// Security Assessment 07.11.2025 - 07.14.2025

Tokenvoting Plugin Upgrade

Aragon

HALBORN

Tokenvoting Plugin Upgrade - Aragon

Prepared by: HALBORN

Last Updated 09/29/2025

Date of Engagement: July 11th, 2025 - July 14th, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	0	1	0	0	1

TABLE OF CONTENTS

1. Introduction

`Aragon` engaged `Halborn` to perform a security assessment of their smart contracts from July 11th, 2025, to July 14th, 2025. The assessment scope was limited to the smart contracts provided to Halborn. Commit hashes and additional details are available in the Scope section of this report.

The `Aragon` codebase in scope consists of feature updates made to the TokenVoting plugin including timestamp-based voting, mint freezing, and token supply exclusion.

2. Assessment Summary

`Halborn` was allocated 2 days for this engagement and assigned 1 full-time security engineer to conduct a comprehensive review of the smart contracts within scope. The engineer is an expert in blockchain and smart contract security, with advanced skills in penetration testing and smart contract exploitation, as well as extensive knowledge of multiple blockchain protocols.

The objectives of this assessment were to:

- Identify potential security vulnerabilities within the smart contracts.
- Verify that the smart contract functionality operates as intended.

In summary, `Halborn` identified several areas for improvement to reduce the likelihood and impact of potential risks, which were partially addressed by the `Aragon team`. The recommendations were as follows:

- `Reduce the storage gap array size by two instead of one.`
- `Explicitly check for both "mode=timestamp" and "mode=blocknumber" in CLOCK_MODE(). If neither is detected, revert with an error to prevent unsupported or ambiguous clock modes from being used.`

3. Caveats

The security assessment was limited to analyzing changes made to files between two commits at https://github.com/aragon/TokenVotingPlugin/compare/30b1218...a703baf.

While these limitations exist, the assessment was conducted to provide a comprehensive evaluation of the protocol's security posture. Nonetheless, these constraints should be considered when reviewing the findings and recommendations.

3.1 Post-Assessment Disclosure

After the initial assessment, `Aragon` reported a vulnerability in the `MinVotingPowerCondition` contract, which integrated with the code in scope, and indirectly affected the code in scope and allowed bypassing proposal creation checks using flash loans.

Halborn performed a post-assessment review of PR14 (commit `590ebd8`), and confirms the remediation is present in that commit.

4. Test Approach And Methodology

`Halborn` conducted a combination of manual code review and automated security testing to balance efficiency, timeliness, practicality, and accuracy within the scope of this assessment. While manual testing is crucial for identifying flaws in logic, processes, and implementation, automated testing enhances coverage of smart contracts and quickly detects deviations from established security best practices.

The following phases and associated tools were employed throughout the term of the assessment:

- Research into the platform's architecture, purpose and use.
- Manual code review and walkthrough of smart contracts to identify any logical issues.
- Comprehensive assessment of the safety and usage of critical Solidity variables and functions within scope that could lead to arithmetic-related vulnerabilities.
- Local testing using custom scripts (`Foundry`).
- Fork testing against main networks (`Foundry`).
- Static security analysis of scoped contracts, and imported functions (`Slither`).

5. 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.

5.1 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

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
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:

$$E = \prod m_e$$

5.2 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

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
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:

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

5.3 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

6. SCOPE

REPOSITORY

(a) Repository: TokenVotingPlugin

(b) Assessed Commit ID: a703baf

(c) Items in scope:

- src/TokenVoting.sol
- src/TokenVotingSetup.sol
- src/TokenVotingSetupZkSync.sol
- src/base/MajorityVotingBase.sol
- src/erc20/GovernanceERC20.sol
- src/erc20/GovernanceWrappedERC20.sol

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

REMEDIATION COMMIT ID:

- aa4445a

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

7. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	1	0	0

INFORMATIONAL
1

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INCORRECT STORAGE GAP REDUCTION COULD LEAD TO STORAGE COLLISION	HIGH	SOLVED - 07/16/2025

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INCOMPLETE EIP-6372 CLOCK MODE DETECTION MAY CAUSE SILENT MISCONFIGURATION	INFORMATIONAL	ACKNOWLEDGED - 07/23/2025

8. FINDINGS & TECH DETAILS

8.1 INCORRECT STORAGE GAP REDUCTION COULD LEAD TO STORAGE COLLISION

// HIGH

Description

The `TokenVoting` contract introduces a new storage variable `excludedAccounts` of type `EnumerableSet.AddressSet`. This newly introduced `AddressSet` is a struct containing a `Set` struct, which itself contains a dynamic array and a mapping:

