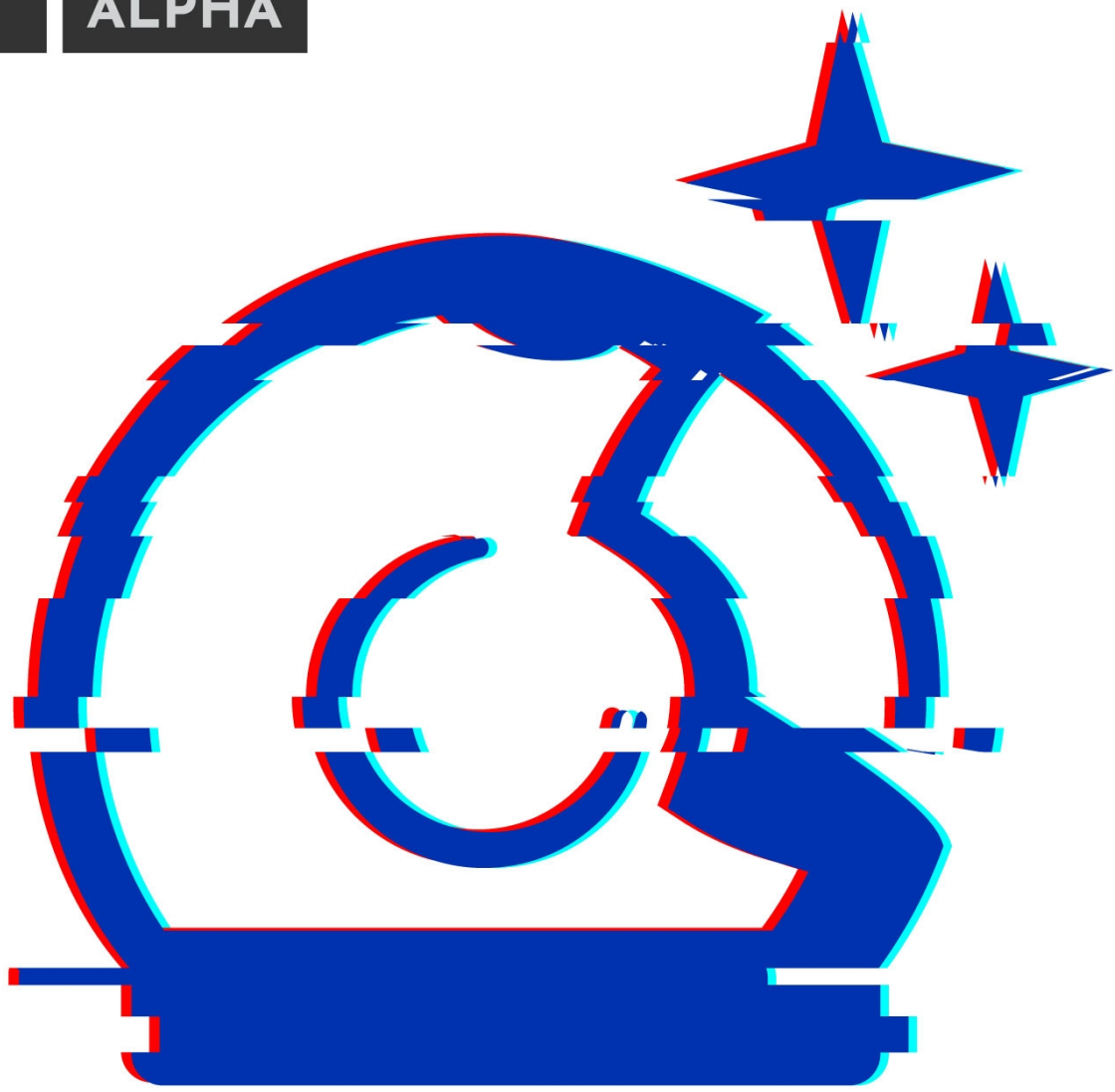
MONO CEROS ALPHA



PROJECT

Meta Governance

CLIENT

Indexed

DATE

January 2021

REVIEWERS

Daniel Luca

@cleanunicorn

Table of Contents

Details

- **Client** Indexed
- **Date** January 2021
- **Lead reviewer** Daniel Luca (@cleanunicorn)
- **Reviewers** Daniel Luca (@cleanunicorn)
- **Repository**: Meta Governance
- **Commit hash** `da7863173ca951be170dae80bb4c90330345860b`
- **Technologies**
  - Solidity
  - Node.JS

Issues Summary

SEVERITY	OPEN	CLOSED

SEVERITY	OPEN	CLOSED
Informational	0	1
Minor	0	4
Medium	0	0
Major	0	0

Executive summary

This report represents the results of the engagement with **Indexed** to review **Meta Governance**.

The review was conducted over the course of **2 days** from **January 24 to January 25, 2020**. A total of **2 person-days** were spent reviewing the code.

Day 1

During the first day, we got familiar with the provided proposal which also serves as documentation.

