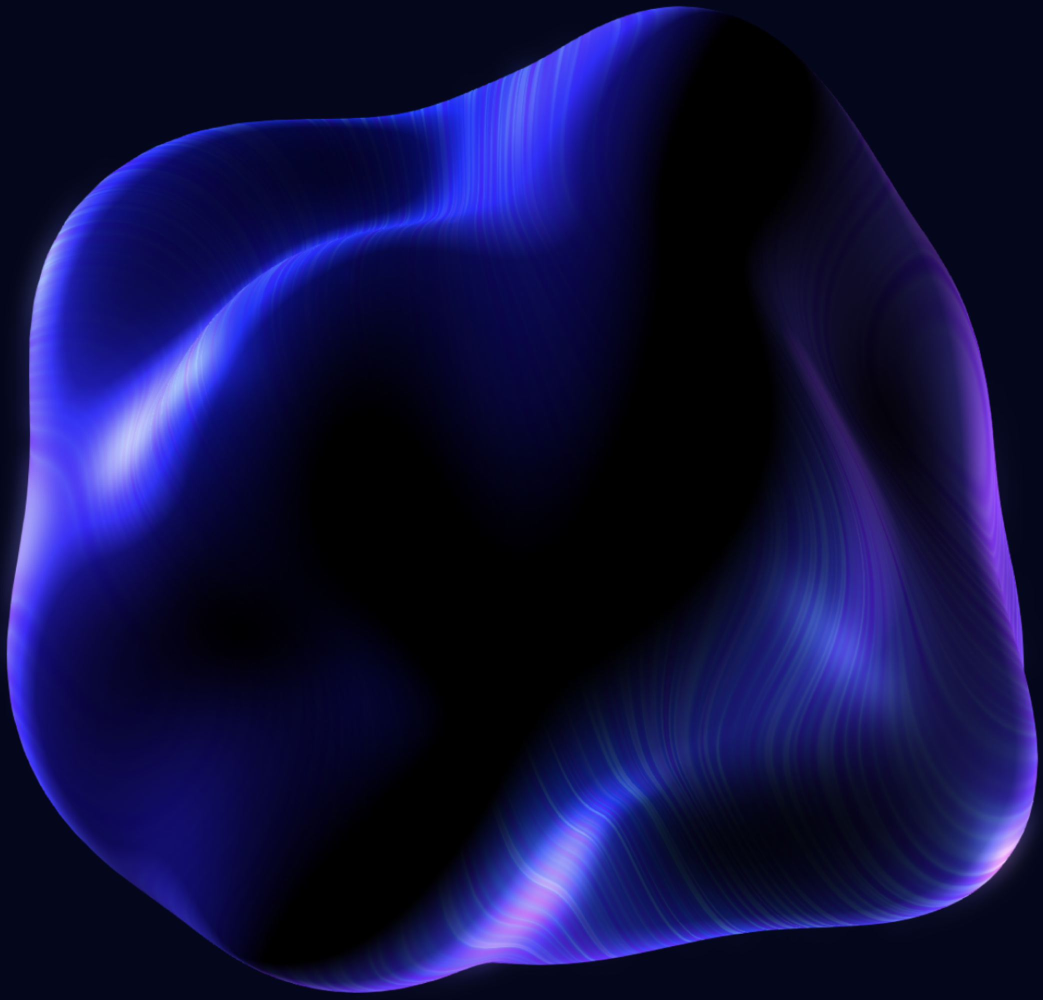
MUNDUS
SECURITY

# Security
# Smart Contract Audit
# MahaDAO Governance



mundus.dev    @mundus_security    Mundus.dev

MUNDUS
SECURITY

# MahaDAO Governance security audit

## Reference information

| | |
|---|---|
| Name | MahaDAO Governance Contracts |
| Language | Solidity |
| Chain | Ethereum mainnet |
| Website | https://mahadao.com/ |
| Documentation | https://docs.mahadao.com/ |
| Reference repositories | https://github.com/MahaDAO/governance-contracts<br>https://github.com/MahaDAO/token |

# Findings summary

## Findings statistics

| **Severity** | **Number** | **Left acknowledged** |
|---|---:|---:|
| **High** | 0 | 0 |
| **Medium** | 3 | 1 |
| Low | 5 | 5 |
| Informational | 13 | 12 |
| Gas | 9 | 9 |
| **Total** | **29** | **26** |