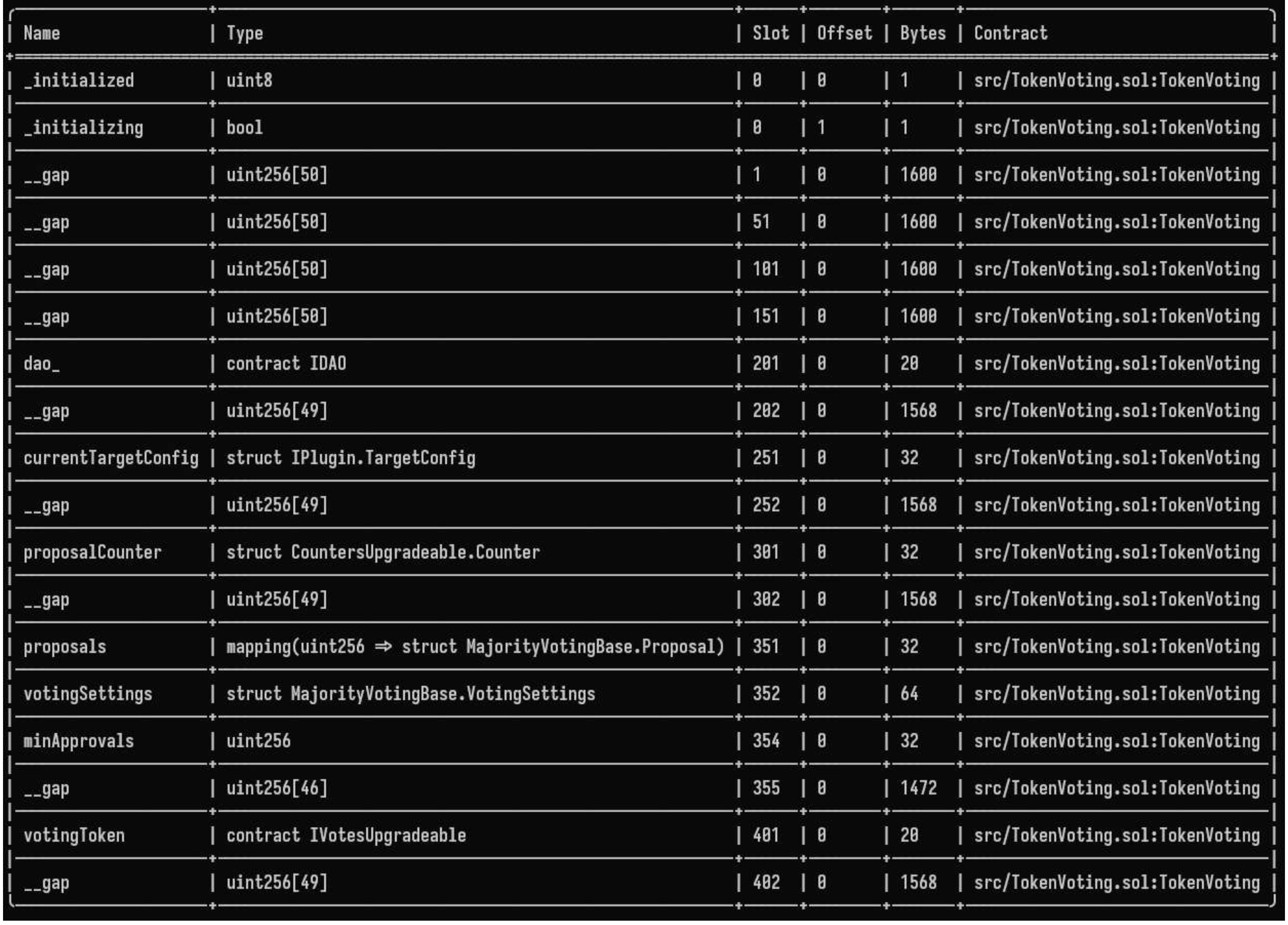
```
struct Set {
    // Storage of set values
    bytes32[] _values;
    // Position of the value in the `values` array, plus 1 because index 0
    // means a value is not in the set.
    mapping(bytes32 => uint256) _indexes;
}
```

Per Solidity **storage layout rules**, this struct occupies **two storage slots.** However, the storage gap `uint256[48] private __gap;` was only reduced by one (from 49 to 48).
This miscalculation causes the storage layout to shift, breaking upgradeability guarantees and potentially corrupting contract state for any future upgrades.