The initial proposal was shared with the community on the Indexed forum:

- https://forum.indexed.finance/t/draft-iip-3-enable-meta-governance/53

The initial proposal was followed up by a Snapshot proposal:

- https://snapshot.page/#/ndx.eth/proposal/QmYzjH27kBiCVHs7vYzQg4nbXiJWyvmN5fbrxANsnTnEZ3

More information about how Snapshot works can be found in their documentation.

Currently there are 10 tokens the indices currently hold: UNI, AAVE, COMP, SNX, CRV, MKR, YFI, UMA, LINK, OMG. Only 2 of those tokens do now hold any governance power, specifically LINK and OMG.

Because most of the tokens held by the index contract have governance power, the proposal wants to allow these tokens to delegate their voting power other contracts that can later participate in their specific community proposals. This unlocks the voting power of held tokens in the current Indexed indices.

As identified in the initial forum proposal, the complexity lies in adapting all types of voting power delegation. Fortunately, a lot of tokens implement Compound-like governance.

Day 2

During the second day we had a meeting with the client where we presented initial findings and discussed details about the implementation.

At the end of the day the report was generated and shared with the client.

Scope

The initial review focused on the Meta Governance identified by the commit hash `da7863173ca951be170dae80bb4c90330345860b`.

We focused on manually reviewing the codebase, searching for security issues such as, but not limited to re-entrancy problems, transaction ordering, block timestamp dependency, exception handling, call stack depth limitation, integer overflow/underflow, self-destructible contracts, unsecured balance, use of origin, gas costly patterns, architectural problems, code readability.

Includes:

- contracts/meta/MetaGovernorCOMP.sol

Excludes:

- everything else

Recommendations

We identified a few possible general improvements that are not security issues during the review, which will bring value to the developers and the community reviewing and using the product.

Issues

[MetaGovernorCOMP] - Consider transforming `castVote` from a wrapper to a function

Status Fixed Severity Minor

Description

When an actor calls `castVote`, the execution is forwarded to the internal function `_castVote`.

code/contracts/meta/MetaGovernorCOMP.sol#L131-L133

```
function castVote(uint256 proposalId, bool support) external {
  return _castVote(msg.sender, proposalId, support);
}
```

This is the only instance where the internal function `_castVote` is called; thus, removing the call with the function's body can improve gas costs, slightly reduce code and increase readability.

Recommendation

Remove the call to `_castVote` with the contents of the internal function `_castVote` directly.

A similar rewrite can be applied to `state`.

code/contracts/meta/MetaGovernorCOMP.sol#L180-L183

```
function state(uint256 proposalId) external view returns (MetaProposalState) {
  MetaProposal storage proposal = proposals[proposalId];
  return _state(proposal);
}
```

And to `_getMetaProposal`.

code/contracts/meta/MetaGovernorCOMP.sol#L151

```
  MetaProposal storage proposal = _getMetaProposal(proposalId);
```

[MetaGovernorCOMP.spec.js] - Some variables are never used

Status Acknowledged Severity Minor

Description

- `ndxTimelock` and `ndxGovernor` are initialized but never used.

[code/test/MetaGovernorCOMP.spec.js#L37](code/test/MetaGovernorCOMP.spec.js#L3

```
const { constants, Contract } = require('ethers')
```

7.

```
let ndx, ndxTimelock, ndxGovernor;
```

- `constants` is never used

code/test/MetaGovernorCOMP.spec.js#L3

```
const { constants, Contract } = require('ethers')
```

- `deployments` is never used

code/test/MetaGovernorCOMP.spec.js#L4

```
const { deployments, ethers } = bre;
```

Recommendation

Remove the unused variables.

[MetaGovernorCOMP] - Consider moving interfaces to separate files

Status Acknowledged Severity Minor

Description

Currently, the `MetaGovernor` contract needs to interact with external contracts. This is why the interfaces are needed. The minimum interfaces are currently defined at the end of the file.

[code/contracts/meta/MetaGovernorCOMP.sol#L202-L226](
code/contracts/meta/MetaGovernorCOMP.sol#L2

```
pragma solidity ^0.6.0;
```

02-L226)

```
interface IGovernorAlpha {
  struct Proposal {
    uint256 id;
    address proposer;
    uint256 eta;
    uint256 startBlock;
    uint256 endBlock;
    uint256 forVotes;
    uint256 againstVotes;
    bool canceled;
    bool executed;
  }

  function proposals(uint256 proposalId) external view returns (Proposal memory);
```

```
  function castVote(uint256 proposalId, bool support) external;
}


interface NdxInterface {
  function getPriorVotes(address account, uint256 blockNumber)
    external
    view
    returns (uint96);
}
```

Recommendation

Consider extracting them into separate files which can be included in `MetaGovernorCOMP.sol` and `MetaGovernorUNI.sol`.

[MetaGovernorCOMP] - Pin Solidity to the latest version

Status Fixed Severity Minor

Description

The current contract specifies a Solidity version of `^0.6.0` but does not specify the exact compiler version.

code/contracts/meta/MetaGovernorCOMP.sol#L2

```
pragma solidity ^0.6.0;
```

This can sometimes create problems, whether it's during the verification of the source code on Etherscan, during the compilation phase, or future problems while reading the source code but not knowing exactly what version was used.

It's a good practice to use the latest, most up to date Solidity version.

Recommendation

Pin Solidity version to `0.6.12`, unless a newer `0.6.x` version is released.

