Cyberscope

Audit Report

Openmesh

March 2024

Repository	https://github.com/Plopmenz/openrd-foundry
Commit	04139b36e213e552ba97308a9e1cb0d52614b94f
Audited by	© cyberscope

Table of Contents

Review

Repository	https://github.com/Plopmenz/openrd-foundry
Commit	04139b36e213e552ba97308a9e1cb0d52614b94f

Audit Updates

Initial Audit	08 Aug 2023
Corrected Phase 2	13 Feb 2024
Corrected Phase 3	26 Mar 2024

Source Files

Filename	SHA256
TasksUtils.sol	1e107ce18d06b1d5070102eb361a0e9752b06704df434e5557a6e492cdc0ee2f
TasksEnsure.sol	b7415dc8ea53be2a905a56f55ac6a9329df3d0b6f03e98f7b077674a92e04938
Tasks.sol	8e8f78e3f05fb02b84dc582aad0176fd4a80dd573dcc0aa99fe624533aeedae0
ITasks.sol	48b8ca60e11899e11f3a4467e8879387c6c3747466848e74a4ae352a31ca26f5
Escrow.sol	244e3b60e6583be2fe3a1e38237817760f0eb0534fc84bc9e3b6de3642770214
trustless-management/TrustlessManagement.sol	0d8e41a7cd08e83e0cef0a42b7e4bbca9beff10c51c308af1d35dd811f61d4da
trustless-management/TagTrustlessManagement.sol	20f164d20ad44a2d53b82e592594a0e3a0f8e912894f6fb1735bdcff6429bba2
trustless-management/ITrustlessManagement.sol	2edea5d79bf40bd4cd82652ed755b49a15b08d9da047665a76a55023588cc86d
trustless-management/IPermissionChecker.sol	42810a9a1cab26d94a30f62b651f3facdff81434a03b49200fb8a7c01954cbe0
trustless-management/IDAOManager.sol	4b98cb34ba39c18783fae50b4f9b8fa595b69ecf483b483d2fc811dea54a884c
trustless-management/IDAOExtensionWithAdmin.sol	c6bce8263796170bc08e27a20b637dd2f73390a98fc159e7ca9c392e0e000c73
trustless-management/ERC721TrustlessManagement.sol	690a6a1f624d96665331b1d3737049c1560dd2e42e0227a621c49e68316cace9

trustless-management/ERC721CountTrustlessManagement.sol	01d663ea8d9c74ba733e59b2c8a347631d95b32c1fb2b0cb75342627fb6dfe9a
trustless-management/ERC20TrustlessManagement.sol	f761827385a3b8f6698c4cbefe28d9a2565bf6c9dd7da2d128700889c4383de6
trustless-management/ERC1155TrustlessManagement.sol	75a713693934e34f164a797a693aaa217350fda42827020330ad48f71ae45544
trustless-management/ERC1155CountTrustlessManagement.sol	d09329d8eb3fb190b2459b5f3179ef7c3a6774471898cacf70f9ebfeb12ca5d4
trustless-management/AddressTrustlessManagement.sol	a4156f72cc13619ca45069344684c701c8bf9c4e301beed5cb98d85da2f8681c
tag-manager/ITagManagerExtended.sol	86ffb017c65fa47de427474e6481298f28f2acd32faa40a74ddce873535290e9
tag-manager/ITagManager.sol	f27d20328bb92177416682ad9d1247c9d1aa380a0ac41b09016b1e348cf22047
tag-manager/ERC721TagManager.sol	6b0a18b6895fe1bb9ecd1ec365a540db7ef1568c29aedae6b3198705de226fa4
smart-account/SmartAccount.sol	f1250c7d56b9df2f90531f96c45cb0b8d11e8cb5d1b8134a3ffef17f23b49895
smart-account/ISmartAccount.sol	572669644cb1ed269b18091556b12df645204d760298e855d2252b0ce7b2b69d
optimistic-actions/OptimisticActions.sol	d6bf44b14a882272091f9aae813d98241de1b7ed16d0874f84ebf167fc572108
optimistic-actions/IOptimisticActions.sol	09f078584d20b3867868a148c3a32ef1ab8ef9a4e762fed28cb541c88161b5be
openrfp/RFPs.sol	43549ba242d6a5340e4582e29e41ce371386f3741033879b9aa1e274925745ce
openrfp/RFPEscrow.sol	74729a3d49a2e9d2e1227e169e2b46047880a6d409b53e9b0bf13d0e76e11bbf

