// Smart Contract Security Assessment

12.02.2024 - 01.03.2025

Aragon OSx 1.4

Aragon

HALBORN

Aragon OSx 1.4 - Aragon

Prepared by: **HALBORN**

Last Updated 02/14/2025

Date of Engagement by: December 2nd, 2024 - January 3rd, 2025

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
6	0	0	1	0	5

TABLE OF CONTENTS

1. Introduction

Aragon engaged Halborn to conduct a security assessment on their smart contracts beginning on December 2nd, 2024 and ending on January 3rd, 2025. The security assessment was scoped to the smart contracts provided to the Halborn team.

Commit hashes and further details can be found in the Scope section of this report.

The Aragon DAO is a set of programmable decentralised autonomous organisation contracts and plugins enabling different types of consensus needed to pass proposals (token voting, multisig...). The current assessment covers a big part of the scope that was already reviewed previously, but also focuses on minor contract upgrades and the addition of a new plugin, the staged proposal plugin.

2. Assessment Summary

The team at Halborn assigned a full-time security engineer to assess the security of the smart contracts. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.
- Ensure upgrades are not breaking the contracts.
- Provide recommendations on how to handle loss of availability in sub bodies of the staged proposal processor.

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were partially addressed by the `Aragon team`. The main ones were the following:

- `Make staged proposal plugin robust against sub bodies denial of availability.`
- `Introduce a buffer between end of voting and limit time for the proposal execution.`
- `Introduce incentives for token voters to hold the tokens after a vote.`
- `Fix unreachable conditions.`

3. Test Approach And Methodology

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the assessment:

- Research into the architecture, purpose, and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could lead to arithmetic related vulnerabilities.
- Manual testing by custom scripts.
- Graphing out functionality and contract logic/connectivity/functions (`solgraph`).
- Static Analysis of security for scoped contract, and imported functions. (`Slither`).
- Local or public testnet deployment (`Foundry`, `Remix IDE`)

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker’s control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: osx

(b) Assessed Commit ID: e0ba7b6

(c) Items in scope:

- osx/packages/contracts/src/core/dao/DAO.sol
- osx/packages/contracts/src/core/dao/IEIP4824.sol
- osx/packages/contracts/src/core/permission/PermissionManager.sol
- osx/packages/contracts/src/core/utils/CallbackHandler.sol
- osx/packages/contracts/src/framework/dao/DAOFactory.sol
- osx/packages/contracts/src/framework/dao/DAORegistry.sol
- osx/packages/contracts/src/framework/plugin/repo/IPluginRepo.sol
- osx/packages/contracts/src/framework/plugin/repo/PluginRepo.sol
- osx/packages/contracts/src/framework/plugin/repo/PluginRepoFactory.sol
- osx/packages/contracts/src/framework/plugin/repo/PluginRepoRegistry.sol
- osx/packages/contracts/src/framework/plugin/repo/placeholder/PlaceholderSetup.sol
- osx/packages/contracts/src/framework/plugin/setup/PluginSetupProcessor.sol
- osx/packages/contracts/src/framework/plugin/setup/PluginSetupProcessorHelpers.sol
- osx/packages/contracts/src/framework/utils/InterfaceBasedRegistry.sol
- osx/packages/contracts/src/framework/utils/RegistryUtils.sol
- osx/packages/contracts/src/framework/utils/ens/ENSMigration.sol
- osx/packages/contracts/src/framework/utils/ens/ENSSubdomainRegistrar.sol
- aragon/osx/commit/fc7403fcef660aa952c7e635e95257ee7f466efb

Out-of-Scope: Third party dependencies., Economic attacks.

FILES AND REPOSITORY

(a) Repository: osx-commons

(b) Assessed Commit ID: 4eca1de

(c) Items in scope:

