Cyberscope

# Audit Report

# OpenRnD

February 2024

| | |
|---|---|
| Repository | https://github.com/Plopmenz/openrd-foundry |
| Commit | 486d10ae0d58ca18e416c831fd4a0ba510ce026a |
| Audited by | © cyberscope |

# Table of Contents

**Table of Contents** **1**
**Review** **3**
Audit Updates 3
Source Files 4
**Overview** **7**
Tasks Contract 7
RFP Contract 7
ERC721TagManager Contract 8
TrustlessManagement Contract 8
OptimisticActions Contract 8
TaskDisputes Contract 8
TaskDrafts Contract 8
TagVoting Contract 9
**Findings Breakdown** **10**
**Diagnostics** **11**
ACV - Access Control Vulnerability 13
Description 13
Recommendation 14
CCR - Contract Centralization Risk 15
Description 15
Recommendation 16
GO - Gas Optimization 17
Description 17
Recommendation 17
IIT - Inefficient Indexing Type 18
Description 18
Recommendation 19
MPS - Metadata Proper Storage 20
Description 20
Recommendation 20
MEE - Missing Events Emission 21
Description 21
Recommendation 21
MSC - Missing Sanity Check 22
Description 22
Recommendation 22
MU - Modifiers Usage 23
Description 23
Recommendation 23

ORM - One-Sided Refund Mechanism 24
Description 24
Recommendation 24
POE - Possible Out-Of-Bounds Error 25
Description 25
Recommendation 26
RSW - Redundant Storage Writes 27
Description 27
Recommendation 27
RC - Repetitive Calculations 28
Description 28
Recommendation 28
L04 - Conformance to Solidity Naming Conventions 29
Description 29
Recommendation 30
L09 - Dead Code Elimination 31
Description 31
Recommendation 32
L14 - Uninitialized Variables in Local Scope 33
Description 33
Recommendation 33
L16 - Validate Variable Setters 34
Description 34
Recommendation 34
L17 - Usage of Solidity Assembly 35
Description 35
Recommendation 35
L19 - Stable Compiler Version 36
Description 36
Recommendation 36
L22 - Potential Locked Ether 37
Description 37
Recommendation 37
**Functions Analysis 38**
**Inheritance Graph 51**
**Flow Graph 52**
**Summary 53**
**Disclaimer 54**
**About Cyberscope 55**

# Review

| | |
|---|---|
| **Repository** | https://github.com/Plopmenz/openrd-foundry |
| **Commit** | 486d10ae0d58ca18e416c831fd4a0ba510ce026a |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 08 Aug 2023 |
| **Corrected Phase 2** | 13 Feb 2024 |

## Source Files

| Filename | SHA256 |
| --- | --- |
| **TasksUtils.sol** | bdb84c268f169ac929dfea6e49457da846a3d742633422d041f6ef63ca8a2d97 |
| **TasksEnsure.sol** | 0c1a3b04e495981743414f135078faf69bc7795fc5c2f7cfae04ff10af148ef1 |
| **Tasks.sol** | 54bac98364b5affc41b6882b4450c5e57bb7245eb520e95f3d41c17e9d7fe1f2 |
| **ITasks.sol** | 984a23eb6f95d80831c522e95c646f22d3f4ce3b1a9db50a3f3eb2e2b46a1107 |
| **Escrow.sol** | 2f3807f1e1c23722755a5c43a33586f5f75be823314aa252f7dd58dcb1a057e9 |
| **lib/trustless-management/TrustlessManagement.sol** | 0d8e41a7cd08e83e0cef0a42b7e4bbca9beff10c51c308af1d35dd811f61d4da |
| **lib/trustless-management/TagTrustlessManagement.sol** | 20f164d20ad44a2d53b82e592594a0e3a0f8e912894f6fb1735bdcff6429bba2 |
| **lib/trustless-management/ITrustlessManagement.sol** | 2edea5d79bf40bd4cd82652ed755b49a15b08d9da047665a76a55023588cc86d |
| **lib/trustless-management/IPermissionChecker.sol** | 42810a9a1cab26d94a30f62b651f3facdff81434a03b49200fb8a7c01954cbe0 |
| **lib/trustless-management/IDAOManager.sol** | 4b98cb34ba39c18783fae50b4f9b8fa595b69ecf483b483d2fc811dea54a884c |
| **lib/trustless-management/IDAOExtensionWithAdmin.sol** | c6bce8263796170bc08e27a20b637dd2f73390a98fc159e7ca9c392e0e000c73 |
| **lib/trustless-management/ERC721TrustlessManagement.sol** | 690a6a1f624d96665331b1d3737049c1560dd2e42e0227a621c49e68316cace9 |