openrfp/IRFPs.sol	8ccb313eeccbe7804fe5395e1579c0df360a5d81b57fee70128630a6d0a532ed
openrd-dao-extensions/TaskDrafts/TaskDrafts.sol	a1aecf884379f39f885f9cbec4b20913a5eec0ac31616117acab7987c4c11476
openrd-dao-extensions/TaskDrafts/ITaskDrafts.sol	9f3e1d274512c18ae60a81debd8cc7e32ed44d843ae095280658dd6b9d7b3802
openrd-dao-extensions/TaskDisputes/TaskDisputes.sol	4a68a360800ed96ca407ff9abbbab3ebb2e98f1aced9e3edf5473f1aca83af8a
openrd-dao-extensions/TaskDisputes/ITaskDisputes.sol	e7e9c27e0c67921fe8fed40a9e44308f7690d13f0e5f9d02b6233cf4bdc3a340
openmesh-admin/OpenmeshENSReverseClaimable.sol	e2a79b4e5e1a2d27b784d20f81ee2fb9b59cf94adf7cdb3b753fc4dbdcead59d
openmesh-admin/OpenmeshAdmin.sol	c627498ab8c98ec1d5c74b50bfd33bfb120ad93671f7b849d59fc4e5afba5ba7
openmesh-admin/Openmesh.sol	32c341f9e2dda53e7aac35e53acbfb9b740e4919775319e96c6b92dcaf83093d
ens-reverse-claimable/ENSReverseClaimable.sol	8e9331b8fc18236b8a15bbde05577a06ba102dfb1b9ba5954f9e7fbe0ac624b2
aragon-tag-voting/TagVotingSetup.sol	1cd8ba04d5b557bbb8aa19d5ad97e708ee229a7858e53025bafae1797bbdb3a7
aragon-tag-voting/TagVoting.sol	1a413f657174d2f328ff58d138bc94ae54ef91ce05245cb84ca37b43902604f4

Overview

Openmesh 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.

Contracts

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.

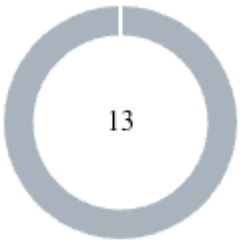
TagVoting Contract

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.

Repositories

Repository	Commit
https://github.com/Plopmenz/openrd-foundry	04139b36e213e552ba97308a9e1cb0d52614b94f
https://github.com/Plopmenz/openrfp	306390e3b7c859dd62f845f6e402899b4c8021a2
https://github.com/Plopmenz/tag-manager	3caf695dd04cebe03f2abe0b535f2521c3317ba3
https://github.com/Plopmenz/trustless-management	d61c3e7c064a47991e036023eb15b99bc4f7704d
https://github.com/Plopmenz/optimistic-actions	8f959d244a212a9af7c5209093768a0066dd1615
https://github.com/Plopmenz/aragon-tag-voting	5a9740d0436efdc4350ef83392909857c2ba2a8b
https://github.com/Plopmenz/openrd-dao-extensions	27e528a8c77fffa246643959b2e7dba64f2d6b80

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	13

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	1	12	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	ACV	Access Control Vulnerability	Acknowledged
	CCR	Contract Centralization Risk	Acknowledged
	MPS	Metadata Proper Storage	Acknowledged
	MEE	Missing Events Emission	Unresolved
	MSC	Missing Sanity Check	Acknowledged
	MU	Modifiers Usage	Acknowledged
	POE	Possible Out-Of-Bounds Error	Acknowledged
	PSS	Potential Staled State	Acknowledged
	RSW	Redundant Storage Writes	Acknowledged
	L04	Conformance to Solidity Naming Conventions	Acknowledged
	L14	Uninitialized Variables in Local Scope	Acknowledged
	L17	Usage of Solidity Assembly	Acknowledged
	L19	Stable Compiler Version	Acknowledged

ACV - Access Control Vulnerability

Criticality	Minor / Informative
Location	optimistic-actions/OptimisticActions.sol#L81 openrd-dao-extensions/TaskDrafts/TaskDrafts.sol#L33 openrd-dao-extensions/TaskDisputes/TaskDisputes.sol#L43
Status	Acknowledged

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.