- osx-commons/contracts/src/dao/IDAO.sol
- osx-commons/contracts/src/executors/Executor.sol
- osx-commons/contracts/src/executors/IExecutor.sol
- osx-commons/contracts/src/permission/PermissionLib.sol
- osx-commons/contracts/src/permission/auth/DaoAuthorizable.sol
- osx-commons/contracts/src/permission/auth/DaoAuthorizableUpgradeable.sol
- osx-commons/contracts/src/permission/auth/auth.sol
- osx-commons/contracts/src/permission/condition/IPermissionCondition.sol
- osx-commons/contracts/src/permission/condition/PermissionCondition.sol
- osx-commons/contracts/src/permission/condition/PermissionConditionUpgradeable.sol
- osx-commons/contracts/src/permission/condition/extensions/RuledCondition.sol
- osx-commons/contracts/src/plugin/IPlugin.sol
- osx-commons/contracts/src/plugin/Plugin.sol
- osx-commons/contracts/src/plugin/PluginCloneable.sol
- osx-commons/contracts/src/plugin/PluginUUPSUpgradeable.sol
- osx-commons/contracts/src/plugin/extensions/governance/Addresslist.sol
- osx-commons/contracts/src/plugin/extensions/membership/IMembership.sol
- osx-commons/contracts/src/plugin/extensions/proposal/IProposal.sol

- osx-commons/contracts/src/plugin/extensions/proposal/Proposal.sol
- osx-commons/contracts/src/plugin/extensions/proposal/ProposalUpgradeable.sol
- osx-commons/contracts/src/plugin/setup/IPluginSetup.sol
- osx-commons/contracts/src/plugin/setup/PluginSetup.sol
- osx-commons/contracts/src/plugin/setup/PluginUpgradeableSetup.sol
- osx-commons/contracts/src/utils/deployment/ProxyFactory.sol
- osx-commons/contracts/src/utils/deployment/ProxyLib.sol
- osx-commons/contracts/src/utils/math/BitMap.sol
- osx-commons/contracts/src/utils/math/Ratio.sol
- osx-commons/contracts/src/utils/math/UncheckedMath.sol
- osx-commons/contracts/src/utils/metadata/MetadataExtension.sol
- osx-commons/contracts/src/utils/metadata/MetadataExtensionUpgradeable.sol
- osx-commons/contracts/src/utils/versioning/IProtocolVersion.sol
- osx-commons/contracts/src/utils/versioning/ProtocolVersion.sol
- osx-commons/contracts/src/utils/versioning/VersionComparisonLib.sol
- aragon/osx-commons/commit/869045b069679fad94d640a8a3e1a1b5538cf0f0

Out-of-Scope:

FILES AND REPOSITORY

(a) Repository: token-voting-plugin

(b) Assessed Commit ID: 02a7dbb

(c) Items in scope:

- token-voting-plugin/packages/contracts/src/ERC20/IERC20MintableUpgradeable.sol
- token-voting-plugin/packages/contracts/src/ERC20/governance/GovernanceERC20.sol
- token-voting-plugin/packages/contracts/src/ERC20/governance/GovernanceWrappedERC20.sol
- token-voting-plugin/packages/contracts/src/ERC20/governance/IGovernanceWrappedERC20.sol
- token-voting-plugin/packages/contracts/src/IMajorityVoting.sol
- token-voting-plugin/packages/contracts/src/MajorityVotingBase.sol
- token-voting-plugin/packages/contracts/src/TokenVoting.sol
- token-voting-plugin/packages/contracts/src/TokenVotingSetup.sol
- token-voting-plugin/packages/contracts/src/VotingPowerCondition.sol
- aragon/token-voting-plugin/commit/9eb9b15986db2fcb9a02c89e3aa99fe8c0292a1c

Out-of-Scope: Third party dependencies and economic attacks.

FILES AND REPOSITORY

(a) Repository: multisig-plugin

(b) Assessed Commit ID: fffc680

(c) Items in scope:

- multisig-plugin/packages/contracts/src/IMultisig.sol
- multisig-plugin/packages/contracts/src/ListedCheckCondition.sol
- multisig-plugin/packages/contracts/src/Multisig.sol
- multisig-plugin/packages/contracts/src/MultisigSetup.sol
- aragon/multisig-plugin/commit/69874ebe11efd0e9687ba24a7e04ba7fac08012d

Out-of-Scope: Third party dependencies and economic attacks.

FILES AND REPOSITORY

(a) Repository: admin-plugin

(b) Assessed Commit ID: 546cfa2

(c) Items in scope:

- admin-plugin/packages/contracts/src/Admin.sol
- admin-plugin/packages/contracts/src/AdminSetup.sol
- aragon/admin-plugin/commit/81db7a26668551a91077fcc23efc10dd937fcfe7

Out-of-Scope: Third party dependencies and economic attacks.