| | |
|---|---|
| **lib/trustless-management/ERC721CountTrustlessManagement.sol** | 01d663ea8d9c74ba733e59b2c8a347631d95b32c1fb2b0cb75342627fb6dfe9a |
| **lib/trustless-management/ERC20TrustlessManagement.sol** | f761827385a3b8f6698c4cbefe28d9a2565bf6c9dd7da2d128700889c4383de6 |
| **lib/trustless-management/ERC1155TrustlessManagement.sol** | 75a713693934e34f164a797a693aaa217350fda42827020330ad48f71ae45544 |
| **lib/trustless-management/ERC1155CountTrustlessManagement.sol** | d09329d8eb3fb190b2459b5f3179ef7c3a6774471898cacf70f9ebfeb12ca5d4 |
| **lib/trustless-management/AddressTrustlessManagement.sol** | a4156f72cc13619ca45069344684c701c8bf9c4e301beed5cb98d85da2f8681c |
| **lib/tag-manager/ITagManagerExtended.sol** | 86ffb017c65fa47de427474e6481298f28f2acd32faa40a74ddce873535290e9 |
| **lib/tag-manager/ITagManager.sol** | f27d20328bb92177416682ad9d1247c9d1aa380a0ac41b09016b1e348cf22047 |
| **lib/tag-manager/ERC721TagManager.sol** | 28dadb35193e17e90e507dfd201f53e7d9cb5b052893f42b5c204737f0d03ab4 |
| **lib/smart-account/SmartAccount.sol** | 9b981c2fa5cd3575a6531f9cab16c13457081d0aa4921209d1860df1377a41be |
| **lib/smart-account/ISmartAccount.sol** | c1c8d0726f94f403c22c9f077fa51c6cbb873959cfa236c1206bca528363891e |
| **lib/optimistic-actions/OptimisticActions.sol** | d6bf44b14a882272091f9aae813d98241de1b7ed16d0874f84ebf167fc572108 |
| **lib/optimistic-actions/IOptimisticActions.sol** | 09f078584d20b3867868a148c3a32ef1ab8ef9a4e762fed28cb541c88161b5be |
| **lib/openrfp/RFPs.sol** | 9dca3b1fab51d4abbddc9312cc416320a9e1bfc379db92d609302063ee0b951b |
| **lib/openrfp/RFPEscrow.sol** | 83867df4b5c076ca051e20cb479152d39c0084c65fed51a4e5b8af6c727a4540 |

| | |
|---|---|
| **lib/openrfp/IRFPs.sol** | 2e301e7ee9306a1a20d414658f9a7a08d629c8dc5af55f2237e11bfc13a62fdf |
| **lib/openrd-dao-extensions/TaskDrafts/TaskDrafts.sol** | e75652d1cbe0bb3b865426a4d4a72351490ac4530edf0ee7a9afcce3a9f36c1e |
| **lib/openrd-dao-extensions/TaskDrafts/ITaskDrafts.sol** | 749666373e4d5fc9842b9438bb9c247ae5a34c5c2afc6143ef1bf789dfae0ab8 |
| **lib/openrd-dao-extensions/TaskDisputes/TaskDisputes.sol** | 36e4662b2eec07009631a16061e6a22743b33df17711b14de6cc0ecf972c588b |
| **lib/openrd-dao-extensions/TaskDisputes/ITaskDisputes.sol** | 193fadde686a867e2394637ff5c33c80756b5f32c7eb5c8ae01f01e64f98bfc0 |
| **lib/openmesh-admin/OpenmeshENSReverseClaimable.sol** | e2a79b4e5e1a2d27b784d20f81ee2fb9b59cf94adf7cdb3b753fc4dbdcead59d |
| **lib/openmesh-admin/OpenmeshAdmin.sol** | c627498ab8c98ec1d5c74b50bfd33bfb120ad93671f7b849d59fc4e5afba5ba7 |
| **lib/openmesh-admin/Openmesh.sol** | e0a89f2d3ae5bd71d58cafecb0c202a565332f8316ff89be8384db05e296c9a6 |
| **lib/ens-reverse-claimable/ENSReverseClaimable.sol** | 6b50558158ba49509231bbfd0be071f5a06334f890e9f26e1c69bd86a8f8dc59 |
| **lib/aragon-tag-voting/TagVotingSetup.sol** | 1f66bfc113177ee6d47e369509b619e05b8266634f1645f894df9fb5c660d5f4 |
| **lib/aragon-tag-voting/TagVoting.sol** | 687c689fd2dd3d78e1ab0ea7895498382a4f3d63674ede2f107519e5a060764b |

# Overview

L3atom has introduced a web3 bounty protocol that exhibits functional resemblances to platforms such as Upwork. However, what sets it apart is its distinctive attribute of executing all tasks within the blockchain. This pioneering methodology not only ensures a secure and transparent space for user collaboration but also empowers users to construct a credible portfolio by means of verifiable interactions on the blockchain.

Cyberscope audited several contracts of the ecosystem. Each contract serves a specific function within the ecosystem, from task management and funding to dispute resolution and DAO governance, ensuring a comprehensive and flexible framework for project development and community engagement.

## Tasks Contract

The Tasks contract introduces functionality for native currency rewards, partial payments, dispute resolution, and transfer of task management. It enhances existing functions with a native budget and reward system, allows partial reward releases, and supports dispute-initiated task completion for fair reward distribution. Task managers can also transfer their responsibilities, facilitating flexible task administration. The Escrow contract, used in conjunction with the Tasks contract, supports the creation and funding of tasks. It enables the allocation of budgets for multiple tasks, accepts fund top-ups at any time, and allows for the retrieval of unused funds. The contract ensures that tasks are only created if sufficient funds are available, streamlining the project funding process.

## RFP Contract

The RFPs contract allows for the submission and funding of project proposals through a single budget. Projects funded by this RFP automatically create corresponding tasks with preapproved applicants. This setup facilitates streamlined project initiation and funding, with the Escrow contract ensuring efficient budget management. The RFPEscrow contract is an extension of the Escrow contract that enables the creation of OpenR&D tasks directly from RFPs, optimizing the task creation process and ensuring direct fund transfer to the tasks, thereby minimizing transaction complexity and fund mismanagement risks.

