Tally Auto Delegate Review

January 4, 2025

Prepared for DappHero Corp

Conducted by:

Kurt Willis (phaze)

Richie Humphrey (devtooligan)

About the Tally Staking Review

Tally is a blockchain-based governance platform that enables organizations to manage decision-making through transparent, decentralized processes. Using the OpenZeppelin Governor framework, it provides infrastructure for onchain voting and proposal systems while ensuring community participation and preventing centralized control.

The Tally Staking protocol allows UNI token holders to stake their tokens and receive stUNI in return, with automated reward claiming and governance power delegation. The auto-delegation functionality enables automated governance participation through configurable rules and thresholds, helping to streamline the governance process while maintaining security and decentralization.

About Offbeat Security

Offbeat Security is a boutique security company providing unique security solutions for complex and novel crypto projects. Our mission is to elevate the blockchain security landscape through invention and collaboration.

Summary & Scope

The src/auto-delegates folder of the `stUNI` repo was reviewed at commit 7aa0b92.

The following **1 contract** was in scope:

- src/auto-delegates/OverwhelmingSupportAutoDelegate.sol

Summary of Findings

The audit focused specifically on the OverwhelmingSupportAutoDelegate contract, which implements configurable rules for automated governance participation. Four low-severity issues, one informational issue, and one gas optimization were identified in this component. While none of these findings present critical risks to the protocol's core operations, implementing the recommended improvements would further strengthen the security and reliability of the contract.

Identifier	Title	Severity	Fixed
L-01	Consider implementing governor contract whitelist	Low	**Response from project team**: "We decided to not implement the fix as we couldn't think of a malicious vote scenario and it adds friction to voting on multiple governors as a new governor needs to be added to the whitelist"
L-02	Missing bounds validation for voting window and sub-quorum parameters	Low	Fixed in PR192
L-03	Precision loss in support threshold calculation	Low	Fixed in PR174
I-01	Missing public view function for vote eligibility checks	Info	Fixed in PR188

G-01	Reduce external calls to governor contract in vote requirement checks	Gas	Fixed in PR189

Additional Recommendations

- Consider using OpenZeppelin's `Ownable2Step` instead of `Ownable` for protection against ownership transfers to mistaken addresses.

Detailed Findings

Low Findings

[L-01] Consider implementing governor contract whitelist

Description

The `castVote()` function currently accepts any address as a governor contract parameter. This could potentially allow votes to be cast on malicious or unintended governor contracts. Adding a whitelist would ensure votes can only be cast on approved governance contracts.

Recommendation

Consider implementing a whitelist mechanism for governor contracts:

```
+ error OverwhelmingSupportAutoDelegate__UnauthorizedGovernor();

+ mapping(IGovernorBravoDelegate => bool) public authorizedGovernors;

+ event GovernorAuthorized(IGovernorBravoDelegate governor);
+ event GovernorUnauthorized(IGovernorBravoDelegate governor);

+ function setGovernorAuthorization(IGovernorBravoDelegate _governor, bool _autho
+   _checkOwner();
+   authorizedGovernors[_governor] = _authorized;
+   if (_authorized) {
+     emit GovernorAuthorized(_governor);
+   } else {
+     emit GovernorUnauthorized(_governor);
+   }
+ }

  function castVote(IGovernorBravoDelegate _governor, uint256 _proposalId) publi
+   if (!authorizedGovernors[_governor]) {
+     revert OverwhelmingSupportAutoDelegate__UnauthorizedGovernor();
+   }
```

```
    if (!_isWithinVotingWindow(_governor, _proposalId)) {
      revert OverwhelmingSupportAutoDelegate__OutsideVotingWindow();
    }
    // ... rest of the function
  }
```

This ensures that votes can only be cast on explicitly authorized governor contracts.

[L-02] Missing bounds validation for voting window and sub-quorum parameters

Description

The `_setVotingWindow()` and `_setSubQuorumBips()` functions lack input validation to ensure the parameters remain within reasonable bounds. Without these checks, it's possible to set these values to extreme or nonsensical values that could disrupt the contract's intended functionality.

Recommendation

Consider adding reasonable bounds checks to both setter functions:

```
+ error OverwhelmingSupportAutoDelegate__InvalidVotingWindow();
+ error OverwhelmingSupportAutoDelegate__InvalidSubQuorumBips();

+ uint256 private constant MIN_VOTING_WINDOW = 1;            // At least 1 block
+ uint256 private constant MAX_VOTING_WINDOW = 50_400;       // About 1 week at 1
+ uint256 private constant MAX_SUB_QUORUM_BIPS = 10_000;     // Cannot exceed 100

  function _setVotingWindow(uint256 _votingWindow) internal {
+   if (_votingWindow < MIN_VOTING_WINDOW || _votingWindow > MAX_VOTING_WINDOW)
+     revert OverwhelmingSupportAutoDelegate__InvalidVotingWindow();
+   }
    emit VotingWindowSet(votingWindow, _votingWindow);
    votingWindow = _votingWindow;
  }

  function _setSubQuorumBips(uint256 _subQuorumBips) internal {
+   if (_subQuorumBips == 0 || _subQuorumBips > MAX_SUB_QUORUM_BIPS) {
+     revert OverwhelmingSupportAutoDelegate__InvalidSubQuorumBips();
+   }
    emit SubQuorumBipsSet(subQuorumBips, _subQuorumBips);
    subQuorumBips = _subQuorumBips;
  }
```

These changes prevent setting values outside of reasonable ranges, similar to how the contract already validates the support threshold.

[L-03] Precision loss in support threshold calculation

Description

In the `_isAboveSupportThreshold()` function, the support percentage calculation divides the left side of the inequality, which can lead to precision loss. For better precision, the multiplication should be performed on the right side of the inequality.

