` the `oldVotableSupply` represents the `votableSupply` of the last checkpoint but for `_updateVotableSupplyAt` it represents the supply of the updated checkpoint. This might lead to confusion for off-chain application since its difficult to distinguish what `oldVotableSupply` parameter represents in given time.

Consider adding new event and emit it in `_updateVotableSupplyAt` function.

Update: *Resolved in pull request #16 at commit 7bd01b6.*

Conclusion

The codebase is quite complex due to multiple inheritances and upgrades of the protocol. It will benefit from incorporating our informational notes and lower-severity recommendations in order to improve its readability and robustness. It is recommended to generate more technical documentation about the Governance V6 and Alligator V5 contracts' integration.