## ERC721TagManager Contract

The ERC721TagManager contract determines whether an address possesses a specific tag, supporting various implementation possibilities. Initially, it utilizes ERC721 NFTs to manage tags, allowing for diverse application scenarios.

## TrustlessManagement Contract

TrustlessManagement acts as a standalone contract replacing traditional Aragon plugins. It allows for advanced on-chain permission settings, including whitelists, blacklists, and custom checks for action execution within a DAO. This contract enables more granular and secure management of permissions and actions.

## OptimisticActions Contract

The OptimisticActions contract facilitates the creation and execution of delayed actions, which can be vetoed by the creator. It integrates with TrustlessManagement to ensure actions are executed securely, supporting use cases like optimistic DAO member payments.

## TaskDisputes Contract

TaskDisputes enables the creation of dispute resolution proposals within a DAO, with a configurable native currency fee to cover the costs associated with dispute resolution. This feature ensures fair and efficient handling of task-related disputes.

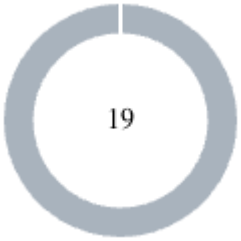
## TaskDrafts Contract

TaskDrafts allows for the creation of proposals to fund or initiate tasks, promoting community-driven project development. Proposals can designate preapproved applicants, encouraging proactive participation in task creation and execution.

## TagVoting Contract

AragonTagVoting and AragonTagVotingSetup replace the TokenListGovernance plugin from V1, enabling DAO creation based on tags managed by the TagManager. This system facilitates department-specific DAO governance, enhancing organizational flexibility and decision-making.

# Findings Breakdown




| | |
|---|---|
| Critical | 0 |
| Medium | 0 |
| Minor / Informative | 19 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 19 | 0 | 0 | 0 |

# Diagnostics

Critical Medium Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| | ACV | Access Control Vulnerability | Unresolved |
| | CCR | Contract Centralization Risk | Unresolved |
| | GO | Gas Optimization | Unresolved |
| | IIT | Inefficient Indexing Type | Unresolved |
| | MPS | Metadata Proper Storage | Unresolved |
| | MEE | Missing Events Emission | Unresolved |
| | MSC | Missing Sanity Check | Unresolved |
| | MU | Modifiers Usage | Unresolved |
| | ORM | One-Sided Refund Mechanism | Unresolved |
| | POE | Possible Out-Of-Bounds Error | Unresolved |
| | RSW | Redundant Storage Writes | Unresolved |
| | RC | Repetitive Calculations | Unresolved |
| | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| | L09 | Dead Code Elimination | Unresolved |

| | | | |
|---|---|---|---|
| ● | L14 | Uninitialized Variables in Local Scope | Unresolved |
| ● | L16 | Validate Variable Setters | Unresolved |
| ● | L17 | Usage of Solidity Assembly | Unresolved |
| ● | L19 | Stable Compiler Version | Unresolved |
| ● | L22 | Potential Locked Ether | Unresolved |

# ACV - Access Control Vulnerability

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | lib/optimistic-actions/OptimisticActions.sol#L81<br>lib/openrd-dao-extensions/TaskDrafts/TaskDrafts.sol#L33<br>lib/openrd-dao-extensions/TaskDisputes/TaskDisputes.sol#L43 |
| **Status** | Unresolved |

## Description

The contracts TaskDisputes, TaskDrafts, and OptimisticActions, designed for DAO interactions, contain vulnerabilities in their admin functions ( `updateManager` and `setAdmin` ). These vulnerabilities stem from the lack of adequate access control mechanisms. A malicious entity can exploit these vulnerabilities by deploying a contract that mimics the expected DAO interface, subsequently gaining unauthorized admin access. By calling `updateManager` or `setAdmin` , the attacker could manipulate the DAO's management structure or admin settings. This unauthorized access would enable the attacker to execute the `asDAO` function with manipulated arguments, potentially leading to adverse actions within the DAO, such as unauthorized actions being taken under the guise of legitimate DAO operations.

```
function updateManager(IDAOManager _manager, uint256 _role) external {
    DaoInfo storage info = daoInfo[IDAO(msg.sender)];
    info.manager = _manager;
    info.role = _role;
}
function setAdmin(IDAO _dao, address _admin) external {
    DAOInfo storage info = daoInfo[_dao];
    _ensureSenderIsAdmin(_dao, info.admin);
    info.admin = _admin;
    emit AdminSet(_dao, _admin);
}
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the contracts can significantly reduce the risk of unauthorized access and manipulation, thereby safeguarding the integrity of DAO operations and assets under their control.

# CCR - Contract Centralization Risk

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | lib/trustless-management/TrustlessManagement.sol#L81,89,97,105,119<br>Tasks.sol#L428 |
| **Status** | Unresolved |

## Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

For instance, the `disable` function within the contract allows a specific address, referred to as the `disabler`, to permanently cease all operations of the contract by setting the `disabler` address to the zero address. This design creates a single point of failure, as the entire contract's functionality hinges on the security of the private keys of the `disabler` address. If these private keys are compromised or stolen, an unauthorized party could irreversibly disable the contract, potentially leading to significant disruptions and loss of funds or assets.