FILES AND REPOSITORY

(a) Repository: staged-proposal-processor-plugin

(b) Assessed Commit ID: dca24e2

(c) Items in scope:

- staged-proposal-processor-plugin/src/StagedProposalProcessor.sol
- staged-proposal-processor-plugin/src/StagedProposalProcessorSetup.sol
- staged-proposal-processor-plugin/src/libraries/Errors.sol
- staged-proposal-processor-plugin/src/utils/PluginSettings.sol
- staged-proposal-processor-plugin/src/utils/SPPRuleCondition.sol
- staged-proposal-processor-plugin/src/utils/TrustedForwarder.sol
- aragon/staged-proposal-processor-plugin/commit/ad73f165e5d7204a37c404d2bd270e401a127dc7

Out-of-Scope: Third party dependencies and economic attacks.

REMEDIATION COMMIT ID:

- b4c1d2b
- b480f34

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	0	5

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
STAGEDPROPOSALPROCESSOR PLUGIN CAN LOSE ABILITY TO ADVANCE	MEDIUM	RISK ACCEPTED - 01/15/2025
SUCEEDED PROPOSALS MAY EXPIRE BEFORE EXECUTION	INFORMATIONAL	ACKNOWLEDGED - 01/21/2025
NO INCENTIVE TO HOLD VOTING TOKENS AFTER A VOTE	INFORMATIONAL	ACKNOWLEDGED - 01/15/2025
UNREACHABLE CONDITION IN STAGEDPROPOSALPROCESSOR	INFORMATIONAL	SOLVED - 11/23/2024
ANYONE CAN REPORT A PROPOSAL RESULT	INFORMATIONAL	ACKNOWLEDGED - 01/15/2025
PERMISSION INCONSISTENCY BETWEEN STAGEDPROPOSALPROCESSORSETUPZKSYNC AND STAGEDPROPOSALPROCESSORSETUP	INFORMATIONAL	SOLVED - 01/31/2025

7. FINDINGS & TECH DETAILS

7.1 STAGEDPROPOSALPROCESSOR PLUGIN CAN LOSE ABILITY TO ADVANCE

// MEDIUM

Description

The **StagedProposalProcessor plugin (SPP)** organizes a proposal lifecycle around different stages, of multiple bodies. For example, stage 1 can be **TokenVotingPlugin** + **MultisigPlugin**, and stage 2 can be **AdminPlugin**. Each stage has a fixed length and waits for the result of the majority of the bodies to advance to the next stage.

Naturally, when installing this plugin, the other installed plugins must be revoked of their rights, since it does not make sense to have a short path (like a Multisig) and a long path (SPP composed of multisig, token voting...) for the same power.

Identified Issues

A few issues can arise from the SPP configuration, as caught by the Aragon team prior to the assessment:

1. **Permanent Unresponsive Body**
    - A body can be permanently unresponding: for example, a multisig where users lost access to their keys, a huge token holder could lose access to their keys, or simply not participate in governance anymore.
2. **Configuration Conflicts**
    - A body configuration or update can conflict with the SPP: for example, the fixed length of the stages in the SPP can differ from the minimum or maximum length of the sub-bodies.
3. **Proposal Creation Failures**
    - If a sub-body's proposal creation is extremely expensive in gas (or has an unexpected revert path), the SPP proposal creation function might fail due to out-of-gas or reverts.

Proposed Solutions

The proposed solution from the Aragon team is to introduce a **recover functionality** allowing the removal of a bricked body and adjust thresholds (either the number of bodies needed for approval or the number of vetoes allowed). This solves the possible loss of availability at the cost of a possible takeover from a plugin with enough rights. For instance, a malicious majority could remove a vetoing sub-body from the next stage just because they "don't like" the potential veto. Similarly, they could remove sub-bodies that are meant to be necessary checks/balances.

Recover Function Design

Having a recover function only callable by an administrator or a multisig, after a certain condition (enough time since the last succeeded proposal, number of failed proposals...), seems to be a simple and robust way of handling the recovery, although still carrying centralization risks (e.g., malicious administrators or loss of keys).

Configuration Validation

One might still argue that the best approach is simply to keep things simple. If the operator or DAO incorrectly sets the stage vote duration to 20 minutes, while the sub-body has a minimum duration of 1 hour, that is a config mistake which should be caught in an update stage validation step before the plugin is used in production.