Recommendation

Consider rewriting the calculation to avoid precision loss:

```
  function _isAboveSupportThreshold(IGovernorBravoDelegate _governor, uint256 _p
    (,,,,, uint256 _forVotes, uint256 _againstVotes,,,) = _governor.proposals(_p
-   return ((_forVotes * BIP) / (_forVotes + _againstVotes)) >= supportThreshold
+   return _forVotes * BIP >= supportThreshold * (_forVotes + _againstVotes);
  }
```

This change maintains better precision by avoiding division operations in the comparison.

Informational Findings

[I-01] Missing public view function for vote eligibility checks

Description

The `OverwhelmingSupportAutoDelegate` contract performs several checks before allowing a vote to be cast in `castVote()`, but these checks are not exposed through a public view function. This makes it harder for integrators to determine if a vote can be cast before attempting the transaction.

The current checks are implemented as internal view functions:

- `_isWithinVotingWindow()`
- `_hasReachedSubQuorum()`
- `_isAboveSupportThreshold()`

Recommendation

Consider adding a public view function that performs all voting eligibility checks:

```
/// @notice Checks if all requirements are met for casting a vote on a proposal
/// @param _governor The Governor contract containing the proposal
/// @param _proposalId The ID of the proposal to check
/// @dev This function reverts if any voting requirement is not met
function checkVoteRequirements(IGovernorBravoDelegate _governor, uint256 _proposa
    if (!authorizedGovernors[_governor]) {
      revert OverwhelmingSupportAutoDelegate__UnauthorizedGovernor();
    }
    if (!_isWithinVotingWindow(_governor, _proposalId)) {
        revert OverwhelmingSupportAutoDelegate__OutsideVotingWindow();
    }
```

```
    if (!_hasReachedSubQuorum(_governor, _proposalId)) {
        revert OverwhelmingSupportAutoDelegate__InsufficientForVotes();
    }
    if (!_isAboveSupportThreshold(_governor, _proposalId)) {
        revert OverwhelmingSupportAutoDelegate__BelowSupportThreshold();
    }
}
```

This function would allow integrators to check vote eligibility before submitting transactions. The `castVote()` function could then be updated to use this validation:

```
function castVote(IGovernorBravoDelegate _governor, uint256 _proposalId) public
+   checkVoteRequirements(_governor, _proposalId);
-   if (!_isWithinVotingWindow(_governor, _proposalId)) {
-       revert OverwhelmingSupportAutoDelegate__OutsideVotingWindow();
-   }
-   if (!_hasReachedSubQuorum(_governor, _proposalId)) {
-       revert OverwhelmingSupportAutoDelegate__InsufficientForVotes();
-   }
-   if (!_isAboveSupportThreshold(_governor, _proposalId)) {
-       revert OverwhelmingSupportAutoDelegate__BelowSupportThreshold();
-   }
    _governor.castVote(_proposalId, FOR);
}
```

Gas Optimizations

[G-01] Reduce external calls to governor contract in vote requirement checks

Description and Recommendations

The internal check functions in `OverwhelmingSupportAutoDelegate` make separate external calls to fetch the same proposal data from the governor contract. This can be optimized by fetching the data once and passing it to the internal functions.

Note: The following example is based on the recommendations provided for L-01 and I-01.

```
 function checkVoteRequirements(IGovernorBravoDelegate _governor, uint256 _propos
+    // Fetch proposal data once
+    (,,,, uint256 endBlock, uint256 forVotes, uint256 againstVotes,,,) = _governo
+
-    if (!_isWithinVotingWindow(_governor, _proposalId)) {
+    if (!_isWithinVotingWindow(endBlock)) {
         revert OverwhelmingSupportAutoDelegate__OutsideVotingWindow();
     }
-    if (!_hasReachedSubQuorum(_governor, _proposalId)) {
+    uint256 quorumVotes = _governor.quorumVotes();
+    if (!_hasReachedSubQuorum(forVotes, quorumVotes)) {
         revert OverwhelmingSupportAutoDelegate__InsufficientForVotes();
```

```
      }
-     if (!_isAboveSupportThreshold(_governor, _proposalId)) {
+     if (!_isAboveSupportThreshold(forVotes, againstVotes)) {
          revert OverwhelmingSupportAutoDelegate__BelowSupportThreshold();
      }
  }

-function _isWithinVotingWindow(IGovernorBravoDelegate _governor, uint256 _propos
-     (,,,, uint256 endBlock,,,,,) = _governor.proposals(_proposalId);
+function _isWithinVotingWindow(uint256 endBlock) internal view returns (bool) {
      return block.number >= (endBlock - votingWindow);
  }

-function _hasReachedSubQuorum(IGovernorBravoDelegate _governor, uint256 _proposa
-     (,,,,, uint256 _forVotes,,,,) = _governor.proposals(_proposalId);
+function _hasReachedSubQuorum(uint256 _forVotes, uint256 quorumVotes) internal
-     return _forVotes >= ((_governor.quorumVotes() * subQuorumBips) / BIP);
+     return _forVotes >= ((quorumVotes * subQuorumBips) / BIP);
  }

-function _isAboveSupportThreshold(IGovernorBravoDelegate _governor, uint256 _pr
-     (,,,,, uint256 _forVotes, uint256 _againstVotes,,,) = _governor.proposals(_
+function _isAboveSupportThreshold(uint256 _forVotes, uint256 _againstVotes) inte
```

This optimization reduces the number of external calls from 4 to 2 (one for proposal data and one for quorum votes). The internal functions are also made more focused by only accepting the specific data they need for their calculations.