```
function disable() external {
    _ensureSenderIsDisabler();
    disabler = address(0);
}
function changeFullAccess(IDAO _dao, uint256 _role, address
_permissionChecker) external {
    DAOInfo storage info = daoInfo[_dao];
    _ensureSenderIsAdmin(_dao, info.admin);
    info.permissions[_role].fullAccess = _permissionChecker;
    emit FullAccessChanged(_dao, _role, _permissionChecker);
}
function changeZoneAccess(IDAO _dao, uint256 _role, address _zone, address
_permissionChecker) external {
    DAOInfo storage info = daoInfo[_dao];
    _ensureSenderIsAdmin(_dao, info.admin);
    info.permissions[_role].zoneAccess[_zone] = _permissionChecker;
    emit ZoneAccessChanged(_dao, _role, _zone, _permissionChecker);
}
...
```

## Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

## GO - Gas Optimization

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | lib/openrfp/RFPs.sol#L205,208 |
| **Status** | Unresolved |

### Description

Gas optimization refers to the process of reducing the amount of gas required to execute a transaction. Gas is the unit of measurement used to calculate the fees paid to miners for including a transaction in a block on the blockchain.

The `j` variable could be post incremented. That way, there will not be a necessity to subtract it by 1 when indexing the `taskReward`.

```
unchecked {
    ++j;
}

if (taskReward[j - 1].nextToken) {
    break;
}
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

# IIT - Inefficient Indexing Type

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L183<br>TasksUtils.sol#L34 |
| **Status** | Unresolved |

## Description

In the `acceptApplications` function, designed to accept multiple applications for a specified task, there is an inconsistency in the data types used for indexing and the application IDs array. The function iterates over an array of application IDs ( `_applicationIds` ) defined as uint32 array, using a loop index ( `i` ) declared as uint256. This discrepancy in data types can result in slightly higher gas costs due to the use of a larger integer type (uint256) than necessary (uint32), especially considering the loop's operation and comparison mechanics.

Similarly, the `_toOffchainTask` function iterates over an array with uint32 type length. However, the indexing value is declared as uint8.

```
for (uint256 i; i < _applicationIds.length;) {
    ...

    unchecked {
        ++i;
    }
}
...
offchainTask.applications = new
OffChainApplication[](task.applicationCount);
for (uint8 i; i < offchainTask.applications.length;) {
    Application storage application = task.applications[i];
```

## Recommendation

To optimize gas consumption and align the data types used within the `acceptApplications` function, the team is recommended to declare the loop index `i` with the same type as the elements of `_applicationIds`, which is uint32. This change reduces the computational overhead associated with type conversion and leverages the more size-appropriate uint32 type for indexing through the application IDs array. Implementing this modification ensures that the data types used for indexing are consistent with the array's element type, leading to more efficient bytecode and potentially reducing gas costs associated with the loop execution.

## MPS - Metadata Proper Storage

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Status** | Unresolved |

### Description

The contract defines structures and function arguments that include IPFS hashes stored as string variables for metadata. Utilizing string types for IPFS hash storage is inherently less gas-efficient due to the dynamic nature of strings, which can incur higher storage and execution costs in smart contracts. Storing these variables or properties as `bytes32` instead of `strings` can significantly reduce the gas costs associated with deploying and interacting with these contracts, especially during frequent read and write operations.

### Recommendation

To optimize gas consumption and improve overall contract efficiency, the team is advised to refactor the contract to store IPFS hashes as `bytes32` instead of `string` variables. This change leverages the fixed size of bytes32 to minimize storage requirements and gas costs. Since IPFS hashes can be encoded into a `bytes32` format, this approach maintains the integrity and accessibility of the hash data while enhancing contract performance.

## MEE - Missing Events Emission

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L430<br>lib/tag-manager/ERC721TagManager.sol#L92,102 |
| **Status** | Unresolved |

### Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
disabler = address(0);
tagData.hasTag[tokenId] = true;
tagData.hasTag[tokenId] = false;
```

### Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

## MSC - Missing Sanity Check

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L71,99,101<br>lib/openrfp/RFPs.sol#L83,84,85<br>lib/optimistic-actions/OptimisticActions.sol#L32 |
| **Status** | Unresolved |

### Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

For isntance, the absence of a check to ensure that `_manager` and `_disputeManager` is not the zero address could lead to a scenario where a task is assigned a manager and/or dispute manager of the zero address.

```
task.deadline = _deadline;
task.manager = _manager;
if (_disputeManager != address(0)) {
    task.disputeManager = _disputeManager;
}

rfp.manager = _manager;
rfp.tasksManager = _tasksManager;
rfp.disputeManager = _disputeManager;
```

### Recommendation

The team is advised to properly check the variables according to the required specifications.

## MU - Modifiers Usage

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | lib/optimistic-actions/OptimisticActions.sol#L42,54<br>lib/openrfp/RFPs.sol#L165,243 |
| **Status** | Unresolved |

### Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
if (_id >= info.requestCount) {
    revert RequestDoesNotExist();
}
if (msg.sender != rfp.manager) {
    revert NotManager();
}
```

### Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

## ORM - One-Sided Refund Mechanism

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L435 |
| **Status** | Unresolved |

### Description

The contract provides a feature to permanently disable its functions through the `disable` method, setting the `disabler` to the zero address and effectively ceasing further operations. Following this action, the `refund` function enables task creators to withdraw funds for tasks that have not been completed. However, this mechanism exclusively covers the refund to the task creators, omitting provisions for compensating task executors for their potential time and effort invested prior to the disablement. This oversight could lead to a scenario where executors are left uncompensated, despite partial or full completion of their tasks, if the contract is disabled before task closure.

```
function refund(uint256 _taskId) external {
    _ensureDisabled();
    Task storage task = _getTask(_taskId);
    _ensureTaskNotClosed(task);
    _refundCreator(task);
}
```

### Recommendation

The team is advised to implement a fair and equitable refund mechanism that considers both task creators and executors. The team could modify the business logic of the `refund` function and ensure both parties (task creators and executors) are eligible for compensation. This might involve distributing the locked funds proportionally based on the task's progress or ensuring a minimum guaranteed payment for the executor. By doing so, the contract mitigates risks for task executors, ensuring they are not disadvantaged in the event of a contract disablement.