Proof of Concept

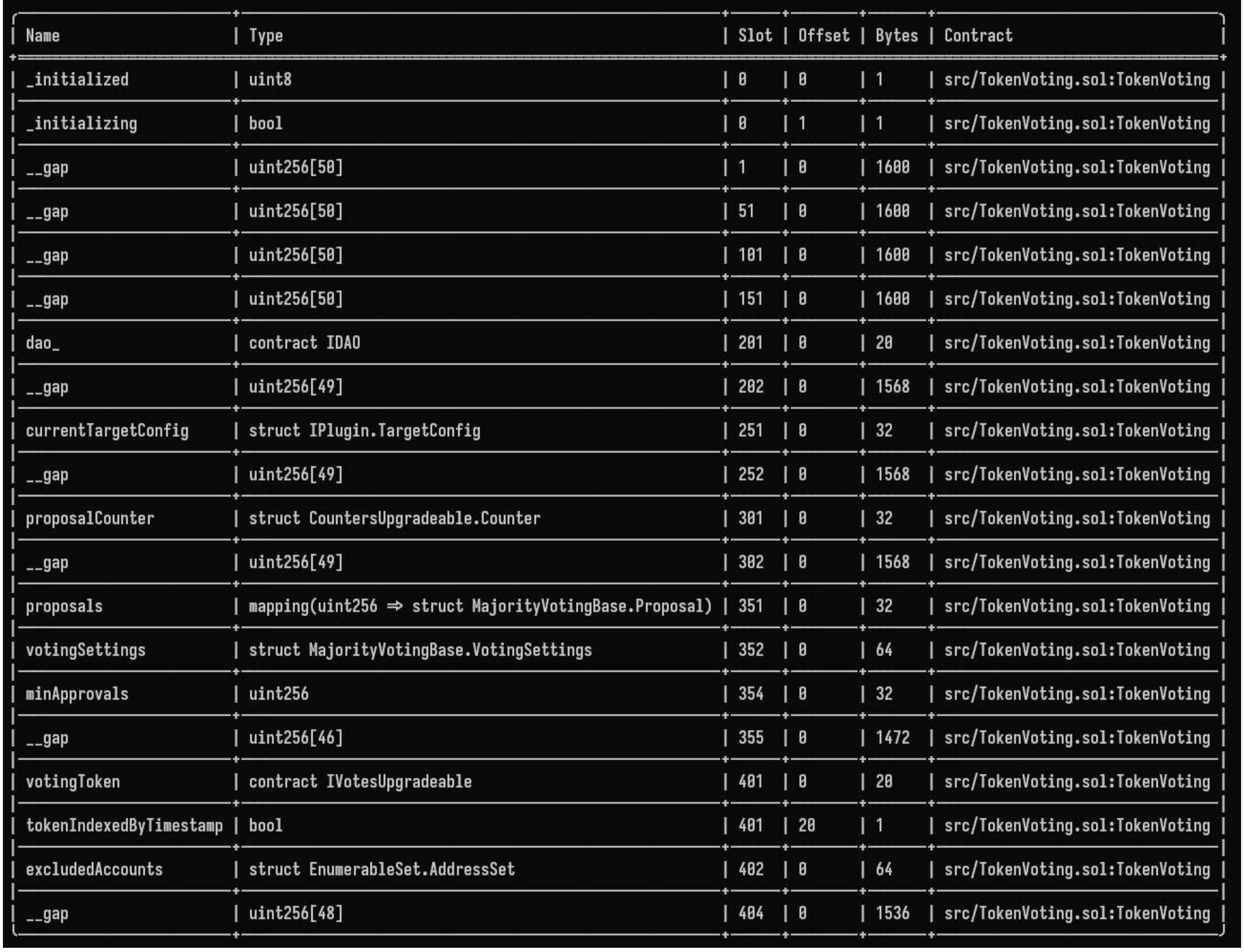
The output of the running the command `forge inspect src/TokenVoting.sol:TokenVoting storage --pretty` clearly demonstrates the difference in storage layout, where the final `__gap` is moved from slot `402` to `404` (2 slots):