Team Update

The team has acknowledged that this cannot be exploited maliciously.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	trustless-management/TrustlessManagement.sol#L81,89,97,105,119
Status	Acknowledged

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.

```
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.

Team Update

The team has acknowledged that the permissions are required for the proper operation of the application.

MPS - Metadata Proper Storage

Criticality	Minor / Informative
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“String is chosen to also be compatible with different kinds of storage (although we will only be using IPFS). For example "https" or "ar" (arweave) could also be used by other parties (or us in the future).”

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	tag-manager/ERC721TagManager.sol#L54,93
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.

```
accountToId[tokenOwner] = tokenId;
accountToId[msg.sender] = tokenId;
```

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#L61,90,92 openrfp/RFPs.sol#L83,84,85 optimistic-actions/OptimisticActions.sol#L32
Status	Acknowledged

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 instance, 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 to 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.

Team Update

The team replied with the following statement:

“I see this as the users responsibility, if they want to make a task with the zero address as manager, I won't stop them. Up to the frontend to give warnings / errors about that.”

MU - Modifiers Usage

Criticality	Minor / Informative
Location	optimistic-actions/OptimisticActions.sol#L42,54 openrfp/RFPs.sol#L165,246
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“Generally modifiers are nice, however imo easier to miss and less flexible than explicit method calls. As for OpenR&D we cannot use modifiers in several places, decided to keep it consistent and not use modifiers anywhere in the "project".”

POE - Possible Out-Of-Bounds Error

Criticality	Minor / Informative
Location	openrfp/RFPs.sol#L183,196
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“Correct this will throw an ugly generic "out of bound" error, the additional code overhead of checking for this is in my opinion not worth the benefit of having a prettier error (and possibly earlier revert). The frontend could do this validation and the error will only be possible to originate from 2 places anyhow, should be quick enough to deduct.”

PSS - Potential Staled State

Criticality	Minor / Informative
Location	tag-manager/ERC721TagManager.sol#L47,92
Status	Acknowledged

Description

Within the contract, the `accountToId` mapping is utilized to store the preferred `tokenId` of users. However, there exists a potential issue where this mapping may contain stale or unsynchronized data if a tokenId is burned or transferred. Specifically, if a tokenId is burned or transferred to another user, the `accountToId` mapping may retain outdated information unless explicitly updated by the affected parties through the `setId` function. As a result, this discrepancy in data synchronization may lead to incorrect behavior within the contract's operations.

```
function addTag(uint256 tokenId, bytes32 tag) external onlyRole(tag) {
    _addTag(tokenId, tag);