[MetaGovernorCOMP] - The Compound vote might go against a user's preference

Status Acknowledged Severity Informational

Description

The `MetaGovernorCOMP` contract tallies the votes of the NDX token holders for a proposal that already exists in the Compound governance contract.

That proposal is retrieved from the Compound governance contract and cached locally, the contract's storage.

code/contracts/meta/MetaGovernorCOMP.sol#L139

```
    IGovernorAlpha.Proposal memory externalProposal = compGovernor.proposals(proposalId);
```

After the proposal active period passed, the `forVotes` are compared with the `againstVotes`.

code/contracts/meta/MetaGovernorCOMP.sol#L194-L196

```
  } else if (proposal.forVotes > proposal.againstVotes) {
    return MetaProposalState.Succeeded;
  }
```

And the result of the tallied votes is executed against the Compound governance.

code/contracts/meta/MetaGovernorCOMP.sol#L126-L127

```
  bool support = state == MetaProposalState.Succeeded;
  compGovernor.castVote(proposalId, support);
```

When the vote is cast, all of the pool's tokens are used to vote in the Compound governance proposal. This means that even if some of the NDX users voted against a proposal, but this proposal passes, all of the deposited tokens will be used as voting power in the Compound governance proposal; even the tokens which were deposited by the users who voted against it.

Recommendation

There's no recommendation for a change, but the users should be aware that their deposited tokens will be used to cast votes on the Compound governance, even if they voted against a `MetaGovernorCOMP` proposal, but it passed (or against but it failed), or if they did not vote at all.

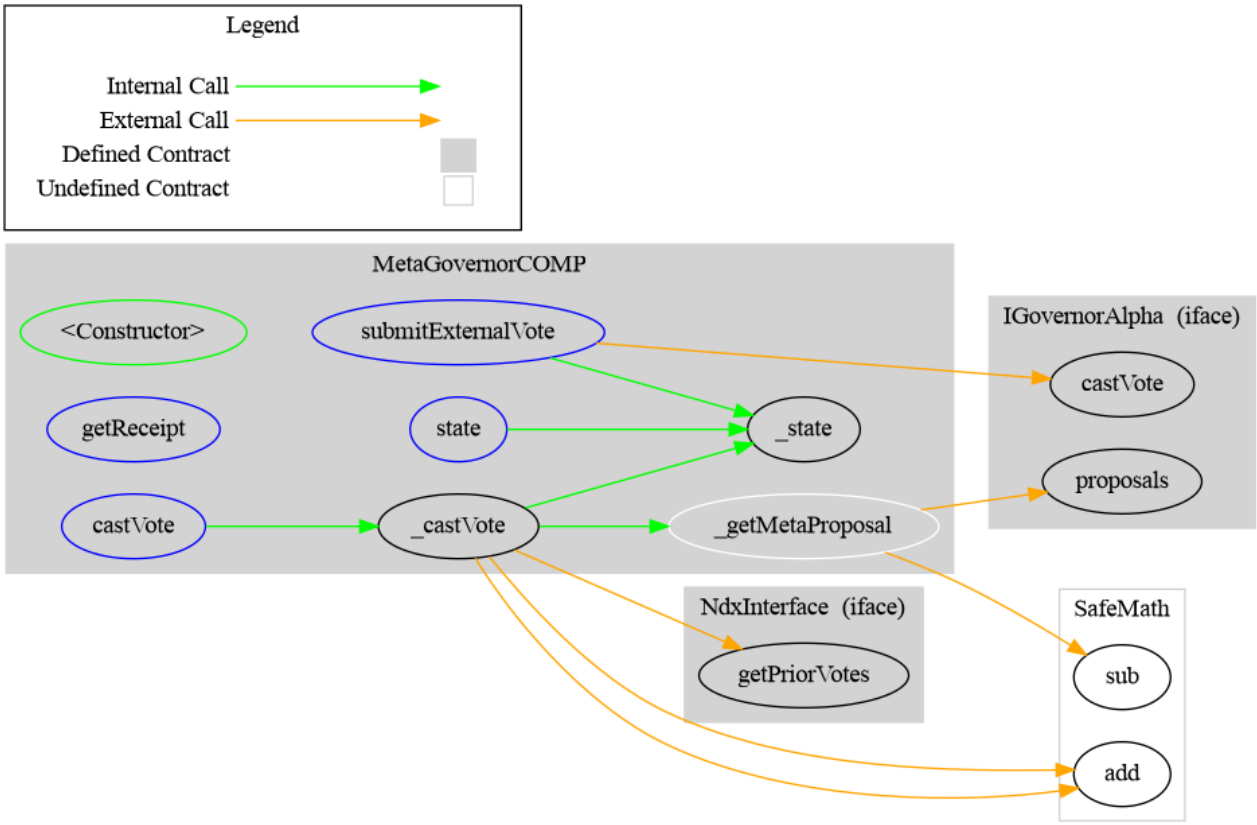
Also, there's no minimum vote limit, so the `MetaGovernorCOMP` proposals can be won even if only 1 vote was cast with 1 voting power.