## Finding Severity breakdown

All vulnerabilities discovered during the source code audit are classified based on their potential severity and have the following classification:

| **Severity** | **Description** |
|---|---|
| **High** | Bugs that can trigger a contract failure or theft of assets. Further recovery is possible only by manual modification of the contract state or replacement of the contract. |
| **Medium** | Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds. |
| Low | Bugs that do not pose significant danger to the project or its users but are recommended to be fixed nonetheless. |
| Informational | All other non-essential recommendations. |
| Gas | Gas optimization recommendations. |

# Project description

## MahaDAO

MahaDAO is a mission to create a decentralized and stable economy. That is driven by the people, for the people.

MahaDAO is a community-powered, decentralized organization on a mission to empower billions with a stable economy through the world's first valuecoin, ARTH.

To do this, MahaDAO uses two tokens to achieve this vision - the governance token MAHA, and the valuecoin ARTH.

## ARTH valuecoin

ARTH is a stablecoin that is designed to appreciate overtime against the US dollar while at the same time it remains relatively stable.

ARTH is minted/burnt using decentralized smart contracts that use ETH as collateral to maintain its peg. The interest rate charged to mint ARTH using ETH is 0%, which makes it very cost-effective for borrowing/lending.

ARTH is fully collateralized with mechanisms that give it a backing of at least 110% in ETH.

MUNDUS
SECURITY

## Scope of work

| Contract | Address |
|---|---|
| BaseV2Bribes | 0x8f362e16a74c2eb564bfbf24dc73bd5ce37d9063 |
| BaseV2Voter<br>*Proxy* | 0x227a445ff220cc9c3584fe77b7dfef6af0b63e8e<br>0xeb99748e91afca94a6289db3b02e7ef4a8f0a22d |
| EmissionController | 0xbd86a195c90cec4606dbc378ea0aa338f674a704 |
| FeesSplitter | 0x9032f1bd0cc645fde1b41941990da85f265a7623 |
| GaugeLP<br>*Proxy* | 0xd2125a722d28c7685aed658a3ddc7b08275b8aeb<br>0x9ee8110c0aacb7f9147252d7a2d95a5ff52f8496 |
| GaugeUniswapV3<br>*Proxy* | 0xa7af7eaa2bf2fbea3fdb90db1a820508ed3f037c<br>0x98e1701f6558dd63481b57926c9f22c64d918c35 |
| MAHATimelockController-14 | 0x43c958affe41d44f0a02ae177b591e93c86adbea |
| MAHATimelockController-30 | 0xb45021f5313b93927699aae6cbe989bccf6b5900 |
| MahaToken | 0x745407c86df8db893011912d3ab28e68b62e49b0 |
| MAHAXGovernor | 0xe7d23c2b3e9148c46cec796f018842ab72d5867f |
| MAHAXLocker | 0xbdd8f4daf71c2cb16cce7e54bb81ef3cfcf5aacb |
| MAHAXStaker | 0x608917f8392634428ec71c6766f3ec3f5cc8f421 |
| MAHAXVetoGovernor | 0x9a7e7b4c2abe3255dec67e3bf2e6b24b46223111 |
| Registry | 0x2684861ba9dada685a11c4e9e5aed8630f08afe0 |
| RenderingContract | 0x9d348281e16218cd8ede9cd8a1bca74e89b410e8 |

