OpenZeppelin | security

Lido Dual Governance Update Audit

LIDO

February 18, 2025

Table of Contents

Summary

Type	Governance
Timeline	From 2025-01-06 To 2025-01-15
Languages	Solidity

Total Issues	9 (4 resolved, 2 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	3 (1 resolved)
Notes & Additional Information	6 (3 resolved, 2 partially resolved)

Scope

We diff-audited the lidofinance/dual-governance repository at the HEAD commit 46d667e against the BASE commit 08e622a.

The following files were in scope:

```
contracts
├── DualGovernance.sol
├── EmergencyProtectedTimelock.sol
├── Escrow.sol
├── Executor.sol
├── ImmutableDualGovernanceConfigProvider.sol
├── ResealManager.sol
├── TimelockedGovernance.sol
├── committees
│   ├── HashConsensus.sol
│   ├── ProposalsList.sol
│   ├── TiebreakerCoreCommittee.sol
│   └── TiebreakerSubCommittee.sol
├── interfaces
│   ├── IDualGovernance.sol
│   ├── IDualGovernanceConfigProvider.sol
│   ├── IEmergencyProtectedTimelock.sol
│   ├── IEscrowBase.sol
│   ├── IExternalExecutor.sol
│   ├── IGovernance.sol
│   ├── IOwnable.sol
│   ├── IRageQuitEscrow.sol
│   ├── IResealManager.sol
│   ├── ISealable.sol
│   ├── ISignallingEscrow.sol
│   ├── IStETH.sol
│   ├── ITiebreaker.sol
│   ├── ITiebreakerCoreCommittee.sol
│   ├── ITimelock.sol
│   ├── IWithdrawalQueue.sol
│   └── IWstETH.sol
├── libraries
│   ├── AssetsAccounting.sol
│   ├── DualGovernanceConfig.sol
│   ├── DualGovernanceStateMachine.sol
│   ├── DualGovernanceStateTransitions.sol
│   ├── EmergencyProtection.sol
│   ├── EnumerableProposals.sol
│   ├── EscrowState.sol
│   ├── ExecutableProposals.sol
│   ├── ExternalCalls.sol
│   ├── Proposers.sol
│   ├── Resealer.sol
│   ├── SealableCalls.sol
│   ├── Tiebreaker.sol
│   ├── TimelockState.sol
│   └── WithdrawalsBatchesQueue.sol
└── types
    ├── Duration.sol
    ├── ETHValue.sol
    ├── IndexOneBased.sol
    ├── PercentD16.sol
    ├── SharesValue.sol
    └── Timestamp.sol
```

Update: *The fix review for this audit has been completed, and all pull requests addressing the reported issues have been merged at [commit 3e0f1ae](#).*

System Overview

The Dual Governance system enables Lido stakers to veto the governance proposals passed by LDO holders. By depositing stETH, wstETH, or Lido Withdrawal NFTs into an Escrow, stakers can delay proposals while retaining the option to exit the protocol. The system transitions between Normal, Veto Signalling, Veto Cooldown, and Rage Quit states which, in turn, determines whether proposal submission and execution are allowed. Tiebreaker logic handles emergency pauses, allowing a special committee to unpause protocol contracts and finalize pending proposals if a deadlock arises.

The Dual Governance contracts were previously audited by OpenZeppelin. This audit focused on changes implemented since the initial review, including:

- Additional checks and validation of input parameters.
- Updated variable names and inline comments to better reflect the code's intention.
- Refactoring of logic and improvements to code quality.
- Removal of the `EmergencyExecutionCommittee` and `EmergencyActivationCommittee` contracts.

Security Model and Privileged Roles

The admin executor is an instance of the `Executor` contract, which is set up with special privileges. Previously, this contract had the following capabilities:

- Configure Dual Governance parameters
- Register or remove proposers
- Set a proposal executor
- Add or remove tiebreaker withdrawal blockers
- Set and revoke committee rights
- Set the `ResealManager` address
- Set the proposal canceller role address
- Control the `EmergencyProtectedTimelock` contract

With this update, the following changes to privileged roles were made:

- A proposal canceller role was added. This role is in charge of calling the `cancelAllPendingProposals` function. Previously, this function was only callable by a proposer that was associated with the admin executor.
- The `ResealManager` address was previously immutable in the `DualGovernance` contract. Now, this address can be set by the admin executor.
- The emergency activation, emergency execution, and reseal committees are no longer part of the Dual Governance system.

It is assumed that all parties who operate and manage the roles of the system act in the best interest of the protocol, follow best practices to protect their access, and are trained to follow security procedures in the event of an emergency.

Details of the roles involved can be found in the system specification.

Low Severity

L-01 Incomplete Minimum Asset Lock Duration Validation

The `setMinAssetsLockDuration` function sets the minimum lock duration of the `Escrow` contract. This function performs a check to ensure that the new lock duration does not exceed the maximum lock duration, which is passed to the function as the `MAX_MIN_ASSETS_LOCK_DURATION` constant.

However, this maximum duration check is not performed during the initialization of the `Escrow` contract. Thus, the lock duration only depends on the `DualGovernance` configuration, which the `Escrow` contract is set up with as a minimum lock duration.

Consider passing down the maximum lock duration value to the `initialize` function and calling the `internal` `setMinAssetsLockDuration` function.

Update: *Resolved in pull request #256 at commit 30015d3.*

L-02 Inconsistent Use of Zero-Address Checks

In `Resealer.sol`, the `setResealManager` function prevents the new manager from being set to the zero address, whereas the `setResealCommittee` function does not have this check.

Consider adding a check to the `setResealCommittee` function to ensure that the committee is not set to the zero address.