Artifacts

Sūrya

Sūrya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Graph



Description

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
MetaGovernorCOMP	Implementation			
L		Public ❗	🛑	NO ❗
L	getReceipt	External ❗		NO ❗
L	submitExternalVote	External ❗	🛑	NO ❗
L	castVote	External ❗	🛑	NO ❗
L	_getMetaProposal	Internal 🔒	🛑	
L	_castVote	Internal 🔒	🛑	
L	state	External ❗		NO ❗

Contract	Type	Bases		
└	_state	Internal 🔒		
IGovernorAlpha	Interface			
└	proposals	External ❗		NO ❗
└	castVote	External ❗	🛑	NO ❗
NdxInterface	Interface			
└	getPriorVotes	External ❗		NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Coverage

```
$ yarn coverage
yarn run v1.22.10
$ buidler coverage --network coverage --solcoverjs ./.solcover.js
> server:            http://127.0.0.1:8555
> ganache-core:      v2.13.2
> solidity-coverage: v0.7.14

Network Info
============
> port:         8555
> network:      coverage


Instrumenting for coverage...
=============================

> distribution/DelegatingVester.sol
> distribution/Reservoir.sol
> distribution/RewardsDistributionRecipient.sol
> distribution/StakingRewards.sol
> distribution/StakingRewardsFactory.sol
> distribution/TreasuryLock.sol
> distribution/TreasuryVester.sol
> meta/MetaGovernorCOMP.sol
> meta/MetaGovernorUNI.sol

Coverage skipped for:
```

```
=====================

> governance/GovernorAlpha.sol
> governance/Ndx.sol
> governance/Timelock.sol
> interfaces/IStakingRewards.sol
> interfaces/IStakingRewardsFactory.sol
> interfaces/ITimelock.sol
> mocks/BaseERC20.sol
> mocks/MockERC20.sol
> mocks/MockPoolFactory.sol
> mocks/MockTimelock.sol
> mocks/ReservoirErrorTrigger.sol
Compiling...


.coverage_contracts/interfaces/IStakingRewards.sol: Warning: SPDX license identifier not provide



.coverage_contracts/interfaces/IStakingRewardsFactory.sol: Warning: SPDX license identifier not



.coverage_contracts/interfaces/ITimelock.sol: Warning: SPDX license identifier not provided in s



.coverage_contracts/mocks/MockPoolFactory.sol: Warning: SPDX license identifier not provided in



.coverage_contracts/mocks/MockTimelock.sol: Warning: SPDX license identifier not provided in sou



.coverage_contracts/mocks/ReservoirErrorTrigger.sol: Warning: SPDX license identifier not provid



.coverage_contracts/interfaces/IStakingRewards.sol:39:23: Warning: This declaration shadows an e
  function initialize(address stakingToken, uint256 rewardsDuration) external;
                      ^------------------^
.coverage_contracts/interfaces/IStakingRewards.sol:34:3: The shadowed declaration is here:
  function stakingToken() external view returns (IERC20);
  ^-----------------------------------------------------^



.coverage_contracts/interfaces/IStakingRewards.sol:39:45: Warning: This declaration shadows an e
  function initialize(address stakingToken, uint256 rewardsDuration) external;
                                            ^---------------------^
```

```
.coverage_contracts/interfaces/IStakingRewards.sol:13:3: The shadowed declaration is here:
  function rewardsDuration() external view returns (uint256);
  ^-------------------------------------------------------^



.coverage_contracts/interfaces/IStakingRewards.sol:54:31: Warning: This declaration shadows an e
  function setRewardsDuration(uint256 rewardsDuration) external;
                              ^---------------------^
.coverage_contracts/interfaces/IStakingRewards.sol:13:3: The shadowed declaration is here:
  function rewardsDuration() external view returns (uint256);
  ^-------------------------------------------------------^



.coverage_contracts/governance/Timelock.sol:9:1: Warning: This contract has a payable fallback f
contract Timelock is ITimelock {
^ (Relevant source part starts here and spans across multiple lines).
.coverage_contracts/governance/Timelock.sol:64:3: The payable fallback function is defined here.
  fallback() external payable {}
  ^----------------------------^



.coverage_contracts/mocks/MockTimelock.sol:11:1: Warning: This contract has a payable fallback f
contract MockTimelock is ITimelock {
^ (Relevant source part starts here and spans across multiple lines).
.coverage_contracts/mocks/MockTimelock.sol:66:3: The payable fallback function is defined here.
  fallback() external payable {}
  ^----------------------------^



.coverage_contracts/governance/GovernorAlpha.sol:331:7: Warning: Using ".value(...)" is deprecat
      timelock.executeTransaction.value(proposal.values[i])(
      ^-------------------------------^



.coverage_contracts/mocks/ReservoirErrorTrigger.sol:20:3: Warning: Function state mutability can
  function triggerAdditionOverflow() public {
  ^ (Relevant source part starts here and spans across multiple lines).

Compiled 43 contracts successfully
All contracts have already been compiled, skipping compilation.


  distribution:DelegatingVester
    √ Constructor fails with invalid vesting times (51ms)
    √ claim:~half (113ms)
    √ claim:all (101ms)
    √ delegate:fail