    // First of your NFTs to get an tag is set as your default (for convenience)
    // Exception: if the tokenId is 0, then any other tokens the account hold
getting tags will overwrite it
    address tokenOwner = collection.ownerOf(tokenId);
    if (accountToId[tokenOwner] == 0) {
        accountToId[tokenOwner] = tokenId;
    }
}
function setId(uint256 tokenId) external {
    accountToId[msg.sender] = tokenId;
}
```

Recommendation

To ensure the integrity and accuracy of the `accountToId` mapping, it is advisable to implement a mechanism for automatic synchronization or update whenever a relevant tokenId transfer or burning operation occurs. Specifically, the team could consider modifying the contract logic to automatically update the `accountToId` mapping whenever a transfer or burn affects the preferred tokenId of a user, thereby eliminating the need for manual intervention by users to maintain synchronization. By enforcing consistent and automated data management practices, the contract can mitigate the risk of data inconsistency and ensure smooth operation in various token-related scenarios.

Team Update

The team has acknowledged that this is the behaviour they want.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Tasks.sol#L376
Status	Acknowledged

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 {
    Task storage task = _getTask(_taskId);
    _ensureTaskIsOpen(task);
    _ensureSenderIsManager(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.

Team Update

The team replied with the following statement:

“Responsibility of the user / frontend, if they want to set the metadata to the exact same value, I dont mind. The only reason I can see them wanting to do this is if they are storing the metadata in a muttible place and want to trigger the metadata changed (to make the indexer cache the new metadata), however the amount of people using this I expect to not be worth the gas usage of calculating the hashes of the strings on every other execution.”

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Tasks.sol#L31,37,51,52,53,54,55,56,125,126,127,128,165,192,208,222,223,224,225,247,271,299,321,332,345,346,347,348,376,387,399,400,401,415,429,438 Escrow.sol#L21
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“Although Solidity best practice does recommend this, I prefer the style to have _ for all function arguments.”

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	TasksUtils.sol#L105,163,228,258,346
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“Tool do not seem to like it, however not initializing uint variables is done on purpose here, as they will start on 0. The extra initialize to 0 has a greater gas costs (although will likely always be removed by the optimizer).”

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	TasksUtils.sol#L380
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“This is part of an OpenZeppelin library, however importing the library caused a bigger contract size, hence it has been copied instead. Assembly is used here for greater gas efficiency.”

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	TasksUtils.sol#L2 TasksEnsure.sol#L2 Tasks.sol#L2 openmesh-admin/OpenmeshENSReverseClaimable.sol#L2 openmesh-admin/Openmesh.sol#L2 ITasks.sol#L2 Escrow.sol#L2 ens-reverse-claimable/ENSReverseClaimable.sol#L2
Status	Acknowledged

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.

Team Update

The team replied with the following statement:

“The ^0.8.0 is to allow other projects to import and compile the contracts for their version (which could be locked by another dependency). From experience having one contract you want to use have a fixed 0.8.17 version, but another be specified as ^0.8.20 is quite a pain. As far as I know all version from 0.8.0 do not have any major concerns for the security of the contract.”

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		
	_ensureTaskClosed	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	-
	increaseReward	External	✓	-
	editMetadata	External	✓	-
	transferManagement	External	✓	-
	completeByDispute	External	✓	-
	partialPayment	External	✓	-
	rescueNative	External	✓	-
	rescue	External	✓	-
	_getTask	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	-
	increaseReward	External	✓	-
	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	External	✓	onlyRole
	setId	External	✓	-
	_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	✓	-

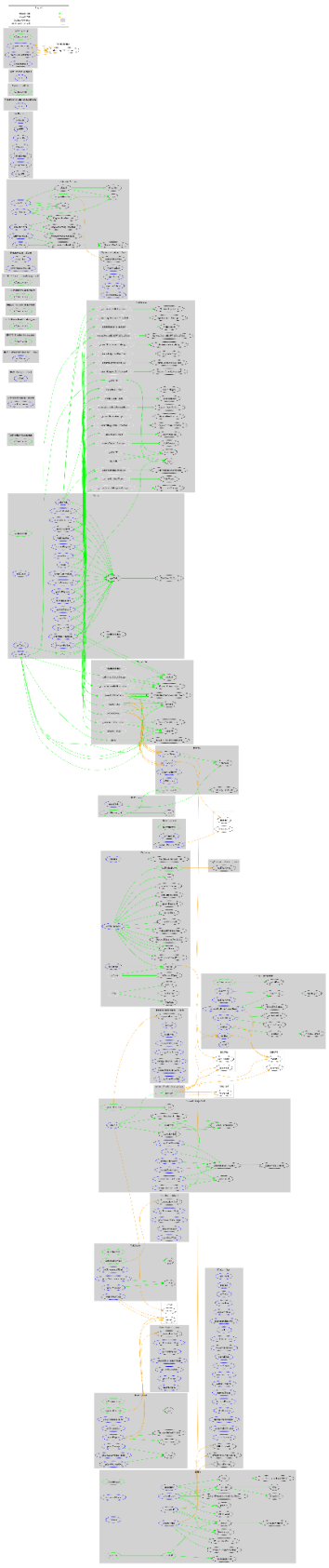
TaskDrafts	Implementation	ERC165, OpenmeshENSReverseClaimable, ITaskDrafts		
		Public	✓	-
	supportsInterface	Public		-
	getGovernancePlugin	External		-
	updateGovernancePlugin	External	✓	-
	updateManager	External	✓	-
	createDraftTask	External	✓	-
ITaskDrafts	Interface			
	getGovernancePlugin	External		-
	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

Openmesh 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