Preventing Harmful Upgrades

It might be interesting to prevent harmful upgrades in sub-bodies. For example, any upgrade on the vote duration that would not match the SPP config should be refused. This could be implemented with a callback from the sub-body to an `upgradeValidator` address (being the SPP contract), checking the validity of the update parameters.

Handling Proposal Creation Failures

In case of failure to create a proposal in a sub-body, a partial success loop could be used, where the SPP tries to create proposals for each body in a loop, keeping track of the successful indices and skipping the failures. The same reasoning could be employed where the plugins expose an `isBricked` method that serves the SPP to exclude the plugin from the approvers or creating proposals on it. Note that this kind of failure should be detected priorly during tests so it should be rare in production.

Safe Partial Creation and Non-Upgradeable Sub-Bodies

Another "middle ground" idea is to add "safe partial creation" or to require that sub-bodies are themselves non-upgradeable once they have been integrated into SPP. That way, parameters can't get out-of-sync mid-flight.

Improved Recover Function Design

The current recover function proposed in the recover-proposal tag does only prevent issues caused by the voting duration, and it might be a better design to prevent durations mismatches with additional verifications during setups or upgrades, rather than a post-issue fix.

Conclusion

Addressing the potential issues with the StagedProposalProcessor involves a combination of introducing robust recovery mechanisms, validating configurations upfront, preventing harmful upgrades, and ensuring proposal creation reliability. While these solutions enhance the resilience and integrity of the governance process, they also introduce certain centralization risks that must be carefully managed.

Code Location

The following snippets show an example where the `TokenVotingPlugin` could refuse the proposal creations.

Enforcing Minimum/Maximum Durations

This snippet shows that durations can be updated by a plugin, independently from the SPP durations.

- token-voting-plugin/packages/contracts/src/MajorityVotingBase.sol:

```
function _updateVotingSettings(VotingSettings calldata _votingSettings) internal virtu
    // ...

    if (_votingSettings.minDuration < 60 minutes) {
        revert MinDurationOutOfBounds({limit: 60 minutes, actual: _votingSettings.minD
    }

    if (_votingSettings.minDuration > 365 days) {
        revert MinDurationOutOfBounds({limit: 365 days, actual: _votingSettings.minDur
    }

    votingSettings = _votingSettings;

    emit VotingSettingsUpdated({
        votingMode: _votingSettings.votingMode,
        supportThreshold: _votingSettings.supportThreshold,
        minParticipation: _votingSettings.minParticipation,
        minDuration: _votingSettings.minDuration,
        minProposerVotingPower: _votingSettings.minProposerVotingPower
    });
}
```

Validating Proposal Start And End Dates

These lines show that dates proposed by the SPP could be considered out of bounds if the durations mismatch.

- token-voting-plugin/packages/contracts/src/MajorityVotingBase.sol:

```
function _validateProposalDates(
    uint64 _start,
    uint64 _end
) internal view virtual returns (uint64 startDate, uint64 endDate) {
    uint64 currentTimestamp = block.timestamp.toUint64();

    if (_start == 0) {
        startDate = currentTimestamp;
    } else {
        startDate = _start;

        if (startDate < currentTimestamp) {
            revert DateOutOfBounds({limit: currentTimestamp, actual: startDate});
        }
    }

    uint64 earliestEndDate = startDate + votingSettings.minDuration;

    if (_end == 0) {
        endDate = earliestEndDate;
    } else {
        endDate = _end;

        if (endDate < earliestEndDate) {
            revert DateOutOfBounds({limit: earliestEndDate, actual: endDate});
        }
    }
}
```

BVSS

AO:A/AC:L/AX:M/R:N/S:C/C:N/A:H/I:N/D:N/Y:N (6.3)

Recommendation

It is recommended to:

- Verify beforehand the compatibility of the voting timelines between the SPP and the sub bodies.
- Give the possibility to bodies to signal their state of availability, so the SPP can ignore them when in a long enough non-available state.

Remediation

RISK ACCEPTED: The **Aragon team** accepted the risk behind this finding. The remediation would introduce too much complexity and potential bugs, while the denial of service can be avoided by a thorough verification of the plugin settings before installing the **StagedProposalProcessor**.