## Findings

| ID | Severity | Description | Status |
|---|---|---|---|
| 01 | **Medium** | Potential DoS in getReward of GaugeLP.sol | Ack. |
| 02 | **Medium** | distributeETH should be non-reentrant in FeesSplitter.sol | Fixed |
| 03 | **Medium** | Arbitrary IERC20 in distributeERC20 public of FeesSplitter.sol | Fixed |
| 04 | Low | Unnecessary events emitted by getReward in BaseV2Bribes.sol | Ack. |
| 05 | Low | Sanity check required for setters in Registry.sol | Ack. |
| 06 | Low | Sanity check required for period initialization in Epoch.sol | Ack. |
| 07 | Low | Sanity check required for amountToSplit in FeesSplitter.sol | Ack. |
| 08 | Low | Sanity check required for _accounts.length in FeesSplitter.sol | Ack. |
| 09 | Informational | Unused imports in GaugeLP.sol | Ack. |
| 10 | Informational | Unused imports in BaseV2Voter.sol | Ack. |
| 11 | Informational | Unused import {INFTLocker} in BaseV2Bribes.sol | Ack. |
| 12 | Informational | onlyOwner modifier for checkPercentages method is excessive in FeesSplitter.sol | Ack. |
| 13 | Informational | Misleading comment in MAHAXVetoGovernor.sol | Ack. |
| 14 | Informational | Misleading comment in MAHAXGovernor.sol | Ack. |
| 15 | Informational | Misleading comment in GaugeLP.sol | Ack. |
| 16 | Informational | Misleading comment in EmissionController.sol | Ack. |
| 17 | Informational | Misleading comment in BaseV2Bribes.sol | Ack. |
| 18 | Informational | Logic duplication in getReward and getRewardForOwner in BaseV2Bribes.sol | Ack. |
| 19 | Informational | Dead code: SafeERC20 in FeesSplitter.sol | Fixed |
| 20 | Informational | Dead code: modifier checkStartTime in Epoch.sol | Ack. |

| ID | Severity | Description | Status |
|---|---|---|---|
| 21 | Informational | Add events to MAHAXLocker.sol | Ack. |
| 22 | Gas | _users.length should be cached in for-loops in MAHAXLocker.sol | Ack. |
| 23 | Gas | tokens.length should be cached in for-loops in BaseV2Bribes.sol | Ack. |
| 24 | Gas | tokenIds.length should be cached in for-loops in GaugeUniswapV3.sol | Ack. |
| 25 | Gas | targets.length should be cached in for-loops in MAHATimelockController.sol | Ack. |
| 26 | Gas | registry should be immutable in MAHAXLocker.sol | Ack. |
| 27 | Gas | registry should be immutable in EmissionController.sol | Ack. |
| 28 | Gas | _percentAllocations.length should be cached in for-loops in FeesSplitter.sol | Ack. |
| 29 | Gas | _gauges.length should be cached in for-loops in BaseV2Voter.sol | Ack. |
| 30 | Gas | balanceOf[account] should be cached in for-loops in GaugeUniswapV3.sol | Ack. |

# Source code audit

## ID-01. **Medium**: Potential DoS in getReward of GaugeLP.sol

### Description

The getReward method of GaugeLP.sol performs calls the distribute method of the **BaseV2Voter** contract (see @audit_1), which in turn calls back the notifyRewardAmount method of the **GaugeLP** contract, if specific criteria are met (see @audit_2).

```
// @audit_1 GaugeLP.sol getReward call to BaseV2Voter
_unlocked = 1;
IGaugeVoterV2(registry.gaugeVoter()).distribute(address(this));
_unlocked = 2;

// @audit_2 BaseV2Voter.sol distribute call to GaugeLP
claimable[_gauge] = 0;
IGauge(_gauge).notifyRewardAmount(registry.maha(), _claimable);
emit DistributeReward(msg.sender, _gauge, _claimable);
```

The getReward method is protected against reentrancy with the require(_unlocked == 0, "reentrancy"); requirement of the lock modifier. Thus, the **BaseV2Voter** contract's call to **GaugeLP** will revert due to _unlocked local variable not being set to 0 (see @audit_1).

### Recommendation

Modify the getReward method of GaugeLP.sol in the following way

```
_unlocked = 0;
IGaugeVoterV2(registry.gaugeVoter()).distribute(address(this));
_unlocked = 1;
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-02. **Medium**: distributeETH should be non-reentrant in FeesSplitter.sol

### Description

The distributeETH method of FeesSplitter.sol performs low-level calls to addresses in a for-loop to distribute ETH. This pattern is prone to reentrancy, e.g. a CREATE2 address with predefined malicious logic could be provided to accounts state variable which later drains out the **FeesSplitter** contract.

### Recommendation

Add nonReentrant modifier to the distributeETH method of FeesSplitter.sol.

### Alleviation

The issue fix was introduced in commit ce2036f053f68fd48ed043f572e6aeb952b3f33d. The distributeETH function now has nonReentrant modifier.

## ID-03. **Medium**: Arbitrary IERC20 in distributeERC20 public of FeesSplitter.sol

### Description

The distributeERC20 method of FeesSplitter.sol has no access control and takes arbitrary ERC20 as its argument. This is a dangerous pattern that should be addressed.

### Recommendation

Modify the distributeERC20 method's logic in any or all of the following ways

- Add nonReentrant and/or onlyOwner modifier to the distributeERC20 method.
- Use Openzeppelin's SafeERC20 library for token transfers.
- Introduce ERC20 whitelist to the **FeesSplitter** contract's storage.

### Alleviation

The issue fix was introduced in commit f9bbbb0d1cd8d2a632405cb2f4adc3fc24417b1f. The distributeERC20 function now uses token.safeTransfer.

## ID-04. Low: Unnecessary events emitted by getReward in BaseV2Bribes.sol

### Description

The getReward and getRewardForOwner methods of BaseV2Bribes.sol transfers rewards per each tokens[i] if _reward > 0. However, the ClaimRewards event is emitted irrespective of reward for a particular token being > 0.

### Recommendation

Construct a _getReward() internal function (see Issue ) and modify its logic in the following way