- **Previous storage layout**:



```
| Name                | Type                                                 | Slot | Offset | Bytes | Contract                        |
+=====================================================================================================================================+
| _initialized        | uint8                                                | 0    | 0      | 1     | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| _initializing       | bool                                                 | 0    | 1      | 1     | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[50]                                          | 1    | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[50]                                          | 51   | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[50]                                          | 101  | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[50]                                          | 151  | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| dao_                | contract IDAO                                        | 201  | 0      | 20    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[49]                                          | 202  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| currentTargetConfig | struct IPlugin.TargetConfig                          | 251  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[49]                                          | 252  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| proposalCounter     | struct CountersUpgradeable.Counter                   | 301  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[49]                                          | 302  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| proposals           | mapping(uint256 => struct MajorityVotingBase.Proposal) | 351  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| votingSettings      | struct MajorityVotingBase.VotingSettings             | 352  | 0      | 64    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| minApprovals        | uint256                                              | 354  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[46]                                          | 355  | 0      | 1472  | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| votingToken         | contract IVotesUpgradeable                           | 401  | 0      | 20    | src/TokenVoting.sol:TokenVoting |
|---------------------+------------------------------------------------------+------+--------+-------+---------------------------------|
| __gap               | uint256[49]                                          | 402  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
+---------------------+------------------------------------------------------+------+--------+-------+---------------------------------+
```

- **New storage layout:**



```
| Name                  | Type                                               | Slot | Offset | Bytes | Contract                         |
+==================================================================================================================================+
| _initialized          | uint8                                              | 0    | 0      | 1     | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| _initializing         | bool                                               | 0    | 1      | 1     | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[50]                                        | 1    | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[50]                                        | 51   | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[50]                                        | 101  | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[50]                                        | 151  | 0      | 1600  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| dao_                  | contract IDAO                                      | 201  | 0      | 20    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[49]                                        | 202  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| currentTargetConfig   | struct IPlugin.TargetConfig                        | 251  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[49]                                        | 252  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| proposalCounter       | struct CountersUpgradeable.Counter                 | 301  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[49]                                        | 302  | 0      | 1568  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| proposals             | mapping(uint256 => struct MajorityVotingBase.Proposal) | 351  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| votingSettings        | struct MajorityVotingBase.VotingSettings           | 352  | 0      | 64    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| minApprovals          | uint256                                            | 354  | 0      | 32    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[46]                                        | 355  | 0      | 1472  | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| votingToken           | contract IVotesUpgradeable                         | 401  | 0      | 20    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| tokenIndexedByTimestamp | bool                                             | 401  | 20     | 1     | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| excludedAccounts      | struct EnumerableSet.AddressSet                    | 402  | 0      | 64    | src/TokenVoting.sol:TokenVoting |
|-----------------------+----------------------------------------------------+------+--------+-------+---------------------------------|
| __gap                 | uint256[48]                                        | 404  | 0      | 1536  | src/TokenVoting.sol:TokenVoting |
```

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:H/D:N/Y:N (7.5)

Recommendation

Reduce the storage gap by two instead of one. Change the gap declaration to:

```
uint256[47] private __gap;
```

Remediation Comment

SOLVED: The **Aragon team** solved this finding in the specified commit by following the mentioned recommendation.

Remediation Hash

https://github.com/aragon/TokenVotingPlugin/commit/aa4445a3b2b421feeed33ad000c7a0e7312f3817

8.2 INCOMPLETE EIP-6372 CLOCK MODE DETECTION MAY CAUSE SILENT MISCONFIGURATION

// INFORMATIONAL

Description

The `_detectTokenClock()` function in the `TokenVoting` contract attempts to determine whether the governance token uses timestamp or block number snapshots by checking `CLOCK_MODE()` and `clock()` from EIP-6372. However, if neither timestamp mode nor matching clock value is detected, the function silently assumes block number mode without explicit validation. This could result in silent misconfiguration if the token does not implement EIP-6372 or uses a custom clock mode, potentially leading to incorrect voting power snapshots.

```
/// @dev Helper function to identify the clock mode used by the given voting token.
function _detectTokenClock() private {
    bool clockModeTimestamp;
    bool clockTimestamp;

    try IERC6372Upgradeable(address(votingToken)).CLOCK_MODE() returns (string memory clockMode) {
        clockModeTimestamp = keccak256(bytes(clockMode)) == keccak256(bytes("mode=timestamp"));
    } catch {}
    try IERC6372Upgradeable(address(votingToken)).clock() returns (uint48 timePoint) {
        clockTimestamp = (timePoint == block.timestamp);
    } catch {}

    if (clockModeTimestamp != clockTimestamp) {
        revert TokenClockMismatch();
    } else if (clockModeTimestamp) {
        tokenIndexedByTimestamp = true;
    } else {
        // Assuming that the token indexes by block number
    }
}
```

BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:L/D:N/Y:N (0.5)

Recommendation

Explicitly check for both `"mode=timestamp"` and `"mode=blocknumber"` in `CLOCK_MODE()`. If neither is detected, revert with an error to prevent unsupported or ambiguous clock modes from being used.

Remediation Comment

ACKNOWLEDGED: The **Aragon team** made a business decision to acknowledge this finding and not alter the contracts.

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.