7.2 SUCEEDED PROPOSALS MAY EXPIRE BEFORE EXECUTION

// INFORMATIONAL

Description

In the token majority voting plugin, a scenario could exist where the majority is reached at the last block before the end of the proposal (`proposal_.parameters.endDate`) and a manual execution of the proposal (not using `_tryEarlyExecution`) would cause the proposal to succeed but switch in a closed state.

Code Location

The following snippets show that the `vote` function checks the `_canVote` function, that checks the `_isProposalOpen` function, to highlight that the end of the vote matches the end limit for the execution.

- token-voting-plugin/packages/contracts/src/MajorityVotingBase.sol:

```
function vote(
    uint256 _proposalId,
    VoteOption _voteOption,
    bool _tryEarlyExecution
) public virtual {
    address account = _msgSender();

    if (!_canVote(_proposalId, account, _voteOption)) {
        revert VoteCastForbidden({
            proposalId: _proposalId,
            account: account,
            voteOption: _voteOption
        });
    }
    _vote(_proposalId, _voteOption, account, _tryEarlyExecution);
}
```

- token-voting-plugin/packages/contracts/src/TokenVoting.sol:

```
function _canVote(
    uint256 _proposalId,
    address _account,
    VoteOption _voteOption
) internal view override returns (bool) {
    Proposal storage proposal_ = proposals[_proposalId];

    // The proposal vote hasn't started or has already ended.
    if (!_isProposalOpen(proposal_)) {
        return false;
    }

    // The voter votes `None` which is not allowed.
    if (_voteOption == VoteOption.None) {
        return false;
    }

    // The voter has no voting power.
    if (votingToken.getPastVotes(_account, proposal_.parameters.snapshotBlock) == 0) {
```

```
            return false;
        }

        // The voter has already voted but vote replacment is not allowed.
        if (
            proposal_.voters[_account] != VoteOption.None &&
            proposal_.parameters.votingMode != VotingMode.VoteReplacement
        ) {
            return false;
        }

        return true;
    }
```

- token-voting-plugin/packages/contracts/src/MajorityVotingBase.sol:

```
function _isProposalOpen(Proposal storage proposal_) internal view virtual returns (bo
    uint64 currentTime = block.timestamp.toUint64();

    return
        proposal_.parameters.startDate <= currentTime &&
        currentTime < proposal_.parameters.endDate &&
        !proposal_.executed;
}
```

Same for the Multisig plugin:

- multisig-plugin/packages/contracts/src/Multisig.sol:

```
function _isProposalOpen(Proposal storage proposal_) internal view returns (bool) {
    uint64 currentTimestamp64 = block.timestamp.toUint64();
    return
        !proposal_.executed &&
        proposal_.parameters.startDate <= currentTimestamp64 &&
        proposal_.parameters.endDate >= currentTimestamp64;
}
```

BVSS

AO:A/AC:L/AX:M/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (1.7)

Recommendation

It is recommended to add a buffer period between the end of votes and the execution limit.

Remediation

ACKNOWLEDGED: The **Aragon team** acknowledged this finding, also mentioning that:

> Introducing a buffer period could result in a scenario where voting is restricted during the buffer, even though the proposal has not yet been executed. Additionally, implementing such a buffer would introduce an unnecessary delay in the governance process, which we aim to avoid.
> Our current design choice ensures no hard delays while achieving the same overall outcome.

```
    if (
        _canExecute(_proposalId) &&
        dao().hasPermission(address(this), _voter, EXECUTE_PROPOSAL_PERMISSION_ID, _ms
    ) {
        _execute(_proposalId);
    }
}
```

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

It is recommended to use a locking mechanism to ensure that the tokens remain frozen until the proposal passes or fails.

Remediation

ACKNOWLEDGED: The **Aragon team** acknowledged this finding, also mentioning that:

> This plugin is designed to be a pure token voting mechanism, focused solely on governance and not on creating incentives to hold the tokens for financial purposes. It’s intentionally meant to separate governance from tokenomics. Additionally, we don’t have any plans in our roadmap to introduce such features in this plugin.

```
        // current tx and will not be needed later on for advancing.
        for (uint256 i = 1; i < _proposalParams.length; ++i) {
            for (uint256 j = 0; j < _proposalParams[i].length; ++j)
                createProposalParams[proposalId][uint16(i)][j] = _proposalParams[i][j];
        }

        _createBodyProposals(
            proposalId,
            0,
            proposal.lastStageTransition,
            _proposalParams.length > 0 ? _proposalParams[0] : new bytes[](0)
        );

        emit ProposalCreated({
            proposalId: proposalId,
            creator: _msgSender(),
            startDate: proposal.lastStageTransition,
            endDate: 0,
            metadata: _metadata,
            actions: _actions,
            allowFailureMap: _allowFailureMap
        });
    }
```

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

It is recommended to review if the unreachable condition is wanted, and in that case, fix the logic to make it accessible.