```

```
    √ delegate (82ms)
    √ setRecipient:fail
    √ setRecipient (111ms)

  distribution:Reservoir
    √ drip() (83ms)
    √ drip() 0 drip rate (79ms)
    √ drip() many blocks (122ms)
    √ drip() more blocks than duration (121ms)
    errors
      √ dripTotal overflow (79ms)
      √ deltaDrip underflow (84ms)
      √ addition overflow (66ms)

  distribution:StakingRewards
    Initialization
      √ does not allow null staking token
      √ does not allow zero duration
      √ sets the staking token (107ms)
      √ can not be initialized twice
    Constructor & Initializer
      √ should set rewards token on constructor
      √ should set rewardsDistribution on constructor
      √ should set staking token on initialize
      √ should set rewardsDuration on initialize
    Function permissions
      √ only owner can call notifyRewardAmount
    lastTimeRewardApplicable()
      √ should return 0
      when updated
        √ should equal current timestamp (67ms)
    rewardPerToken()
      √ should return 0
      √ should be > 0 (486ms)
    stake()
      √ staking increases staking balance (446ms)
      √ cannot stake 0
    earned()
      √ should be 0 when not staking
      √ should be > 0 when staking (545ms)
      √ rewardRate should increase if new rewards come before DURATION ends (360ms)
      √ rewards token balance should rollover after DURATION (654ms)
    getReward()
      √ should increase rewards token balance (919ms)
      √ gives nothing for user with no owed rewards (255ms)
    getRewardForDuration()
      √ should increase rewards token balance (125ms)
    withdraw()
      √ cannot withdraw if nothing staked
      √ should increases lp token balance and decreases staking balance (876ms)
      √ cannot withdraw 0
    exit()
```

```
      √ should retrieve all earned and increase rewards bal (1110ms)
    notifyRewardAmount()
      √ Reverts if the provided reward is greater than the balance. (49ms)
      √ Reverts if the provided reward is greater than the balance, plus rolled-over balance. (1
    recoverERC20
      √ Reverts if not called by owner
      √ Reverts if token is rewards or staking token
      √ Recovers tokens (107ms)
    setRewardsDuration()
      √ should revert if not called by owner
      √ should revert if duration is 0
      √ should increase rewards duration before starting distribution (50ms)
      √ should revert when setting setRewardsDuration before the period has finished (460ms)
      √ should update when setting setRewardsDuration after the period has finished (626ms)
      √ should update when setting setRewardsDuration after the period has finished (1221ms)

  distribution:StakingRewardsFactory
    Constructor & Settings
      √ Rejects genesis earlier than block timestamp
      √ STAKING_REWARDS_IMPLEMENTATION_ID
      √ poolFactory
      √ proxyManager
      √ rewardsToken
      √ uniswapFactory
      √ weth
    notifyRewardAmounts()
      √ Reverts if there are no staking pools
      √ Notifies all the pools of their rewards (381ms)
    getStakingRewards()
      √ Reverts if the staking token provided does not have a rewards pool
    deployStakingRewardsForPool()
      √ Only allows owner to call deployStakingRewardsForPool
      √ Reverts if the staking token is not an index lp token
Returned address 0x7A8F76a5E3d51eC7607D8d36bD5A42aae697696F
Event address 0x7A8F76a5E3d51eC7607D8d36bD5A42aae697696F
      √ Allows the owner to deploy a staking pool for an index lp token (145ms)
      √ Fails duplicate deployment without calling proxy manager
      Staking Info
        √ computeStakingRewardsAddress()
        √ getStakingRewards()
        √ stakingTokens()
        √ stakingRewardsInfoByStakingToken()
      StakingRewards
        √ rewardsToken()
        √ stakingToken()
      notifyRewardAmount()
        √ Fails if the staking genesis timestamp has not passed
        √ Fails if the factory does not have sufficient tokens (42ms)
        √ Fails if the staking pool does not exist (65ms)
        √ Notifies the pool of its rewards if there are pending rewards (132ms)
        √ Does nothing if there are no pending rewards (65ms)
      setRewardsDuration()
```

```
      √ Reverts if not called by owner
      √ Reverts if stakingToken has no pool
      √ Updates the duration (77ms)
    recoverERC20()
      √ Reverts if token is rewards or staking token
      √ Recovers tokens (116ms)
    deployStakingRewardsForPoolUniswapPair
      √ Only allows owner to call deployStakingRewardsForPoolUniswapPair
      √ Reverts if the staking token is not an index lp token
      √ Reverts if index token is null address (49ms)
      √ Reverts if token is weth (43ms)
      √ Allows the owner to deploy a staking pool for an index lp token <-> weth uniswap pair (1
      √ Fails duplicate deployment without calling proxy manager
      Staking Info
        √ computeStakingRewardsAddress()
        √ getStakingRewards()
        √ stakingTokens()
        √ stakingRewardsInfoByStakingToken()