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

> *Acknowledged. This is the expected behavior.*
>
> *In the future, there is a chance that the use of `ISealable` contracts will be abandoned, so it is necessary to have the ability to disable the reseal functionality. `ResealManager` is an essential part of `DualGovernance`, so if it is set to the zero address, it may cause reverts in the system's functioning. Therefore, it was decided that the reseal functionality can be disabled by setting the committee to the zero address.*

L-03 `ResealManager` Could Make Use of `SealableCalls` Library

The `Tiebreaker` contract uses the `SealableCalls` library function `callGetResumeSinceTimestamp` to catch potential reverts and return a boolean indicating whether the call succeeded. In contrast, the `ResealManager` contract calls the `ISealable` interface directly, which could cause a generic revert.

Consider using the `SealableCalls` library in `ResealManager` to revert more specifically.

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

> *Acknowledged. This is the expected behavior.*
>
> *For the Tiebreaker functionality, it is critically important to handle reverts of `ISealable`, for example, in cases where the contract implementation has been updated and no longer conforms to the `ISealable` interface. For `ResealManager`, it is necessary to explicitly determine whether a transaction has reverted or not.*

Notes & Additional Information

N-01 Code Quality Improvements

Throughout the codebase, multiple opportunities for code quality improvement were identified:

- The calldata for contract calls is encoded using `abi.encodeWithSelector` which is not type-safe. Instead, consider using `abi.encodeCall`.
- In `Escrow.sol`, the `lockUnstETH`, `unlockUnstETH`, and `markUnstETHFinalized` functions perform actions on the `unstETHIds` input array. However, `markUnstETHFinalized` does not revert given an empty input array, unlike the other functions. For consistency, consider implementing the same check in `markUnstETHFinalized` as well.
- The `setProposerExecutor` and `unregisterProposer` functions of `DualGovernance.sol` ensure that they can only be called by the admin executor and that the admin executor remains with a valid proposer afterwards. This is done by querying the admin executor address once from the timelock and then once by using

`msg.sender`. Consider using `msg.sender` instead of querying the timelock in `setProposerExecutor` to save some gas.

Consider applying the above suggestions to improve the quality of the codebase.

Update: *Partially resolved in pull request #258 at commit f721d2f. The Lido team stated:*

> *We decided to not change everything related to `abi.encodeWithSelector`.*

N-02 Misleading Documentation

In the `DualGovernanceConfig` library, the `firstSealRageQuitSupport` field of the `Context` struct is described as the following:

> The percentage of the total stETH supply that must be reached in the Signalling Escrow to transition Dual Governance from the Normal state to the VetoSignalling state

However, the VetoSignalling state can also be accessed from the Veto Cooldown and Rage Quit states.

Consider updating the documentation to reflect the specification.

Update: *Resolved in pull request #260 at commit ca3ea75.*

N-03 Incomplete Docstrings

The `cancelAllPendingProposals` function of the `TimelockedGovernance` contract does not describe the purpose of its boolean return value. This value indicates the success of the function.

For improved code clarity, consider documenting the return value of `cancelAllPendingProposals`.

Update: *Resolved in pull request #261 at commit d03d9d1.*

N-04 Incorrect Interfaces

Some of the publicly declared constants and function signatures in the provided interfaces do not align with their respective implementation contracts.

For the `IDualGovernance` interface:

- In the `registerProposer`, `unregisterProposer`, `isProposer`, and `getProposer` functions, the parameter `account` should be `proposerAccount`.
- In the `isExecutor` function, the `address` parameter should be `executor`.
- The `setProposalsCanceller` and `getProposalsCanceller` functions are not defined.

Additionally:

- The `DUAL_GOVERNANCE` `public` constant is not defined in the `IEscrowBase` interface.
- The `MIN_EXECUTION_DELAY` `public` constant is not defined in the `IEmergencyProtectedTimelock` interface.

To avoid any unexpected behavior, ensure that all contracts correctly implement their public interfaces.

Update: *Partially resolved in pull request #262 at commit d4fde93. The Lido team stated:*

> *We decided to not add constants to interfaces.*

N-05 Use `calldata` Instead of `memory`

When working with `external` function parameters, reading arguments directly from `calldata` is more gas-efficient than storing them in `memory`. As a read-only region containing the arguments of incoming `external` calls, `calldata` is cheaper and more gas-efficient for these parameters.

Throughout the codebase, multiple instances where function parameters could use `calldata` instead of `memory` were identified:

- In `Escrow.sol`, the `unstETHIds` parameters [1, 2, 3].
- In `HashConsensus.sol`, the `membersToRemove` parameter.

Consider using `calldata` as the data location for the parameters of `external` functions to optimize gas usage.

Update: *Acknowledged, not resolved.*

N-06 Typographical Errors

Throughout the codebase, multiple instances of typographical errors were identified:

- The codebase mostly refers to proposals as "cancelled", but sometimes writes "canceled". Consider sticking to one.
- In line 8 of `TimelockState.sol`, "configuration related the state" should be "configuration related to the state".
- In line 166 of `Proposers.sol`, "Checks if an `proposerAccount`" should be "Checks if a `proposerAccount`".

Consider correcting the above-listed typographical errors to improve the clarity and readability of the codebase.

Update: *Resolved in pull request #265 at commit c19a82b.*

Conclusion

After conducting a thorough review of the modifications made to the Dual Governance contracts, we identified several opportunities for minor enhancements to the codebase. However, due to the complexity of the system we recommend applying further testing methods like Formal Verification and Fuzzing. Throughout the audit, the Lido team was highly responsive, promptly addressing our inquiries and providing us with comprehensive documentation and detailed explanations about the system's design and functionality.