Remediation

SOLVED: The issue was fixed in the specified commit.

Remediation Hash

https://github.com/aragon/staged-proposal-processor-plugin/commit/b4c1d2b37d3efb065297b696830b36c04085df5d

7.5 ANYONE CAN REPORT A PROPOSAL RESULT

// INFORMATIONAL

Description

In the StagedProposalProcessor plugin, each plugin (token voting, multisig, admin...) participating in the main proposal stage creates independent proposals that are programmed to call the StagedProposalProcessor's reportProposalResult function as a callback one they are passed and executed, so that the latter can record the status of each sub plugin result.

It was found that this function is not restricted by access control, unlike most of the plugins that priorly distribute rights for each actors. The current logic does not allow any user to modify status of real proposals but could be causing confusion in external systems working with the events.

Code Location

- staged-proposal-processor-plugin/src/StagedProposalProcessor.sol:

```
function reportProposalResult(
    uint256 _proposalId,
    uint16 _stageId,
    ResultType _resultType,
    bool _tryAdvance
) external virtual {
    Proposal storage proposal = proposals[_proposalId];

    if (!_proposalExists(proposal)) {
        revert Errors.NonexistentProposal(_proposalId);
    }

    uint16 currentStage = proposal.currentStage;

    // Ensure that result can not be submitted
    // for the stage that has not yet become active.
    if (_stageId > currentStage) {
        revert Errors.StageIdInvalid(currentStage, _stageId);
    }

    _processProposalResult(_proposalId, _stageId, _resultType);

    if (_tryAdvance && _canProposalAdvance(_proposalId)) {
        // If it's the last stage, only advance(i.e execute) if
        // caller has permission. Note that we don't revert in
        // this case to still allow the records being reported.
        if (
            proposal.currentStage != stages[proposal.stageConfigIndex].length - 1 ||
            hasExecutePermission()
        ) {
            _advanceProposal(_proposalId);
        }
    }
}
```

- staged-proposal-processor-plugin/src/StagedProposalProcessor.sol:

```
function _processProposalResult(
    uint256 _proposalId,
    uint16 _stageId,
    ResultType _resultType
) internal virtual {
    address sender = _msgSender();

    bodyResults[_proposalId][_stageId][sender] = _resultType;
    emit ProposalResultReported(_proposalId, _stageId, sender);
}
```

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

It is recommended to add a role to restrict proposal result reporting to only the subbodies.

Remediation

ACKNOWLEDGED: The **Aragon team** acknowledged this finding.

7.6 PERMISSION INCONSISTENCY BETWEEN STAGEDPROPOSALPROCESSORSETUPZKSYNC AND STAGEDPROPOSALPROCESSORSETUP

// INFORMATIONAL

Description

The **StagedProposalProcessorSetupZkSync.sol** contract is copied from **StagedProposalProcessorSetup.sol**, aimed for zkSync deployment. It was found that the `CANCEL_PERMISSION_ID` permission was granted by the zkSync setup contract, but not the original.

This creates inconsistencies between the StagedProposalProcessors deployed on both environments.

Code Location

- StagedProposalProcessorSetupZkSync.sol:

```
permissions[6] = PermissionLib.MultiTargetPermission({
    operation: _op,
    where: _spp,
    who: ANY_ADDR,
    condition: PermissionLib.NO_CONDITION,
    permissionId: Permissions.CANCEL_PERMISSION_ID
});
```

BVSS

AO:A/AC:L/AX:M/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (1.7)

Recommendation

It is recommended to review both files and keep the wanted version of the permissions.

Remediation

SOLVED: The issue was fixed in the specified commit.

Remediation Hash

https://github.com/aragon/staged-proposal-processor-plugin/commit/b480f34c40390a929a2c9259d161fc36c88ac419

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project’s integrity and addressing potential vulnerabilities introduced by code modifications.