      StakingRewards
        √ rewardsToken()
        √ stakingToken()
      notifyRewardAmount()
        √ Fails if the factory does not have sufficient tokens
        √ Fails if the staking pool does not exist (51ms)
        √ Notifies the pool of its rewards if there are pending rewards (150ms)
        √ Does nothing if there are no pending rewards (59ms)
    increaseStakingRewards
      √ Can only be called by owner
      √ Reverts if amount is zero
      √ Reverts if pool does not exist
      √ Reverts if pool has pending rewards (58ms)
      √ Reverts if pool is still active (113ms)
      √ Reverts if factory has insufficient balance
      √ Succeeds when the pool is finished (157ms)

  distribution:TreasuryLock
    Constructor
      √ Reverts if recipient is null
      √ Reverts if token is null
      √ Reverts if unlockDate is too soon
      √ Sets the correct values (73ms)
    claim()
      √ Reverts if unlock date has not passed
      √ Transfers balance (124ms)

  distribution:TreasuryVester
    √ Constructor fails with invalid vesting times (54ms)
    √ claim:fail
    √ claim:~half (71ms)
    √ claim:all (70ms)
    √ setRecipient:fail
    √ setRecipient (115ms)
```

```
GovernorAlpha
  √ ndx
  √ timelock
  √ governor
  voting period
    √ votingPeriod initialized to 2880
    √ permanentVotingPeriod set to 17280
    √ setPermanentVotingPeriod: reverts if too early
    √ setPermanentVotingPeriod: adjusts voting period when allowed (53ms)

MetaGovernorCOMP
  castVote
    √ rejects if caller has no votes
    √ casts vote (123ms)
    √ stores the correct start and end block
    √ records caller vote
    √ does not set voteSubmitted
    √ creates receipt
    √ rejects duplicate vote
    √ rejects if proposal not active (12517ms)
  state
    √ Active (98ms)
    √ Defeated (6287ms)
    √ Succeeded (6450ms)
    √ Null / Not ready (131ms)
    √ Executed (6141ms)
  submitExternalVote
    rejection
      √ rejects if proposal does not exist
      √ rejects if proposal not ready (94ms)
    vote for
      √ submits to governor (6263ms)
    vote against
      √ submits to governor (6201ms)

MetaGovernorUNI
  castVote
    √ rejects if caller has no votes
    √ casts vote (119ms)
    √ stores the correct start and end block
    √ records caller vote
    √ does not set voteSubmitted
    √ creates receipt
    √ rejects duplicate vote
    √ rejects if proposal not active (12488ms)
  state
    √ Active (99ms)
    √ Defeated (6431ms)
    √ Succeeded (6809ms)
    √ Null / Not ready (131ms)
    √ Executed (6320ms)
```

```
    submitExternalVote
      rejection
        √ rejects if proposal does not exist
        √ rejects if proposal not ready (97ms)
      vote for
        √ submits to governor (6504ms)
      vote against
        √ submits to governor (6418ms)

  Ndx
    √ permit() (189ms)
    √ nested delegation (301ms)
    Constructor & Settings
      √ totalSupply()
      √ Gave supply to address in constructor
      √ Set the minter as the governor contract
      √ Sets the mint timestamp
    setMinter()
      √ reverts if not called by owner
      √ sets the minter (106ms)
    mint()
      √ reverts if not called by owner
      √ reverts if called before mintingAllowedAfter
      √ reverts if target is null address
      √ reverts if amount exceeds 96 bits
      √ reverts if amount exceeds 10% supply
      √ mints tokens to target (87ms)


  171 passing (2m)

-----------------------------------|----------|----------|----------|----------|----------------
File                               |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines
-----------------------------------|----------|----------|----------|----------|----------------
 distribution/                     |      100 |      100 |      100 |      100 |
  DelegatingVester.sol             |      100 |      100 |      100 |      100 |
  Reservoir.sol                    |      100 |      100 |      100 |      100 |
  RewardsDistributionRecipient.sol |      100 |      100 |      100 |      100 |
  StakingRewards.sol               |      100 |      100 |      100 |      100 |
  StakingRewardsFactory.sol        |      100 |      100 |      100 |      100 |
  TreasuryLock.sol                 |      100 |      100 |      100 |      100 |
  TreasuryVester.sol               |      100 |      100 |      100 |      100 |
 meta/                             |      100 |      100 |      100 |      100 |
  MetaGovernorCOMP.sol             |      100 |      100 |      100 |      100 |
  MetaGovernorUNI.sol              |      100 |      100 |      100 |      100 |
-----------------------------------|----------|----------|----------|----------|----------------
All files                          |      100 |      100 |      100 |      100 |
-----------------------------------|----------|----------|----------|----------|----------------

> Istanbul reports written to ./coverage/ and ./coverage.json
> solidity-coverage cleaning up, shutting down ganache server
Done in 136.51s.
```