# POE - Possible Out-Of-Bounds Error

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | lib/openrfp/RFPs.sol#L184,200 |
| **Status** | Unresolved |

## Description

As part of the `acceptProject` function the contract calculates the budget for every task within a project by iterating over arrays to assign native rewards and ERC20-based rewards. It constructs `taskNativeReward` and `taskReward` arrays based on `nativeRewardCount` and `rewardCount`, respectively. During this process, it assigns rewards using the `_nativeReward` and `_reward` input arrays. However, an issue arises when the lengths of `nativeRewardCount` and `rewardCount` exceed the lengths of the `_nativeReward` and `_reward` input arrays. The contract does not validate their length is less than or equal to their corresponding count values. This discrepancy can lead to an index out-of-bounds error, causing the transaction to revert, as the code attempts to access elements beyond the input arrays' limits.

```
for (uint8 i; i < taskNativeReward.length;) {
    taskNativeReward[i] = ITasks.NativeReward(project.nativeReward[i].to,
_nativeReward[i]);
    taskNativeBudget += _nativeReward[i];
    ...
}
...
uint8 j;
for (uint8 i; i < taskBudget.length;) {
    IERC20 erc20 = rfp.budget[i];
    uint96 projectBudget;
    while (j < taskReward.length) {
        taskReward[j] = project.reward[j];
        taskReward[j].amount = _reward[j];
        projectBudget += _reward[j];
        ...
    }
    ...
}
```

## Recommendation

To prevent this index out-of-bounds error and ensure robust handling of array sizes, the team is advised to implement additional safeguards and code adjustments such as validating the input array lengths. Before processing the arrays, the team could add validation checks to ensure that the lengths of the `_nativeReward` and `_reward` input arrays are at least as large as `project.nativeRewardCount` and `project.rewardCount`, respectively. If the validation fails, the contract could revert the transaction with a clear error message indicating the mismatch in expected array lengths. By incorporating these safeguards, the contract can avoid potential out-of-bounds errors, improving its reliability and user experience by ensuring that tasks are budgeted correctly without causing transaction reverts due to array size mismatches.

## RSW - Redundant Storage Writes

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L368 |
| **Status** | Unresolved |

## Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function editMetadata(uint256 _taskId, string calldata _newMetadata)
external {
    _ensureNotDisabled();
    Task storage task = _getTask(_taskId);
    _ensureSenderIsManager(task);

    _ensureTaskIsOpen(task);

    task.metadata = _newMetadata;

    emit MetadataChanged(_taskId, _newMetadata);
}
```

## Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

## RC - Repetitive Calculations

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L78,109 |
| **Status** | Unresolved |

### Description

The contract contains methods with multiple occurrences of the same calculation being performed. The calculation is repeated without utilizing a variable to store its result, which leads to redundant code, hinders code readability, and increases gas consumption. Each repetition of the calculation requires computational resources and can impact the performance of the contract, especially if the calculation is resource-intensive.

```
task.nativeBudget = _toUint96(msg.value);
emit TaskCreated(
    taskId, _metadata, _deadline, _manager, _disputeManager, msg.sender,
_toUint96(msg.value), _budget, escrow
);
```

### Recommendation

To address this finding and enhance the efficiency and maintainability of the contract, it is recommended to refactor the code by assigning the calculation result to a variable once and then utilizing that variable throughout the method. By storing the calculation result in a variable, the contract eliminates the need for redundant calculations and optimizes code execution.

Refactoring the code to assign the calculation result to a variable has several benefits. It improves code readability by making the purpose and intent of the calculation explicit. It also reduces code redundancy, making the method more concise, easier to maintain, and gas effective. Additionally, by performing the calculation once and reusing the variable, the contract improves performance by avoiding unnecessary computations

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Tasks.sol#L39,45,59,60,61,62,63,64,136,137,138,139,177,205,222,237,238,239,240,263,289,318,341,354,368,381,395,396,397,413,435<br>Escrow.sol#L21 |
| **Status** | Unresolved |

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256 _taskId
uint256[] memory _taskIds
string calldata _metadata
uint64 _deadline
address _manager
address _disputeManager
ERC20Transfer[] calldata _budget
PreapprovedApplication[] calldata _preapprove
NativeReward[] calldata _nativeReward
Reward[] calldata _reward
uint32[] calldata _applicationIds
uint32 _applicationId
uint8 _submissionId
SubmissionJudgement _judgement

...
```

## Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

## L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | TasksUtils.sol#L15,78,132,191,203,220,289,314,378<br>TasksEnsure.sol#L10,16,22,28,34,41,47,55,61,67,73,79,85,91,97,103,109,115,122,129 |
| **Status** | Unresolved |

### Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _toOffchainTask(Task storage task) internal view returns
(OffChainTask memory offchainTask) {
        offchainTask.metadata = task.metadata;
        offchainTask.deadline = task.deadline;
        offchainTask.executorApplication = task.executorApplication;
        offchainTask.creator = task.creator;
        offchainTask.manager = task.manager;
...
        for (uint8 i; i < offchainTask.cancelTaskRequests.length;) {
            offchainTask.cancelTaskRequests[i] =
task.cancelTaskRequests[i];
            unchecked {
                ++i;
            }
        }
    }