```
if (_reward > 0) {
    _safeTransfer(tokens[i], _owner, _reward);
    emit ClaimRewards(_owner, tokens[i], _reward);
}
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-05. Low: Sanity check required for setters in Registry.sol

### Description

The setters for state variables of type address in Registry.sol, e.g. the setMAHA method, should exercise isContract sanity check.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-06. Low: Sanity check required for period initialization in Epoch.sol

### Description

Each epoch timestamp in `Epoch.sol` is calculated as the product of epoch number, `_getNextEpoch()`, and the epoch period, `period`. Subsequently, other methods execute their corresponding logic via comparing this product with `block.timestamp`. Thus, the `period` cannot be less than the maximum time delta between consequent blocks. Adding a sanity check for `period` being larger than some reasonable time delta is required.

### Recommendation

Add `require(_period >= 86400)` to the constructor and `setPeriod` method of `Epoch.sol`.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-07. Low: Sanity check required for amountToSplit in FeesSplitter.sol

### Description

Sanity check for `amountToSplit > PERCENTAGE_SCALE/min(percentAllocations)` is required for the `distributeETH` and `distributeERC20` methods of `FeesSplitter.sol`.

### Recommendation

Addition of `require(amountToSplit > PERCENT_SCALE)` to the `distributeETH` and `distributeERC20` methods of `FeesSplitter.sol` should suffice.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-08. Low: Sanity check required for `_accounts.length` in `FeesSplitter.sol`

### Description

Sanity check is required for lengths of `accounts` and `percentAllocations` state variables in the constructor and `updateSplit` method of `FeesSplitter.sol`.

### Recommendation

Add `require(_accounts.length == _percentAllocations.length)` to constructor and `updateSplit` method of `FeesSplitter.sol`.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-09. Informational: Unused imports in `GaugeLP.sol`

### Description

`IBribe` and `INFTLocker` are unused imports in `GaugeLP.sol` and should be removed.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-10. Informational: Unused imports in `BaseV2Voter.sol`

### Description

`IBribeFactory` and `IUniswapV2Pair` are unused imports in `BaseV2Voter.sol` and should be removed.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-11. Informational: Unused `import {INFTLocker}` in `BaseV2Bribes.sol`

### Description

`INFTLocker` is an unused import in `BaseV2Bribes.sol` and should be removed.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-12. Informational: `onlyOwner` modifier for `checkPercentages` method is excessive in `FeesSplitter.sol`

### Description

The `onlyOwner` modifier for `function checkPercentages view` is excessive as it does not deal with contract's storage on its own. The `onlyOwner` modifier should be removed and the `checkPercentages` method should be declared `pure`.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-13. Informational: Misleading comment in MAHAXVetoGovernor.sol

### Description

MAHAXVetoGovernor.sol contains misleading comment on the quorum method (see @audit).

```
/**                              @audit
 * @dev Returns the quorum for a block number,
 * in terms of number of votes: `supply * numerator / denominator`.
 */
function quorum(uint256 blockNumber)
    public
    view
    override
    returns (uint256)
{
    return _quorum; // @audit
}
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-14. Informational: Misleading comment in MAHAXGovernor.sol

### Description

MAHAXGovernor.sol contains misleading comment on the quorum(uint256) method (see @audit).