Tests

```
$ yarn test
yarn run v1.22.10
$ buidler test
Compiling...


contracts/interfaces/IStakingRewards.sol: Warning: SPDX license identifier not provided in sourc



contracts/interfaces/IStakingRewardsFactory.sol: Warning: SPDX license identifier not provided i



contracts/interfaces/ITimelock.sol: Warning: SPDX license identifier not provided in source file



contracts/mocks/MockPoolFactory.sol: Warning: SPDX license identifier not provided in source fil



contracts/mocks/MockTimelock.sol: Warning: SPDX license identifier not provided in source file.



contracts/mocks/ReservoirErrorTrigger.sol: Warning: SPDX license identifier not provided in sour



contracts/interfaces/IStakingRewards.sol:39:23: Warning: This declaration shadows an existing de
  function initialize(address stakingToken, uint256 rewardsDuration) external;
                      ^------------------^
contracts/interfaces/IStakingRewards.sol:34:3: The shadowed declaration is here:
  function stakingToken() external view returns (IERC20);
  ^-------------------------------------------------------^



contracts/interfaces/IStakingRewards.sol:39:45: Warning: This declaration shadows an existing de
  function initialize(address stakingToken, uint256 rewardsDuration) external;
                                            ^----------------------^
contracts/interfaces/IStakingRewards.sol:13:3: The shadowed declaration is here:
  function rewardsDuration() external view returns (uint256);
  ^-----------------------------------------------------------^
```

```
contracts/interfaces/IStakingRewards.sol:54:31: Warning: This declaration shadows an existing de
  function setRewardsDuration(uint256 rewardsDuration) external;
                              ^---------------------^
contracts/interfaces/IStakingRewards.sol:13:3: The shadowed declaration is here:
  function rewardsDuration() external view returns (uint256);
  ^-----------------------------------------------------------^




contracts/governance/Timelock.sol:9:1: Warning: This contract has a payable fallback function, b
contract Timelock is ITimelock {
^ (Relevant source part starts here and spans across multiple lines).
contracts/governance/Timelock.sol:64:3: The payable fallback function is defined here.
  fallback() external payable {}
  ^----------------------------^




contracts/mocks/MockTimelock.sol:11:1: Warning: This contract has a payable fallback function, b
contract MockTimelock is ITimelock {
^ (Relevant source part starts here and spans across multiple lines).
contracts/mocks/MockTimelock.sol:66:3: The payable fallback function is defined here.
  fallback() external payable {}
  ^----------------------------^




contracts/governance/GovernorAlpha.sol:331:7: Warning: Using ".value(...)" is deprecated. Use "{
      timelock.executeTransaction.value(proposal.values[i])(
      ^-------------------------------^




contracts/mocks/ReservoirErrorTrigger.sol:20:3: Warning: Function state mutability can be restri
  function triggerAdditionOverflow() public {
  ^ (Relevant source part starts here and spans across multiple lines).

Compiled 43 contracts successfully



  distribution:DelegatingVester
    √ Constructor fails with invalid vesting times
    √ claim:~half
    √ claim:all
    √ delegate:fail
    √ delegate
    √ setRecipient:fail
    √ setRecipient (38ms)

  distribution:Reservoir
    √ drip()
    √ drip() 0 drip rate
```

```
    √ drip() many blocks
    √ drip() more blocks than duration
    errors
      √ dripTotal overflow
      √ deltaDrip underflow
      √ addition overflow

  distribution:StakingRewards
    Initialization
      √ does not allow null staking token
      √ does not allow zero duration
      √ sets the staking token
      √ can not be initialized twice
    Constructor & Initializer
      √ should set rewards token on constructor
      √ should set rewardsDistribution on constructor
      √ should set staking token on initialize
      √ should set rewardsDuration on initialize
    Function permissions
      √ only owner can call notifyRewardAmount
    lastTimeRewardApplicable()
      √ should return 0
      when updated
        √ should equal current timestamp
    rewardPerToken()
      √ should return 0
      √ should be > 0 (53ms)
    stake()
      √ staking increases staking balance (41ms)
      √ cannot stake 0
    earned()
      √ should be 0 when not staking
      √ should be > 0 when staking (51ms)
      √ rewardRate should increase if new rewards come before DURATION ends
      √ rewards token balance should rollover after DURATION (71ms)
    getReward()
      √ should increase rewards token balance (76ms)
      √ gives nothing for user with no owed rewards
    getRewardForDuration()
      √ should increase rewards token balance
    withdraw()
      √ cannot withdraw if nothing staked
      √ should increases lp token balance and decreases staking balance (61ms)
      √ cannot withdraw 0
    exit()
      √ should retrieve all earned and increase rewards bal (79ms)
    notifyRewardAmount()
      √ Reverts if the provided reward is greater than the balance.
      √ Reverts if the provided reward is greater than the balance, plus rolled-over balance. (4
    recoverERC20
      √ Reverts if not called by owner
      √ Reverts if token is rewards or staking token
```

```
      √ Recovers tokens
    setRewardsDuration()
      √ should revert if not called by owner
      √ should revert if duration is 0
      √ should increase rewards duration before starting distribution
      √ should revert when setting setRewardsDuration before the period has finished (48ms)
      √ should update when setting setRewardsDuration after the period has finished (57ms)