...
```

## Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L14 - Uninitialized Variables in Local Scope

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | TasksUtils.sol#L105,163,228,258,346 |
| **Status** | Unresolved |

### Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint8 j
uint96 paidOut
```

### Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

## L16 - Validate Variable Setters

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | Escrow.sol#L49 |
| **Status** | Unresolved |

### Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
to.transfer(amount)
```

### Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

## L17 - Usage of Solidity Assembly

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | TasksUtils.sol#L380 |
| **Status** | Unresolved |

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            // Cleans the upper 96 bits of the `implementation` word, then
packs the first 3 bytes
            // of the `implementation` address with the bytecode before
the address.
            mstore(0x00, or(shr(0xe8, shl(0x60, implementation)),
0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000))
            // Packs the remaining 17 bytes of `implementation` with the
bytecode after the address.
            mstore(0x20, or(shl(0x78, implementation),
0x5af43d82803e903d91602b57fd5bf3))
            instance := create(0, 0x09, 0x37)
        }
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

## L19 - Stable Compiler Version

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | TasksUtils.sol#L2<br>TasksEnsure.sol#L2<br>Tasks.sol#L2<br>lib/openmesh-admin/OpenmeshENSReverseClaimable.sol#L2<br>lib/openmesh-admin/Openmesh.sol#L2<br>lib/ens-reverse-claimable/ENSReverseClaimable.sol#L2<br>ITasks.sol#L2<br>Escrow.sol#L2 |
| **Status** | Unresolved |

## Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

## Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

## L22 - Potential Locked Ether

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ITasks.sol#L257 |
| **Status** | Unresolved |

### Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
function createTask(
        string calldata _metadata,
        uint64 _deadline,
        address _manager,
        address _disputeManager,
        ERC20Transfer[] calldata _budget,
        PreapprovedApplication[] calldata _preapprove
    ) external payable returns (uint256 taskId);
```

### Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **TasksUtils** | Implementation | TasksEnsure | | |
| | _toOffchainTask | Internal | | |
| | _ensureRewardBellowBudget | Internal | | |
| | _setRewardBellowBudget | Internal | ✓ | |
| | _increaseNativeBudget | Internal | ✓ | |
| | _increaseBudget | Internal | ✓ | |
| | _payoutTask | Internal | ✓ | |
| | _refundCreator | Internal | ✓ | |
| | _payoutTaskPartially | Internal | ✓ | |
| | clone | Internal | ✓ | |
| | | | | |
| **TasksEnsure** | Implementation | ITasks | | |
| | _ensureTaskIsOpen | Internal | | |
| | _ensureTaskIsTaken | Internal | | |
| | _ensureTaskNotClosed | Internal | | |
| | _ensureSenderIsManager | Internal | | |
| | _ensureSenderIsDisputeManager | Internal | | |
| | _ensureSenderIsExecutor | Internal | | |

| | | | | |
|---|---|---|---|---|
| | _ensureRewardEndsWithNextToken | Internal | | |
| | _ensureApplicationExists | Internal | | |
| | _ensureSenderIsApplicant | Internal | | |
| | _ensureApplicationIsAccepted | Internal | | |
| | _ensureSubmissionExists | Internal | | |
| | _ensureSubmissionNotJudged | Internal | | |
| | _ensureJudgementNotNone | Internal | | |
| | _ensureCancelTaskRequestExists | Internal | | |
| | _ensureRequestNotAccepted | Internal | | |
| | _ensureRequestAccepted | Internal | | |
| | _ensureRequestNotExecuted | Internal | | |
| | _toUint8 | Internal | | |
| | _toUint32 | Internal | | |
| | _toUint96 | Internal | | |
| | | | | |
| **Tasks** | Implementation | TasksUtils, OpenmeshENSReverseClaimable | | |
| | | Public | ✓ | OpenmeshENSReverseClaimable |
| | taskCount | External | | - |
| | getTask | Public | | - |
| | getTasks | External | | - |
| | createTask | External | Payable | - |
| | applyForTask | External | ✓ | - |

| | | | | |
|---|---|---|---|---|
| | acceptApplications | External | ✓ | - |
| | takeTask | External | ✓ | - |
| | createSubmission | External | ✓ | - |
| | reviewSubmission | External | ✓ | - |
| | cancelTask | External | ✓ | - |
| | acceptRequest | External | ✓ | - |
| | executeRequest | External | ✓ | - |
| | extendDeadline | External | ✓ | - |
| | increaseBudget | External | Payable | - |
| | editMetadata | External | ✓ | - |
| | transferManagement | External | ✓ | - |
| | completeByDispute | External | ✓ | - |
| | partialPayment | External | ✓ | - |
| | disable | External | ✓ | - |
| | refund | External | ✓ | - |
| | _getTask | Internal | | |
| | _ensureNotDisabled | Internal | | |
| | _ensureDisabled | Internal | | |
| | _ensureSenderIsDisabler | Internal | | |
| | | | | |
| **ITasks** | Interface | | | |
| | taskCount | External | | - |
| | getTask | External | | - |