```
/**                                @audit
  * @dev Returns the quorum for a block number,
  * in terms of number of votes: `supply * numerator / denominator`.
  */
function quorum(uint256) public view override returns (uint256) {
    return _quorum; // @audit
}
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-15. Informational: Misleading comment in GaugeLP.sol

### Description

GaugeLP.sol contains the same comment on supplyCheckpoints and rewardPerTokenCheckpoints state variables.

```
/// @notice A record of balance checkpoints for each token, by index
mapping(uint256 => SupplyCheckpoint) public supplyCheckpoints;

/// @notice A record of balance checkpoints for each token, by index
mapping(address => mapping(uint256 => RewardPerTokenCheckpoint))
    public rewardPerTokenCheckpoints;
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-16. Informational: Misleading comment in EmissionController.sol

### Description

The allocateEmission method of EmissionController.sol contains misleading comment about token transfer approval, which doe not take place in this method. This is likely related to the code base upgrade from BaseV1Voter.sol to BaseV2Voter.sol.

```
// approve token and notify the gauge voter
IERC20(registry.maha()).transfer(registry.gaugeVoter(), balanceToSend);
IGaugeVoter(registry.gaugeVoter()).notifyRewardAmount(balanceToSend);
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-17. Informational: Misleading comment in BaseV2Bribes.sol

### Description

GaugeLP.sol contains the same comment on supplyCheckpoints and rewardPerTokenCheckpoints state variables.

```
/// @notice A record of balance checkpoints for each token, by index
mapping(uint256 => SupplyCheckpoint) public supplyCheckpoints;

/// @notice A record of balance checkpoints for each token, by index
mapping(address => mapping(uint256 => RewardPerTokenCheckpoint))
    public rewardPerTokenCheckpoints;
```

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-18. Informational: Logic duplication in getReward and getRewardForOwner in BaseV2Bribes.sol

### Description

The getReward and getRewardForOwner methods of BaseV2Bribes.sol are unnecessary duplicates of one another. Consider defining a _getReward internal function and getReward and getRewardForOwner wrappers for it.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-19. Informational: Dead code: SafeERC20 in FeesSplitter.sol

### Description

OpenZeppelin's library SafeERC20 is imported but never used in FeesSplitter.sol.

### Alleviation

See issue ID-02 fix.

## ID-20. Informational: Dead code: modifier checkStartTime in Epoch.sol

### Description

checkStartTime is an unused modifier in Epoch.sol and should be removed.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-21. Informational: Add events to MAHAXLocker.sol

### Description

The setRoyaltyInfo and setMinLockAmount methods of MAHAXLocker.sol should emit corresponding events.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-22. Gas: _users.length should be cached in for-loops in MAHAXLocker.sol

### Description

_users.length should be cached in the uploadUsers method of MAHAXLocker.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-23. Gas: tokens.length should be cached in for-loops in BaseV2Bribes.sol

### Description

tokens.length should be cached for gas savings in getReward and getRewardForOwner methods of BaseV2Bribes.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-24. Gas: tokenIds.length should be cached in for-loops in GaugeUniswapV3.sol

### Description

tokenIds.length should be cached for gas savings in the isIdsWithinRange and claimFeesMultiple methods of GaugeUniswapV3.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-25. Gas: targets.length should be cached in for-loops in MAHATimelockController.sol

### Description

targets.length should be cached for gas savings in the scheduleBatch and executeBatch methods of MAHATimelockController.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-26. Gas: registry should be immutable in MAHAXLocker.sol

### Description

registry state variable of MAHAXLocker.sol should be immutable for gas savings.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-27. Gas: registry should be immutable in EmissionController.sol

### Description

registry state variable of EmissionController.sol should be immutable for gas savings.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-28. Gas: _percentAllocations.length should be cached in for-loops in FeesSplitter.sol

### Description

_percentAllocations.length should be cached for gas savings in the checkPercentages method of FeesSplitter.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-29. Gas: _gauges.length should be cached in for-loops in BaseV2Voter.sol

### Description

_gauges.length should be cached for gas savings in distribute(address[] memory) and updateFor(address[] memory methods of BaseV2Voter.sol.

### Alleviation

This issue is acknowledged by the MahaDAO team.

## ID-30. Gas: `balanceOf[account]` should be cached in for-loops in `GaugeUniswapV3.sol`

### Description

`balanceOf[account]` should be cached for gas savings in the `getReward(address account, address[] memory)` method of `GaugeUniswapV3.sol`.

### Alleviation

This issue is acknowledged by the MahaDAO team.

# Disclaimers

## Mundus disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

## Technical disclaimers

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.