      √ should update when setting setRewardsDuration after the period has finished (89ms)

  distribution:StakingRewardsFactory
    Constructor & Settings
      √ Rejects genesis earlier than block timestamp
      √ STAKING_REWARDS_IMPLEMENTATION_ID
      √ poolFactory
      √ proxyManager
      √ rewardsToken
      √ uniswapFactory
      √ weth
    notifyRewardAmounts()
      √ Reverts if there are no staking pools
      √ Notifies all the pools of their rewards (70ms)
    getStakingRewards()
      √ Reverts if the staking token provided does not have a rewards pool
    deployStakingRewardsForPool()
      √ Only allows owner to call deployStakingRewardsForPool
      √ Reverts if the staking token is not an index lp token
Returned address 0x41EbF7cf9fFa6f23A960d0b0aD91Ec901664E2d3
Event address 0x41EbF7cf9fFa6f23A960d0b0aD91Ec901664E2d3
      √ Allows the owner to deploy a staking pool for an index lp token
      √ Fails duplicate deployment without calling proxy manager
      Staking Info
        √ computeStakingRewardsAddress()
        √ getStakingRewards()
        √ stakingTokens()
        √ stakingRewardsInfoByStakingToken()
      StakingRewards
        √ rewardsToken()
        √ stakingToken()
      notifyRewardAmount()
        √ Fails if the staking genesis timestamp has not passed
        √ Fails if the factory does not have sufficient tokens
        √ Fails if the staking pool does not exist
        √ Notifies the pool of its rewards if there are pending rewards
        √ Does nothing if there are no pending rewards
      setRewardsDuration()
        √ Reverts if not called by owner
        √ Reverts if stakingToken has no pool
        √ Updates the duration
    recoverERC20()
      √ Reverts if token is rewards or staking token
      √ Recovers tokens
    deployStakingRewardsForPoolUniswapPair
```

```
    √ Only allows owner to call deployStakingRewardsForPoolUniswapPair
    √ Reverts if the staking token is not an index lp token
    √ Reverts if index token is null address
    √ Reverts if token is weth
    √ Allows the owner to deploy a staking pool for an index lp token <-> weth uniswap pair
    √ Fails duplicate deployment without calling proxy manager
    Staking Info
      √ computeStakingRewardsAddress()
      √ getStakingRewards()
      √ stakingTokens()
      √ stakingRewardsInfoByStakingToken()
    StakingRewards
      √ rewardsToken()
      √ stakingToken()
    notifyRewardAmount()
      √ Fails if the factory does not have sufficient tokens
      √ Fails if the staking pool does not exist
      √ Notifies the pool of its rewards if there are pending rewards
      √ Does nothing if there are no pending rewards
  increaseStakingRewards
    √ Can only be called by owner
    √ Reverts if amount is zero
    √ Reverts if pool does not exist
    √ Reverts if pool has pending rewards
    √ Reverts if pool is still active
    √ Reverts if factory has insufficient balance
    √ Succeeds when the pool is finished (40ms)

distribution:TreasuryLock
  Constructor
    √ Reverts if recipient is null
    √ Reverts if token is null
    √ Reverts if unlockDate is too soon
    √ Sets the correct values
  claim()
    √ Reverts if unlock date has not passed
    √ Transfers balance

distribution:TreasuryVester
  √ Constructor fails with invalid vesting times
  √ claim:fail
  √ claim:~half
  √ claim:all
  √ setRecipient:fail
  √ setRecipient

GovernorAlpha
  √ ndx
  √ timelock
  √ governor
  voting period
    √ votingPeriod initialized to 2880
```

```
      √ permanentVotingPeriod set to 17280
      √ setPermanentVotingPeriod: reverts if too early
      √ setPermanentVotingPeriod: adjusts voting period when allowed

  MetaGovernorCOMP
    castVote
      √ rejects if caller has no votes
      √ casts vote
      √ stores the correct start and end block
      √ records caller vote
      √ does not set voteSubmitted
      √ creates receipt
      √ rejects duplicate vote
      √ rejects if proposal not active (2069ms)
    state
      √ Active
      √ Defeated (931ms)
      √ Succeeded (929ms)
      √ Null / Not ready
      √ Executed (882ms)
    submitExternalVote
      rejection
        √ rejects if proposal does not exist
        √ rejects if proposal not ready
      vote for
        √ submits to governor (883ms)
      vote against
        √ submits to governor (948ms)

  MetaGovernorUNI
    castVote
      √ rejects if caller has no votes
      √ casts vote
      √ stores the correct start and end block
      √ records caller vote
      √ does not set voteSubmitted
      √ creates receipt
      √ rejects duplicate vote
      √ rejects if proposal not active (1866ms)
    state
      √ Active
      √ Defeated (969ms)
      √ Succeeded (984ms)
      √ Null / Not ready
      √ Executed (1042ms)
    submitExternalVote
      rejection
        √ rejects if proposal does not exist
        √ rejects if proposal not ready
      vote for
        √ submits to governor (946ms)
      vote against
```

```
        √ submits to governor (943ms)


  Ndx
    √ permit() (41ms)
    √ nested delegation (65ms)
    Constructor & Settings
      √ totalSupply()
      √ Gave supply to address in constructor
      √ Set the minter as the governor contract
      √ Sets the mint timestamp
    setMinter()
      √ reverts if not called by owner
      √ sets the minter
    mint()
      √ reverts if not called by owner
      √ reverts if called before mintingAllowedAfter
      √ reverts if target is null address
      √ reverts if amount exceeds 96 bits
      √ reverts if amount exceeds 10% supply
      √ mints tokens to target



  171 passing (25s)

Done in 32.36s.
```

License

This report falls under the terms described in the included LICENSE.