| | | | | |
|---|---|---|---|---|
| | getTasks | External | | - |
| | createTask | External | Payable | - |
| | applyForTask | External | ✓ | - |
| | acceptApplications | External | ✓ | - |
| | takeTask | External | ✓ | - |
| | createSubmission | External | ✓ | - |
| | reviewSubmission | External | ✓ | - |
| | cancelTask | External | ✓ | - |
| | acceptRequest | External | ✓ | - |
| | executeRequest | External | ✓ | - |
| | extendDeadline | External | ✓ | - |
| | increaseBudget | External | Payable | - |
| | editMetadata | External | ✓ | - |
| | transferManagement | External | ✓ | - |
| | completeByDispute | External | ✓ | - |
| | partialPayment | External | ✓ | - |
| | | | | |
| **Escrow** | Implementation | | | |
| | | External | Payable | - |
| | | External | Payable | - |
| | __Escrow_init | Public | Payable | - |
| | transfer | External | ✓ | - |
| | transferNative | External | ✓ | - |

| | | | | |
|---|---|---|---|---|
| **TrustlessManagement** | Implementation | ERC165, ENSReverseClaimable, ITrustlessManagement | | |
| | supportsInterface | Public | | - |
| | hasRole | Public | | - |
| | isAllowed | Public | | - |
| | asDAO | External | ✓ | - |
| | setAdmin | External | ✓ | - |
| | changeFullAccess | External | ✓ | - |
| | changeZoneAccess | External | ✓ | - |
| | changeZoneBlacklist | External | ✓ | - |
| | changeFunctionAccess | External | ✓ | - |
| | changeFunctionBlacklist | External | ✓ | - |
| | _checkPermission | Internal | | |
| | _functionId | Internal | | |
| | _ensureSenderIsAdmin | Internal | | |
| | | | | |
| **TagTrustlessManagement** | Implementation | TrustlessManagement | | |
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **ITrustlessManagement** | Interface | IDAOManager | | |
| | hasRole | External | | - |

| | | | | |
|---|---|---|---|---|
| | isAllowed | External | | - |
| | changeFullAccess | External | ✓ | - |
| | changeZoneAccess | External | ✓ | - |
| | changeFunctionBlacklist | External | ✓ | - |
| | changeFunctionAccess | External | ✓ | - |
| | changeZoneBlacklist | External | ✓ | - |
| | | | | |
| **IPermissionChecker** | Interface | | | |
| | checkPermission | External | | - |
| | | | | |
| **IDAOManager** | Interface | IDAOExtensionWithAdmin | | |
| | asDAO | External | ✓ | - |
| | | | | |
| **IDAOExtensionWithAdmin** | Interface | | | |
| | setAdmin | External | ✓ | - |
| | | | | |
| **ERC721TrustlessManagement** | Implementation | TrustlessManagement | | |
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **ERC721CountTrustlessManagement** | Implementation | TrustlessManagement | | |

| | | | | |
|---|---|---|---|---|
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **ERC20TrustlessManagement** | Implementation | TrustlessManagement | | |
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **ERC1155TrustlessManagement** | Implementation | TrustlessManagement | | |
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **ERC1155CountTrustlessManagement** | Implementation | TrustlessManagement | | |
| | | Public | ✓ | - |
| | hasRole | Public | | - |
| | | | | |
| **AddressTrustlessManagement** | Implementation | TrustlessManagement | | |
| | hasRole | Public | | - |
| | | | | |
| **ITagManagerExtended** | Interface | ITagManager | | |
| | totalTagHavers | External | | - |
| | | | | |
| **ITagManager** | Interface | | | |

| | | | | |
|---|---|---|---|---|
| | hasTag | External | | - |
| | | | | |
| **ERC721TagManager** | Implementation | AccessControl, ITagManagerExtended | | |
| | | Public | ✓ | - |
| | hasTag | External | | - |
| | totalTagHavers | External | | - |
| | addTag | External | ✓ | onlyRole |
| | removeTag | External | ✓ | onlyRole |
| | removeTagFromBurnedToken | External | ✓ | - |
| | setRoleAdmin | Public | ✓ | onlyRole |
| | _addTag | Internal | ✓ | |
| | _removeTag | Internal | ✓ | |
| | | | | |
| **SmartAccount** | Implementation | Ownable, Multicall, ISmartAccount | | |
| | | Public | ✓ | Ownable |
| | performCall | External | ✓ | onlyOwner |
| | performDelegateCall | External | ✓ | onlyOwner |
| | | | | |
| **ISmartAccount** | Interface | | | |
| | performCall | External | ✓ | - |
| | performDelegateCall | External | ✓ | - |

| | | | | |
|---|---|---|---|---|
| **OptimisticActions** | Implementation | ERC165, IOptimisticActions | | |
| | supportsInterface | Public | | - |
| | createAction | External | ✓ | - |
| | rejectAction | External | ✓ | - |
| | executeAction | External | ✓ | - |
| | setExecuteDelay | External | ✓ | - |
| | setAdmin | External | ✓ | - |
| | _ensureSenderIsAdmin | Internal | | |
| | _toUint64 | Internal | | |
| | | | | |
| **IOptimisticActions** | Interface | IDAOExtensionWithAdmin | | |
| | createAction | External | ✓ | - |
| | rejectAction | External | ✓ | - |
| | executeAction | External | ✓ | - |
| | setExecuteDelay | External | ✓ | - |
| | | | | |
| **RFPs** | Implementation | OpenmeshENSReverseClaimable, IRFPs | | |
| | | Public | ✓ | - |
| | | External | Payable | - |
| | rfpCount | External | | - |

| | | | | |
|---|---|---|---|---|
| | getRFP | Public | | - |
| | getRFPs | Public | | - |
| | createRFP | External | Payable | - |
| | submitProject | External | ✓ | - |
| | acceptProject | External | ✓ | - |
| | emptyRFP | External | ✓ | - |
| | _getRFP | Internal | | |
| | _toOffchainRFP | Internal | | |
| | clone | Internal | ✓ | |
| | _toUint8 | Internal | | |
| | | | | |
| **RFPEscrow** | Implementation | Escrow | | |
| | __RFPEscrow_init | Public | Payable | - |
| | createTask | External | ✓ | - |
| | | | | |
| **IRFPs** | Interface | | | |
| | rfpCount | External | | - |
| | getRFP | External | | - |
| | getRFPs | External | | - |
| | createRFP | External | Payable | - |
| | submitProject | External | ✓ | - |
| | acceptProject | External | ✓ | - |
| | emptyRFP | External | ✓ | - |

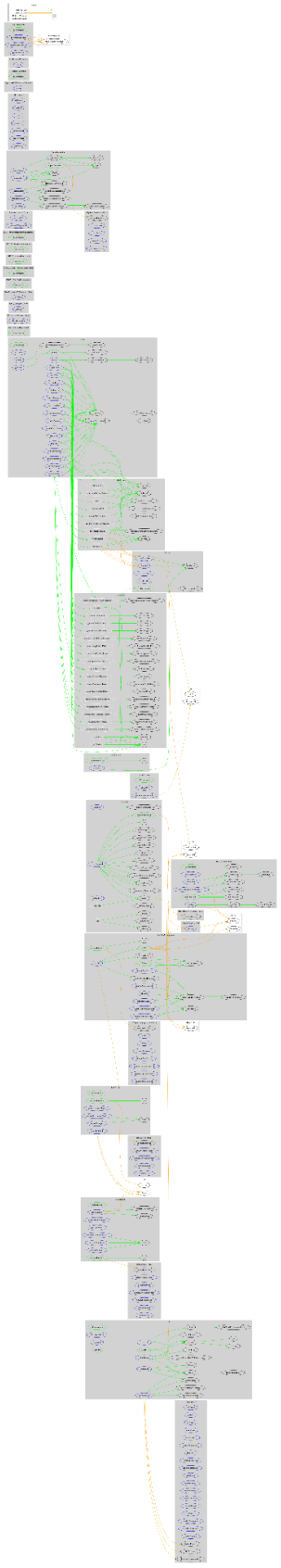
| | | | | |
|---|---|---|---|---|
| **TaskDrafts** | Implementation | ERC165, OpenmeshENSReverseClaimable, ITaskDrafts | | |
| | | Public | ✓ | - |
| | supportsInterface | Public | | - |
| | getGovernancePlugin | External | | - |
| | updateGovernancePlugin | External | ✓ | - |
| | updateManager | External | ✓ | - |
| | createDraftTask | External | ✓ | - |
| | | | | |
| **ITaskDrafts** | Interface | | | |
| | getGovernancePlugin | External | | - |
| | updateGovernancePlugin | External | ✓ | - |
| | updateManager | External | ✓ | - |
| | createDraftTask | External | ✓ | - |
| | | | | |
| **TaskDisputes** | Implementation | ERC165, OpenmeshENSReverseClaimable, ITaskDisputes | | |
| | | Public | ✓ | - |
| | supportsInterface | Public | | - |
| | getGovernancePlugin | External | | - |
| | getDisputeCost | External | | - |

| | | | | |
|---|---|---|---|---|
| | updateGovernancePlugin | External | ✓ | - |
| | updateDisputeCost | External | ✓ | - |
| | updateManager | External | ✓ | - |
| | createDispute | External | Payable | - |
| | | | | |
| **ITaskDisputes** | Interface | | | |
| | getGovernancePlugin | External | | - |
| | getDisputeCost | External | | - |
| | updateGovernancePlugin | External | ✓ | - |
| | updateDisputeCost | External | ✓ | - |
| | updateManager | External | ✓ | - |
| | createDispute | External | Payable | - |
| | | | | |
| **OpenmeshENSReverseClaimable** | Implementation | Openmesh, ENSReverseClaimable | | |
| | owner | External | | - |
| | | | | |
| **OpenmeshAdmin** | Implementation | SmartAccount | | |
| | | Public | ✓ | SmartAccount |
| | | | | |
| **Openmesh** | Implementation | | | |
| | | | | |
| **ENSReverseClaimable** | Implementation | | | |

| | | | | |
|---|---|---|---|---|
| | owner | External | | - |
| | | | | |
| **TagVotingSetup** | Implementation | PluginUpgradeableSetup | | |
| | | Public | ✓ | PluginUpgradeableSetup |
| | prepareInstallation | External | ✓ | - |
| | prepareUninstallation | External | | - |
| | prepareUpdate | External | ✓ | - |
| | | | | |
| **TagVoting** | Implementation | MajorityVotingBase, IMembership | | |
| | initialize | External | ✓ | initializer |
| | totalVotingPower | Public | | - |
| | createProposal | External | ✓ | - |
| | isMember | External | | - |
| | _vote | Internal | ✓ | |
| | _canVote | Internal | | |
| | hasTag | Internal | | |

# Inheritance Graph

# Flow Graph



# Summary

Deeplink contracts implement a token, NFT, utility, escrow, and governance mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

# Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

**The Cyberscope team**

https://www.cyberscope.io